Christos Kaklamanis
Kirk Pruhs (Eds.)

LNCS 8447

Approximation and Online Algorithms

11th International Workshop, WAOA 2013
Sophia Antipolis, France, September 5–6, 2013
Revised Selected Papers

Lecture Notes in Computer Science 8447

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Christos Kaklamanis Kirk Pruhs (Eds.)

Approximation and Online Algorithms

11th International Workshop, WAOA 2013
Sophia Antipolis, France, September 5-6, 2013
Revised Selected Papers

Springer

Volume Editors

Christos Kaklamanis
University of Patras
Computer Technology Institute
and Press “Diophantus”
26504 Rio, Greece
E-mail: kakl@ceid.upatras.gr

Kirk Pruhs
University of Pittsburgh
210 South Bouquet Street
Pittsburgh, PA 15260, USA
E-mail: kirk@cs.pitt.edu

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-319-08000-0 e-ISBN 978-3-319-08001-7
DOI 10.1007/978-3-319-08001-7
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014940940

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The 11th Workshop on Approximation and Online Algorithms (WAOA 2013) focused on the design and analysis of algorithms for online and computationally hard problems. Both kinds of problems have a large number of applications in a wide variety of fields. WAOA 2013 took place in Sophia Antipolis, France, during September 5–6, 2013. The workshop was part of the ALGO 2013 event that also hosted ESA, WABI, IPEC, ALGOSENSORS, MASSIVE, and ATMOS. The previous WAOA workshops were held in Budapest (2003), Rome (2004), Palma de Mallorca (2005), Zurich (2006), Eilat (2007), Karlsruhe (2008), Copenhagen (2009), Liverpool (2010), Saarbrücken (2011), and Leicester (2012). The proceedings of these previous WAOA workshops have appeared as LNCS volumes 2909, 3351, 3879, 4368, 4927, 5426, 5893, 6534, 7164, and 7846, respectively.

Topics of interest for WAOA 2013 were: algorithmic game theory, algorithmic trading, coloring and partitioning, competitive analysis, computational advertising, computational finance, cuts and connectivity, geometric problems, graph algorithms, inapproximability results, mechanism design, natural algorithms, network design, packing and covering, paradigms for the design and analysis of approximation and online algorithms, parameterized complexity, real-world applications, and scheduling problems. In response to the call for papers, we received 33 submissions. Each submission was reviewed by at least three referees. The submissions were mainly judged on originality, technical quality, and relevance to the topics of the conference. Based on the reviews, the Program Committee selected 14 papers. This volume contains final revised versions of these papers.

We would also like to thank all the authors who submitted papers to WAOA 2013 as well as Ola Svensson, who gave an invited talk on "New Approaches for Approximating TSP." Furthermore, we are grateful to the local organizers of ALGO 2013: Frédéric Cazals, Agnès Cortell, David Coudert, Olivier Devillers, Joanna Moulierac, and Monique Teillaud (Chair).

March 2014

Christos Kaklamanis
Kirk Pruhs

Organization

Program Co-chairs

Christos Kaklamanis	University of Patras and CTI, Greece
Kirk Pruhs	University of Pittsburgh, USA

Program Committee

Vincenzo Bonifaci	IASI-CNR, Italy
Niv Buchbinder	Tel Aviv University, Israel
Matthias Englert	University of Warwick, UK
Leah Epstein	University of Haifa, Israel
Bruno Escoffier	Université Paris Dauphine, France
Dimitris Fotakis	National Technical University of Athens, Greece
Fabrizio Grandoni	University of Lugano, Switzerland
Anupam Gupta	Carnegie Mellon University, USA
Łukasz Jeż	University of Wrocław, Poland
Sungjin Im	Duke University, USA
Christos Kaklamanis (Co-chair)	University of Patras and CTI, Greece
Bodo Manthey	University of Twente, The Netherlands
Luca Moscardelli	University of Chieti-Pescara, Italy
Benjamin Moseley	Toyota Technological Institute at Chicago, USA
Viswanath Nagarajan	IBM Research, USA
Debmalya Panigrahi	Duke University, USA
Kirk Pruhs (Co-chair)	University of Pittsburgh, USA
Adi Rosen	CNRS and Université Paris Diderot, France
Anastasios Sidiropoulos	UIUC, USA
Rene Sitters	Vrije Universiteit Amsterdam, The Netherlands
Kavitha Telikepalli	Tata Institute of Fundamental Research, India
Kasturi Varadarajan	University of Iowa, USA

Additional Reviewers

Marek Adamczyk	Paul Bonsma
Haris Angelidakis	Denise Duma
Marcin Bienkowski	Alina Ene

Andreas Galanis
Laurent Gourves
Csanad Imreh
Nikos Karanikolas
Robert Kleinberg
Maria Kyropoulou
Asaf Levin
Nutan Limaye
George Mertzios
Jerome Monnot
Katarzyna Paluch
Dror Rawitz
Marc Renault
Bartosz Rybicki
Alejandro Salinger
Florian Sikora
Aravindan Vijayaraghavan

Table of Contents

Min-Sum 2-Paths Problems

Trevor Fenner[1], Oded Lachish[1], and Alexandru Popa[2]

[1] Birkbeck, University of London, London, UK
{trevor,oded}@dcs.bbk.ac.uk
[2] Faculty of Informatics, Masaryk University, Brno, Czech Republic
popa@fi.muni.cz

Abstract. An *orientation* of an undirected graph G is a directed graph obtained by replacing each edge $\{u, v\}$ of G by exactly one of the arcs (u, v) or (v, u). In the *min-sum k-paths orientation problem*, the input is an undirected graph G and ordered pairs (s_i, t_i), where $i \in \{1, 2, \ldots, k\}$. The goal is to find an orientation of G that minimizes the sum over every $i \in \{1, 2, \ldots, k\}$ of the distance from s_i to t_i.

In the *min-sum k edge-disjoint paths problem* the input is the same, however the goal is to find for every $i \in \{1, 2, \ldots, k\}$ a path between s_i and t_i so that these paths are edge-disjoint and the sum of their lengths is minimum. Note that, for every fixed $k \geq 2$, the question of **NP**-hardness for the min-sum k-paths orientation problem and the min-sum k edge-disjoint paths problem have been open for more than two decades. We study the complexity of these problems when $k = 2$.

We exhibit a PTAS for the min-sum 2-paths orientation problem. A by-product of this PTAS is a reduction from the min-sum 2-paths orientation problem to the min-sum 2 edge-disjoint paths problem. The implications of this reduction are: (i) an **NP**-hardness proof for the min-sum 2-paths orientation problem yields an **NP**-hardness proof for the min-sum 2 edge-disjoint paths problem, and (ii) any approximation algorithm for the min-sum 2 edge-disjoint paths problem can be used to construct an approximation algorithm for the min-sum 2-paths orientation problem with the same approximation guarantee and only an additive polynomial increase in the running time.

1 Introduction

In communications, *Multihoming* is the process of communicating through more than one connection. The goal is to increase communication reliability. Now imagine that each connection must be made between two distinct entities, for example, if a customer has numerous internet providers, each with a distinct entry point that requires a connection to a distinct end-point, see [1,8]. This is the case we deal with here.

In order to optimize reliability when using multiple connections a natural goal is that the channels are disjoint. We model the problem of determining whether such channels exist with the *k edge-disjoint paths problem*, where the input is an instance consisting of a graph and pairs of vertices $\{s_i, t_i\}$, where $i \in \{1, 2, \ldots, k\}$, and the goal is to find k edge-disjoint paths between the k pairs $\{s_i, t_i\}$. Robertson and Seymour proved in [7] that, for fixed k, this problem is in $\mathbf{P}$.

C. Kaklamanis and K. Pruhs (Eds.): WAOA 2013, LNCS 8447, pp. 1–11, 2014.

However, just having k edge-disjoint paths is often not sufficient. A natural requisite is that the paths found are optimized according to some condition. Such conditions can be minimum maximal length or minimum sum of lengths. These conditions lead to two optimization problems: the first is known as the *min-max k edge-disjoint paths problem*; and the latter as the *min-sum k edge-disjoint paths problem*. In [6] Li et al. show that the min-max k edge-disjoint paths problem is **NP**-hard, even when $k = 2$ and $\{s_1, t_1\} = \{s_2, t_2\}$. In contrast, the question of **NP**-hardness of the min-sum k edge-disjoint paths problem for fixed $k \geq 2$ has been open for more than twenty years.

An *orientation* of an undirected graph G is a directed graph obtained by replacing each edge $\{u, v\}$ of G by exactly one of the arcs (u, v) or (v, u). In the *min-sum k-paths orientation problem*, the input instance is an undirected graph G and ordered pairs (s_i, t_i), where $i \in \{1, 2, \ldots, k\}$. The goal is to find an orientation of G in which the sum over all $i \in \{1, 2, \ldots, k\}$ of the distance from s_i to t_i is minimized. The min-sum k-paths orientation problem is a relaxation of the min-sum k edge-disjoint paths problem in the following sense: if the requirement for a path between s_i and t_i for each $i \in \{1, 2, \ldots, k\}$ is replaced by the requirement for an unsplittable flow of size 1 from s_i to t_i for each $i \in \{1, 2, \ldots, k\}$ and these flows may share edges if they are in the same direction, then we get the min-sum k-paths orientation problem. We note that the question of **NP**-hardness for the min-sum k-paths orientation problem, for fixed $k \geq 2$, has also been open for more than twenty years. In this paper we focus on the min-sum 2-paths orientation problem and its relation with the min-sum 2 edge-disjoint paths problem.

There have been a number of results for the min-sum k edge-disjoint paths problem. Zhang and Zhao [10] have shown that in general graphs for general k the min-sum k edge-disjoint paths problem is FP^{NP}-complete. They gave a bicriteria approximation algorithm for the problem. There have also been a number of results for the min-sum 2 edge-disjoint paths problem. Zhang and Zhao have shown that this problem has a constant factor approximation. Kobayashi and Sommer [5] showed that the problem is in $\mathbf{P}$ if G is planar and s_1, t_1, s_2 and t_2 are on at most two faces of the graph. Kammer et al. [4] showed that it is in $\mathbf{P}$ if G is a chordal graph. For a comprehensive discussion of results, see Kobayashi and Sommer [5].

Finally, the min-sum k-paths orientation problem has been studied by Hassin and Megiddo [2]. There they showed that this problem is $\mathbf{NP}$-hard for general k. They also studied the *min-max k paths-orientation problem*. They proved that this problem is $\mathbf{NP}$-hard even for $k = 2$. In [3], Ito et al. also studied these two problems. They showed that, for unrestricted k, the min-sum k-paths orientation problem does not have a polynomial time algorithm with an approximation factor of 2 or less, unless $\mathbf{P} = \mathbf{NP}$. They presented approximation algorithms for restricted variations of this problem, for example, for certain classes of graphs, such as cacti.

In this paper, we exhibit a PTAS for the min-sum 2-paths orientation problem. A by-product of this PTAS is a reduction from the min-sum 2-paths orientation problem to the min-sum 2 edge-disjoint paths problem. The implications of this reduction are: (i) that an $\mathbf{NP}$-hardness proof for the min-sum 2-paths orientation problem yields an $\mathbf{NP}$-hardness proof for the min-sum 2 edge-disjoint paths problem, and (ii) that any approximation algorithm for the min-sum 2 edge-disjoint paths problem can be used

to construct an approximation algorithm for the min-sum 2-paths orientation problem with the same approximation guarantee and only an additive polynomial increase in the running time. Our results suggest that if indeed the min-sum 2-paths orientation problem is **NP**-hard, then proving this may be more difficult than it seems because of the implication for the min-sum 2 edge-disjoint paths problem. The reduction also implies, according to results by Kobayashi and Sommer [5] and Kammer et al. [4] for the min-sum 2 edge-disjoint paths problem, that the orientation problem is in **P** if G is chordal or if it is planar and s_1, t_1, s_2 and t_2 are on at most two faces of the graph.

One of the central ingredients we use is a structural lemma that states that for any given input instance (G, s_1, t_1, s_2, t_2), if there exists an orientation in which the distances from s_1 to t_1 and from s_2 to t_2 are both finite there exists an optimal orientation with two min-sum directed paths, one from s_1 to t_1 and the other from s_2 to t_2, such that either (i) these directed paths are arc-disjoint, or (ii) the directed paths are not arc-disjoint and their common edges form a directed-path. We obtain the reduction to the min-sum 2 edge-disjoint paths problem by showing that if, on the same input instance, we execute an algorithm for min-sum 2 edge-disjoint problem and an algorithm that works if (ii) holds, then the best result is optimal. We obtain the PTAS in a similar manner, by showing that a PTAS exists for instances on which (i) holds.

2 Preliminaries

We use $[k]$ to denote the set $\{1, 2, \ldots, k\}$. An undirected *graph* is an ordered pair $G = (V, E)$, where V is a set of *vertices* and E is a set of *edges*, each edge being a subset of V of size two. A *directed graph* is an ordered pair $\boldsymbol{G} = (V, \boldsymbol{E})$, where V is a set of vertices and $\boldsymbol{E}$ is a set of ordered pairs of vertices of V called *arcs*. We use the notation $V(G)$ for the set of vertices of G or $\boldsymbol{G}$ and $E(G)$ for the set of edges of G, and $E(\boldsymbol{G})$ for the set of arcs of $\boldsymbol{G}$. When clear from the context we use n instead of $|V(G)|$.

Definition 1. [Orientation] *An **orientation** of an undirected graph $G = (V, E)$ is a directed graph $\boldsymbol{H} = (V, \boldsymbol{E})$ such that, for every $\{u, v\} \in E$, either $(u, v) \in \boldsymbol{E}$ or $(v, u) \in \boldsymbol{E}$, but not both. We use the notation $\boldsymbol{H}_G$ to denote that $\boldsymbol{H}$ is an orientation of G.*

A *path* P or a *dipath* $\boldsymbol{P}$ in G or $\boldsymbol{G}$, respectively, is a tuple $(u_1, u_2, \ldots, u_k) \in V^k$ such that for every $i \in [k-1]$ we have that $\{u_i, u_{i+1}\} \in E(G)$ or $(u_i, u_{i+1}) \in E(\boldsymbol{G})$, respectively, and $u_1, u_2, \ldots, u_k$ are all distinct. The path $(u, \ldots, v)$ in G is a path **between** u **and** v. The dipath $(u, \ldots, v)$ in $\boldsymbol{G}$ is a dipath **from** u **to** v. We use the notation $P_{u,v}$ to indicate that the path is between u and v, and the notation $\boldsymbol{P}_{u,v}$ to indicate that the dipath is from u to v. A *cycle* in G is a tuple $C = (u_1, u_2, \ldots, u_k, u_1) \in V^{k+1}$ such that $(u_1, u_2, \ldots, u_k)$ is a path and $\{u_k, u_1\} \in E(G)$. Note that we often consider a path to be a subgraph.

A path $P' = (u_1, \ldots, u_\ell)$ in a graph is a *subpath* of the path $P = (v_1, \ldots, v_k)$ if there exists $i \in [k-\ell+1]$ such that $(u_1, u_2, \ldots, u_\ell) = (v_i, v_{i+1}, \ldots, v_{i+\ell-1})$. A graph (V', E') is a *subgraph* of a graph (V, E) if $V' \subseteq V$ and $E' \subseteq E$.

The *length* of P or $\boldsymbol{P}$, denoted by $len(P)$ or $len(\boldsymbol{P})$, respectively, is $k-1$. The *distance* between u and v in $V(G)$, denoted by $dist_G(u, v)$, is the length of a shortest

path between u and v if such a path exists, and $dist_G(u,v) = \infty$ otherwise. The *distance* between a pair of paths P and P' in G, denoted by $dist_G(P,P')$, is the minimal distance between a vertex in $V(P)$ and a vertex in $V(P')$.

The *distance* from $u \in V(G)$ to $v \in V(G)$ in a directed graph $\boldsymbol{G}$, denoted by $dist_{\boldsymbol{G}}(u,v)$, is the length of a shortest dipath from u to v if such a dipath exists, and $dist_{\boldsymbol{G}}(u,v) = \infty$ otherwise. When the graph under consideration is clear from context, we simply write $dist(u,v)$.

Definition 2. **[$B_G(v,x)$]** *Let G be a graph, $v \in V(G)$ and $x > 0$. Then $B_G(v,x)$ is the subset of $E(G)$ containing all the edges $\{u,w\} \in E(G)$ such that $dist_G(v,u) < x$ and $dist_G(v,w) < x$.*

Definition 3. [Instance] *An **instance** is an ordered tuple (G, s_1, t_1, s_2, t_2) such that G is an undirected graph and s_1, t_1, s_2 and t_2 are vertices in $V(G)$.*

Problem 1 (Min-Sum 2 Edge-Disjoint Paths). Given an instance (G, s_1, t_1, s_2, t_2), find edge disjoint paths P_{s_1,t_1} and P_{s_2,t_2} such that $len(P_{s_1,t_1}) + len(P_{s_2,t_2})$ is minimum.

2.1 The Min-Sum *2* Paths Orientation Problem

Problem 2 (Min-Sum 2 Paths Orientation). Given an instance (G, s_1, t_1, s_2, t_2), find an orientation $\boldsymbol{H}_G$ of G that minimizes $dist_{\boldsymbol{H}_G}(s_1,t_1) + dist_{\boldsymbol{H}_G}(s_2,t_2)$. We call such an orientation an ***optimal orientation***.

Definition 4. **[OPT]** *Let (G, s_1, t_1, s_2, t_2) be an instance. We define $OPT(G, s_1, t_1, s_2, t_2) = dist_{\boldsymbol{H}_G}(s_1,t_1) + dist_{\boldsymbol{H}_G}(s_2,t_2)$ for any optimal orientation $\boldsymbol{H}_G$. We write OPT when the instance under consideration is clear from the context.*

We make the following definition in order to recast the problem in terms of undirected graphs.

Definition 5. [Non-conflicting paths and optimal paths] *Let G be an undirected graph and $x_1, y_1, x_2, y_2 \in V(G)$. Paths P_{x_1,y_1} and P_{x_2,y_2} in G are **non-conflicting** if there exists an orientation $\boldsymbol{H}_G$ in which $\boldsymbol{P}_{x_1,y_1} = P_{x_1,y_1}$ and $\boldsymbol{P}_{x_2,y_2} = P_{x_2,y_2}$ and are **optimal** if they are non-conflicting and $len(P_{x_1,y_1}) + len(P_{x_2,y_2}) = OPT(G, x_1, y_1, x_2, y_2)$ for the instance (G, x_1, y_1, x_2, y_2).*

Observe that for any optimal orientation $\boldsymbol{H}_G$ for an instance (G, s_1, t_1, s_2, t_2) any two shortest dipaths $(s_1, \ldots, t_1)$ and $(s_2, \ldots, t_2)$ in $\boldsymbol{H}_G$ are an optimal pair of paths and in particular a non-conflicting pair of paths. We note that checking whether two paths are non-conflicting can easily be done in polynomial time. By the following observation, we see that, in order to show that $OPT(G, s_1, t_1, s_2, t_2) \leq k$, it is sufficient to find non-conflicting paths P_{s_1,t_1} and P_{s_2,t_2} such that $len(P_{s_1,t_1}) + len(P_{s_2,t_2}) \leq k$.

Observation 1. *Let (G, s_1, t_1, s_2, t_2) be an instance. If P_{s_1,t_1} and P_{s_2,t_2} are non-conflicting, then $OPT(G, s_1, t_1, s_2, t_2) \leq len(P_{s_1,t_1}) + len(P_{s_2,t_2})$.*

Without loss of generality, we always make the following assumption:

Assumption 2. *For every given instance (G, s_1, t_1, s_2, t_2), we assume that $OPT < \infty$, G is connected and that s_1, t_1, s_2, t_2 are distinct.*

We may make this assumption since it is easy to decide whether $OPT = \infty$ and the problem on an instance (G, s_1, t_1, s_2, t_2) such that s_1, t_1, s_2 and t_2 are not distinct can be easily reduced to the problem on an instance $(G', s'_1, t'_1, s'_2, t'_2)$ where s'_1, t'_1, s'_2 and t'_2 are distinct.

3 Algorithm Overview and Definitions

We start by giving an algorithm that finds an optimal pair of paths for a restricted set of instances. Afterwards we explain how to obtain our claimed results by extending this algorithm.

Let (G, s_1, t_1, s_2, t_2) be an instance that has an optimal pair of edge-disjoint paths P_{s_1,t_1} and P_{s_2,t_2} such that $dist(P_{s_1,t_1}, P_{s_2,t_2}) > dist(s_1, t_1)/2$. Consequently, any shortest path between s_1 and t_1 does not intersect P_{s_2,t_2}. For such an instance finding an optimal pair of paths can be done as follows: (i) find a shortest path P'_{s_1,t_1} (ii) let G' be the graph resulting from removing the edges of P'_{s_1,t_1} from G, and (iii) find a shortest path P'_{s_2,t_2} in G'. We refer to this as the *simple algorithm*.

Observe that G' contains all the edges of P_{s_2,t_2}, since P'_{s_1,t_1} and P_{s_2,t_2} are edge disjoint. Hence, $len(P'_{s_2,t_2}) \leq len(P_{s_2,t_2})$. Since P'_{s_1,t_1} is also a shortest path $len(P'_{s_1,t_1}) \leq len(P_{s_1,t_1})$. Consequently, P'_{s_1,t_1} and P'_{s_2,t_2} are an optimal pair of paths.

We have demonstrated that, if an instance has optimal pair that are sufficiently far from each other, then the problem of finding an optimal pair of paths requires only polynomial time. The distance between the paths of an optimal pair is crucial for our results. Hence, we make the following definition.

Definition 6. **[$\Delta(G, s_1, t_1, s_2, t_2)$ and $\delta(G, s_1, t_1, s_2, t_2)$]** *Let (G, s_1, t_1, s_2, t_2) be an instance. We define $\Delta(G, s_1, t_1, s_2, t_2)$ and $\delta(G, s_1, t_1, s_2, t_2)$ to be the maximum and minimum, respectively, of $dist_G(P_{s_1,t_1}, P_{s_2,t_2})/OPT$ over all optimal pairs of paths P_{s_1,t_1} and P_{s_2,t_2}. We write just Δ and δ when the instance under consideration is clear from the context.*

Obviously, $\delta \leq \Delta$. Note that if $\Delta(G, s_1, t_1, s_2, t_2) > 1/2$, then we can use the simple algorithm to find an optimal pair of paths. For our results we need something stronger. We next describe an algorithm, similar in essence to the simple algorithm, which for input $\epsilon > 0$ and instance (G, s_1, t_1, s_2, t_2) finds an optimal pair of paths in time $n^{O(1/\epsilon)}$ if $\epsilon < \Delta$. Our final algorithm is a slight variation of this.

Let $\epsilon > 0$ such that $\epsilon < \Delta$ and suppose that P_{s_1,t_1} and P_{s_2,t_2} are an optimal pair of paths $\Delta \cdot OPT$ apart. Suppose also that we have a set of $h = O(1/\epsilon)$ vertices $u_1, u_2, \ldots, u_h \in V(P_{s_1,t_1})$, where $u_1 = s_1$, $u_h = t_1$ and $dist_{P_{s_1,t_1}}(u_i, u_{i+1}) < \epsilon \cdot OPT$ for each $i \in [h-1]$. Now apply the following algorithm, which we call the *basic algorithm*: (i) find a shortest path P'_{s_1,t_1} in the graph $\left(V(G), \bigcup_{i\in[h]} B_G(u_i, \epsilon \cdot OPT)\right)$, (ii) find a shortest path P'_{s_2,t_2} in the graph $\big(V(G), E(G) \setminus E(P'_{s_1,t_1})\big)$. We show that P'_{s_1,t_1} and P'_{s_2,t_2} are an optimal pair of paths.

First observe that all the edges of P_{s_1,t_1} are contained in $\bigcup_{i\in[h]} B_G(u_i, \epsilon \cdot OPT)$ and hence $len(P'_{s_1,t_1}) \leq len(P_{s_1,t_1})$. Since $\epsilon < \Delta$, by Definition 6, P_{s_2,t_2} and $B_G(u_i, \epsilon \cdot OPT)$ are edge-disjoint for each $i \in [h]$. Thus, P'_{s_1,t_1} and P_{s_2,t_2} are also edge-disjoint. It follows, from the simple algorithm, that P'_{s_1,t_1} and P'_{s_2,t_2} are an optimal pair of paths.

Therefore for the rest of this section we assume that $\epsilon \geq \Delta$. In order to deal with this case, we now prove a structural result that states that any non-trivial instance is of at least one of the following two types.

Definition 7. [Disjoint Instance and Intersecting Instance] *An instance (G, s_1, t_1, s_2, t_2) is **disjoint** if it has an optimal pair of paths P_{s_1,t_1} and P_{s_2,t_2} that are edge-disjoint. An instance (G, s_1, t_1, s_2, t_2) is **intersecting** if it has an optimal pair of paths P_{s_1,t_1} and P_{s_2,t_2} that are not edge-disjoint and whose common edges form a subpath of both P_{s_1,t_1} and P_{s_2,t_2}.*

The proof of the following lemma is left for the full version.

Lemma 1. *Let (G, s_1, t_1, s_2, t_2) be an instance for which $OPT < \infty$, then (G, s_1, t_1, s_2, t_2) is disjoint or intersecting (or both).*

Suppose that P_{s_1,t_1} and P_{s_2,t_2} are an optimal pair of paths that either: (i) are edge-disjoint and $dist(P_{s_1,t_1}, P_{s_2,t_2}) = \Delta \cdot OPT$, or (ii) all their common edges form a path P_{m_0,m_1}. In case (i) let m_0 and m_1 be such that $dist(m_0, m_1) = \delta \cdot OPT$ where m_0 is on one of the paths P_{s_1,t_1} and P_{s_2,t_2} and m_1 is on the other. We note that $m_0 = m_1$ if $\delta = 0$. Since there are less than n^2 potential pairs, we can assume we have one since we can try each pair in turn. The advantage of knowing such a pair $\{m_0, m_1\}$ is that every shortest path between m_0 and m_1 is edge disjoint from both P_{s_1,t_1} and P_{s_2,t_2}. We call such a pair a *pivot.* More formally:

Definition 8. ***[Pivot]** Let (G, s_1, t_1, s_2, t_2) be an instance. A **pivot** is a pair $\{m_0, m_1\}$ such that one of the following holds:*

1. *(G, s_1, t_1, s_2, t_2) is disjoint and $dist_G(m_0, m_1) = \delta \cdot OPT$ for some optimal pair of edge-disjoint paths P_{s_1,t_1} and P_{s_2,t_2}, where m_0 is in one of these paths and m_1 is in the other, or*
2. *(G, s_1, t_1, s_2, t_2) is intersecting with optimal paths whose common edges form a path P_{m_0,m_1}.*

*If Case 1 holds, then $\{m_0, m_1\}$ is a **disjoint-pivot**, and if Case 2 holds, then $\{m_0, m_1\}$ is an **intersecting-pivot**. In both cases P_{m_0,m_1} is necessarily a shortest path.*

Let $\{m_0, m_1\}$ be a pivot and P_{m_0,m_1} be a shortest path. The naive way of proceeding is to use a min-cost single source flow algorithm for a flow of size 4 as follows: (i) let G' be obtained from G by adding a vertex a and four edges: two between a and m_0 and two between a and m_1; (ii) solve the min-cost single source flow with a being the source, s_1, t_1, s_2 and t_2 being the targets, and all edges of G' having capacity and cost 1. This will result in four min-sum edge-disjoint paths P_{x_1,m_0}, P_{x_2,m_0}, P_{x_3,m_1}, P_{x_4,m_1}, where $\{x_1, x_2, x_3, x_4\} = \{s_1, t_1, s_2, t_2\}$. Now the natural conjecture is that a non-conflicting

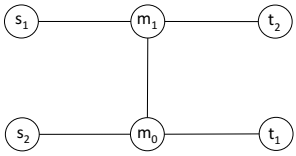

Fig. 1. [Naive Attempt]

pair of paths as required can be found in the graph consisting of all vertices and edges of these four paths and of P_{m_0,m_1}. However, this idea does not work if the configuration obtained is like that in Figure 1. Thus, a different strategy is required. The strategy we use in Section 4 is to first find a min-sum edge-disjoint pair P_{s_1,m_i}, $P_{t_2,m_{1-i}}$, where $i \in \{0,1\}$ and then a min-sum edge-disjoint pair P_{s_2,m_j}, $P_{t_1,m_{j-i}}$, where $j \in \{0,1\}$. We shall show that the graph consisting of the vertices and edges of these four paths and of P_{m_0,m_1} is sufficient for finding the required non-conflicting pair of paths.

In Section 5, we introduce the algorithm that works when $\epsilon < \Delta$ and in Section 6 we prove the main results.

4 Algorithm 1

We introduce here the algorithm for the case that Δ is small ($\Delta \leq \epsilon$) and hence δ is even smaller.

Theorem 3. *Let $P^*_{s_1,t_1}$ and $P^*_{s_2,t_2}$ be the paths returned by Algorithm 1 on instance (G, s_1, t_1, s_2, t_2). Then $P^*_{s_1,t_1}$ and $P^*_{s_2,t_2}$ are non-conflicting and $len(P^*_{s_1,t_1}) +$*

Algorithm 1

Input: instance (G, s_1, t_1, s_2, t_2)

- Iterate over all pairs of vertices $\{m_0, m_1\} \subseteq V$
  1. $P_{m_0,m_1} \longleftarrow$ an arbitrary shortest path between m_0 and m_1
  2. $G' \longleftarrow (V(G), E(G) \setminus E(P_{m_0,m_1}))$
  3. In G', find min-sum edge-disjoint paths P'_{s_1,m_i} and $P'_{t_2,m_{1-i}}$, where $i \in \{0,1\}$
  4. In G', find min-sum edge-disjoint paths P'_{s_2,m_j} and $P'_{t_1,m_{1-j}}$, where $j \in \{0,1\}$
  5. Define Q to be the undirected graph such that
     (a) $V(Q) = V(P'_{s_1,m_i}) \cup V(P'_{t_2,m_{1-i}}) \cup V(P'_{s_2,m_j}) \cup V(P'_{t_1,m_{1-j}}) \cup V(P_{m_0,m_1})$
     (b) $E(Q) = E(P'_{s_1,m_i}) \cup E(P'_{t_2,m_{1-i}}) \cup E(P'_{s_2,m_j}) \cup E(P'_{t_1,m_{1-j}}) \cup E(P_{m_0,m_1})$
  6. Using the method explained in Lemma 2, find non-conflicting paths $P^{m_0,m_1}_{s_1,t_1}$ and $P^{m_0,m_1}_{s_2,t_2}$ in Q such that

$$len(P^{m_0,m_1}_{s_1,t_1}) + len(P^{m_0,m_1}_{s_2,t_2}) \leq$$
$$len(P'_{s_1,m_i}) + len(P'_{t_2,m_{1-i}}) + len(P'_{s_2,m_j}) + len(P'_{t_1,m_{1-j}}) + 2 \cdot len(P_{m_0,m_1})$$

Output: The paths $P^{m_0,m_1}_{s_1,t_1}$ and $P^{m_0,m_1}_{s_2,t_2}$ that minimize $len(P^{m_0,m_1}_{s_1,t_1}) + len(P^{m_0,m_1}_{s_2,t_2})$

$len(P^*_{s_2,t_2}) \leq (1+2\delta) \cdot OPT$. *The running time of Algorithm 1 is bounded by a polynomial function of* n.

[Note that if the input instance is intersecting, then $\delta = 0$ and hence Algorithm 1 returns an optimal pair of paths.]

Proof. Finding the paths in Steps 3 and 4 can be done by reducing the problem to finding edge-disjoint paths from a single vertex as follows. Add to the graph G', computed in Step 2, a vertex a and edges $\{a, m_0\}$ and $\{a, m_1\}$. Then find a pair of min-sum edge-disjoint paths each from a to s_1 and t_2 in Step 3 and s_2 and t_1 in Step 4. According to Yang et al. [9] this requires a running time of $O(n^2)$. By Lemma 2 below, Step 6 requires a running time that is polynomial in n. Since all other steps also require at most a polynomial in n running time and there are fewer than n^2 iteration of Steps 1 to 6, the overall running time is polynomial in n.

Suppose that $\{m_0, m_1\}$ is a pivot with associated optimal pair of paths P_{s_1,t_1} and P_{s_2,t_2} and that P_{m_0,m_1} is the shortest path found in Step 1. Let $\xi = len(P'_{s_1,m_i}) + len(P'_{t_2,m_{1-i}}) + len(P'_{s_2,m_j}) + len(P'_{t_1,m_{1-j}}) + 2 \cdot len(P'_{m_0,m_1})$.

As noted earlier, when $\{m_0, m_1\}$ is a disjoint-pivot, then P_{m_0,m_1} does not share any edges with P_{s_1,t_1} and P_{s_2,t_2}. Consequently, the paths found Step 3 and 4 have overall at most OPT edges. Hence, $\xi \leq OPT \cdot (1+2\delta)$. By Lemma 2, the paths $P^{m_0,m_1}_{s_1,t_1}$ and $P^{m_0,m_1}_{s_2,t_2}$ found in Step 6 are non-conflicting and $len(P^{m_0,m_1}_{s_1,t_1}) + len(P^{m_0,m_1}_{s_2,t_2}) \leq \xi \leq (1+2\delta) \cdot OPT$.

If $\{m_0, m_1\}$ is an intersecting-pivot, then by definition, all the edges that P_{m_0,m_1} shares with P_{s_1,t_1} or P_{s_2,t_2} are edges common to both. Consequently, the paths found in Step 3 and 4 have overall at most $OPT - 2 \cdot len(P_{m_0,m_1})$ edges. Hence, $\xi \leq OPT$. In this case, by Lemma 2,the paths $P^{m_0,m_1}_{s_1,t_1}$ and $P^{m_0,m_1}_{s_2,t_2}$ found in Step 6 are non-conflicting and $len(P^{m_0,m_1}_{s_1,t_1}) + len(P^{m_0,m_1}_{s_2,t_2}) \leq \xi$. Consequently, since $\xi \leq OPT$, these paths are an optimal pair of paths.

The proof of the following lemma is left for the journal version.

Lemma 2. *Let* (G, s_1, t_1, s_2, t_2) *be the input to Algorithm 1. Assume that* Q, P'_{t_1,m_i}, $P'_{t_2,m_{1-i}}$, P'_{s_2,m_j}, $P'_{t_1,m_{1-j}}$ *and* P_{m_0,m_1} *are as computed by Algorithm 1 in an iteration using* $\{m_0, m_1\}$. *Let* $\xi = len(P'_{s_1,m_i}) + len(P'_{t_2,m_{1-i}}) + len(P'_{s_2,m_j}) + len(P'_{t_1,m_{1-j}}) + 2 \cdot len(P_{m_0,m_1})$. *Then there exists a procedure that runs in time polynomial in* n *that finds non-conflicting paths* $P^{m_0,m_1}_{s_1,t_1}$ *and* $P^{m_0,m_1}_{s_2,t_2}$ *in* Q *with* $len(P^{m_0,m_1}_{s_1,t_1}) + len(P^{m_0,m_1}_{s_2,t_2}) \leq \xi$.

5 Algorithm 2

The input to Algorithm 2 consists of an instance (G, s_1, t_1, s_2, t_2), $\gamma > 0$ and $d \in [n]$. The additional parameter d is required for using this algorithm in both the additive and multiplicative approximation modes. We prove here that, if $\gamma \cdot OPT \leq \gamma d \leq \Delta \cdot OPT$, then Algorithm 2 returns an optimal pair of paths in time $(n/(\gamma d))^{O(1/\gamma)} \cdot poly(n)$.

Algorithm 2 is a variation of the basic algorithm described in Section 3, which works when the input instance has an optimal pair of paths that are far from each other. It is

used because it has a better running time when OPT is large, which is essential for the additive approximation. We next explain how it differs from the basic algorithm.

Suppose that the input (G, s_1, t_1, s_2, t_2) satisfies $\gamma \cdot OPT \leq \gamma d \leq \Delta \cdot OPT$ and that P_{s_1,t_1} and P_{s_2,t_2} are an optimal pair of paths that are $\Delta \cdot OPT$ apart. Recall that the basic algorithm, required finding specific vertices $u_1, u_2, \ldots, u_h$ in P_{s_1,t_1}. These vertices can be found via exhaustive search over all relevant subsets of $V(G)$. Algorithm 2 is almost the same as the basic algorithm except that the vertices $u_1, u_2, \ldots, u_h$ are selected from a subset of $V(G)$, which we call *representatives*, and this subset may be significantly smaller than $V(G)$. The relevant parameters for choosing this set are γ and d.

A set of representatives S has the property that every vertex in $V(G)$ is very close to a vertex in S. Consequently, the approach used in the basic algorithm will work when we use representatives. We now formally define the set of representatives and prove that such a set always exists. We then present the algorithm and prove its correctness.

Definition 9. **[$Rep_G(\ell)$]** *Given G and $\ell > 0$, let $Rep_G(\ell)$ be an arbitrary subset of $V(G)$ such that: (i) for every $u \in V(G)$, there exists $v \in Rep_G(\ell)$ such that $dist(u, v) < \ell$; and (ii) $|Rep_G(\ell)| \leq 2n/\ell$.*

Lemma 3. *For every connected graph G and $\ell > 0$, there exists a set $Rep_G(\ell)$ satisfying Definition 9.*

Proof. Initially set $Rep_G(\ell) = \{u\}$, where u is an arbitrary vertex from $V(G)$. Afterwards add vertices to $Rep_G(\ell)$ in the following manner. If there is a vertex in $V(G) \setminus Rep_G(\ell)$ whose distance from every other vertex in $Rep_G(\ell)$ is greater than ℓ, then add it to $Rep_G(\ell)$, otherwise stop. This process eventually ends since $V(G)$ is finite. Every vertex in $V(G)$ has distance not exceeding ℓ to some vertex in $Rep_G(\ell)$ because either it is in the set or it was not added. Thus, the minimum distance between any pair of distinct vertices in $Rep_G(\ell)$ is ℓ. Therefore, since G is connected, if $|Rep_G(\ell)| > 1$, then for all $v \in Rep_G(\ell)$ there are at least $\lceil \ell/2 \rceil$ distinct vertices (including v itself) whose distance from v is less than their distance to any other vertex in $Rep_G(\ell)$. Consequently, $|Rep_G(\ell)| \leq 2n/\ell$.

Algorithm 2

Input: an instance (G, s_1, t_1, s_2, t_2), $\gamma > 0$ and $d \in [n]$

1. $P^*_{s_1,t_1} \longleftarrow \emptyset$, $P^*_{s_2,t_2} \longleftarrow \emptyset$
2. Iterate over all $S^* \subseteq Rep_G(\gamma d/4)$ such that $|S^*| \leq \lceil 8/\gamma \rceil$
   (a) $P'_{s_1,t_1} \longleftarrow$ an arbitrary shortest path in $\left(V(G), \bigcup_{v \in S^* \cup \{s_1,t_1\}} B_G(v, \gamma d/2)\right)$ between s_1 and t_1, if one exists, and $\emptyset$ otherwise
   (b) $P'_{s_2,t_2} \longleftarrow$ an arbitrary shortest path in $(V, E(G) \setminus E(P'_{s_1,t_1}))$ path between s_2 and t_2, if one exists, and $\emptyset$ otherwise
   (c) If $P^*_{s_1,t_1}$ and $P^*_{s_2,t_2}$ are empty, then $P^*_{s_1,t_1} \longleftarrow P'_{s_1,t_1}$, $P^*_{s_2,t_2} \longleftarrow P'_{s_2,t_2}$
   (d) If P'_{s_1,t_1} and P'_{s_2,t_2} are both non-empty and $len(P'_{s_1,t_1}) + len(P'_{s_2,t_2}) < len(P^*_{s_1,t_1}) + len(P^*_{s_2,t_2})$, then $P^*_{s_1,t_1} \longleftarrow P'_{s_1,t_1}$, $P^*_{s_2,t_2} \longleftarrow P'_{s_2,t_2}$

Output: $P^*_{s_1,t_1}, P^*_{s_2,t_2}$

Theorem 4. *Let $\gamma > 0$ and $d \in [n]$. Assume that Algorithm 2 is executed with parameters $(G, s_1, t_1, s_2, t_2), \gamma$ and d. If (G, s_1, t_1, s_2, t_2) is disjoint and $\gamma \cdot OPT \leq \gamma d \leq \Delta \cdot OPT$, then Algorithm 2 will return an optimal pair of paths. The running time of Algorithm 2 is $(n/(\gamma d))^{O(1/\gamma)} \cdot poly(n)$.*

Proof. The running time follows, since the iteration in Step 2 is executed $O(n/(\gamma d))$ choose $O(1/\gamma)$ times. The other steps in the algorithm only increase the running time by a multiplicative factor that is polynomial in n.

Let P_{s_1,t_1}, P_{s_2,t_2} be an optimal pair such that $dist(P_{s_1,t_1}, P_{s_2,t_2}) \geq \Delta \cdot OPT \geq \gamma d$. Since $|V(P_{s_1,t_1})| \leq OPT \leq d$, by Lemma 3, there exists $U = Rep_{P_{s_1,t_1}}(\gamma d/4)$, where $|U| \leq \lceil 8/\gamma \rceil$. Consider $Rep_G(\gamma d/4)$ as selected in the execution of Algorithm 2. For each $u \in U$, by Definition 9, we can choose $q_u \in Rep_G(\gamma d/4)$ such that $dist_G(q_u, u) < \gamma d/4$. Let $Q = \{q_u \mid u \in U\}$, then clearly $|Q| \leq |U|$. Step 2 of Algorithm 2 checks every subset of $Rep_G(\gamma d/4)$ of size at most $\lceil 8/\gamma \rceil$. Hence $S^* = Q$ for some iteration of Step 2. Immediately after executing the iteration of Step 2 where $S^* = Q$, let P'_{s_2,t_2} and P'_{s_1,t_1} be the paths found in Steps 2a and 2b, respectively.

We observe that, by the definition of U, for every vertex $v \in V(P_{s_1,t_1})$ there exists a vertex $u \in U$ such that $dist(v, u) \leq \gamma d/4$. By the choice of Q, for every $u \in U$ there exists $q \in Q$ such that $dist(q_u, u) \leq \gamma d/4$. Consequently, by the triangle inequality, for every vertex $v \in V(P_{s_1,t_1})$ there exists a vertex $q_u \in Q$ such that $dist(v, q_u) < \gamma d/2$. Hence, since $Q = S^*$, $E(P_{s_1,t_1}) \subseteq \bigcup_{v \in S^* \cup \{s_1,t_1\}} B_G(v, \gamma d/2)$ and therefore $len(P'_{s_1,t_1}) \leq len(P_{s_1,t_1})$.

We also observe that by triangle inequality, P_{s_2,t_2} and $\bigcup_{v \in S^* \cup \{s_1,t_1\}} B_G(v, \gamma d/2)$ are edge-disjoint since $\gamma d \leq \Delta \cdot OPT$ and $dist(q_u, P_{s_1,t_1}) \leq \gamma d/4$, for every $q_u \in Q$. Thus, P'_{s_1,t_1} and P_{s_2,t_2} are edge-disjoint. It follows as in the proof for the basic algorithm, in Section 3 following Definition 6, that P'_{s_1,t_1} and P'_{s_2,t_2} are an optimal pair of paths.

6 Main Results

We start this section by proving the reduction from the min-sum 2-paths orientation problem to the min-sum 2 edge-disjoint paths problem. Afterwards we prove the additive approximation result and we conclude the section by proving the multiplicative approximation result.

Theorem 5. *If there exists an approximation algorithm for the min-sum 2 edge-disjoint paths problem with time complexity $T(n)$, then there exists an algorithm for the min-sum 2-paths orientation problem with time complexity $T(n) + poly(n)$ and the same quality of approximation.*

Proof. Given an instance (G, s_1, t_1, s_2, t_2), we solve the min-sum 2-paths orientation problem as follows: (i) execute Algorithm 1 with input (G, s_1, t_1, s_2, t_2); (ii) execute the approximation algorithm for the min-sum 2 edge-disjoint paths problem with input (G, s_1, t_1, s_2, t_2); and then (iii) return an arbitrary best solution.

If the input instance is intersecting then, by Theorem 3, Algorithm 1 returns an optimal pair of paths. If the input instance is not intersecting then, by Lemma 1, it is disjoint.

So G has an optimal pair of edge-disjoint paths. Thus, the approximation algorithm for the min-sum 2 edge-disjoint paths returns the required pair of paths.

Theorem 6. *There exists an algorithm that given an instance* (G, s_1, t_1, s_2, t_2) *and* $\alpha > 0$*, returns non-conflicting paths* P_{s_1,t_1} *and* P_{s_2,t_2} *such that* $len(P_{s_1,t_1}) + len(P_{s_2,t_2}) \leq OPT + 2\alpha n$*, in time* $(1/\alpha)^{\tilde{O}(1/\alpha)} \cdot poly(n)$.

Proof. To obtain the required paths we perform the following steps: (i) execute Algorithm 1 with input (G, s_1, t_1, s_2, t_2); (ii) execute Algorithm 2 with input (G, s_1, t_1, s_2, t_2), α and n; and then (iii) return an arbitrary best solution.

The bound on the running time is immediate from Theorem 3 and Theorem 4. By Theorem 3, Algorithm 1 returns a pair of non-conflicting paths $P^*_{s_1,t_1}$ and $P^*_{s_2,t_2}$ whose sum of lengths does not exceed $(1 + 2\delta) \cdot OPT$. Note that if $\delta > 0$, then $OPT \leq n$. Thus, if $\delta \cdot OPT \leq \alpha n$, then $(1 + 2\delta) \cdot OPT \leq OPT + 2\alpha n$ and hence the theorem holds when $\delta \cdot OPT \leq \alpha n$. Suppose that $\delta \cdot OPT > \alpha n$ and therefore, $\alpha \cdot OPT \leq \alpha n \leq \delta \cdot OPT \leq \Delta \cdot OPT$ and hence by Theorem 4, Algorithm 2 will return an optimal pair of paths. Consequently, the theorem holds in general.

The proof of the following theorem is left for the full version.

Theorem 7. *There exists an algorithm that, given an instance* (G, s_1, t_1, s_2, t_2) *and* $\gamma > 0$*, returns non-conflicting paths* P_{s_1,t_1} *and* P_{s_2,t_2} *such that* $len(P_{s_1,t_1}) + len(P_{s_2,t_2}) \leq (1 + 2\gamma) \cdot OPT$*, in time* $(n/(\gamma \cdot OPT))^{O(1/\gamma)} \cdot poly(n)$.

References

1. Han, J., Jahanian, F.: Impact of path diversity on multi-homed and overlay networks. In: DSN 2004, p. 29. IEEE Computer Society (2004)
2. Hassin, R., Megiddo, N.: On orientations and shortest paths. Linear Algebra Appl. 114–115, 589–602 (1989)
3. Ito, T., Miyamoto, Y., Ono, H., Tamaki, H., Uehara, R.: Route-enabling graph orientation problems. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 403–412. Springer, Heidelberg (2009)
4. Kammer, F., Tholey, T.: The k-disjoint paths problem on chordal graphs. In: Paul, C., Habib, M. (eds.) WG 2009. LNCS, vol. 5911, pp. 190–201. Springer, Heidelberg (2010)
5. Kobayashi, Y., Sommer, C.: On shortest disjoint paths in planar graphs. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 293–302. Springer, Heidelberg (2009)
6. Li, C., McCormick, T.S., Simich-Levi, D.: The complexity of finding two disjoint paths with min-max objective function. Discrete Appl. Math. 26(1), 105–115 (1989)
7. Robertson, N., Seymour, P.D.: Graph minors. XIII. The disjoint paths problem. J. Comb. Theory Ser. B 63(1), 65–110 (1995)
8. Vasudevan, V., Andersen, D.G., Zhang, H.: Understanding the AS-level path disjointness provided by multi-homing. Technical Report CMU-CS-07-141. Carnegie Mellon University (2007)
9. Yang, B., Zheng, S.Q.: Finding min-sum disjoint shortest paths from a single source to all pairs of destinations. In: Cai, J.-Y., Cooper, S.B., Li, A. (eds.) TAMC 2006. LNCS, vol. 3959, pp. 206–216. Springer, Heidelberg (2006)
10. Zhang, P., Zhao, W.: On the complexity and approximation of the min-sum and min-max disjoint paths problems. In: Chen, B., Paterson, M., Zhang, G. (eds.) ESCAPE 2007. LNCS, vol. 4614, pp. 70–81. Springer, Heidelberg (2007)

Low Dimensional Embeddings of Doubling Metrics

Ofer Neiman

Department of Computer Science, Ben-Gurion University of the Negev, Israel
neimano@cs.bgu.ac.il

Abstract. We study several embeddings of doubling metrics into low dimensional normed spaces, in particular into ℓ_2 and ℓ_∞. Doubling metrics are a robust class of metric spaces that have low intrinsic dimension, and often occur in applications. Understanding the dimension required for a concise representation of such metrics is a fundamental open problem in the area of metric embedding. Here we show that the n-vertex Laakso graph can be embedded into constant dimensional ℓ_2 with the best possible distortion, which has implications for possible approaches to the above problem.

Since arbitrary doubling metrics require high distortion for embedding into ℓ_2 and even into ℓ_1, we turn to the ℓ_∞ space that enables us to obtain arbitrarily small distortion. We show embeddings of doubling metrics and their "snowflakes" into low dimensional ℓ_∞ space that simplify and extend previous results.

1 Introduction

In this paper we study embeddings of doubling metric spaces into low dimension normed spaces. A metric space (X, d) has doubling constant λ if any ball can be covered by λ balls of half its radius. A family of metrics is called *doubling* if the doubling constant of every member is bounded by some universal constant. The past decade has seen a surge of interest in doubling metrics, mainly because numerous algorithmic tasks are (approximately) tractable in such metrics, e.g. routing in networks [CGMZ05], low stretch spanners [GR08], nearest neighbor search and approximate distance oracles [HPM06,GK11], traveling salesperson problem [BGK12] and more.

Embedding into normed spaces is a very useful paradigm for representing and analyzing data. Since the cost of many data processing tasks depend exponentially on the dimension (the "curse of dimensionality"), it is often crucial to obtain a low dimension in the host space. The doubling constant of the metric captures in some sense the intrinsic dimension of the metric, and the logarithm of the doubling constant is known as the *doubling dimension* [GKL03]. Indeed, there are numerous results on low dimensional embedding of doubling metrics, and in what follows we review some of them. Recall that an embedding of a metric space (X, d) into ℓ_p^D is a map $f : X \to \mathbb{R}^D$, and the distortion of f is defined as

$$\max_{x \neq y \in X} \left\{ \frac{\|f(x) - f(y)\|_p}{d(x, y)} \right\} \cdot \max_{x \neq y \in X} \left\{ \frac{d(x, y)}{\|f(x) - f(y)\|_p} \right\} .$$

Several results only hold for a "snowflake" version of the metric: The $1 - \alpha$ snowflake of (X, d) is the metric $(X, d^{1-\alpha})$ with $0 < \alpha < 1$ (that is, every distance is raised to power $1 - \alpha$).

C. Kaklamanis and K. Pruhs (Eds.): WAOA 2013, LNCS 8447, pp. 12–23, 2014.

Euclidean Embeddings: Assouad [Ass83] showed that if (X, d) is λ-doubling then $(X, d^{1-\alpha})$ can be embedded into constant dimensional Euclidean space with constant distortion, where the constants depend only on λ and on α. He conjectured that such a result is possible also when $\alpha = 0$ (i.e. the original metric), but this was disproved by Semmes [Sem96]. In the computer science community, [GKL03] gave a comprehensive study on embedding doubling metrics. Among other results, they showed that n-point doubling metric spaces can be embedded with tight distortion $O(\sqrt{\log n})$ into Euclidean space (in contrast with arbitrary metrics that may require $\Omega(\log n)$ distortion [LLR95]). [KLMN05] showed an embedding with optimal dependence on the doubling constant (the lower bound was given by [JLM09]). The "price" paid for obtaining optimal distortion is that the dimension of all these embeddings is at least $\Omega(\log n)$. Following the intuition that the doubling dimension should be related to the dimension of the host space, [ABN08] showed that for any $\epsilon > 0$, λ-doubling metrics can be embedded into $O((\log \lambda)/\epsilon)$ dimensional Euclidean space with distortion $O(\log^{1+\epsilon} n)$. Both [ABN08,CGT10] exhibited a tradeoff between distortion and dimension: as the dimension ranges from $O(\log \log n)$ to $O(\log n)$, the distortion ranges from $O(\log n)$ to $O(\sqrt{\log n})$. However, the following is still open:

Question 1. Does every doubling metric on n points embeds into $O(1)$ dimensional ℓ_2 space with distortion $O(\sqrt{\log n})$?

Here we (arguably) show some evidence for a positive answer to this question, by providing an embedding of the metric induced by an n-vertex Laakso graph into constant dimensional Euclidean space with distortion $O(\sqrt{\log n})$. The Laakso graph G_k is a series-parallel graph with 6^k edges, $\Theta(6^k)$ vertices, and its doubling constant is at most 6 (see Section 3 for a definition of the Laakso graph), it was first introduced by [Laa02]. This graph seems difficult for ℓ_2 embedding and low dimensional embeddings. In particular, it is known that the metric induced by the n-vertex Laakso graph requires $n^{\Omega(1/\beta^2)}$ dimensions for a β distortion embedding into ℓ_1 [LMN05] (following the results of [BC05,LN04]). Also, this metric requires distortion at least $\Omega(\sqrt{\log n})$ for any embedding into ℓ_2 [GKL03]. So it seems surprising that allowing distortion $O(\sqrt{\log n})$ the embedding only requires 3 dimensions[1].

Theorem 1. *For any positive integer m, there exists an embedding of the metric induced by G_m into 3 dimensional ℓ_2 space with distortion $O(\sqrt{m})$.*

The proof of Theorem 1 appears in Section 3.

Embedding into ℓ_∞: The distortion of the above results is often undesirably high, in particular for application areas, where it is useful to have arbitrarily low distortion. Obtaining low distortion was shown to be impossible for ℓ_2 by [Sem96,Laa02,GKL03], and for ℓ_1 by [CK10,CKN09,LS11], where for the former the lower bound is a tight $\Omega(\sqrt{\log n})$ and for the latter $\Omega(\sqrt{\log n/\log\log n})$. Another natural candidate space is the ℓ_∞ space. In [GKL03] it was shown that for any $\epsilon > 0$, any doubling metric space (X, d) on n points embeds into $\ell_\infty^{O(\log n)}$ with distortion $1 + \epsilon$. While the explicit proof and the dependence on the parameters ϵ and λ was not specified there, the proof was

[1] It is quite conceivable that 2 dimensions suffice, we used 3 to simplify the analysis.

based on a variation of Bourgain's embedding and an application of the Lovász Local Lemma. In this paper we give a very simple proof of this result that does not require the Local Lemma, and has the best possible dependence on ϵ and the doubling constant λ, up to a constant in the exponent. Another advantage is that our construction only requires building nets, which can be implemented efficiently in near linear time [HPM06]. The result is in fact a simple adaptation of the methods introduced by [HPM06].

Theorem 2. *For any $0 < \epsilon \leq 1$, any finite metric space (X, d) on n points with doubling constant λ embeds into ℓ_∞^D with distortion $1 + \epsilon$ where $D = \lambda^{\log(1/\epsilon)+O(1)} \log n$.*

The proof of Theorem 2 appears in Section 4.

Snowflake embeddings: Following the result of Assouad, there were several extensions for the snowflakes of doubling metrics. [GKL03] provided an improved dependence of the distortion and the dimension on the doubling constant λ in Assouad's result. The dependence on α in the dimension was further improved in [ABN08], and finally was completely removed in [NN12] (in the range $0 < \alpha < 1/2$). [HPM06], among other algorithmic results on doubling metrics, showed an embedding of $(X, d^{1/2})$ into ℓ_∞ of dimension $\lambda^{O(\log(1/\epsilon))}$, which is then used for distance labeling. More recently, [GK11] showed a dimension reduction result for a snowflake of Euclidean subsets that are doubling, and [BRS11] obtained similar result. For the ℓ_∞ host, they showed a $1 + \epsilon$ distortion embedding for a $1 - \alpha$ snowflake with $\lambda^{O(\log(1/\epsilon)+\log\log\lambda)}/(\alpha(1 - \alpha))$ dimensions. The proof of [GK11] ingeniously combined many "hammers" such as the Johnson-Lindenstrauss dimension reduction, padded decompositions, a Gaussian transform and smoothing techniques. In this work we improve slightly the result of [GK11] for embedding doubling snowflakes into ℓ_∞, and generalize the embedding result of [HPM06] to arbitrary snowflaking parameter α. Perhaps more importantly, the construction and analysis given here are arguably simpler that those of [GK11], and admit an efficient implementation.

Theorem 3. *For any $0 < \epsilon \leq 1/20$, $0 < \alpha < 1$, and any finite metric space (X, d) on n points with doubling constant λ, there exists an embedding of the snowflake $(X, d^{1-\alpha})$ into ℓ_∞^D with distortion $1 + \epsilon$ where $D = \lambda^{\log(1/\epsilon)+O(1)}/(\alpha(1 - \alpha))$.*

The proof of Theorem 3 appears in the full version.

Dimension Reduction for Doubling Subsets: Assouad's result (embedding doubling snowflakes into constant dimensional Euclidean space with constant distortion) cannot be extended to arbitrary doubling metrics as mentioned above. One of the major open problems in the area of metric embedding is whether his result can be extended to doubling subsets of Euclidean space. That is,

Question 2. Does every doubling subset of ℓ_2 embeds into constant dimensional ℓ_2 space with constant distortion?

This question was raised by [LP01,GKL03], and also referred to in other works such as [ABN08,CGT10,GK11,NN12]. A possible approach for finding a counterexample, mentioned in [NN12], is to use the image under Euclidean embedding of a known "difficult" doubling metric. If it can be shown that a certain n-point doubling metric has

the following properties: 1) It has an ℓ_2 embedding with distortion $O(\sqrt{\log n})$ in which its image is doubling, and 2) Any embedding of this metric into constant dimensional ℓ_2 requires $\omega(\sqrt{\log n})$ distortion, then it would provide a negative answer to the above question.

A natural candidate for such a doubling metric, used in [CK10,CKN09] to prove non-embeddability in ℓ_1 of negative type metrics, is the Heisenberg group $\mathbb{H}$ equipped with the Carnot-Carathéodory metric. It was shown in [NN12] that it satisfies the first property. Another possible "difficult" metric is the Laakso graph, however the result stated in Theorem 1 rules out this example. In fact, a positive answer to Question 1 would rule out this approach entirely.

2 Preliminaries

Let (X, d) be a finite metric space, with $|X| = n$. We shall assume w.l.o.g that $d(x, y) \geq 1$ for all $x, y \in X$. The diameter of (X, d) is $\mathrm{diam}(X) = \max_{x,y \in X}\{d(x, y)\}$. A ball around $x \in X$ with radius $r \geq 0$ is defined as $B(x, r) = \{z \in X \mid d(x, z) \leq r\}$. The doubling constant of (X, d) is the minimal integer λ such that for all $x \in X$ and $r > 0$, the ball $B(x, 2r)$ can be covered by λ balls of radius r. The doubling dimension of (X, d) is defined as $\dim(X) = \log_2 \lambda$. A family of metric spaces is called doubling if there is a constant K such that every metric in the family has doubling constant at most K. An r-net of (X, d) is a set of points $N \subseteq X$ satisfying: 1) For all $u, v \in N$, $d(u, v) > r$, and 2) $\bigcup_{u \in N} B(u, r) = X$. It is well known that a simple greedy algorithm can provide an r-net.

3 Low Dimensional Embedding of the Laakso Graph

In this section we prove Theorem 1. For integer $k \geq 0$ let G_k be the k-th level Laakso graph, defined as follows: G_0 consist of a single edge, G_k is defined by replacing every edge of G_{k-1} with the graph on six edges and six vertices depicted in Figure 1, such that the vertices a, b correspond to the original endpoints of the edge. The edge lengths in G_k are 4^{-k} for all edges. For a pair of vertices that were edges in G_i, we abuse notation and call them *level i edges*. A level i edge e is a child of a level $i - 1$ edge e' if it is one of the six edges that replaced e'. This defines an (partial) inheritance relation on the edges of different levels. Note that any edge at level $k > i$ has a unique level i ancestor.

We label the edges of G_k by a sequence $l \in L^k$, where $L = \{0, 1, -1, 2, -2, 3\}$, such that for $1 \leq i \leq k$, l_i is the position of the level i ancestor of the edge, depicted in Figure 1. The vertices created in level k are labeled by a string in $L^{k-1} \times \{s, t, u, v\}$. In particular, each edge of level $k - 1$ creates 4 new vertices, if the label of the edge was $l \in L^{k-1}$, then the new vertices will be labeled by $l \circ s$, $l \circ t$, $l \circ u$, and $l \circ v$ (where for strings w, w', $w \circ w'$ denotes their concatenation).

We write $\| \cdot \|$ for the standard Euclidean norm.

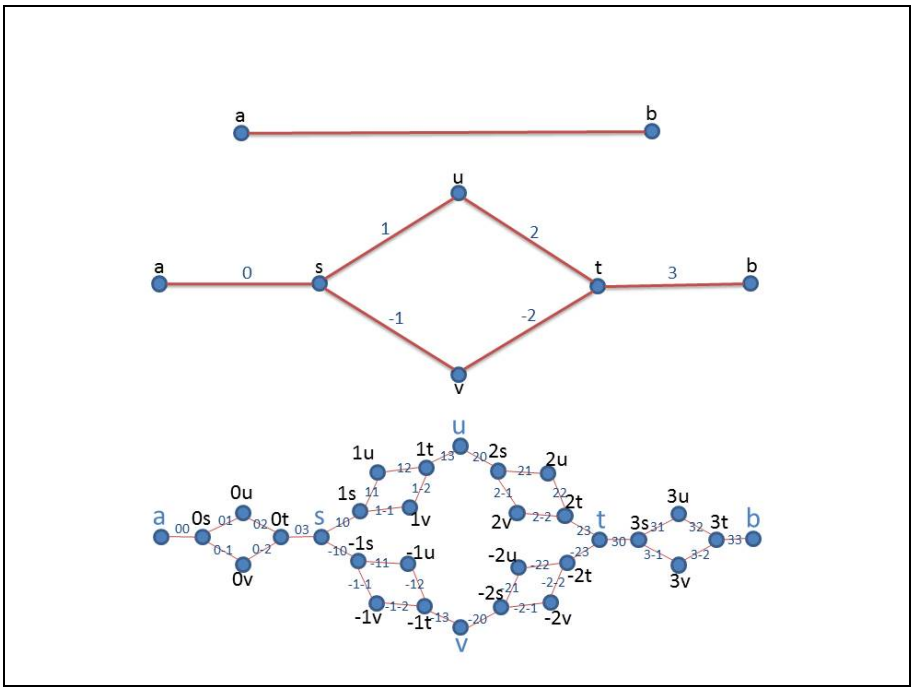

Fig. 1. Naming vertices and edges of Laakso graph

3.1 Construction of the Embedding

Consider the graph G_n, with shortest path metric d, and fix $D = 1/\sqrt{n}$. First define an embedding $g : V(G_n) \to \mathbb{R}$ by $g(x) = d(x, a)$, where a is the left vertex of G_0. We define the embedding $f : V(G_n) \to \mathbb{R}^2$ recursively as follows. In the case $k = 0$ where a, b are the two endpoints of the single edge of level 0, define $f(a) = (0, 0)$, $f(b) = (1, 0)$. Fix some integer $1 \leq k \leq n$. Now, let $\{a, b\}$ be any level $k - 1$ edge, let s, t, u, v be the new four vertices created from it in level k. Inductively, f is already defined on both a and b, so let $z = f(b) - f(a)$. Finally let $\bar{z}$ be one of the two unit vectors orthogonal to z in $\mathbb{R}^2$ (chosen arbitrarily). Define

$$
\begin{aligned}
f(s) &= f(a) + z/4 \\
f(t) &= f(a) + 3z/4 \\
f(u) &= f(a) + z/2 + D4^{-k} \cdot \bar{z} \\
f(v) &= f(a) + z/2 - D4^{-k} \cdot \bar{z}\,.
\end{aligned}
$$

In some sense the embedding g is just a projection of the graph into the line, and its sole purpose is to provide contribution for the edges. The difficulty in embedding

the Laakso graph comes from handling the diagonals (each diagonal is composed of the two vertices whose labels are $p \circ u$ and $p \circ v$ for some $k \geq 0$ and $p \in L^k$). The map f provides sufficient contribution for the diagonals, the price is that we expand slightly the four inner edges (e.g. $\{p \circ s, p \circ u\}$). Intuitively, since f provides only $1/\sqrt{n}$ fraction of the distance between the diagonals, and uses an orthogonal vector to the parent edge's vector, we get that the distance between the images of the edge's endpoints is increased only by a factor of $1/n$. Thus even n levels of recursion will not generate a large expansion.

3.2 Analysis of the Embedding

The first step is to bound the distortion of the edges (of all levels), which yields an upper bound on the expansion of the embedding.

Claim. For any integer $0 \leq k \leq n$ and any level k edge $\{x, y\} \in E(G_k)$,

$$d(x, y) \leq \|f(x) - f(y)\| \leq 2d(x, y) \ .$$

Proof. We prove by induction on k that if $\{x, y\}$ is level k edge, then

$$4^{-k} \leq \|f(x) - f(y)\| \leq \sqrt{1 + kD^2} \cdot 4^{-k} \ . \tag{1}$$

The base case $k = 0$ is true by definition. For the inductive step, let $\{a, b\}$ be a level $k-1$ edge with $d(a, b) = 4^{-(k-1)}$, and let $z = f(b) - f(a)$. By the induction hypothesis

$$4^{-(k-1)} \leq \|z\| \leq \sqrt{1 + (k-1)D^2} \cdot 4^{-(k-1)} \ . \tag{2}$$

Consider the six edges created from $\{a, b\}$ in level k that are depicted in Figure 1. First observe that for the edge $\{a, s\}$, by definition $\|f(s) - f(a)\| = \|z\|/4$ so it satisfies (1). The same holds for the edge $\{b, t\}$. Consider now the edge $\{s, u\}$, using that $z, \bar{z}$ are orthogonal suggests the following bound,

$$\begin{aligned} \|f(u) - f(s)\|^2 &= \|z/4 + D4^{-k} \cdot \bar{z}\|^2 \\ &= \|z/4\|^2 + \|D4^{-k} \cdot \bar{z}\|^2 \\ &= \|z\|^2/16 + (D4^{-k})^2 \end{aligned}$$

Using (2) it holds that $\|z\| \geq 4^{-(k-1)}$ and thus $\|f(u) - f(s)\| \geq \|z\|/4 \geq 4^{-k}$. For the upper bound, note that by (2)

$$\|f(u) - f(s)\|^2 \leq (1 + (k-1)D^2) \cdot 4^{-2(k-1)}/16 + D^2 4^{-2k} = (1 + kD^2) \cdot 4^{-2k} \ .$$

The same calculation holds for the edges $\{s, v\}$, $\{u, t\}$ and $\{v, t\}$. This concludes the proof of (1). Using that $k \leq n$, we see that $\sqrt{1 + kD^2} \leq \sqrt{1 + 1} < 2$, proving the claim.

Lemma 1. *For any $x, y \in V(G_n)$,*

$$\|(f \oplus g)(x) - (f \oplus g)(y)\| \leq 3d(x, y) \ .$$

Proof. Let $x = u_0, u_1, \ldots, u_t = y$ be a shortest path in G_n connecting x to y. By the triangle inequality and Claim 3.2,

$$\|f(x) - f(y)\| \leq \sum_{i=1}^{t} \|f(u_i) - f(u_{i-1})\| \leq 2 \sum_{i=1}^{t} d(u_i, u_{i-1}) = 2d(x, y) \ . \qquad (3)$$

Using the triangle inequality it follows that

$$|g(x) - g(y)| = |d(x, a) - d(y, a)| \leq d(x, y) \ .$$

The main effort will be showing that the contraction of $f \oplus g$ is bounded by $O(1/D)$. Observe that the vertices u, v of any basic structures in any level already suffer contraction of $\Theta(1/D)$. If we consider two vertices x, y of distance $d(x, y) \approx 4^{-j}$, then in level j they have different ancestor edges, and at least intuitively they should get a contribution of $D \cdot 4^{-j}$ from the embedding of level j. However, in their final embedding, x, y may "get closer" to each other because we use few dimensions. We first focus on the case where $g(x) = g(y)$, and so the contribution must entirely come from the f embedding. The following lemma shows there is indeed sufficient contribution from the critical level, and the main issue is showing that this contribution does not completely cancel out.

Lemma 2. *Let $x, y \in V(G_n)$ be such that $g(x) = g(y)$, then*

$$\|f(x) - f(y)\| \geq D \cdot d(x, y)/32 \ .$$

Proof. First observe that since x, y have the same g value they must have been created in the same level, and denote this level by m. Abusing notation, denote by $x, y \in L^{m-1} \times \{s, t, u, v\}$ the labels of the vertices. For any $1 \leq i \leq m$ let $p_i = x_1 \circ \cdots \circ x_{i-1}$ be the label of the level $i - 1$ ancestor edge of x. Observe that since $g(x) = g(y)$ it must be for every $i \in [m - 1]$ that $|x_i| = |y_i|$. First consider the case that $x_i = y_i$ for all $i \in [m]$, then in fact x, y are the u, v vertices created from the edge labeled p_m, and by definition of f, $\|f(x) - f(y)\| = 2D \cdot 4^{-m} = D \cdot d(x, y)$.

Now assume that there is an index $i \in [m - 1]$ such that $x_i \neq y_i$, and let j be the minimal such index. We shall assume w.l.o.g that $x_j = 1$, $y_j = -1$ (the case $x_j = 2$ and $y_j = -2$ is symmetric). Let $s' = p_j \circ s$ and $t' = p_j \circ t$. Let k be the smallest integer satisfying $j < k < m$ and such that at least one of x_k, y_k is different from 0 (or different from 3 if it was the case that $x_j = 2$, $y_j = -2$). If no such k exists put $k = m$. Assume w.l.o.g that $x_k \neq 0$. Roughly speaking, j is the index of the scale in which x, y are separated into different recursive components, however since in the scales i from $j + 1$ to $k - 1$, $x_i = y_i = 0$, both are still close to s' and thus to each other. The final distance between x, y is about 4^{-k}. To see this, note that $d(x, y) = d(x, s') + d(y, s')$, and thus $d(x, y) \geq d(x, s') \geq 4^{-k}$. On the other hand, since for $j < i < k$, $x_i = y_i = 0$, then $d(x, s') \leq d(p_{k-1} \circ s, s') = 4^{-(k-1)}$ and similarly for $d(y, s')$, so $d(x, y) < 4^{-(k-2)}$. It remains to show that $\|f(x) - f(y)\| \geq D \cdot 4^{-k}/2$.

Let ℓ be the line passing through the endpoints of p_j. We will prove that $f(x)$ is at least $D \cdot 4^{-k}/2$ away from any point on the line ℓ. To this end, we prove by induction on m that the Euclidean distance of $f(x)$ from the line ℓ is at least

$$D \cdot (4^{-k} - \sum_{i=k+1}^{m} 4^{-i}) \,, \tag{4}$$

furthermore, $f(x)$ is on the same side of ℓ as $f(u')$ where $u' = p_j \circ u$.

The base case is when $m = k$. As $x_j = 1$, the end points of the level j ancestor of x are $s' = p_j \circ s$ and $u' = p_j \circ u$. By definition $f(s')$ lies right on the line ℓ connecting the images of the endpoints of p_j, and u' is at distance $D \cdot 4^{-j}$ from ℓ (since we use an orthogonal vector to ℓ). Let $z' = f(u') - f(s')$. By definition of f, for all $i > j$ we have that $f(p_i \circ s) = f(s') + z'/4^{i-j}$, in particular $f(p_{k-1} \circ s) = f(s') + z'/4^{k-1-j}$, which suggests that the distance of $f(p_{k-1} \circ s)$ from the line ℓ is equal to $D \cdot 4^{-j}/4^{k-1-j} = D/4^{k-1}$. As x is one of the four vertices created from the edge p_k whose end points are s' and $p_{k-1} \circ s$, it can be verified by the definition of the embedding f that its distance to ℓ is at least $1/4$ of the distance of $p_{k-1} \circ s$ to ℓ, that is at least $D/4^k$, as required.

Next we prove the inductive step. Let q, r be the level $m-1$ vertices which are the end points of the edge labeled p_m, and let ℓ' denote the line passing through $f(q)$ and $f(r)$. By the induction hypothesis the distance of both $f(q)$ and $f(r)$ from ℓ is at least $D \cdot (4^{-k} - \sum_{i=k+1}^{m-1} 4^{-i})$ and both are on the same side of ℓ. This suggests that every point on ℓ', in particular $p_m \circ s$ and $p_m \circ t$, is at least $D \cdot (4^{-k} - \sum_{i=k+1}^{m-1} 4^{-i})$ away from ℓ. It remains to argue about $p_m \circ u$ and $p_m \circ v$, which by definition are embedded by f at distance $D/4^m$ from ℓ', which means their distance to ℓ can be closer than that of $f(q), f(r)$ by at most $D/4^m$, which concludes the proof of (4).

Since $\sum_{i=k+1}^{m} 4^{-i} \leq 4^k/2$ we have that $f(x)$ is at least $D \cdot 4^{-k}/2$ away from ℓ. If $j < k(y) < m$ is the minimal such that $y_{k(y)} \neq 0$ (and $k(y) = m$ if there is no such value), then a analogous argument will show that $f(y)$ is at least $D \cdot 4^{-k(y)}/2$ away from ℓ on the side of $f(v')$ where $v' = p_j \circ v$. In particular, this suggests that $\|f(x) - f(y)\| \geq D \cdot 4^{-k}/2$, as required.

It remains to bound the contraction for an arbitrary pair x, y.

Lemma 3. *For any $x, y \in V(G_n)$,*

$$\|(f \oplus g)(x) - (f \oplus g)(y)\| \geq D \cdot d(x,y)/128 \,. \tag{5}$$

Proof. First consider the case that $|g(x) - g(y)| \geq D \cdot d(x,y)/128$, then clearly (5) holds. Otherwise, $|g(x) - g(y)| < D \cdot d(x,y)/128$, and w.l.o.g assume that $g(x) < g(y)$. In this case, let $y' \in G_n$ be any point on a shortest path connecting y to a such that $g(x) = g(y')$. Then

$$d(y,y') = g(y) - g(y') = g(y) - g(x) \leq D \cdot d(x,y)/128, \tag{6}$$

thus also

$$d(x,y') \geq d(x,y) - d(y,y') \geq 3d(x,y)/4 \,. \tag{7}$$

Using Lemma 2 on x, y' it follows that,

$$\begin{aligned}\|f(x)-f(y)\| &\geq \|f(x)-f(y')\|-\|f(y')-f(y)\| \\ &\overset{(3)}{\geq} D\cdot d(x,y')/32-2d(y',y) \\ &\overset{(6)\wedge(7)}{\geq} 3D\cdot d(x,y)/128-D\cdot d(x,y)/64 \\ &= D\cdot d(x,y)/128\,.\end{aligned}$$

The proof of Theorem 1 follows from Lemma 1 and Lemma 3

4 Embedding Doubling Metrics to Low Dimensional ℓ_∞

In this section we prove Theorem 2. Let us first remark that the dependence of the dimension D on the parameters is essentially tight (up to a constant in the exponent), that is $D \geq \lambda + (1/\epsilon)^{\Omega(1)} + \Omega(\log n)$: First, the $\log n$ term cannot be improved, because [GKL03] showed an $\Omega(\sqrt{\log n})$ lower bound on the distortion when embedding doubling metrics into ℓ_2. Under the ℓ_2 norm our embedding has distortion at most $(1+\epsilon)\sqrt{D}$, so when ϵ and λ are constants it must be that $D = \Omega(\log n)$. Second, a linear dependence on λ in the dimension is necessary, because for $\epsilon = 1$, say, the dimension of a normed space in which any n-point metric embeds with distortion 2 must be $\Omega(n) = \Omega(\lambda)$ [Mat02]. In the full version we show that there must be a polynomial dependence on $1/\epsilon$ as well.

4.1 Construction

For simplicity of presentation we first handle the case in which the aspect ratio of the metric is at most n, that is, $\mathrm{diam}(X) < n$, the general case is deferred to the full version. For each $0 \leq i < \log n$ take a r_i-net N_i, where $r_i = \epsilon \cdot 2^{i-2}$. Fix some net N_i, and for an integer $k > 0$ define a *spread-partition* $P_i(k)$ as a partition of N_i into k clusters $N_{i0}, N_{i1}, \ldots, N_{i(k-1)}$, such that each cluster is well spread. Formally, for all $0 \leq j \leq k-1$, if $u, v \in N_{ij}$ then

$$d(u,v) \geq 5 \cdot 2^i\,. \tag{8}$$

Note that N_{ij} is not necessarily a net of N_i, as it may not satisfy the covering property of nets.

Claim. Fix $k = \lambda^{6+\log(1/\epsilon)}$. For all $0 \leq i < \log n$ there exists a spread-partition $P_i(k)$.

Proof. To construct $P_i(k)$, first greedily choose a maximal $N_{i0} \subseteq N_i$ that satisfy (8). For any $0 < j \leq k-1$, after choosing $N_{i0}, ..., N_{i(j-1)}$, greedily choose a maximal $N_{ij} \subseteq N_i \setminus (N_{i0} \cup ... \cup N_{i(j-1)})$ that satisfy (8). We claim that after k iterations N_i must be exhausted. Seeking contradiction, assume that $u \in N_i$ was not covered by any N_{ij}, and consider $B = B(u, 5 \cdot 2^i)$. By using the doubling property iteratively, the ball B can be covered by $\lambda^{\log(5\cdot 2^i/(r_i/2))}$ balls of radius $r_i/2$, each of these small balls can contain at most one point from N_i. As $\lambda^{\log(5\cdot 2^i/(r_i/2))} < k$, we conclude that for some $0 \leq j \leq k-1$, N_{ij} does not contain any point from B, but then by maximality it should have contained u, a contradiction.

Next we define the embedding $f : X \to \mathbb{R}^D$ with $D = k \log n$, where k is defined as in Claim 4.1. Let $f_{ij}(x) = d(x, N_{ij})$, and

$$f(x) = \bigoplus_{i=1}^{\log n} \bigoplus_{j=0}^{k-1} f_{ij}(x) \ .$$

(We use the convention that if $N_{ij} = \emptyset$ then $d(x, N_{ij}) = 0$).

4.2 Proof

Fix some $x, y \in X$. By the triangle inequality we have that for any N_{ij}, $d(x, N_{ij}) - d(y, N_{ij}) \leq d(x, y)$, so that for any $1 \leq i \leq \log n$ and $0 \leq j \leq k-1$, $f_{ij}(x) - f_{ij}(y) \leq d(x, y)$. By symmetry of x, y this suggests that

$$|f_{ij}(x) - f_{ij}(y)| \leq d(x, y) \ . \tag{9}$$

Next we show that there are i, j such that $f_{ij}(x) - f_{ij}(y) \geq d(x, y)(1 - \epsilon)$. Let $1 \leq i \leq \log n$ be such that $2^{i-1} \leq d(x, y) < 2^i$, and let $0 \leq j \leq k - 1$ be such that $d(x, N_{ij}) \leq r_i$ (such a j must exist because N_i is an r_i-net). Denote by $u \in N_{ij}$ the point satisfying $d(x, N_{ij}) = d(x, u)$.

We claim that $d(y, N_{ij}) = d(y, u)$. To see this, first observe that $d(y, u) \leq d(y, x) + d(x, u) \leq 2^i + r_i < (5/4) \cdot 2^i$. Consider any other $v \in N_{ij}$, by the construction of N_{ij}, $d(v, u) \geq 5 \cdot 2^i$, so $d(y, v) \geq d(u, v) - d(y, u) > 5 \cdot 2^i - (5/4) \cdot 2^i > (5/4) \cdot 2^i > d(y, u)$. Thus it follows that $d(y, N_{ij}) = d(y, u) \geq d(y, x) - d(x, u) \geq d(x, y) - r_i$. We conclude that

$$f_{ij}(y) - f_{ij}(x) \geq (d(x, y) - r_i) - r_i = d(x, y) - \epsilon \cdot 2^{i-1} \geq d(x, y)(1 - \epsilon) \ . \tag{10}$$

The proof of Theorem 2 follows directly from (9) and (10).

Acknowledgements. The author is grateful to Moses Charikar, Michael Elkin and Lee-Ad Gottlieb for fruitful discussions. This work is supported by ISF grant No. (523/12) and by the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°303809.

References

ABN08. Abraham, I., Bartal, Y., Neiman, O.: Embedding metric spaces in their intrinsic dimension. In: Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, Philadelphia, PA, USA, pp. 363–372. Society for Industrial and Applied Mathematics (2008)

Ass83. Assouad, P.: Plongements lipschitziens dans $\mathbb{R}^n$. Bull. Soc. Math. France 111(4), 429–448 (1983)

BC05. Brinkman, B., Charikar, M.: On the impossibility of dimension reduction in l_1. J. ACM 52(5), 766–788 (2005)

BGK12. Bartal, Y., Gottlieb, L.-A., Krauthgamer, R.: The traveling salesman problem: low-dimensionality implies a polynomial time approximation scheme. In: Proceedings of the 44th Symposium on Theory of Computing, STOC 2012, New York, NY, USA, pp. 663–672. ACM (2012)

BRS11. Bartal, Y., Recht, B., Schulman, L.J.: Dimensionality reduction: beyond the johnson-lindenstrauss bound. In: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, pp. 868–887. SIAM (2011)

CGMZ05. Chan, T.-H.H., Gupta, A., Maggs, B.M., Zhou, S.: On hierarchical routing in doubling metrics. In: Proc. 16th Ann. ACM-SIAM Symposium on Discrete Algorithms, SODA (2005)

CGT10. Chan, T.-H.H., Gupta, A., Talwar, K.: Ultra-low-dimensional embeddings for doubling metrics. J. ACM 57(4) (2010)

CK10. Cheeger, J., Kleiner, B.: Differentiating maps into L^1, and the geometry of BV functions. Annals of Math 171(2), 1347–1385 (2010)

CKN09. Cheeger, J., Kleiner, B., Naor, A.: A $(\log n)^{\Omega(1)}$ integrality gap for the sparsest cut sdp. In: Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, Washington, DC, USA, pp. 555–564. IEEE Computer Society (2009)

GK11. Gottlieb, L.-A., Krauthgamer, R.: A nonlinear approach to dimension reduction. In: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, pp. 888–899. SIAM (2011)

GKL03. Gupta, A., Krauthgamer, R., Lee, J.R.: Bounded geometries, fractals, and low-distortion embeddings. In: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2003, Washington, DC, USA, pp. 534–543. IEEE Computer Society (2003)

GR08. Gottlieb, L.-A., Roditty, L.: Improved algorithms for fully dynamic geometric spanners and geometric routing. In: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, Philadelphia, PA, USA, pp. 591–600. Society for Industrial and Applied Mathematics (2008)

HPM06. Har-Peled, S., Mendel, M.: Fast construction of nets in low-dimensional metrics and their applications. SIAM Journal on Computing 35(5), 1148–1184 (2006)

JLM09. Jaffe, A., Lee, J.R., Moharrami, M.: On the optimality of gluing over scales. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX and RANDOM 2009. LNCS, vol. 5687, pp. 190–201. Springer, Heidelberg (2009)

KLMN05. Krauthgamer, R., Lee, J.R., Mendel, M., Naor, A.: Measured descent: A new embedding method for finite metrics. Geometric and Functional Analysis 15(4), 839–858 (2005)

Laa02. Laakso, T.J.: Plane with A_∞-weighted metric not bi-Lipschitz embeddable to $\mathbb{R}^N$. Bull. London Math. Soc. 34(6), 667–676 (2002)

LLR95. Linial, N., London, E., Rabinovich, Y.: The geometry of graphs and some of its algorithmic applications. Combinatorica 15(2), 215–245 (1995)

LMN05. Lee, J.R., Mendel, M., Naor, A.: Metric structures in l1: dimension, snowflakes, and average distortion. Eur. J. Comb. 26(8), 1180–1190 (2005)

LN04. Lee, J.R., Naor, A.: Embedding the diamond graph in l_p and dimension reduction in l_1, pp. 745–747 (2004)

LP01. Lang, U., Plaut, C.: Bilipschitz embeddings of metric spaces into space form. Geometriae Dedicata 87(1-3), 285–307 (2001)

LS11. Lee, J.R., Sidiropoulos, A.: Near-optimal distortion bounds for embedding doubling spaces into l1. In: Proceedings of the 43rd annual ACM symposium on Theory of computing, STOC 2011, New York, NY, USA, pp. 765–772. ACM (2011)

Mat02. Matoušek, J.: Lectures on discrete geometry. Springer, New York (2002)
NN12. Naor, A., Neiman, O.: Assouad's theorem with dimension independent of the snowflaking. Revista Matematica Iberoamericana 28(4), 1–21 (2012)
Sem96. Semmes, S.: On the nonexistence of bilipschitz parameterizations and geometric problems about a_∞ weights. Revista Matemática Iberoamericana 12, 337–410 (1996)

Degree-Constrained Graph Orientation: Maximum Satisfaction and Minimum Violation*

Yuichi Asahiro[1], Jesper Jansson[2], Eiji Miyano[3], and Hirotaka Ono[4]

[1] Department of Information Science, Kyushu Sangyo University, Higashi-ku, Fukuoka 813-8503, Japan
asahiro@is.kyusan-u.ac.jp
[2] Laboratory of Mathematical Bioinformatics, Institute for Chemical Research, Kyoto University, Gokasho, Uji, Kyoto 611-0011, Japan
jj@kuicr.kyoto-u.ac.jp
[3] Department of Systems Design and Informatics, Kyushu Institute of Technology, Iizuka, Fukuoka 820-8502, Japan
miyano@ces.kyutech.ac.jp
[4] Department of Economic Engineering, Kyushu University, Higashi-ku, Fukuoka 812-8581, Japan
hirotaka@en.kyushu-u.ac.jp

Abstract. A *degree-constrained graph orientation* of an undirected graph G is an assignment of a direction to each edge in G such that the outdegree of every vertex in the resulting directed graph satisfies a specified lower and/or upper bound. Such graph orientations have been studied for a long time and various characterizations of their existence are known. In this paper, we consider four related optimization problems introduced in [4]: For any fixed non-negative integer W, the problems Max W-Light, Min W-Light, Max W-Heavy, and Min W-Heavy take as input an undirected graph G and ask for an orientation of G that maximizes or minimizes the number of vertices with outdegree at most W or at least W. The problems' computational complexities vary with W. Here, we resolve several open questions related to their polynomial-time approximability and present a number of positive and negative results.

1 Introduction

Let $G = (V, E)$ be an undirected (multi-)graph. An *orientation of* G is a function that maps each undirected edge $\{u, v\}$ in E to one of the two possible directed edges (u, v) and (v, u). For any orientation Λ of G, define $\Lambda(E) = \bigcup_{e \in E}\{\Lambda(e)\}$ and let $\Lambda(G)$ denote the directed graph $(V, \Lambda(E))$. For any vertex $u \in V$, the *outdegree of* u *under* Λ is defined as $d^+_\Lambda(u) = |\{(u, v) : (u, v) \in \Lambda(E)\}|$, i.e., the number of outgoing edges from u in $\Lambda(G)$. For any non-negative integer W, a vertex $u \in V$ is called W*-light in* $\Lambda(G)$ if $d^+_\Lambda(u) \leq W$, and W*-heavy in* $\Lambda(G)$ if

* Supported by KAKENHI Grant Numbers 21680001, 23500020, 25104521, and 25330018 and The Hakubi Project at Kyoto University.

C. Kaklamanis and K. Pruhs (Eds.): WAOA 2013, LNCS 8447, pp. 24–36, 2014.

$d^+_\Lambda(u) \geq W$. For any $U \subseteq V$, if all the vertices in U are W-light (resp., W-heavy), we say that U is W-light (resp., W-heavy).

The optimization problems Max W-Light, Min W-Light, Max W-Heavy, and Min W-Heavy, where W is any fixed non-negative integer, were introduced in [4]. In each problem, the input is an undirected (multi-)graph $G = (V, E)$ and the objective is to output an orientation Λ of G such that:

- Max W-Light: $\left|\{u \in V : d^+_\Lambda(u) \leq W\}\right|$ is maximized
- Min W-Light: $\left|\{u \in V : d^+_\Lambda(u) \leq W\}\right|$ is minimized
- Max W-Heavy: $\left|\{u \in V : d^+_\Lambda(u) \geq W\}\right|$ is maximized
- Min W-Heavy: $\left|\{u \in V : d^+_\Lambda(u) \geq W\}\right|$ is minimized

We write $n = |V|$ and $m = |E|$ for the input graph G.

The *degree of u in G* is denoted by $d(u)$. We define $\delta = \min\{d(u) \mid u \in V\}$ and $\Delta = \max\{d(u) \mid u \in V\}$. For any $U \subseteq V$, the subgraph induced by U is denoted by $G[U]$.

Observe that Max W-Light and Min $(W+1)$-Heavy are *supplementary problems* in the sense that an exact algorithm for one gives an exact algorithm for the other but their polynomial-time approximability properties may differ. The same observation holds for the pair Min W-Light and Max $(W+1)$-Heavy.

The computational complexities of Max W-Light, Min W-Light, Max W-Heavy, and Min W-Heavy were studied for different values of W in [4]. As observed in [4], the special case of Max 0-Light is equivalent to the well-known Maximum Independent Set problem (and the supplementary problem Min 1-Heavy is equivalent to Minimum Vertex Cover). Thus, allowing the value of W to vary yields a natural generalization of Maximum Independent Set and Minimum Vertex Cover. In many cases, however, the (in)tractability and the (in)approximability remained unknown. In this paper, we establish several new results on the polynomial-time approximability of these problems.

New Results: Below is a summary of previous results from [4] and the new results presented in this paper (see Table 1 for a summary). Due to space limitations, many technical details will be deferred to the full version of the paper.

- Max W-Light: It is known that Max 0-Light cannot be approximated within a ratio of $n^{1-\varepsilon}$ for any positive constant ε in polynomial time unless $\mathcal{P} = \mathcal{NP}$ [4,31]. Theorem 6 of Sect. 4 proves that for every fixed $W \geq 1$, Max W-Light cannot be approximated within $(n/W)^{1-\varepsilon}$ in polynomial time unless $\mathcal{P} = \mathcal{NP}$. On the positive side, Theorem 7 of Sect. 4 provides a polynomial-time $n/(2W+1)$-approximation algorithm for Max W-Light.
- Min W-Heavy: Min 1-Heavy cannot be approximated within 1.3606 in polynomial time unless $\mathcal{P} = \mathcal{NP}$ [4,9]. Theorem 5 of Sect. 4 extends this inapproximability result to hold for Min W-Heavy for every fixed $W \geq 2$. We also show how to approximate Min W-Heavy within a ratio of $\log(\Delta - W + 1)$ in polynomial time for every fixed $W \geq 2$ in Theorem 2 of Sect. 3.

Table 1. Summary of the results from [4] and the new results in this paper

W	Max W-Light	Min $(W+1)$-Heavy
$=0$	Identical to Maximum Independent Set [4]	Identical to Minimum Vertex Cover [4]
≥ 1	Solvable in $O(n)$ time for trees [4] $(n/(2W+1))$-approx. (Theorem 7) $(n/W)^{1-\varepsilon}$-inapprox. (Theorem 6)	Solvable in $O(n)$ time for trees [4] $\log(\Delta - W)$-approx. (Theorem 2) 1.3606-inapprox. (Theorem 5)

W	Min W-Light	Max $(W+1)$-Heavy
$=0$	Solvable in $O(m^{3/2})$ time [4]	Solvable in $O(m^{3/2})$ time [4]
≥ 0	Solvable in $O(n)$ time for trees [4] Solvable in $O(n^2)$ time for outerplanar graphs [4]	Solvable in $O(n)$ time for trees [4] Solvable in $O(n^2)$ time for outerplanar graphs [4]
≥ 1	$(W+1)$-approx. [4] $\log(W+1)$-approx. (Theorem 1)	$O(n^2)$-time 2-approx. for planar graphs [4] $O(m)$-time $(W+2)$-approx. [4]
≥ 2	NP-hard for planar graphs [4]	NP-hard for planar graphs [4]
large	$(\log(W+1) - O(\log\log(W+1)))$ -inapprox. (Theorem 4)	$W^{1-\varepsilon}$-inapprox. (Theorem 3)

- Min W-Light: A polynomial-time $(W+1)$-approximation algorithm was given in [4]. Theorem 1 of Sect. 3 improves the approximation ratio to $\log(W+1)$ for any $W \geq 1$. Moreover, Theorem 4 in Sect. 4 shows that for sufficiently large W, Min W-Light is $\mathcal{NP}$-hard to approximate within $\log(W+1) - O(\log\log(W+1))$, implying that our $\log(W+1)$-approximation is almost tight.
- Max W-Heavy: It was shown in [4] that Max 1-Heavy and Min 0-Light are in $\mathcal{P}$, but Max W-Heavy and Min $(W-1)$-Light are $\mathcal{NP}$-hard for every fixed $W \geq 3$. An open problem from [4] was to determine the computational complexity of Max 2-Heavy and Min 1-Light. Now, consider two special cases: (i) $\Delta \leq 3$ and (ii) $\delta \geq 4$. Corollary 3 of Sect. 5 and Proposition 5 of Sect. 2 demonstrate that Max 2-Heavy and Min 1-Light can be solved in polynomial time for (i) and (ii), respectively. Also, Theorem 3 in Sect. 4 proves that for sufficiently large W, Max W-Heavy is $\mathcal{NP}$-hard to approximate within $W^{1-\varepsilon}$ for any $\varepsilon > 0$. The best previously known polynomial-time approximation ratio was $W+1$ [4].

Motivation: Graph orientations that optimize certain objective functions involving the resulting directed graph or that satisfy some special property such as acyclicity [26] or k-edge connectivity [8,21,24] have many applications to graph theory, combinatorial optimization, scheduling (load balancing), resource allocation, and efficient data structures. For example, an orientation that minimizes the maximum outdegree [2,7,10,29] can be used to support fast vertex adjacency

queries in a sparse graph by storing each edge in exactly one of its two incident vertices' adjacency lists while ensuring that all adjacency lists are short [7]. There are many optimization criteria for graph orientation other than these. See [3] or chapter 61 in [27] for more details and additional references.

On the other hand, *degree-constrained* graph orientations [12,13,15,19] arise when a lower degree bound $W^l(v)$ and an upper degree bound $W^u(v)$ for each vertex v in the graph are specified, and the outdegree of v in any valid graph orientation is required to lie in the interval $W^l(v)..W^u(v)$. Obviously, a graph does not always have such an orientation, and in this case, one might want to compute an orientation that best fits the outdegree constraints according to some well-defined criteria [3,4]. In case $W^l(v) = 0$ and $W^u(v) = W$ for every vertex v in the input graph, where W is a non-negative integer, and the objective is to maximize (resp., minimize) the number of vertices that satisfy (resp., violate) the outdegree constraints, then we obtain MAX W-LIGHT (resp., MIN $(W+1)$-HEAVY). Similarly, if $W^l(v) = W$ and $W^u(v) = \infty$ for every vertex v in the input graph, then we obtain MAX W-HEAVY and MIN $(W-1)$-LIGHT.

2 Preliminaries

For a graph G, we denote its vertex set and edge set by $V(G)$ and $E(G)$, respectively. For any fixed integer $W \geq 0$, an orientation of a graph is called a *W-orientation* if and only if the maximum outdegree is at most W. If a W-orientation exists, we say that the graph is *W-orientable*. For any $S \subseteq V$, we write $E(S)$ to denote the subset of edges whose both endpoints belong to S. Also, for any two disjoint subsets $S, T \subseteq V$, we write $E(S,T)$ to denote the subset of all edges such that one endpoint belongs to S and the other T. The ratio $|E(S)|/|S|$ is called the *density of S*. The *maximum density* D_G of a graph G is defined by $D_G = \max_{S \subseteq V} \left\lceil \frac{|E(S)|}{|S|} \right\rceil$ [1]. We denote a subgraph of G whose vertex set and edge set are respectively $V(G) \setminus S$ and $E(V(G) \setminus S)$ by $G \setminus S$. Finally, an orientation Λ of an undirected graph G is called an *Eulerian orientation* if $d^+_\Lambda(v) = d(v) - d^+_\Lambda(v)$, i.e., if the outdegree equals the indegree for every vertex.

It is known [12] that finding the maximum density of any graph is equivalent to finding the smallest integer W such that the graph is W-orientable:

Proposition 1 ([12]). *Any graph G is W-orientable if and only if $D_{G'} \leq W$ for all induced subgraphs G' in G.*

The following immediate consequence plays an important role in the paper. Note that the orientation referred to in Proposition 2 is an Eulerian orientation.

Proposition 2. *The complete graph K_{2W+1} has an orientation in which the indegree and outdegree of every vertex are equal to W.*

[1] The ceiling function gives the maximum degree of the vertices in the subgraph induced by S, where the maximum degree is an integer here.

Proposition 3 (p. 91 of [27]). *Given a graph G with all degrees even, an Eulerian orientation of G can be found in $O(m)$ time.*

The following proposition extends the notion of density D_G for our problems:

Proposition 4. *Consider a graph G and an orientation Λ of G, and assume that m' edges in $E(U, V(G) \setminus U)$ for a subset U of vertices are oriented outward from U to $V(G) \setminus U$ in Λ. Then, the average outdegree of the vertices in U is $(|E(U)| + m')/|U|$. As a result, there exists a vertex $v \in U$ such that $d^+_\Lambda(v) \geq \lceil (|E(U)| + m')/|U| \rceil$.*

For restricted instances, Max $(W+1)$-Heavy and Min W-Light can be solved in polynomial time. The fundamental idea of the algorithm is (i) first insert matching edges between odd degree vertices, and then (ii) orient the edges along with an Eulerian tour.

Proposition 5. *If the minimum degree δ of the input graph G satisfies $W+1 \leq \lfloor \delta/2 \rfloor$, an $O(m)$-time algorithm finds an optimal orientation for* Max $(W+1)$-Heavy *and* Min W-Light*, under which no vertex is W-light.*

Let $\delta^* = \max_\Lambda \min_v d^+_\Lambda(v)$. Since the algorithm in Proposition 5 outputs an orientation under which the minimum outdegree is at least $\lfloor \delta/2 \rfloor$, it always holds that $\delta^* \geq \lfloor \delta/2 \rfloor$. A known polynomial-time algorithm from [1] named Exact-1-MaxMinO outputs an orientation under which the minimum outdegree is δ^*, which gives the following corollary:

Corollary 1. *The algorithm* Exact-1-MaxMinO *outputs an optimal orientation for* Max $(W+1)$-Heavy *and* Min W-Light *when $W + 1 \leq \lfloor \delta/2 \rfloor$.*

An analogous discussion gives the following proposition and corollary, utilizing the polynomial-time algorithm Reverse from [5]:

Proposition 6. *If Δ satisfies $W \geq \lceil \Delta/2 \rceil$, then* Min $(W+1)$-Heavy *and* Max W-Light *can be solved in $O(m)$ time.*

Corollary 2. *The algorithm* Reverse *outputs an optimal orientation for* Min $(W+1)$-Heavy *and* Max W-Light *when $W \geq \lceil \Delta/2 \rceil$.*

3 Greedy Algorithms for Min W-Light and Min $(W+1)$-Heavy

In this section, for general W, we present greedy algorithms for Min W-Light and for Min $(W+1)$-Heavy, which use the same framework, but different criterion functions are adopted.

Here we explain the main idea of the greedy algorithm for Min W-Light. Our algorithm sequentially chooses vertices to be removed as violating vertices (W-light vertices). We refer by S the temporary vertices to be removed in Min W-Light, that is, S starts from $\emptyset$ and the size of S increases one-by-one by a greedy manner until $V(G) \setminus S$ becomes $(W+1)$-heavy. The criterion of the

greedy algorithm is defined by the following problem and its polynomial time solvability:

Problem Attainment of $(W+1)$-Heavy Orientation ($P_1(G,W,S)$)

$$\begin{aligned} \max \quad & \sum_{v \in V \setminus S} \min\{W+1, d^+_\Lambda(v)\} \\ \text{subject to} \quad & \Lambda \in \mathcal{A}(G) \ , \end{aligned}$$

where $\mathcal{A}(G)$ is the set of all orientations on G.

Since $P_1(G,W,S)$ can be solved via the maximum flow problem, we obtain the following lemma.

Lemma 1. Attainment of $(W+1)$-Heavy Orientation *($P_1(G,W,S)$) can be solved in $O(m^{1.5}\min\{m^{0.5}, \log m \log W\})$ time.*

Proof. The problem $P_1(G,W,S)$ can be reduced to the following maximum flow problem which can be solved in $O(m^{1.5}\min\{m^{0.5}, \log m \log W\})$ time [14,18,22]: For graph G, we construct network $\mathcal{N}(G,W,S)$, where the set of vertices is $\{s,t\} \cup E(G) \cup V(G)$ and the set of arcs is $\{(s,e) \mid e \in E\} \cup \{(e,u),(e,v) \mid e = \{u,v\} \in E(G)\} \cup \{(u,t) \mid u \in V(G)\}$. The capacities of the arcs are defined by

$$\begin{aligned} & cap((s,e)) = 1 && \text{for } e \in E(G), \\ & cap((e,u)) = 1 && \text{for } u \in e \in E(G), \text{ and} \\ & cap((u,t)) = \begin{cases} 0 & \text{for } u \in S, \\ W+1 & \text{for } u \in V(G) \setminus S \ . \end{cases} \end{aligned}$$

We can see that the objective value of $P_1(G,W,S)$ corresponds to the flow value of the network. In fact, flow with size one from s to $e(=\{u,v\})$ goes through either u or v (exactly one of u and v) by the flow integrality. This is interpreted as follows: $e = \{u,v\}$ is oriented as (u,v) if the flow via e goes through u, and (v,u) otherwise; the value of flow via u is considered the minimum of $W+1$ and the outdegree of u of the corresponding orientation. Thus, the optimal value of $P_1(G,W,S)$ is obtained by solving the maximum flow problem on $\mathcal{N}(G,W,S)$. □

By the optimality of the maximum flow, there is a simple characterization of an optimal orientation.

Lemma 2. *Λ is an optimal orientation of $P_1(G,W,S)$ if and only if there is not a directed path on Λ of $G \setminus S$ from any $(W+2)$-heavy vertex in $V \setminus S$ or a vertex in S to W-light vertex in $V \setminus S$.*

As mentioned above, we design a greedy algorithm that uses the optimal value of $P_1(G,W,S)$ as a criterion. Let $g_1(S)$ be the optimal value of $P_1(G,W,S)$ plus $|S|(W+1)$. It is easy to see that $g_1(S) = g_1(V)$ if $G \setminus S$ is $(W+1)$-heavy. Thus, by using this $g_1(S)$, Min W-Light can be formulated as $\min_{S \subseteq V}\{|S| \mid g_1(S) = g_1(V)\}$. We can show the following lemma.

Lemma 3. *$g_1(S)$ is a non-decreasing submodular function, that is, it satisfies that (non-decreasingness) $g_1(S \cup \{i\}) - g_1(S) \geq 0$ for any $i \in V \setminus S$, and (submodularity) $g_1(S) + g_1(T) \geq g_1(S \cap T) + g_1(S \cup T)$ for any $S, T \subseteq V$.*

Proof. For two disjoint subsets $S, S' \subseteq V$ of vertices, let us denote

$$\alpha(S, S') = \min\{ \sum_{v \in S'} \min\{W+1, d^+_\Lambda(v)\} \mid \Lambda \in OptO(P_1(G, W, S))\} \ ,$$

where $OptO(P_1(G, W, S))$ is the set of all optimal orientations of $P_1(G, W, S)$. To prove this lemma, we first show that

$$g_1(S \cup S') - g_1(S) = |S'|(W+1) - \alpha(S, S') \tag{1}$$

holds for any disjoint $S, S' \subseteq V$. Let $\Lambda_{S,S'}$ be an orientation that achieves $\alpha(S, S')$. We can see that $\Lambda_{S,S'}$ is also an optimal orientation of $P_1(G, W, S \cup S')$. In fact, by Lemma 2 and the optimality of $\Lambda_{S,S'}$ for $P_1(G, W, S)$, there is no directed path from $(W+2)$-heavy vertex in $V \setminus S$ or a vertex in S to W-light vertex in $\Lambda_{S,S'}$. Also, there exists no directed path from a vertex in S' to a W-light vertex in $V \setminus (S \cup S')$, otherwise it contradicts that $\Lambda_{S,S'}$ minimizes $\sum_{v \in S'} \min\{W+1, d^+_\Lambda(v)\}$. These imply the optimality of $\Lambda_{S,S'}$ for $P_1(G, W, S \cup S')$. Thus, we have

$$\begin{aligned} g_1(S) &= |S|(W+1) + \sum_{v \in V \setminus S} \min\{W+1, d^+_{\Lambda_{S,S'}}(v)\} \\ &= (|S \cup S'| - |S'|)(W+1) + \sum_{v \in V \setminus (S \cup S')} \min\{W+1, d^+_{\Lambda_{S,S'}}(v)\} \\ &\quad + \sum_{v \in S'} \min\{W+1, d^+_{\Lambda_{S,S'}}(v)\} \\ &= g_1(S \cup S') - |S'|(W+1) + \sum_{v \in S'} \min\{W+1, d^+_{\Lambda_{S,S'}}(v)\} \ , \end{aligned}$$

which is equivalent to (1). Note that the second equality in the above is based on the fact that S and S' are disjoint. By (1), $g_1(S \cup \{i\}) - g_1(S) = (W+1) - \alpha(S, \{i\}) \geq 0$ holds for any $i \in V \setminus S$, which implies the non-decreasing property of g_1.

We are ready to prove the submodularity of g_1. An equivalent condition of the submodularity of g_1 is that

$$g_1(S \cup \{i\}) - g_1(S) \geq g_1(S \cup \{i, j\}) - g_1(S \cup \{j\}) \tag{2}$$

for any $S \subseteq V$ and any $i, j \in V \setminus S$. By (1), we have

$$\begin{aligned} g_1(S \cup \{i\}) - g_1(S) &= (W+1) - \alpha(S, \{i\}), \\ g_1(S \cup \{j\}) - g_1(S) &= (W+1) - \alpha(S, \{j\}), \text{ and} \\ g_1(S \cup \{i, j\}) - g_1(S) &= 2(W+1) - \alpha(S, \{i, j\}) \ . \end{aligned}$$

Here, it is easy to see that $\alpha(S, \{i\}) + \alpha(S, \{j\}) \leq \alpha(S, \{i, j\})$ holds. This implies (2), the submodularity of g_1. □

It is known that optimization problems that form $\min_{S \subseteq V}\{|S| \mid g(S) = g(V)\}$ can be approximated within a $\log(\max_{i \in V}\{g(\{i\}) - g(\emptyset)\})$ factor by the following greedy algorithm, if g is a non-decreasing submodular function [30].

1. Set $S = \emptyset$.
2. Find an $i \in V \setminus S$ that maximizes $g(S \cup \{i\}) - g(S)$, and update $S := S \cup \{i\}$.
3. If $g(S) = g(V)$, then output S and halt. Otherwise, goto 2.

In our case, g_1 is a non-decreasing submodular function from Lemma 3, so it can be approximated within a $\log(\max_{i \in V}\{g_1(\{i\}) - g_1(\{\emptyset\})\}) \leq \log(W+1)$ factor by the greedy algorithm. In our case, this algorithm can be executed by n iterations of Step 2. Step 2 is done in $O(m^{1.5} \min\{m^{0.5}, \log m \log W\}) + O(mn)$ time, where $O(m^{1.5}\{m^{0.5}, \log m \log W\})$ time is for computing the maximum flow for $g_1(\emptyset)$ based on Lemma 1, and $O(mn)$ is n-times finding an augmenting path to compute $g_1(S \cup \{i\})$ from $g_1(S)$. We obtain the following theorem:

Theorem 1. Min W-Light *can be approximated within a factor of* $\log(W+1)$ *in* $O((mn + m^{1.5} \min\{m^{0.5}, \log m \log W\})n)$ *time.*

As for Min $(W+1)$-Heavy, we can obtain the similar theorem as follows, though we need to be a little more careful because we use the minimum cost flow for the proof, and it is not as simple as the maximum flow.

Theorem 2. Min $(W+1)$-Heavy *can be approximated within a factor of* $\log(\Delta - W)$ *in polynomial time.*

4 (In)approximability of the Problems

In this section, we give several results on the (in)approximability of the four problems, Max W-Heavy, Min W-Light, Min W-Heavy, and Max W-Light in this order.

In [4], the $\mathcal{NP}$-hardness of Max W-Heavy is shown for $W \geq 3$, however, no inapproximability results are known. The next theorem gives an inapproximability of Max W-Heavy for a sufficiently large W:

Theorem 3. *For* $W = \Omega(n^{1/3})$, Max W-Heavy *cannot be approximated within a factor of* $W^{1-\varepsilon}$ *in polynomial time for any constant* $\varepsilon > 0$ *unless* $\mathcal{P} = \mathcal{NP}$.

It should be noted that the proof of this theorem is based on the hardness of Max Independent Set. An important condition here is $W \geq \Delta$ of an instance of Max Independent Set. Since Max Independent Set is $\mathcal{NP}$-hard when $\Delta \geq 3$, the proof implies that Max W-Heavy is $\mathcal{NP}$-hard also when $W \geq 3$, i.e., we cannot show the hardness of Max W-Heavy for the case $W = 2$.

Next we give an inapproximability of Min W-Light here:

Theorem 4. Min 2-Light *and* Min 3-Light *cannot be approximated within a constant factor 100/99 and 53/52, respectively, in polynomial time unless* $\mathcal{P} = \mathcal{NP}$. *Furthermore, for sufficiently large* W, Min W-Light *cannot be approximated within a factor of* $\log(W+1) - O(\log \log W)$ *in polynomial time unless* $\mathcal{P} = \mathcal{NP}$.

Since Min 1-Heavy is equivalent to Min Vertex Cover [4], it can be approximated within a ratio of $2 - \Theta(1/\sqrt{\log n})$ [16]. Also, in this paper, we designed $O(\log(\Delta - W))$-approximation algorithm for Min W-Heavy in Theorem 2. On the other hand, the following inapproximability of Min W-Heavy can be also shown.

Theorem 5. *For every fixed $W \geq 1$,* Min W-Heavy *cannot be approximated within a ratio of* 1.3606 *in polynomial time unless $\mathcal{P} = \mathcal{NP}$.*

Proof. Since Min 1-Heavy is equivalent to Minimum Vertex Cover[4], Min 1-Heavy cannot be approximated within a ratio of 1.3606 in polynomial time unless $\mathcal{P} = \mathcal{NP}$ [9]. The hardness of approximating Min W-Heavy for every fixed $W \geq 2$ is shown by a gap-reserving reduction from Minimum Vertex Cover. Let $G = (V(G), E(G))$ be an input graph of Minimum Vertex Cover with n vertices. Then, we construct a graph $H = (V(H), E(H))$ of Min W-Heavy from G. Let $OPT(G)$ and $OPT'(H)$ denote the values of optimal solutions for G of Minimum Vertex Cover and for H of Min W-Heavy, respectively. Let $V(G) = \{v_1, v_2, \cdots, v_n\}$ of n vertices in G. The constructed graph H has n subgraphs H_1 through H_n. Each subgraph H_i consists of one vertex $u_{i,0}$ and a complete graph K^i_{2W-1} of $2W-1$ vertices, $u_{i,1}$ through $u_{i,2W-1}$. The vertex $u_{i,0}$ is connected to $W-1$ vertices $u_{i,1}$ through $u_{i,W-1}$. That is, the number of edges in the subgraph H_i is $(2W-1)(2W-2)/2 + (W-1) = 2W(W-1)$. If $\{v_i, v_j\}$ in G of Minimum Vertex Cover, then H_i and H_j are connected by an edge $\{u_{i,0}, u_{j,0}\}$. This reduction can be done in polynomial time. In the following we show that this reduction can completely preserve the approximation gap of $\alpha = 1.3606$ in Minimum Vertex Cover, i.e., $OPT(G) \leq k$ if and only if $OPT'(H) \leq k$ holds.

The following simple observation plays a key role in this proof: Now suppose that $\{v_i, v_j\} \in E(G)$. Then, consider the subgraph $G[V(H_i) \cup V(H_j)]$ induced by $V(H_i)$ and $V(H_j)$ connected by the edge $\{u_{i,0}, u_{j,0}\}$. One can see that $G[V(H_i) \cup V(H_j)]$ contains $|V(H_i)| + |V(H_j)| = 4W$ vertices and $|E(H_i)| + |E(H_j)| + 1 = 4W(W-1) + 1$ edges; the density of $G[V(H_i) \cup V(H_j)]$ is larger than $W-1$. This means that the maximum density is at least W so that at least one vertex in $G[V(H_i) \cup V(H_j)]$ must be W-heavy.

(Only-if part) Consider a vertex cover $S \subseteq V(G)$ with size at most k of G. Then we can give the following orientation of H: For the internal edges of K^i_{2W-1} in the ith subgraph H_i for every $i = 1, 2, \cdots, n$, we give an arbitrary orientation in which every vertex has outdegree $W-1$ by Proposition 2. The number of edges between $u_{i,0}$ and the complete graph K^i_{2W-1} in H_i is $W-1$, and those are oriented from $u_{i,0}$ to $W-1$ vertices in K^i_{2W-1}. At this moment, the outdegree of $u_{i,0}$ is exactly $W-1$. For an edge $\{u_{i,0}, u_{j,0}\}$ between H_i and H_j where $v_i \in S$ and $v_j \in V \setminus S$, we orient it from $u_{i,0}$ to $u_{j,0}$. If both vertices v_i and v_j are in S, then the edge $\{u_{i,0}, u_{j,0}\}$ is oriented arbitrarily. Since at least one vertex in $\{u_{i,0}, u_{j,0}\}$ between H_i and H_j is in S, the outdegree of a vertex in $V(G) \setminus S$ is $W-1$. The number of W-heavy vertices is at most k.

(If part) Consider an orientation Λ such that the number of W-heavy vertices in H is at most k. As observed above, at least one vertex in the subgraph induced

by two subgraphs H_i and H_j corresponding to two vertices in an edge $\{v_i, v_j\}$ in G is W-heavy. If the W-heavy vertex is in H_i, then we select the vertex v_i into the subset S of vertices. Otherwise, the vertex v_j is selected into S. Then, at least one endpoint of every edge in $E(G)$ must be in S. Thus, S is a vertex cover of G and $|S| \leq k$ holds by the assumption. □

Since Max 0-Light is equivalent to Max Independent Set [4], it cannot be approximated within a factor of $n^{1-\varepsilon}$ [31] while it can be approximated within a factor of $n(\log\log n)^2/(\log n)^3$ [11]. In the following we give the inapproximability and the approximability of Max W-Light for $W \geq 1$:

Theorem 6. *For every fixed $W \geq 1$,* Max W-Light *cannot be approximated within a factor of $(n/W)^{1-\varepsilon}$ in polynomial time unless $\mathcal{P} = \mathcal{NP}$.*

The following algorithm runs in linear time. Although it is quite simple, its approximation ratio is almost tight due to the inapproximability ratio of $\Omega((n/W)^{1-\varepsilon})$ above.

1. Pick any $\min\{2W+1, n\}$ vertices in the input G. Let the set of the chosen vertices be U.
2. Apply the algorithm in Prop. 6 to $G[U]$.
3. Orient the edges in $E \setminus E(U)$ connecting to any vertex in U towards U.
4. Orient the remaining edges arbitrarily.

Theorem 7. *There is a linear time $n/(2W+1)$-approximation algorithm for* Max W-light.

5 Degree-Bounded Graphs

In this section, the obtained results for input graphs with bounded degrees are briefly summarized.

First we can obtain a polynomial time 2-approximation algorithm by a slight modification to the one in Prop. 5; the main idea of the modification is to choose appropriate pairs of vertices having odd degrees, when inserting matching edges. Recall that if $\Delta = 2W$, then the problem Max W-Light can be solved in polynomial time by Corollary 2.

Theorem 8. *If $\Delta = 2W+1$, there is a polynomial time* 2*-approximation algorithm for* Max W-Light.

Next theorem shows the $\lfloor \frac{\Delta}{2} \rfloor$-approximability for Max 2-Heavy. The algorithm roughly works as follows: (i) first it obtains a line graph $L(G)$ of the input graph G, (ii) finds a maximum matching in $L(G)$, then (iii) converts the obtained matching to an orientation of G. Here an important property is that the size of the maximum matching in $L(G)$ guarantees the number of 2-heavy vertices in the resulted directed graph.

Theorem 9. *There is a polynomial time* $\lfloor \Delta/2 \rfloor$*-approximation algorithm for* Max 2-Heavy.

Based on this theorem, the following corollary holds, which shows one side of the complexity of Max 2-Heavy and Min 1-Light; it is unknown whether Max 2-Heavy and Min 1-Light are $\mathcal{NP}$-hard or not for general.

Corollary 3. Max 2-Heavy *and* Min 1-Light *can be solved in polynomial time when* $\Delta \leq 3$.

6 Concluding Remarks

In this paper, we have derived several new results on the complexity of Max W-Light, Min W-Light, Max W-Heavy, and Min W-Heavy. As for one technical aspect, we remark that the proof of the submodularity in Sect. 3 might be simplified using matroid theory. We would also like to note here that the 2-approximation algorithm for Feedback Vertex Set [6] gives a fundamental idea for a polynomial-time 2-approximation algorithm for Min 2-Heavy.

An interesting open question is whether Max 2-Heavy (or Min 1-Light) is $\mathcal{NP}$-hard for general graphs. Furthermore, there are still many gaps between the known polynomial-time approximability and inapproximability bounds for the problems; investigating stricter thresholds is a further research topic.

The problems were defined on unweighted graphs. A natural generalization is to let the vertices be weighted and try to minimize (or maximize) the total weights of heavy (or light) vertices. Under this generalization, designing algorithms becomes harder in general, but some of the presented approximation algorithms (e.g., the ones in Sect. 3) can easily be adjusted to the weighted version with the same approximation guarantees. Alternatively, the problems can be generalized by allowing the edges to be weighted, in which the outdegree of a vertex is defined by the total weights of outgoing edges.

References

1. Asahiro, Y., Jansson, J., Miyano, E., Ono, H.: Graph orientation to maximize the minimum weighted outdegree. International Journal of Foundations of Computer Science 22(3), 583–601 (2011)
2. Asahiro, Y., Jansson, J., Miyano, E., Ono, H., Zenmyo, K.: Approximation algorithms for the graph orientation minimizing the maximum weighted outdegree. Journal of Combinatorial Optimization 22(1), 78–96 (2011)
3. Asahiro, Y., Jansson, J., Miyano, E., Ono, H.: Upper and lower degree bounded graph orientation with minimum penalty. In: Proc. of CATS 2012. CRPIT Series, vol. 128, pp. 139–146 (2012)
4. Asahiro, Y., Jansson, J., Miyano, E., Ono, H.: Graph orientations optimizing the number of light or heavy vertices. In: Mahjoub, A.R., Markakis, V., Milis, I., Paschos, V.T. (eds.) ISCO 2012. LNCS, vol. 7422, pp. 332–343. Springer, Heidelberg (2012)

5. Asahiro, Y., Miyano, E., Ono, H., Zenmyo, K.: Graph orientation algorithms to minimize the maximum outdegree. International Journal of Foundations of Computer Science 18(2), 197–215 (2007)
6. Bafna, V., Berman, P., Fujito, T.: Constant ratio approximations of the weighted feedback vertex set problem for undirected graphs. In: Staples, J., Katoh, N., Eades, P., Moffat, A. (eds.) ISAAC 1995. LNCS, vol. 1004, pp. 142–151. Springer, Heidelberg (1995)
7. Chrobak, M., Eppstein, D.: Planar orientations with low out-degree and compaction of adjacency matrices. Theoretical Computer Science 86(2), 243–266 (1991)
8. Chung, F.R.K., Garey, M.R., Tarjan, R.E.: Strongly connected orientations of mixed multigraphs. Networks 15(4), 477–484 (1985)
9. Dinur, I., Safra, S.: On the hardness of approximating minimum vertex cover. Annals of Mathematics 162(1), 439–485 (2005)
10. Ebenlendr, T., Krčál, M., Sgall, J.: Graph balancing: A special case of scheduling unrelated parallel machines. In: Proc. of SODA 2008, pp. 483–490 (2008), Journal version: Graph balancing: A special case of scheduling unrelated parallel machines. Algorithmica (June 2012) published online doi:10.1007/s00453-012-9668-9
11. Feige, U.: Approximating maximum clique by removing subgraphs. SIAM Journal on Discrete Mathematics 18(2), 219–225 (2004)
12. Frank, A., Gyárfás, A.: How to orient the edges of a graph? In: Combinatorics, vol. I, pp. 353–364. North-Holland (1978)
13. Gabow, H.N.: Upper degree-constrained partial orientations. In: Proc. of SODA 2006, 554–563 (2006)
14. Goldberg, A.V., Rao, S.: Beyond the flow decomposition barrier. Journal of the ACM 45(5), 783–797 (1998)
15. Hakimi, S.L.: On the degrees of the vertices of a directed graph. Journal of the Franklin Institute 279(4), 290–308 (1965)
16. Karakostas, G.: A better approximation ratio for the vertex cover problem. ACM Transactions on Algorithms 5(4), Article 41(2009)
17. Kowalik, Ł.: Approximation scheme for lowest outdegree orientation and graph density measures. In: Asano, T. (ed.) ISAAC 2006. LNCS, vol. 4288, pp. 557–566. Springer, Heidelberg (2006)
18. King, V., Rao, S., Tarjan, R.E.: A faster deterministic maximum flow algorithm. J. Algorithms 23, 447–474 (1994)
19. Landau, H.G.: On dominance relations and the structure of animal societies: III The condition for a score structure. Bulletin of Mathematical Biophysics 15(2), 143–148 (1953)
20. Lovász, L.: Graph minor theory. Bulletin of the American Mathematical Society 43, 75–86 (2005)
21. Nash-Williams, C., St, J.A.: On orientations, connectivity and odd-vertex-pairings in finite graphs. Canadian Journal of Mathematics 12(4), 555–567 (1960)
22. Orlin, J.B.: Max flows in $O(nm)$ time, or better. In: Proc. of STOC 2013, pp. 765–774 (2013)
23. Picard, J.-C., Queyranne, M.: A network flow solution to some nonlinear 0-1 programming problems with application to graph theory. Networks 12, 141–159 (1982)
24. Robbins, H.E.: A theorem on graphs, with an application to a problem of traffic control. The American Mathematical Monthly 46(5), 281–283 (1939)
25. Robertson, N., Seymour, P.D.: Graph minors. XX. Wagner's conjecture. Journal of Combinatorial Theory Ser.B 92(2), 325–357 (2004)
26. Stanley, R.P.: Acyclic orientations of graphs. Discrete Mathematics 5(2), 171–178 (1973)

27. Schrijver, A.: Combinatorial Optimization. Springer (2003)
28. Vazirani, V.V.: Approximation Algorithms. Springer (2001)
29. Venkateswaran, V.: Minimizing maximum indegree. Discrete Applied Mathematics 143(1-3), 374–378 (2004)
30. Wolsey, L.A.: An analysis of the greedy algorithm for the submodular set covering Problem. Combinatorica 2(4), 385–393 (1982)
31. Zuckerman, D.: Linear degree extractors and the inapproximability of Max Clique and Chromatic Number. Theory of Computing 3(1), 103–128 (2007)

On the MAX MIN VERTEX COVER Problem*

Nicolas Boria[1], Federico Della Croce[2,3], and Vangelis Th. Paschos[4,5]

[1] Dalle Molle Institute for Artificial Intelligence (IDSIA), Manno, Switzerland
nicolas.boria@supsi.ch
[2] D.A.I., Politecnico di Torino, Italy
federico.dellacroce@polito.it
[3] CNR, IEIIT, Torino, Italy
[4] PSL Research University, Université Paris-Dauphine, LAMSADE CNRS UMR 7243
paschos@lamsade.dauphine.fr
[5] Institut Universitaire de France

Abstract. We address the MAX MIN VERTEX COVER problem, which is the maximization version of the well studied MIN INDEPENDENT DOMINATING SET problem, known to be **NP**-hard and highly inapproximable in polynomial time. We present tight approximation results for this problem on general graphs, namely a polynomial approximation algorithm which guarantees an $n^{-1/2}$ approximation ratio, while showing that unless $\mathbf{P} = \mathbf{NP}$, the problem is inapproximable within ratio $n^{\varepsilon-(1/2)}$ for any strictly positive ε. We also analyze the problem on various restricted classes of graph, on which we show polynomiality or constant-approximability of the problem. Finally, we show that the problem is fixed-parameter tractable with respect to the size of an optimal solution, to treewidth and to the size of a maximum matching.

1 Introduction

In the MIN INDEPENDENT DOMINATING SET problem, also called MIN MAX INDEPENDENT SET, given a graph $G(V, E)$, we are asked to determine a minimum size vertex-subset that is simultaneously independent and dominating. This problem, although polynomially solvable in strongly chordal graphs [20], has been proved to be inapproximable within $n^{1-\varepsilon}$, for any $\epsilon > 0$, not only in general graphs [1] but also in restricted graph classes as, for instance, the circle graphs [2]. Also, and probably due to this fact, exact solution of MIN INDEPENDENT DOMINATING SET in general or in restricted classes of graphs by moderately exponential algorithms has received a growing attention in the past years [3–5]. This problem has also been tackled using exponential approximation techniques [5]. Finally, it is shown to be very hard from a parameterized complexity point of view since it is **W[2]**-hard [6].

* Research supported by the French Agency for Research under the program TODO, ANR-09-EMER-010, by a Lagrange fellowship of the Fondazione CRT, Torino, Italy, and by the Swiss National Science Foundation project 200020_144491/1 "Approximation Algorithms for Machine Scheduling Through Theory and Experiments".

C. Kaklamanis and K. Pruhs (Eds.): WAOA 2013, LNCS 8447, pp. 37–48, 2014.

Surprisingly, to the best of our knowledge, the natural symmetric problem, the MAX MIN VERTEX COVER problem, where the goal is to compute a minimal (for exclusion) vertex cover of maximum size, has not been addressed yet. Knowing the direct applications of MIN INDEPENDENT DOMINATING SET in terms of ad-hoc wireless networks, it seems natural to study the symmetric version, where instead of minimizing the number of servers, one wishes to maximize the number of clients. This problem obviously has the same characteristics as its minimization counterpart in terms of **NP**-hardness and exact computation, but might have different behaviours in terms of approximability and parameterized complexity (as in the case of the pair MAX INDEPENDENT SET - MIN VERTEX COVER).

We show in this paper that, while also highly inapproximable, MAX MIN VERTEX COVER is better approximable than its mate, since it can be approximately solved in polynomial time within ratio $n^{-1/2}$, where n is the size of the input graph. This result is matched by an inapproximability bound of $n^{\varepsilon-(1/2)}$ that can be extended also to an $O(1/\Delta)$ inapproximability bound, where Δ is the maximum degree of the input graph. We also match it to an $O(3/2\Delta)$ approximation ratio achieved by a natural greedy algorithm. We also prove that, unlike MIN INDEPENDENT DOMINATING SET, MAX MIN VERTEX COVER is in **FPT**, the class of fixed-parameter tractable problems not only with respect to the standard parameter, i.e., the value of the optimum, but also with respect to the cardinality of a maximum matching (that is smaller than the value of the optimum). Let us note that both MIN WEIGHTED DOMINATING SET and MAX WEIGHTED INDEPENDENT SET are polynomially solvable in graphs with bounded treewidth [7–9] (and, actually, fixed parameter tractable with respect to the treewidth of the input graph [10]). With similar dynamic programming techniques, it can be shown that also both WEIGHTED MAX MIN VERTEX COVER and WEIGHTED MIN INDEPENDENT DOMINATING SET are fixed parameter tractable with respect to the treewidth. Since the techniques used for obtaining this result are quite similar to those in [10], the proof of the result is omitted.

2 Approximation of MAX MIN VERTEX COVER in General Graphs

We give in this section inapproximability upper bounds matched by lower bounds achieved in polynomial time for MAX MIN VERTEX COVER. We first study ratios functions of n and then functions of Δ.

Proposition 1. *For any positive constant ε,* MAX MIN VERTEX COVER *is inapproximable within ratio $O(n^{\varepsilon-(1/2)})$ unless $\mathbf{P} = \mathbf{NP}$.*

Proof. First, recall that MAX INDEPENDENT SET has been proved to be inapproximable within ratio $n^{\varepsilon-1}$ for a given $\varepsilon \in (0,1)$ [12] unless $\mathbf{P} = \mathbf{NP}$.

Consider an unweighted instance of MAX INDEPENDENT SET given by a graph $G(V,E)$. Out of this instance of MAX INDEPENDENT SET, we build an instance $H(V \cup S, E')$ of MAX MIN VERTEX COVER in the following way: for each vertex v of V one adds $n+1$ vertices connected only to v in H, while the

inner edges of the set V are left unchanged. In other words, graph H is obtained by adding an independent set S, of order $n^2 + n$ to the initial graph G, and connecting $n + 1$ vertices of the independent set to each vertex v.

Note that the graph H can be built in polynomial time, and has precisely $n^2 + 2n$ vertices. Denote by $\text{opt}(G)$ an optimal independent set in G, and by $\text{opt}(H)$ an optimal vertex cover in H.

Figure 1 provides an example of the construction where $\text{opt}(G)$ is the set of circled vertices, and $\text{opt}(H)$ the set of black vertices.

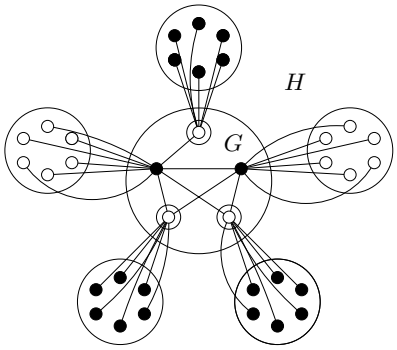

Fig. 1. An example of the reduction in Proposition 1

.

Consider a vertex cover $\text{SOL}(H)$ that has cardinality $\text{sol}(H)$ in H. First, notice that H admits a maximal matching of n edges, that consists of taking, for each vertex of V, one edge linking this vertex to one of its neighbors in S. Hence, any vertex cover in H takes at least n vertices, i.e., $\text{sol}(H) \geqslant n$.

Notice also that, for any vertex v of V that does not belong to $\text{SOL}(H)$, then $\text{SOL}(H)$ must take all its neighbors in S, that is $n + 1$ vertices. Moreover the set $V \setminus \text{SOL}(H)$ of vertices of V that do not belong to $\text{SOL}(H)$ defines an independent set in G with $n - |\text{sol}(H) \cap V|$ vertices. In other words, one can assert that any solution $\text{SOL}(H)$ of cardinality $\text{sol}(H)$ in H can be easily transformed into an independent set $\text{SOL}(G)$ in G of cardinality:

$$\text{sol}(G) \geqslant \frac{\text{sol}(H) - n}{n} \tag{1}$$

Conversely, the existence of a maximal independent set of size h in G induces the existence of a minimal vertex cover of size $nh + n$ in H. It suffices to consider the following vertex cover: all vertices of V that do not belong in the independent set ($n - h$ vertices), and all vertices of S linked to a vertex of the independent set ($h(n + 1)$ vertices). Therefore, it holds that a minimal vertex cover of size $n \cdot \text{opt}(G) + n$ exists in H. In other words, $\text{opt}(H) \geqslant n \cdot \text{opt}(G) + n$.

Now, for some constant positive $\rho < 1$, suppose that there exists a polynomial time algorithm `A` for MAX MIN VERTEX COVER that guarantees an approximation ratio $n^{-\rho}$, and suppose that a solution $\text{SOL}(H)$ has been computed by this

algorithm on graph H. Reminding that graph H has $O(n^2)$ vertices, the approximation ratio guaranteed by $\mathtt{A}$ on H turns to be $n^{-2\rho}$. Then, it holds that:

$$\mathrm{sol}(H) \geqslant n^{-2\rho} \cdot \mathrm{opt}(H) \tag{2}$$

By combining (1) and (2), one can assert that $\mathrm{SOL}(H)$ can easily be transformed into an independent set $\mathrm{SOL}(G)$ in G of value:

$$\begin{aligned} \mathrm{sol}(G) &\geqslant \frac{n^{-2\rho} \cdot \mathrm{opt}(H) - n}{n} \;\geqslant\; n^{-2\rho-1} \cdot \mathrm{opt}(H) - 1 \\ &\geqslant n^{-2\rho} \cdot \mathrm{opt}(G) + n^{-2\rho} - 1 \;\geqslant\; n^{-2\rho} \cdot \mathrm{opt}(G) \end{aligned}$$

where the last inequality holds for n big enough.

Hence, the existence of an $n^{-\rho}$-approximation algorithm $\mathtt{A}$ for MAX MIN VERTEX COVER induces the existence of an $n^{-2\rho}$-approximation algorithm for MAX INDEPENDENT SET, that would consist of:

- building the instance H of MAX MIN VERTEX COVER out of the instance G of MAX INDEPENDENT SET;
- running the algorithm $\mathtt{A}$ on the instance H that outputs a solution $\mathrm{SOL}(H)$;
- returning the solution $V \setminus \mathrm{SOL}(H)$ for the initial problem.

Since, for any constant $0 < \varepsilon \leqslant 1$, the existence of an $n^{1-\varepsilon}$-approximation algorithm for MAX WEIGHTED INDEPENDENT SET induces $\mathbf{P} = \mathbf{NP}$ [12], an $n^{-\rho}$-approximation algorithm for MAX MIN VERTEX COVER can exist only subject to the condition that $n^{-2\rho} \leqslant n^{1-\varepsilon}$ for any $0 < \varepsilon \leqslant 1$. This leads to $\rho \leqslant \varepsilon - 1/2$, which concludes the proof.

Observe now that the order of graph H in the gap-reduction of Proposition 1 is $O(n^2)$, while the maximum degree of H is $O(n)$. Then, the following inapproximability bound also can be immediately derived.

Corollary 1. MAX MIN VERTEX COVER *is inapproximable in polynomial time within ratios* $O(\Delta^{\epsilon-1})$*, for any* $\epsilon > 0$.

Let us now recall the following very classical and obvious observation that will be used later.

Remark 1. Denoting by M a maximum matching of a graph G, any vertex cover (a fortiori a minimal one) of G uses at least $|M|$ vertices (since, at least one distinct vertex is needed per one edge of M).

Lemma 1. *Consider a graph* $G(V, E)$ *and an independent set* S *of* G*. Denote by* $\Gamma(S)$ *the set of neighbors of* S*, and* $V' = V \setminus (S \cup \Gamma(S))$*. Finally, denote by* $SOL(G')$ *a minimal vertex cover on the induced subgraph* $G[V']$ *and by* $\mathrm{sol}(G')$ *its cardinality. It holds that* $\Gamma(S) \cup SOL(G')$ *is a feasible solution for* MAX MIN VERTEX COVER.

Proof. First, let us prove that $\Gamma(S) \cup \mathrm{sol}\,(V')$ is a vertex cover: all edges of $S \times \Gamma(S)$ and $\Gamma(S) \times V'$ are covered by vertices of $\Gamma(S)$, and all edges inside

the induced subgraph $G[V']$ are covered by $\mathrm{SOL}(G')$. By hypothesis, $S \times S = \emptyset$, so that $\Gamma(S) \cup \mathrm{SOL}(G')$ is, indeed, a vertex cover.

Then, let us establish the minimality of such a vertex cover: on the one hand, no vertex of $\Gamma(S)$ can be removed, as they all cover an edge linked to a vertex of S (and no vertex of S is in the vertex cover), and on the other hand, $\mathrm{SOL}(G')$ is a minimal vertex cover on a subgraph of G, so that none of its vertices can be removed without uncovering an edge.

Proposition 2. MAX MIN VERTEX COVER *is approximable within ratio* $n^{-1/2}$ *in polynomial time.*

Proof. Consider a graph $G(V, E)$, with $|V| = n$. Let $\Gamma(x)$ be the set of neighbors of a given vertex x and, given $V' \subseteq V$, let $G[V']$ be the subgraph of G induced by the set of vertices V'. Consider the following approximation algorithm for MAX MIN VERTEX COVER:

- compute a maximum matching M;
- among the matched vertices, let x be the one with the maximal number of exposed neighbors;
- compute a minimal vertex cover on $G[V']$ with a greedy algorithm, where $V' = V \setminus (\{x\} \cup \Gamma(x))$, and denote it by $\mathrm{SOL}(G')$;
- output $\mathrm{SOL}(G) = \Gamma(x) \cup \mathrm{SOL}(G')$.

First, by Lemma 1, we can assert that the solution returned by our approximation algorithm is feasible. Then notice that the algorithm runs in polynomial time, all steps of the algorithm are so: step 1 can be performed in $O(n^{2.376})$ time by the algorithm presented in [13], identifying vertex x is done in $O(n)$, and building a minimal vertex cover is done in $O(n^2)$ (starting from the whole set of vertices, the greedy algorithm deletes them one by one as long as the solution remains a vertex cover, when no vertex can be deleted, the remaining set is a minimal vertex cover). Finally, without loss of generality, let us suppose that the graph has no isolated vertices, since such vertices obviously make the problem easier to approximate.

Let us now analyze the approximation guarantee of this algorithm. Given the maximum matching M computed at the first step of the algorithm, denote by P the set of unmatched vertices of V with respect to M (i.e., the set of vertices of V that are not endpoints of M), which obviously forms an independent set. Finally, set $p = |P|$ and $m = |M|$. Our analysis is based upon the following *maximality argument* that will be used also in Proposition 3.

Notice that each edge (v_i, v_j) of M is linked to a set of vertices $P_{ij} \subseteq P$, so that $P_{ij} \subset \Gamma(v_i)$, or $P_{ij} \subset \Gamma(v_j)$. Indeed, suppose v_i has some neighbor v_k in P not linked to v_j, and v_j some neighbor v_l in P not linked to v_i. Then, by deleting (v_i, v_j) from M and adding (v_i, v_k) and (v_j, v_l) to it, one could produce a matching with $m + 1$ edges, so that M would not be a maximum matching. In other words, there exists a covering of P by m sets P_{ij}, each of them been included in the neighborhood of a single matched vertex. For the algorithm, this implies that the vertex x picked at step 2 has at least p/m neighbors in P.

The algorithm includes the whole neighborhood of this vertex x, which might also include some matched vertices. Suppose that $\Gamma(x)$ contains exactly h matched vertices. Then it holds that:

$$|\Gamma(x)| \geqslant h + \frac{p}{m} = h + \frac{n-2m}{m} \tag{3}$$

We now bound the value of $\mathrm{SOL}(G')$. Among the edges of the matching M, at least $m-h$ still exist in the subgraph $G[V']$. Indeed, this subgraph is obtained by deleting from G the vertex x together with all its neighbors, and all edges incident to these vertices. It is clear that, by deleting h vertices from M, only h edges are deleted from it. Thus, $G[V']$ contains a matching with $m-h$ edges, so that any vertex cover in $G[V']$ has at least $m-h$ vertices. A fortiori, this also holds for the vertex cover computed at step 3 of the algorithm; so:

$$\mathrm{sol}\,(G')\,| \geqslant m-h \tag{4}$$

Combining (3) and (4), we finally get the following bound on the value of the solution computed by the algorithm: $\mathrm{sol}(G) \geqslant m + \frac{n}{m} - 2 \geqslant \sqrt{n}$, where the last inequality results from a simple case analysis on the value of m with respect to $\sqrt{n}$: if $m \geqslant \sqrt{n}$, then the first term of the sum is at least $\sqrt{n}$ and the second at least 2 (as $m \leqslant n/2$). In the opposite case ($m < \sqrt{n}$), the second term of the sum is at least $\sqrt{n}$, where $m \geqslant 2$ (if $m = 1$, the graph is a star, and the problem is polynomial).

Considering that $\mathrm{opt}(G) \leqslant n$, the algorithm clearly guarantees an $n^{-1/2}$ approximation ratio, and the proof is concluded.

The following proposition, provides a lower bound, function of the maximum degree of the input graph, for the approximation ratio of MAX MIN VERTEX COVER.

Proposition 3. MAX MIN VERTEX COVER *is polynomially approximable within ratio* $3/2\Delta$*, where* Δ *is the maximum degree of the input graph. Furthermore, in bounded-degree graphs, regular graphs and graphs admitting a perfect matching* MAX MIN VERTEX COVER *is in* ***APX***.

Proof. Denote by d_i the degree of a vertex $v_i \in V$, by d the average degree of G and, as previously, by M a maximum matching of G, by m the cardinality of M and by p the cardinality of the set $P = V \setminus V(M)$ of the exposed vertices of V with respect to M.

The *maximality argument* stated in the proof of Proposition 2, has the following consequence for sets M and P:

> *for an edge* $(v_i, v_j) \in M$*, if one, say* v_i*, of its endpoints has more than one exposed neighbor, then* v_j *has no exposed neighbour at all; in the opposite case, an augmenting path would occur; in other words, in the case that a matching edge* (v_i, v_j) *is incident to some edge* (v, u) *with either* $v_i = v$*, or* $v_j = v$*, and* $u \in P$*, it holds that* $|(\Gamma(v_i) \cup \Gamma(v_j)) \cap P| \leqslant 1$.

Suppose that there exist a set M' of m' edges of M, whose one endpoint is adjacent to some exposed vertex of G with respect to M. Obviously, $p \leqslant m' \cdot \Delta$. Since $n = 2m + p$, we get, using the quoted consequence above, $n = 2m + p \leqslant 2(m - m') + (\Delta + 1)m' = 2m + (\Delta - 1)m' \leqslant (\Delta + 1)m$; hence:

$$m \geqslant \frac{n}{\Delta + 1} \tag{5}$$

By the seminal Turán's Theorem, every maximal (for inclusion) independent set of G has size at least $n/(d + 1)$; consequently:

$$\mathrm{opt}(G) \leqslant \frac{dn}{d + 1} \tag{6}$$

Combining (5) and (6) and taking into account Remark 1, the following holds for the approximation of every minimal vertex cover:

$$\frac{\mathrm{sol}(G)}{\mathrm{opt}(G)} \geqslant \frac{d + 1}{d(\Delta + 1)} \simeq \frac{d + 1}{d\Delta} \tag{7}$$

for arbitrarily large values of Δ. Also the following fact holds for any vertex cover of a graph $G(V, E)$ of order n.

> *Any vertex cover (a fortiori a minimal one) C guarantees approximation ratio at least $(d + 1)/2\Delta$ for* MAX MIN VERTEX COVER.

In fact, since C covers E, it holds that $\sum_{v_i \in C} d_i \geqslant |E|$. Also, $\sum_{v_i \in C} d_i \leqslant \Delta|C|$ and $|E| = nd/2$. Putting all this together, we get:

$$\mathrm{sol}(G) = |C| \geqslant \frac{nd}{2\Delta} \tag{8}$$

Combining (8) and (6), we derive:

$$\frac{\mathrm{sol}(G)}{\mathrm{opt}(G)} \geqslant \frac{d + 1}{2\Delta} \tag{9}$$

Ratio in (9) is increasing with d, while in (7) is decreasing with d. Equality holds for $d = 2$, which derives ratio $3/2\Delta$.

3 Parameterized Analysis

We prove in this section that, continuing the asymmetry between MIN INDEPENDENT DOMINATING SET and MAX MIN VERTEX COVER, the later is fixed parameter tractable when parameterized by the standard parameter, i.e., the cardinality opt of a maximum minimal vertex cover.

Proposition 4. MAX MIN VERTEX COVER *can be solved in* $O^*(4^{\mathrm{opt}/3})$.

Proof. Let, as previously, $\Gamma(v_i)$ be the neighborhood of vertex v_i. First, notice that if all vertices have degree $\leqslant 2$, then the problem becomes straightforwardly polynomially solvable by dynamic programming. Then, we assume that there exists a vertex v_j such that $d_j \geqslant 3$. Notice also that for each vertex v_i at least one vertex among the set $\Gamma(v_i) \cup \{v_i\}$ cannot be part of the vertex cover or else that vertex cover would not be minimal. For this, just observe that if $\Gamma(v_i) \cup \{v_i\}$ is included in the solution, v_i can be removed, since its incident edges are covered by the vertices of $\Gamma(v_i)$. We consider a branch and reduce approach where in each branch a vertex is excluded from the vertex cover and its neighbors are then necessarily included. We point out that such branch guarantees that all vertex covers generated will be minimal. We branch on vertex v_j according to the following exhaustive cases.

Case 1. $d_j \geqslant 3$ and all $v_i \in \Gamma(v_j)$ have degree $d_i \geqslant d_j$. We generate $|\Gamma(v_j)| + 1$ branches as follows: in one branch, vertex v_j is excluded from the vertex cover and correspondingly all its neighbors are included; in all other branches one of the vertices $v_i \in \Gamma(v_j)$ is excluded while all $v_k \in \Gamma(v_i)$ are included. This corresponds to $|d_j + 1|$ branches where in each branch at least $|d_j|$ vertices are included in the vertex cover. The worst-case occurs for $|d_j| = 3$, where we have four branches each including 3 vertices in the vertex set. Correspondingly, the complexity is $O^*(4^{\mathrm{opt}/3}) = O^*(1.5874^{\mathrm{opt}})$.

Case 2. $d_j \geqslant 3$ and there exists $v_i \in \Gamma(v_j)$ with $d_i = 2$. Three subcases occur with respect to the degree of the other neighbor v_k of v_i.

Subcase 2 (a). If $d_k \geqslant 3$, then either v_i or v_j or v_k are excluded from the vertex cover and correspondingly their neighbors are included in the vertex cover. Then, the recursion is at least $T(\mathrm{opt}) \leqslant T(\mathrm{opt} - 2) + 2T(\mathrm{opt} - 3)$ and the worst-case complexity is $O^*(1.5214^{\mathrm{opt}})$.

Subcase 2 (b). If $d_k = 2$, v_j may or may not be adiacent to v_k. If v_j and v_k are adjacent, then a branch on v_j can be performed: either v_j is excluded from the vertex cover and correspondingly its neighbors (at least three) are included in the vertex cover, or v_j is included in the vertex cover and arbitrarily v_i (v_k) is excluded from the vertex cover and v_k (v_i) is included in the vertex cover. Then, the recursion is at least $T(\mathrm{opt}) \leqslant T(\mathrm{opt} - 2) + T(\mathrm{opt} - 3)$ and the worst-case complexity is $O^*(1.3248^{\mathrm{opt}})$. Alternatively, v_j and v_k are non adjacent and v_k is adjacent to another vertex v_l. Then, either v_i is excluded from the vertex cover (and its two neighbors are included), or v_k is excluded from the vertex cover (and again its two neighbors are included), or both v_i and v_k are included in the vertex cover and correspondingly v_j and v_l are excluded from the vertex cover. In this last case, the other neighbors of v_j and v_l (that may possibly coincide) must be included in the vertex cover and globally at least four vertices must be included in the vertex cover. Correspondingly, the recursion is at least $T(\mathrm{opt}) \leqslant 2T(\mathrm{opt}-2)+T(\mathrm{opt}-4)$ and the worst-case complexity is $O^*(1.5538^{\mathrm{opt}})$ (notice that this last branch does not even occur if v_j and v_l are adjacent).

Subcase 2 (c). If $d_k = 1$, then v_i and v_j cannot be both included in the vertex cover as such solution is not better than the one with v_j and v_k included in the vertex cover and v_i excluded from the vertex cover. Then either v_i is

excluded from the vertex and two vertices (v_i andv_j) are included in the vertex cover, or v_j is excluded from the vertex cover and all its neighbors (at least three vertices) are included in the vertex cover. Correspondingly, the recursion is at least $T(\text{opt}) \leqslant T(\text{opt}-2)+T(\text{opt}-3)$ and the worst-case complexity is $O^*(1.3248^{\text{opt}})$.

Case 3. $d_j \geqslant 3$ and there exists $v_i \in \Gamma(v_j)$ with $d_i = 1$. We generate 2 branches where either v_i is excluded from the vertex cover and v_j is included, or v_j is excluded from the vertex and all its neighbors are included. Correspondingly, the recursion is at least $T(\text{opt}) \leqslant T(\text{opt}-1)+T(\text{opt}-3)$ and the worst-case complexity is $O^*(1.4656^{\text{opt}})$.

Overall, the worst-case is attained in case 1 with complexity $O^*(4^{\text{opt}/3}) = O^*(1.5874^{\text{opt}})$.

In what follows, we further strengthen the result of Proposition 4, showing that MAX MIN VERTEX COVER is FPT even when parameterized by the cardinality of a maximum matching M of the input graph (recall that $m \leqslant \text{opt}(G)$).

Proposition 5. MAX MIN VERTEX COVER *can be solved in* $O^*(3^m)$ *where* m *is the cardinality of a maximum matching of the input graph.*

Proof. Consider a general graph $G(V,E)$, and a maximum matching $M \subseteq E$ on G. All exposed vertices obviously form an independent set, that we denote by S. We also denote, as previously, by $V(M)$ the set of matched vertices.

We show that any feasible solution $\text{SOL}(G)$ for MAX MIN VERTEX COVER can be unequivocally characterized by its subset of matched vertices. Consider any subset $\text{SOL}(G) \cap V(M)$ of $V(M)$ known to be the subset of a unknown feasible solution $\text{SOL}(G)$. There actually exists a single solution $\text{SOL}(G)$ which admits $\text{SOL}(G) \cap V(M)$ as subset of matched vertices. Indeed denote by $\hat{S}$ the subset of S containing all exposed vertices incident to a matched vertex that does not belong in $\text{SOL}(G) \cap V(M)$. Then, the whole set $\hat{S}$ must be part of $\text{SOL}(G)$ in order to make it feasible. Conversely, all exposed vertices that do not belong to $\hat{S}$ cannot belong to $\text{SOL}(G)$, because they would make the solution non minimal: by definition, all of their neighbors already belong in the vertex cover.

Therefore, by identifying the subset $\text{OPT} \cap V(M)$, where OPT denotes a maximum minimal vertex cover, one would be able to reconstruct the whole solution OPT by simply adding to $\text{OPT} \cap V(M)$ all exposed vertices incident to a matched vertex not in the vertex cover.

Finally, notice that for each edge of the matching M, any vertex cover (a fortiori the optimal one) must take at least one endpoint of this edge. So, for each edge, any solution can take one endpoint, or the other, or both endpoints, that is three possibilities. Hence, there are at most 3^m vertex covers in the subgraph induced by the matched vertices.

Consider the following algorithm:

- compute a maximum matching M;
- build all 3^m possible vertex-covers $V_i \subseteq V(M)$ among the matched vertices;

– complete each of these vertex-covers by adding all vertices of S incident to a vertex not in the vertex cover;
– output the maximal feasible solution.

It is clear that through the exhaustive search performed at step 2 of the algorithm, the subset $\mathrm{OPT} \cap V(M)$ will be found, and when completed by exposed vertices at step 3, the optimal solution will be produced. Hence, an optimal solution can be computed in $O^*(3^m)$, and the proof is concluded.

Taking into account that $m \leqslant \tau$, the cardinality of a minimum vertex cover of the input graph, the following corollary immediately holds.

Corollary 2. MAX MIN VERTEX COVER *parameterized by* τ *is FPT.*

Let us now quickly point out how combination of Propositions 4 and 5 allows us to handle interesting trade-offs between parameterization and approximation. Indeed, we shall show that approximation ratios for MAX MIN VERTEX COVER, unachievable in polynomial time (unless an unlikely complexity condition holds), can be achieved in parameterized time. This issue has been already studied in [14–18], etc., for several problems, as MIN VERTEX COVER, STEINER TREE, MIN EDGE DOMINATING SET, several restricted versions of MIN HITTING SET, etc.

Revisit Proposition 5 and remark that if $m < (\log 1.5874/\log 3)\mathrm{opt}(G) \approx 0.42\mathrm{opt}(G)$, then the parameterized algorithm of Proposition 5 runs faster than that of Proposition 4 while, if $m \geqslant 0.42\mathrm{opt}(G)$, any minimal vertex cover (a fortiori the one of Proposition 3) achieves ratio greater than, or equal to, 0.42, ratio "forbidden" in polynomial time. For instance, we can guarantee ratio 0.1 in parameterized time less than $O^*(1162^{\mathrm{opt}})$, much smaller than $O^*(1.5874^{\mathrm{opt}})$, or even, approximation ratio 0,4 in time less than $O^*(1.552^{\mathrm{opt}})$ that always remains less than $O^*(1.5874^{\mathrm{opt}})$.

We conclude the section by showing that the same kind of trade-off can me made combining Proposition 4 and Proposition 1 [5] in order to get approximation results unachievable in polynomial time through exponential algorithms running faster than the currently best known exact algorithms. Recall that Proposition 1 in [5] claims that *for any positive* $\epsilon \leqslant 5$, MIN MAX INDEPENDENT SET *is* $(1+\epsilon)$*-approximable in time* $O^*(1.3351^{(1-(\epsilon/168))n})$.

If $\mathrm{opt}(G) < 0.626(1-(\epsilon/168))n$, then the algorithm of Proposition 4 computes a maximum minimal vertex cover of G in time smaller than $O^*(1.3351^n)$, which is the best worst-case complexity known for MIN MAX INDEPENDENT SET and, consequently, for MAX MIN VERTEX COVER. Suppose now that $\mathrm{opt}(G) \geqslant 0.626(1-(\epsilon/168))n$. In this case the $(1+\epsilon)$-approximation algorithm of Proposition 1 in [5] (indeed this algorithm can be seen as a kind of *moderately exponential approximation schema*) can be transformed a moderately exponential approximation schema for MAX MIN VERTEX COVER.

Denote by $\mathrm{opt}'(G)$ the size of a minimum dominating set in G and use the algorithm of Proposition 1 [5] in order to get an $(1+\epsilon)$-approximate independent

dominating set S. Obviously, the set $C = V \setminus S$ is a minimal vertex cover of G. The approximation ratio of C is:

$$\frac{|C|}{\mathrm{opt}(G)} = \frac{n - |S|}{n - \mathrm{opt}'(G)} \geqslant \frac{n - (1+\epsilon)\mathrm{opt}'(G)}{n - \mathrm{opt}'(G)} \tag{10}$$

The last expression in (10) decreases with $\mathrm{opt}'(G)$; since $\mathrm{opt}(G) \geqslant 0.626(1 - (\epsilon/168))n$, $\mathrm{opt}'(G) \leqslant (0.374 + (0.626\epsilon/168))n$ and setting it in the last term of (10) we get after some easy but tedious algebra that $|C|/\mathrm{opt}(G) \geqslant 1 + \epsilon'$ for some ϵ' that only depends on ϵ.

Let us finally note, that the same reasoning can be applied even with respect to future improved exact algorithms (just do the same analysis simply parameterizing the bases of the exponentials, i.e., using, for instance γ^n and δ^{opt} instead of 1.3351^n and 1.5874^{opt}).

References

1. Halldórsson, M.M.: Approximating the minimum maximal independence number. Inform. Process. Lett. 46, 169–172 (1993)
2. Damian-Iordache, M., Pemmaraju, S.V.: Hardness of approximating independent domination in circle graphs. In: Aggarwal, A.K., Pandu Rangan, C. (eds.) ISAAC 1999. LNCS, vol. 1741, pp. 56–69. Springer, Heidelberg (1999)
3. Gaspers, S., Liedloff, M.: A branch-and-reduce algorithm for finding a minimum independent dominating set in graphs. In: Fomin, F.V. (ed.) WG 2006. LNCS, vol. 4271, pp. 78–89. Springer, Heidelberg (2006)
4. Gaspers, S., Kratsch, D., Liedloff, M.: Exponential time algorithms for the minimum dominating set problem on some graph classes. In: Arge, L., Freivalds, R. (eds.) SWAT 2006. LNCS, vol. 4059, pp. 148–159. Springer, Heidelberg (2006)
5. Bourgeois, N., Della Croce, F., Escoffier, B., Paschos, V.T.: Fast algorithms for min independent dominating set. Discrete Appl. Math. 161, 558–572 (2013)
6. Downey, R.G., Fellows, M.R.: Parameterized complexity. Monographs in Computer Science. Springer, New York (1999)
7. Arnborg, S.: Efficient algorithms for combinatorial problems on graphs with bounded decomposability, A survey. BIT Numerical Mathematics 25, 1–23 (1985)
8. Arnborg, S., Proskurowski, A.: Linear time algorithms for NP-hard problems restricted to partial k-trees. Discrete Appl. Math. 23, 11–24 (1989)
9. Bodlaender, H.L.: Dynamic programming on graphs with bounded treewidth. In: Lepistö, T., Salomaa, A. (eds.) ICALP 1988. LNCS, vol. 317, pp. 105–118. Springer, Heidelberg (1988)
10. van Rooij, J.M.M., Bodlaender, H.L., Rossmanith, P.: Dynamic programming on tree decompositions using generalised fast subset convolution. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 566–577. Springer, Heidelberg (2009)
11. Boria, N., Della Croce, F., Paschos, V.: On the MAX MIN VERTEX COVER problem. Cahier du LAMSADE 343, LAMSADE (2013), `http://www.lamsade.dauphine.fr`
12. Lund, C., Yannakakis, M.: The approximation of maximum subgraph problems. In: Lingas, A., Carlsson, S., Karlsson, R. (eds.) ICALP 1993. LNCS, vol. 700, pp. 40–51. Springer, Heidelberg (1993)
13. Mucha, M., Sankowski, P.: Maximum matchings via gaussian elimination. In: Proc. FOCS 2004, pp. 248–255 (2004)

14. Cai, L., Huang, X.: Fixed-parameter approximation: Conceptual framework and approximability results. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 96–108. Springer, Heidelberg (2006)
15. Chen, Y.-J., Grohe, M., Grüber, M.: On parameterized approximability. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 109–120. Springer, Heidelberg (2006)
16. Downey, R.G., Fellows, M.R., McCartin, C.: Parameterized approximation problems. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 121–129. Springer, Heidelberg (2006)
17. Escoffier, B., Monnot, J., Paschos, V.T., Xiao, M.: New results on polynomial inapproximability and fixed parameter approximability of edge dominating set. In: Thilikos, D.M., Woeginger, G.J. (eds.) IPEC 2012. LNCS, vol. 7535, pp. 25–36. Springer, Heidelberg (2012)
18. Fellows, M.R., Kulik, A., Rosamond, F., Shachnai, H.: Parameterized approximation via fidelity preserving transformations. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part I. LNCS, vol. 7391, pp. 351–362. Springer, Heidelberg (2012)
19. Farber, M.: Independent domination in chordal graphs. Oper. Res. Lett. 1, 134–138 (1982)
20. Farber, M.: Domination, independent domination, and duality in strongly chordal graphs. Discrete Appl. Math. 7, 115–130 (1984)
21. Okamoto, Y., Uno, T., Uehara, R.: Linear-time counting algorithms for independent sets in chordal graphs. In: Kratsch, D. (ed.) WG 2005. LNCS, vol. 3787, pp. 433–444. Springer, Heidelberg (2005)

On Fixed Cost k-Flow Problems*

MohammadTaghi Hajiaghayi[1,**], Rohit Khandekar[2], Guy Kortsarz[3,***], and Zeev Nutov[4]

[1] University of Maryland, College Park, MD
hajiagha@cs.umd.edu.
[2] Knight Capital Group, Jersey City, NJ
rkhandekar@gmail.com
[3] Rutgers University–Camden, NJ
guyk@camden.rutgers.edu
[4] The Open University of Israel
nutov@openu.ac.il

Abstract. In the Fixed Cost k-Flow problem, we are given a graph $G = (V, E)$ with edge-capacities $\{u_e \mid e \in E\}$ and edge-costs $\{c_e \mid e \in E\}$, source-sink pair $s, t \in V$, and an integer k. The goal is to find a minimum cost subgraph H of G such that the minimum capacity of an st-cut in H is at least k. We show that Group Steiner is a special case of Fixed Cost k-Flow, thus obtaining the first polylogarithmic lower bound for the problem; this also implies the first non constant lower bounds for the Capacitated Steiner Network and Capacitated Multicommodity Flow problems. We then consider two special cases of Fixed Cost k-Flow. In the Bipartite Fixed-Cost k-Flow problem, we are given a bipartite graph $G = (A \cup B, E)$ and an integer $k > 0$. The goal is to find a node subset $S \subseteq A \cup B$ of minimum size $|S|$ such G has k pairwise edge-disjoint paths between $S \cap A$ and $S \cap B$. We give an $O(\sqrt{k \log k})$ approximation for this problem. We also show that we can compute a solution of optimum size with $\Omega(k/\text{polylog}(n))$ paths, where $n = |A| + |B|$. In the Generalized-P2P problem we are given an undirected graph $G = (V, E)$ with edge-costs and integer charges $\{b_v : v \in V\}$. The goal is to find a minimum-cost spanning subgraph H of G such that every connected component of H has non-negative charge. This problem originated in a practical project for shift design [10]. Besides that, it generalizes many problems such as Steiner Forest, k-Steiner Tree, and Point to Point Connection. We give a logarithmic approximation algorithm for this problem. Finally, we consider a related problem called Connected Rent or Buy Multicommodity Flow and give a $\log^{3+\epsilon} n$ approximation scheme for it using Group Steiner techniques.

* Part of this work was done at DIMACS. We thank DIMACS for their hospitality. A preliminary version appeared in archive [13] in 2011.
** Supported in part by NSF CAREER award 1053605, ONR YIP award N000141110662, DARPA/AFRL award FA8650-11-1-7162, and University of Maryland Research and Scholarship Award (RASA). The author is also with AT&T Labs– Research, Florham Park, NJ.
*** Supported in part by NSF grant number 434923.

C. Kaklamanis and K. Pruhs (Eds.): WAOA 2013, LNCS 8447, pp. 49–60, 2014.

1 Introduction

1.1 Problems Considered

Graphs in this paper are undirected, unless stated otherwise. For a graph H with edge capacities u_e (a default capacity of an edge is 1) let $\lambda_H(A, B)$ denote the max-flow/min-cut value between A and B in H. We study variants of the following network design problems from [7].

FIXED COST k-FLOW
Instance: A graph $G = (V, E)$ with edge-capacities $\{u_e \mid e \in E\}$ and edge-costs $\{c_e \mid e \in E\}$, source-sink pair $s, t \in V$, and an integer k.
Objective: Find a minimum cost subgraph H of G such that $\lambda_H(s, t) \geq k$.

We study a particular case of FIXED COST k-FLOW on a bipartite graph $G = (A \cup B, E)$, where all edges between s and A and between t and B exist and have infinite capacity and cost 1, and all the other edges have cost 0 and arbitrary capacity. A slightly simpler version was suggested to us by Deeparnab Chakrabarty in a personal communication. Our version can be casted as follows.

BIPARTITE FIXED-COST k-FLOW
Instance: A bipartite graph $G = (A \cup B, E)$ with integral edge-capacities $\{u_e : e \in E\}$, and an integer $k > 0$.
Objective: Find a minimum size set $S \subseteq A \cup B$ such that $\lambda_G(S \cap A, S \cap B) \geq k$.

The following special case of FIXED COST k-FLOW was shown in [7] to include several well studied problems, such as STEINER FOREST, k-STEINER TREE, and POINT TO POINT CONNECTION.

GENERALIZED POINT TO POINT CONNECTION (GENERALIZED-P2P)
Instance: An undirected graph $G = (V, E)$ with edge-costs $\{c_e \mid e \in E\}$, a subpartition V^+, V^- of V, and integer *charges* $\{b_v > 0 : v \in V^+\}$ and $\{b_v < 0 : v \in V^-\}$.
Objective: Find a minimum-cost spanning subgraph H of G such that $b(H') := \sum_{v \in H'} b_v \geq 0$ holds for every connected component H' of H.

To see that this problem is a special case of FIXED COST k-FLOW note that it is equivalent to the following problem. We are given a graph $G = (V, E)$, and disjoint sets $S \subseteq V$ of sources and $T \subseteq V$ of sinks. Every source or sink $v \in S \cup T$ is associated with a number b_v and such that $\sum_{t \in T} b_t \geq \sum_{s \in S} b_s$. The goal is to send b_s flow units from every $s \in S$ to the sinks, such that every sink t will receive at most b_t flow units. The main point is that all the graph edges have infinite capacity. Thus the entire graph is (a possibly expensive) feasible solution. Because the edges have infinite capacity, a solution H is feasible if and only if $\sum_{t \in C \cap T} b_t \geq \sum_{s \in C \cap S} b_s$ for every connected component C of H. This is the same as the POINT TO POINT CONNECTION problem.

FIXED COST k-FLOW is a particular case of the following two problems. In these problems we are given a graph $G = (V, E)$ with edge-capacities $\{u_e \mid e \in E\}$

and edge-costs $\{c_e \mid e \in E\}$, and requirements $\{r_{ij} \mid i, j \in V\}$. The goal is to find a minimum cost subgraph H of G such that for every $i, j \in V$, r_{ij} units of ij-flow can be sent in H. In Capacitated Multicommodity Flow the flows should be sent simultaneously for all commodities, while in Capacitated Steiner Network they are sent separately. In the case of one source and one sink both problems reduce to the Fixed Cost k-Flow problem.

Our last problem is related to Capacitated Multicommodity Flow with rooted requirements, and it is motivated by the following scenario. We are given an undirected graph $G = (V, E)$ with edge-capacities $\{u_e \mid e \in E\}$ and edge-costs $\{c_e \mid e \in E\}$, and flow demands $\{d_v : v \in V\}$ to a single sink t, where the flows should be delivered simultaneously. We have two options with regards to every edge e. First, we can *rent* $h(e) \leq u_e$ capacity units of e, and pay c_e per unit. The cost incurred in this case is $h(e) \cdot c_e$. The second possibility is to *buy* e, and then e can be assigned infinite capacity. Naturally, buying an edge is more expensive than renting a capacity unit of that edge. The cost incurred in this case is $M \cdot c_e$, where M is a large number called the *cost inflation factor*. (For simplicity we choose a uniform cost inflation factor, but in general there may be unrelated higher costs for buying then for renting, and our algorithms can handle this more general case as well.) Namely, we should determine a set E' of bought edges, which are assigned infinite capacity, and bandwidths $h(e) \leq u_e$ for edges in $E \setminus E'$. The overall cost is $M \cdot c(E') + \sum_{e \in E \setminus E'} h(e) \cdot c_e$.

Over this scenario, we add a requirement that the the set E' of bought edges form a connected graph (i.e., a tree) $G' = (V', E')$ that includes t. A similar constraint appears in several other problems, such as Connected Dominating Set, Connected Facility Location, and others.

Connected Rent or Buy Multicommodity Flow
Instance: A graph $G = (V, E)$ with with edge-capacities $\{u_e \mid e \in E\}$ and edge-costs $\{c_e \mid e \in E\}$, a cost inflation number $M > 0$, a sink node t, and demands $\{d_v : v \in V\}$.
Objective: Find a connected subgraph $G' = (V', E')$ of G containing t, and bandwidths $\{h(e) \leq u_e : e \in E \setminus E'\}$, such that after the edges in E' are given infinite capacity, and each $e \in E \setminus E'$ is given capacity $h(e)$, every $v \in V \setminus V'$ can deliver d_v flow units to t, where the flows should be delivered simultaneously. Minimize $M \cdot c(E') + \sum_{e \in E \setminus E'} h(e) c_e$.

1.2 Previous Work and Our Results

Directed Fixed Cost k-Flow was shown to be Label-Cover hard by Even et al. [7]. The same hardness result was rediscovered independently by Chakrabarty et al. [4]. Carr et al. [3] observed that the natural cut-LP has an unbounded integrality gap. They strengthened the cut-LP by adding so-called Knapsack-Cover inequalities, and obtained constant ratio approximation algorithms for some special graph topologies. However, in the general case, the integrality gap of the cut-LP enhanced by Knapsack-Cover inequalities is $\Theta(n^2)$. In [4] ratio $O(\log n)$ is obtained for the case of uniform requirements $r_{ij} = k$ for all $i, j \in V$.

In Capacitated Steiner Network with *soft capacities* every edge can be selected in multiple copies. For this variant, [4] give an an $\Omega(\log \log n)$ hardness, and an $O(\log k)$-approximation algorithm, where $k = |\{ij : r_{ij} > 0\}|$.

In the Group Steiner problem we are given an undirected graph $G = (V, E)$ with edge-costs $\{c_e \mid e \in E\}$, and a collection of groups $g_1, g_2, \ldots, g_k \subseteq V$. The goal is to find a minimum cost subtree H of G that contains at least one node from every group. In the Group Steiner on Trees problem, G is a tree rooted at a node r, every group is a subset of the leaves (a leaf may belong to many groups), and H should be a subtree of T rooted at r that contains at least one leaf from every group. Garg, Konjevod, and Ravi [9] present an $O(\log N \cdot \log k)$-approximation algorithm for Group Steiner on Trees where k is the number of groups, and N is the maximum size of a group. A combinatorial $O(\log^{2+\epsilon} n)$-approximation algorithm for the problem is given in [6], and a primal dual algorithm with a similar ratio is given in [16]. Halperin and Krauthgamer [14] prove that unless NP $\subseteq$ ZTIME $\left(n^{\log^c n}\right)$ for some constant c, for every constant $\epsilon > 0$, Group Steiner on Trees admits no $O(\log^{2-\epsilon} n)$ approximation.

In Sections 2 we give an approximation ratio preserving reduction from Group Steiner on Trees to the Fixed Cost k-Flow problem, thus obtaining the following result, that also implies the first non constant lower bound for Capacitated Steiner Network and Capacitated Multicommodity Flow.

Theorem 1. Fixed Cost k-Flow *admit no* $O\left(\log^{2-\epsilon} n\right)$ *approximation for any constant* $\epsilon > 0$, *unless* NP $\subseteq$ ZTIME $\left(n^{\log^c n}\right)$ *for some constant* c. *Consequently, the same hardness result holds for both* Capacitated Multicommodity Flow *and* Capacitated Steiner Network.

Recently, two years after the archive version [13] of this paper appeared, Chakrabarty, Krishnaswamy, Li, and Narayanan [5], improved our hardness result by showing that Fixed Cost k-Flow is Label-Cover hard. The methods in [5] are closely related to the ideas in our hardness result, namely, avoiding long paths in the solution.

Our results for Bipartite Fixed-Cost k-Flow and Generalized-P2P are given in the following two theorems, proved in Sections 3 and 4, respectively.

Theorem 2. Bipartite Fixed-Cost k-Flow *admits an* $O(\sqrt{k \ln k})$*-approximation algorithm. The problem also admits a* bicriteria *approximation algorithm that finds* $S \subseteq A \cup B$ *with* $|S| \leq |OPT|$ *such that* $\lambda_G(S \cap A, S \cap B) = \Omega(k/\text{polylog}(n))$.

Theorem 3. Generalized-P2P *with* $b(V) := \sum_{v \in V} b_v = 0$ *admits a* 2*-approximation algorithm. Furthermore, if* $b(V)$ *is polynomially bounded in* $n = |V|$, *then* Generalized-P2P *admits an exact algorithm on instances when the input graph is a tree, and an approximation algorithm with ratio* $O(\log \min\{n', 2 + b(V)\})$ *on general graphs, where* $n' = |V^+ \cup V^-|$.

As was mentioned, Generalized-P2P generalizes the Steiner Forest, the k-Steiner Tree, and the Point to Point Connection problems. Our algorithm gives a *single* algorithm for all these problems, but with a logarithmic ratio. It would be interesting to find a constant ratio approximation algorithm for

Generalized-P2P, as it would give a unifying constant ratio algorithm for both Steiner Forest and k-Steiner Tree.

We showed that the Group Steiner on Trees is a special case of Fixed Cost k-Flow. Conversely, we show that techniques applied previously to Group Steiner [6,2], can be applied to the related Connected Rent or Buy Multicommodity Flow problem. In the full version we prove the following theorem.

Theorem 4. Connected Rent or Buy Multicommodity Flow *admits an* $O(\log^{3+\epsilon} n)$*-approximation scheme, assuming all instance numbers are polynomial in* $n = |V|$.

2 Hardness of Fixed Cost k-Flow (Theorem 1)

Given an instance $(G = (V, E), \{c_e \geq 0 \mid e \in E\}, r, \{S_1, \ldots, S_k\})$ of Group Steiner on Trees, we construct an instance of Fixed Cost k-Flow as follows (see Figure 1 for an illustration). For a positive integer k, let $[k] = \{1, \ldots, k\}$. Construct a graph $G_+ = (V_+, E_+)$ from G by adding some new nodes and edges as follows. Let $V_+ = V \cup \{s\} \cup \{g_i \mid i \in [k]\}$ and $E_+ = E \cup F$ where $F = \{\{s, v\} \mid v \in \cup_{i\in[k]} S_i\} \cup \{\{v, g_i\} \mid v \in S_i, i \in [k]\} \cup \{\{g_i, r\} \mid i \in [k]\}$. Each edge $e \in E$ is assigned cost c_e and capacity $u_e = \infty$. Each edge $e = \{s, v\}$ for $v \in \cup_i S_i$ is assigned cost $c_e = 0$ and capacity $u_e = |\{i \mid v \in S_i, i \in [k]\}|$, i.e., the number of groups v belongs to. Each edge $e = \{v, g_i\}$ for $v \in S_i, i \in [k]$ is assigned cost $c_e = 0$ and capacity $u_e = 1$. Each edge $e = \{g_i, r\}$ for $i \in [k]$ is assigned cost $c_e = 0$ and capacity $u_e = |S_i| - 1$, i.e., one less than the number of nodes in the group S_i. Finally we set sink as $t = r$ and demand as $d = \sum_{i\in[k]} |S_i| = \sum_{v\in V} |\{i \mid v \in S_i, i \in [k]\}|$.

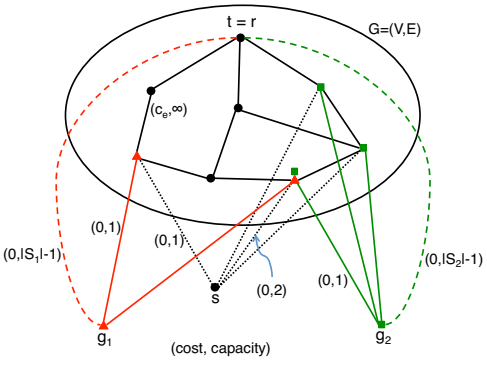

Fig. 1. The instance of Fixed Cost k-Flow created in the reduction from Group Steiner on Trees. The labels on the edges denote (cost, capacity). Not all labels are shown in the figure.

Now we show the following one-to-one correspondence between the feasible solutions of the original Group Steiner on Trees instance and that of the created Fixed Cost k-Flow instance.

Lemma 1. *There exists a solution for the* Group Steiner on Trees *instance of cost at most* C *if, and only if, there exists a solution for* Fixed Cost k-Flow

instance of cost at most C. Furthermore, the solution to Group Steiner on Trees *can be computed in polynomial time from that to the* Fixed Cost k-Flow *instance, and vice versa.*

Proof. Let subtree $T = (V_T, E_T)$ be a solution of cost C to the Group Steiner on Trees instance. Let $H = E_T \cup F$ be a subgraph of G_+. Since all edges in F have cost 0, the cost of H is also C. We now argue that H forms a feasible solution to the Fixed Cost k-Flow instance, i.e., a flow of d units can be routed from s to t in H. We start by routing flow of $u_{\{s,v\}} = |\{i \mid v \in S_i, i \in [k]\}|$ units from s path from it to r in the tree T. This flow can be supported since received flow to each g_i for which $v \in S_i$ along the most $|S_i| - 1$ units of flow from all the nodes $v \in S_i$. This is because at most $|S_i| - 1$ nodes in S_i do not belong to T, which along edge $\{g_i, r\}$ of capacity $|S_i| - 1$. Thus indeed H forms a feasible solution to the Fixed Cost k-Flow instance.

Now let H be a solution of cost C to the Fixed cost flow instance. Since all edges in F have zero cost, we can assume that $F \subseteq H$, without loss of generality. It is enough to prove that $i \in [k]$. Suppose this is not true for some group S_j for $j \in [k]$. We extract an s-t-cut in graph H with capacity strictly less than d contradicting the existence of flow of value d from s to t in H. Let $U \subseteq V$ denote the set of nodes connected to some node in S_j in $H \cap E$ and let $U = \{s, g_j\} \cup U$. Note that $s \in U$ while from our assumption $t \notin U$. We now prove the following claim.

Claim. The total capacity of edges in H that leave U is strictly less than d.

Proof. It is easy to note that all the edges in H that leave U are (1) $\{g_j, r\}$ with capacity $|S_j| - 1$, (2) $\{v, g_i\}$ with capacity 1, for all $i \neq j$ and $v \in S_i \cap U$, and (3) $\{s, v\}$ with capacity $|\{i \mid v \in S_i, i \in [k]\}|$ for all $v \in V \setminus U$. Thus the total capacity of these edges is

$$
\begin{aligned}
&|S_j| - 1 + \sum_{i \neq j} \sum_{v \in S_i \cap U} 1 + \sum_{v \in V \setminus U} |\{i \mid v \in S_i, i \in [k]\}| \\
&= |S_j| - 1 + \sum_{v \in U} |\{i \mid v \in S_i, i \in [k], i \neq j\}| + \sum_{v \in V \setminus U} |\{i \mid v \in S_i, i \in [k]\}| \\
&= -1 + \sum_{v \in U} |\{i \mid v \in S_i, i \in [k]\}| + \sum_{v \in V \setminus U} |\{i \mid v \in S_i, i \in [k]\}| \\
&= -1 + \sum_{v \in V} |\{i \mid v \in S_i, i \in [k]\}| = d - 1.
\end{aligned}
$$

This finishes the proof of the claim. □

The above claim implies that $H \cap E$ indeed contains a path from some node in S_i to r for each $i \in [k]$, establishing that it is a feasible solution to Group Steiner on Trees. From the reduction, it is also clear that the solution to Group Steiner on Trees can be computed in polynomial time from that to the Fixed Cost k-Flow instance, and vice versa. This completes the proof of Lemma 1. □

Theorem 1 now follows from Lemma 1 and the hardness result for Group Steiner on Trees given in [14].

3 Bipartite Fixed-Cost k-Flow (Theorem 2)

3.1 An $O(\sqrt{k \log k})$ Approximation Algorithm

For $S \subseteq A \cup B$ let $f(S) = \min\{\lambda_G(S \cap A, S \cap B), k\}$, where we define $f(\emptyset) = 0$. Consider the following algorithm.

Algorithm Greedy

1. $S \leftarrow \emptyset$.
2. While $f(S) < k$ do:
   Add to S a node pair $P = \{a, b\}$ such that $\frac{f(S \cup P) - f(S)}{c(P)}$ is maximum.
3. Return S.

For the analysis, consider a generic Covering Problem defined as follows. Let $f : 2^U \mapsto \mathbb{Z}$ and $c : 2^U \mapsto \mathbb{R}^+$ be two increasing set functions on a groundset U given by an evaluation oracle, where c is also subadditive. The goal is to find $S \subseteq U$ with $f(S) = f(U)$ such that $c(S)$ is minimum. Let OPT be an optimal solution. A ρ-greedy algorithm starts with $S \leftarrow \emptyset$ and while $f(S) < f(U)$ repeatedly adds to S a set $P \subseteq U$ that satisfies the *density condition*

$$\frac{f(S \cup P) - f(S)}{c(P)} \geq \frac{1}{\rho} \cdot \frac{f(U) - f(S)}{c(OPT)} \ .$$

If at each iteration a set P as above is computed in polynomial time in $|U|$, then the algorithm runs in time polynomial in $|U|$ and $\ln(f(U) - f(\emptyset))$, and computes a solution S such that $c(S) \leq \rho(\ln(f(U) - f(\emptyset)) + 1) \cdot c(OPT)$.

Note that in the context of the Bipartite Fixed-Cost k-Flow problem with $U = A \cup B$, the functions $f(S), c(S)$ satisfy the needed assumptions; both are increasing, the function c is subadditive, and $f(S) = f(U)$ if and only if S is a feasible solution. Also note that $f(U) - f(\emptyset) = k$.

We show that for Bipartite Fixed-Cost k-Flow there exists a pair $P = \{a, b\}$ that satisfies the density condition with $\rho = |OPT| \leq 2k$. For simplicity of the analysis, we consider uncapacitated multigraphs, by replacing every edge e of capacity u_e by u_e parallel edges. Let $S^* = OPT \setminus S$ and let

$$\mathcal{P} = \{\{a, b\} : a \in S^* \cap A, b \in OPT\} \cup \{\{a, b\} : a \in OPT \cap A, b \in S^* \cap B\} \ .$$

We have

$$\begin{aligned} \sum_{P \in \mathcal{P}} c(P) &= c(S^* \cap B) \cdot |OPT \cap A| + c((OPT \setminus S) \cap A) \cdot |(OPT \cap B| \\ &\leq c(OPT) \cdot |OPT \cap A| + c((OPT) \cdot |(OPT \cap B| = c(OPT) \cdot |OPT| \ . \end{aligned}$$

By the Menger's theorem, there exists a set of k edge-disjoint paths between $OPT \cap A$ and $OPT \cap B$, where each path is between some $a \in OPT \cap A$ and $b \in OPT \cap B$. For $P \in \mathcal{P}$ let $f^*(P)$ be the number of paths between the two nodes of P. At most $f(S)$ paths may connect pairs not in $\mathcal{P}$. This implies

$$\sum_{P \in \mathcal{P}} (f(S) - f(S \cup P)) \geq \sum_{P \in \mathcal{P}} f^*(P) \geq k - f(S) \ .$$

Thus by an averaging argument there exists $P \in \mathcal{P}$ such that

$$\frac{f(S \cup P) - f(S)}{c(P)} \geq \frac{1}{|OPT|} \cdot \frac{k - f(S)}{c(OPT)} .$$

Consequently, the greedy algorithm above computes a solution S such that $c(S) \leq |OPT|(\ln k + 1)c(OPT)$.

By the same argument we have that if $f(S) < k$, there exists a pair P such that $f(S \cup P) - f(S) \geq 1$. This implies that at the end of the algorithm $|S| \leq 2k$.

In the case of unit costs, we have $c(OPT) = |OPT|$ and $c(S) = |S|$. If $|OPT| \geq \sqrt{2k/(\ln k + 1)}$ then $|S| \leq 2k \leq \sqrt{2k(\ln k + 1)}|OPT|$. If $|OPT| \leq \sqrt{2k/(\ln k + 1)}$ then $|S| \leq |OPT| \cdot (\ln k + 1) \cdot |OPT| \leq \sqrt{2k(\ln k + 1)}|OPT|$. Thus in the case of unit costs, the algorithm has approximation ratio $\sqrt{2k(\ln k + 1)}$.

3.2 A Bicriteria Approximation Algorithm

We reduce the problem to tree instances using the theorem of Harrelson, Hildrum, and Rao [15]. A tree decomposition $\mathcal{T}$ of V is a sequence $\Pi_0, \ldots, \Pi_d$ of partitions of V, where $\Pi_0 = \{V\}$, $\Pi_d = \{\{v\} : v \in V\}$ and each Π_i is a refinement of Π_{i-1}. Such tree decomposition can be represented by a rooted tree, which we also denote by $\mathcal{T}$. The root of $\mathcal{T}$ is $\{V\}$. The nodes in layer i are the sets in Π_i and the leaves correspond to the sets in Π_d, i.e., the nodes in V. The edges of the tree go between the consecutive layers and are given by set inclusion. If $G = (V, E)$ is a graph with edge capacities u_e, then the weight of an edge (S, T) of $\mathcal{T}$ is $w(S, T) = u(\delta_G(S))$.

Now consider an instance of multi-commodity flow demands $M = \{d_{ij} \geq 0 \mid i, j \in V\}$ between pairs of nodes. Let $c_G(M)$ (resp., $c_{\mathcal{T}}(M)$) denote the minimum maximum edge-congestion under which M can be routed in G (resp., $\mathcal{T}$). Harrelson et al. [15] proved the following theorem.

Theorem 5 (Harrelson et al. [15]). *In time polynomial in $n = |V|$, one can compute a tree decomposition $\mathcal{T}$ with depth $d = O(\log n)$ such that for any multi-commodity flow instance M, we have*

- *$c_{\mathcal{T}}(M) \leq c_G(M)$, and*
- *given a routing of M with maximum edge-congestion $c_{\mathcal{T}}(M)$ in $\mathcal{T}$, we can compute in polynomial time, a routing of M with maximum edge-congestion $O(\log^2 n \log\log n) \cdot c_{\mathcal{T}}(M)$ in G.*

We use the above theorem to compute a tree decomposition $\mathcal{T}$ for the input graph $G = (A \cup B, E)$. It is easy to see that the optimum solution of the Bipartite Fixed-Cost k-Flow instance in G induces a solution A^*, B^* in the tree $\mathcal{T}$ of the same value and so that we can route at least k units of flow between A^* and B^* in $\mathcal{T}$. We next give an exact algorithm to find sets $A' \subseteq A, B' \subseteq B$ in $\mathcal{T}$ with minimum $|A'| + |B'|$ so that we can route k units of flow between them. The optimum solution A', B' in $\mathcal{T}$, in turn, induces a solution A', B' in G of value at most that of the optimum such that we can route $\Omega(k/\log^2 n \log\log n)$ flow.

The algorithm on tree instances uses dynamic programming. For each node $u \in \mathcal{T}$ and values $0 \leq F, F^+, F^- \leq k$, we use $S^+(u, F, F^+)$ (resp., $S^-(u, F, F^-)$)

to denote the minimum of $|A'| + |B'|$ such that there exist subsets $A' \subseteq A$ and $B' \subseteq B$ in the subtree $\mathcal{T}_u$ of $\mathcal{T}$ hanging below u so that we can route a flow of F units between A' and B' and send out (resp., bring in) a flow of F^+ (resp., F^-) units from nodes in A' (resp., from u) to u (resp., to nodes in B'), using only the edges in $\mathcal{T}_u$. It is easy to compute S^+ and S^- values for leaf nodes $u \in \mathcal{T}$. Furthermore, given a non-leaf node $v \in \mathcal{T}$ and its children $u_1, \ldots, u_p$, it is easy to compute S^+ and S^- values for v from the corresponding values for its children. Finally, we read off the value of $S^+(r, k, 0)$ (or, equivalently $S^-(r, k, 0)$) (where r is the root of $\mathcal{T}$) to compute the optimum solution.

4 Generalized-P2P (Theorem 3)

4.1 An Exact Algorithm on Trees

Here we show how to solve Generalized-P2P optimally on instances when the input graph is a tree T, using dynamic programming. Root T at some node s. By adding zero-charge nodes and zero-cost edges to T if necessary, we can assume that T is a binary tree. If a node v has one child, then we add an additional child to v of charge 0, connected by an edge of cost 0. If v has at least 3 children, we add a binary tree rooted at v whose leaves are the children of v. In this binary tree, each leaf u is connected to to its parent by an edge of cost c_{uv}; non-leaf edges have cost 0 and non-leaf nodes distinct from v have charge 0. It is easy to see that the problem essentially remains unchanged by this modification.

For $v \in V$ let T_v denote the subtree of T that consist of v and its descendants. For an integer $B \in [b(V^-), b(V^+)]$ let $T(v, B)$ be the minimum-cost of a subgraph H of T_v satisfying the following:

- The connected component in H containing v has total charge B.
- Every other connected component in H has non-negative total charge.

If there is no subgraph H satisfying the above conditions, then $T(v, B) = \infty$. The optimal solution value is $\min\{T(s, B) \mid B \geq 0\}$. The dynamic program computes quantities $T(v, B)$ for all $v \in T$ and integer $B \in [b(V^-), b(V^+)]$. Since each b_u is polynomially bounded, the number of such quantities is polynomial. We assume that the corresponding minimum-cost subgraph H is also stored in the dynamic program table.

The quantities $T(v, B)$ can be computed as follows. If v is a leaf, then computing $T(v, B)$ is trivial. For an internal node v, we compute $T(v, B)$ as follows. Let u_1 and u_2 be the two children of v. Depending on the set $F \subseteq \{vu_1, vu_2\}$ picked to the solution, we get four possibilities.

1. $F = \emptyset$. Then $T(v, B) = \min\{T(u_1, B_1) + T(u_2, B_2) \mid B_1, B_2 \geq 0\}$ if $B = b_v$, and $T(v, B) = \infty$ otherwise.
2. $F = \{vu_1\}$. Then $T(v, B) = \min\{c_{vu_1} + T(u_1, B_1) + T(u_2, B_2) \mid B_2 \geq 0\}$ if $B = b_v + B_1$, and $T(v, B) = \infty$ otherwise.
3. $F = \{vu_2\}$. Then $T(v, B) = \min\{c_{vu_2} + T(u_2, B_2) + T(u_1, B_1) \mid B_1 \geq 0\}$ if $B = b_v + B_2$, and $T(v, B) = \infty$ otherwise.
4. $F = \{vu_1, vu_2\}$. Then $T(v, B) = \min\{c_{vu_1} + T(u_1, B_1) + c_{vu_2} + T(u_2, B_2)\}$ if $B = b_v + B_1 + B_2$, and $T(v, B) = \infty$ otherwise.

Among these possibilities, we pick the minimum-cost solution corresponding to each value of the charge of the connected component containing v.

4.2 A 2-Approximation Algorithm for the Case $b(V) = 0$

Our 2-approximation algorithm generalizes the algorithm of [12] which is the case $b_v \in \{-1, 0, 1\}$. We say an edge e *covers* a set S if e has exactly one endnode in S; we say that an edge-set/graph covers a set family $\mathcal{F}$ if for every $S \in \mathcal{F}$ there is an edge in H covering S. Given a set-family $\mathcal{F}$ and an edge-set H the residual set-family $\mathcal{F}_H$ consists of the members of $\mathcal{F}$ not covered by H. Recall that a set-family $\mathcal{F}$ is *uncrossable* if for any $X, Y \in \mathcal{F}$ at least one of the following holds: $X \cap Y, X \cup Y \in \mathcal{F}$ or $X \setminus Y, Y \setminus X \in \mathcal{F}$. It is known and easy to see that if $\mathcal{F}$ is uncrossable, so is $\mathcal{F}_H$, for any edge-set H.

Goemans et al. [11] give a primal-dual 2-approximation algorithm for the problem of finding a minimum-cost edge-cover of an uncrossable set-family $\mathcal{F}$. A polynomial time implementation of this algorithm requires only that for any edge-set H, the minimal members of the residual set-family $\mathcal{F}_H$ can be computed in polynomial time (but $\mathcal{F}$ itself may not be given explicitly). Now the 2-approximation algorithm follows from the following lemma.

Lemma 2. *Given an instance of* Generalized-P2P *with* $b(V) = 0$, *let* $\mathcal{F} = \{S \subseteq V \mid b(S) \neq 0\}$. *Then the following holds.*

(i) *An edge-set* $H \subseteq E$ *is a feasible solution to* Generalized-P2P *if, and only if,* H *covers* $\mathcal{F}$.
(ii) *For any edge set* $H \subseteq E$, S *is an inclusion-minimal members of* $\mathcal{F}_H$ *if, and only if* S *is a connected component of the graph* (V, H) *and* $b(S) \neq 0$.
(iii) $\mathcal{F}$ *is uncrossable.*

Proof. Parts (i) and (ii) are straightforward, so we prove only part (iii). Let $X, Y \in \mathcal{F}$, so $b(X), b(Y) \neq \emptyset$. We will show that if $X \cap Y \notin \mathcal{F}$ or if $X \cup Y \notin \mathcal{F}$, then $X \setminus Y, Y \setminus X \in \mathcal{F}$. Suppose that $X \cap Y \notin \mathcal{F}$, so $b(X \cap Y) = 0$. Then $b(X \setminus Y) = b(X) - b(X \cap Y) = b(X) \neq 0$ and $b(Y \setminus X) = b(Y) - b(Y \cap X) = b(Y) \neq 0$; hence $X \setminus Y, Y \setminus X \in \mathcal{F}$. Suppose that $X \cup Y \notin \mathcal{F}$, so $b(X \cup Y) = 0$. Then $b(X \setminus Y) = b(X \cup Y) - b(Y) = -b(Y) \neq 0$ and $b(Y \setminus X) = b(X \cup Y) - b(X) = -b(X) \neq 0$; hence $X \setminus Y, Y \setminus X \in \mathcal{F}$. □

4.3 An $O(\log |V^+ \cup V^-|)$-Approximation Algorithm

We already proved that Generalized-P2P can be solved optimally on tree instances. We next reduce the general problem to the case when the input graph is a tree with a loss of $O(\log n')$ factor in the approximation ratio, where $n' = |V^+ \cup V^-|$. This is achieved as follows. Consider the shortest-path metric on $V' = V^+ \cup V^-$ w.r.t. the edge-costs c_e. We probabilistically embed this metric into a tree metric T, c' with $O(\log n')$ distortion using the results of Bartal [1] and Fakcharoenphol, Rao and Talwar [8]. There is a one-to-one correspondence between V' and the set L of leaves of T. The resulting instance of Generalized-P2P on T inherits the charges on the leaves of T from the original charges on

nodes of V', while the charge of internal nodes of T is 0. We compute an optimal solution to the obtained tree instance, and return the corresponding subgraph H of G. Note that any feasible solution with cost C for the original instance induces a solution with cost $O(C \log n')$ for the new instance on tree T. Similarly any feasible solution with cost C for the new instance induces a solution with cost C for the original instance. Hence the approximation ratio is bounded by the distortion of the reduction, which is $O(\log n')$.

Now consider the augmentation version of the problem, when we are give an edge subset $E' \subseteq E$ of cost 0. Then we can contract every connected component F of (V, E') into a single node v_F with charge $b(v_F) = b(F)$. Thus the approximation ratio in this case is $O(\log n')$, where n' is the number of connected components with non-zero charge in the graph (V, E').

4.4 An $O(\log(2 + b(V)))$-Approximation Algorithm

The main novelty in this result is that the ratio becomes smaller as $b(V)$ becomes smaller. In general, $b(V)$ may be very small as compared to $|V^- \cup V^+|$.

Lemma 3. *There exists a polynomial time algorithm that given an instance of* Generalized-P2P *computes an edge set $E' \subseteq E$ of cost $\leq 4\tau^*$, where τ^* denotes the optimal solution value, such that the number n' of connected components with non-zero charge in the graph (V, E') is at most $4b(V)$.*

Proof. Fix a parameter τ, which is an estimate for τ^*. Create an instance of Generalized-P2P with total charge zero by adding a new node s with charge $-b(V)$ and connecting s to each node in V^+ by an edge of cost $\tau/b(V)$. Then apply the 2-approximation algorithm for the case $b(V) = 0$. The new instance admits a solution of cost at most $\tau^* + b(V) \cdot (\tau/b(V)) = \tau^* + \tau$, by taking an optimal solution to the original instance with edges that connect s to at most $b(V)$ nodes in V^+. Thus the procedure returns an edge-set of cost at most $2(\tau^* + \tau)$. Consequently, if $\tau \geq \tau^*$ then the procedure returns an edge-set of cost at most 4τ, and the number of edges incident to s is at most $4\tau/(\tau/b(V)) = 4b(V)$. Using binary search, we find the minimum integer τ for which the procedure returns an edge-set E'' of cost 4τ. Then $c(E'') \leq 4\tau \leq 4\tau^*$ and the number of edges in E'' incident to s is at most $4b(V)$. Let E' be obtained from E'' by removing the edges incident to s. Then $c(E') \leq c(E) \leq 4\tau^*$, and the number n' of connected components in (V, E') with non-zero-charge is at most the degree of s w.r.t. E'', hence at most $4b(V)$, as claimed. □

The entire algorithm has two steps. At step 1 we compute an edge set E' as in the above lemma. Step 2 applies the $O(\log n'))$-approximation algorithm from the previous section to compute an augmenting edge-set $F \subseteq E \setminus E'$ such that $E' \cup F$ is a feasible solution. The solution cost is bounded by $c(E') + c(F) = O(\tau^*) + O(\log n') \cdot \tau^* = O(\log(2 + b(V))) \cdot \tau^*$.

References

1. Bartal, Y.: On approximating arbitrary metrices by tree metrics. In: STOC, pp. 161–168 (1998)
2. Calinescu, G., Zelikovsky, A.: The polymatroid steiner problems. J. Comb. Optim. 9(3), 281–294 (2005)
3. Carr, R., Fleischer, L., Leung, V., Phillips, C.: Strengthening integrality gaps for capacitated network design and covering problems. In: SODA, pp. 106–115 (2000)
4. Chakrabarty, D., Chekuri, C., Khanna, S., Korula, N.: Approximability of capacitated network design. In: Günlük, O., Woeginger, G.J. (eds.) IPCO 2011. LNCS, vol. 6655, pp. 78–91. Springer, Heidelberg (2011)
5. Chakrabarty, D., Krishnaswamy, R., Li, S., Narayanan, S.: Capacitated network design on undirected graphs. In: Raghavendra, P., Raskhodnikova, S., Jansen, K., Rolim, J.D.P. (eds.) APPROX/RANDOM 2013. LNCS, vol. 8096, pp. 71–80. Springer, Heidelberg (2013)
6. Chekuri, C., Even, G., Kortsarz, G.: A greedy approximation algorithm for the group Steiner problem. Discrete Appl. Math. 154(1), 15–34 (2006)
7. Even, G., Kortsarz, G., Slany, W.: On network design problems: fixed cost flows and the covering steiner problem. ACM Trans. Algorithms 1, 74–101 (2005)
8. Fakcharoenphol, J., Rao, S., Talwar, K.: A tight bound on approximating arbitrary metrics by tree metrics. J. Comput. System Sci. 69(3), 485–497 (2004)
9. Garg, N., Konjevod, G., Ravi, R.: A polylogarithmic approximation algorithm for the group Steiner tree problem. J. Algorithms 37(1), 66–84 (2000)
10. Gaspero, L.D., Gärtner, J., Kortsarz, G., Musliu, N., Schaerf, A., Slany, W.: The minimum shift design problem. Annals OR 155(1), 79–105 (2007)
11. Goemans, M.X., Goldberg, A.V., Plotkin, S.A., Shmoys, D.B., Tardos, E., Williamson, D.P.: Improved approximation algorithms for network design problems. In: SODA, pp. 223–232 (1994)
12. Goemans, M.X., Williamson, D.P.: A general approximation technique for constrained forest problems. SIAM J. Comput. 24(2), 296–317 (1995)
13. Hajiaghayi, M., Khandekar, R., Kortsarz, G., Nutov, Z.: Combinatorial algorithms for capacitated network design. CoRR, abs/1108.1176 (2011)
14. Halperin, E., Krauthgamer, R.: Polylogarithmic inapproximability. In: STOC, pp. 585–594 (2003)
15. Harrelson, C., Hildrum, K., Rao, S.: A polynomial-time tree decomposition to minimize congestion. In: SPAA, pp. 34–43 (2003)
16. Zosin, L., Khuller, S.: On directed steiner trees. In: SODA, pp. 59–63 (2002)

Approximating the Quadratic Knapsack Problem on Special Graph Classes*

Ulrich Pferschy and Joachim Schauer

University of Graz, Department of Statistics and Operations Research,
Universitaetsstr. 15, A-8010 Graz, Austria
{pferschy,joachim.schauer}@uni-graz.at

Abstract. We study the classical quadratic knapsack problem (QKP) on special graph classes. In this case the quadratic terms of the objective function are present only for certain pairs of knapsack items. These pairs are represented by the edges of a graph G=(V,E) whose vertices represent the knapsack items. We show that QKP permits an FPTAS on graphs of bounded treewidth and a PTAS on planar graphs and more generally on H-minor free graphs. The latter result is shown by adopting a technique of Demaine et al. (2005). We will also show strong NP-hardness of QKP on graphs that are 3-book embeddable, a natural graph class that is related to planar graphs. In addition we will argue that the problem might have a bad approximability behaviour on all graph classes containing large cliques (under certain complexity assumption used for showing hardness results for the densest k-subgraph problem).

Keywords: quadratic knapsack problem, densest k-subgraph, approximation algorithm, H-minor free.

1 Introduction

In the standard 0-1 knapsack problem (KP) we are given a set of n items each with an integer profit p_j and weight w_j. We look for a subset of items with maximum profit whose total weight does not exceed a given capacity c. If some pairs of items i, j are interdependent and generate a certain synergy, we gain an additional non-negative integer profit p_{ij} if both i and j are included in the solution set. This defines the *Quadratic Knapsack Problem* (QKP), see e.g. Kellerer et al. (2004, Sec.12) or Pisinger (2007).

To represent which items are in relation to each other, we introduce a graph $G = (V, E)$ with $|V| = n$ and $|E| = m$. Every vertex $v \in V$ corresponds uniquely to an item and an edge $(u, v) \in E$ indicates that the two corresponding items yield an additional profit, if they are both included in the solution. We will use vertex and item interchangeably. Using binary variables x_j with $x_j = 1$ iff item

* This research was funded by the Austrian Science Fund (FWF): P23829.

C. Kaklamanis and K. Pruhs (Eds.): WAOA 2013, LNCS 8447, pp. 61–72, 2014.

j is included in the solution, QKP can be defined as follows:

$$(QKP) \quad \max \sum_{i=1}^{n} p_i x_i + \sum_{(i,j)\in E} p_{ij} x_i x_j \tag{1}$$

$$\text{s.t.} \sum_{i=1}^{n} w_i x_i \leq c \tag{2}$$

$$x_i \in \{0,1\}, \quad i = 1,\ldots,n \tag{3}$$

QKP is a challenging strongly $\mathcal{NP}$-hard problem. Indeed, the notoriously hard maximum clique problem can be reduced to it: Given a graph G, we set $w_j = 1$ and $p_j = 0$ for all j and assign profits $p_{ij} = 1$ for all $(i,j) \in E$. Solving QKP with $c = k$, it follows that G contains a clique of size k iff the optimal solution of QKP is $\frac{k(k-1)}{2}$. Going through all values of k identifies the maximum clique.

There is a wide range of literature presenting exact solution algorithms for QKP. The currently best performing exact algorithm was given by Pisinger et al. (2007). On the other hand, very little is know about the approximation of QKP. It is an open question whether a constant approximation ratio for QKP is possible. Note that for the modified problem, where also negative profit values are allowed, Rader Jr. and Woeginger (2002) showed that no constant approximation ratio can be achieved in polynomial time (under $\mathcal{P}\neq\mathcal{NP}$).

A natural approach to fill this void is the consideration of QKP restricted to graphs with special properties. So far, the only result in this direction is an FPTAS for QKP on series parallel graphs based on dynamic programming given by Rader Jr. and Woeginger (2002). On the other hand, they show that QKP on so-called vertex series parallel graphs is strongly $\mathcal{NP}$-hard and thus does not permit an FPTAS (under $\mathcal{P}\neq\mathcal{NP}$).

1.1 Connections to the Densest k-Subgraph Problem

It is common in the literature that optimization problems with bad approximation behaviour on general graphs are studied on certain restricted graph classes, e.g. trees, graphs of bounded treewidth, planar graphs, chordal graphs and comparability graphs (amongst others). For QKP however this strategy might already fail on proper interval graphs and thus on chordal graphs as well as on many other basic graph classes due to the following connection to the densest k-subgraph problem.

The densest k-subgraph problem on a general graph $G = (V, E)$ asks for an induced subgraph $G' = (V', E')$ of G, where $|V'| = k$ and $|E'|$ is maximized.

From a hardness of approximation point of view DkS is a notorious problem. The best known hardness result under $\mathcal{P} \neq \mathcal{NP}$ is the strong $\mathcal{NP}$-hardness derived from the maximum clique problem (cf. Feige et al. (2001)). However under stronger complexity assumptions several inapproximability results were shown in the last years, mostly by using and developing fairly involved techniques: Feige (2002) ruled out the existence of a PTAS based on an assumption

dealing with average-case hardness of random 3-SAT. Khot (2006) ruled out the existence of a PTAS under the assumption that $\mathcal{NP}$ does not have randomized subexponential time algorithms. Alon et al. (2011) showed that under a hardness assumption on random k-AND formulas, there is no constant factor approximation for DkS. They also pointed out that Raghavendra et al. (2010) showed the same result under the Small Set Expansion Conjecture. Alon et al. (2011) even proved superconstant hardness of approximation results for DkS under an hardness assumptions dealing with the Hidden Clique problem.

From an approximation point of view any QKP instance on an n vertex graph G can be modelled by the complete graph K_n. Make G complete by adding all missing edges and assign a profit of 0 to them. Therefore any DkS instance I can be transformed into a QKP instance J on K_n in the following way: for each vertex in I introduce a vertex with weight 1 and profit 0 in J. For each edge in I add the same edge to J with profit 1. Introduce all missing edges with profit 0 in J in order to get a K_n and set the capacity $c = k$. Clearly this simple transformation is approximation preserving. Therefore we can immediately state the following result.

Theorem 1. *QKP is at least as hard to approximate on any graph class containing graphs with n vertices and cliques of size n^{ε} (for some constant ε) as the densest k-subgraph problem on general graphs.*

Hence finding a *good* approximation algorithm for QKP on one of following very prominent and basic graph classes would break one of the cited hardness assumption (depending on the quality of the approximation): proper interval graphs and all superclasses such as chordal graphs; graphs of bounded clique-width; comparability graphs.

1.2 Contributions of This Paper

We will make considerable progress in answering the question of approximability for QKP. Our contribution is threefold:

1. An FPTAS for QKP on graphs of bounded treewidth (which includes the class of series parallel graphs) is given in Section 2.
2. For graphs that do not include any fixed graph H as a minor, a PTAS is derived in Section 3. This includes planar graphs and also gives a PTAS for the densest k-subgraph problem on this class of graphs.

These two contributions are the first meaningful approximation results for QKP on special graph classes since Rader Jr. and Woeginger (2002).

3. In Section 4 we show that QKP on 3-book embeddable graphs is strongly $\mathcal{NP}$-hard.

The result on 3-book embeddable graphs is important since this is the first hardness result for QKP which has no connection to the maximum clique problem or closely related variants.

k-book embeddable graphs generalize the concept of planarity in a natural way. Planar graphs are very interesting for DkS, since the complexity status of DkS remains open on them (cf. Chen et al. (2011)) [1]. Note that the standard reduction for showing $\mathcal{NP}$-hardness of DkS, i.e. the reduction from the maximum clique problem, does not work on planar graphs.

Overbay (2007) proved that the embedding of a complete graph on n vertices needs a book with $\lceil \frac{n}{2} \rceil$ pages. Therefore, a k-book embeddable graph with constant k can have a maximum clique of size at most $2k$ and can be found by enumeration. But this means that the reduction from maximum clique can not work on k-book embeddable graphs with constant k.

The existence of an FPTAS for QKP on planar graphs and, more general, on any H-minor-free graph remains open. However such an FPTAS would optimally solve DkS on planar graphs and thus resolve this long standing open problem.

2 QKP on Graphs of Bounded Treewidth

In this section we discuss an FPTAS for graphs of bounded treewidth. It can be shown that, given a tree-decomposition of constant treewidth k, QKP can be solved by dynamic programming in $O(nP^2)$ time, where P is an upper bound on the optimal solution value. The details of this algorithm are related to Pferschy and Schauer (2009, Sec. 2) and are omitted due to space restrictions.

Theorem 2. *There is an Algorithm AlgQBT for QKP on graphs of treewidth k which has a running time of $O(2^k nP^2)$ and requires $O(2^k nP)$ space given a nice tree-decomposition with $O(n)$ vertices and treewidth k.* □

Next, we apply standard rounding arguments to this dynamic programming algorithm to derive an FPTAS (cf. Kellerer et al. (2004, Sec. 2.6) or Rader Jr. and Woeginger (2002)). The profits p_j and p_{ij} are replaced by scaled profits $\tilde{p}_j := \lfloor \frac{p_j}{K} \rfloor$ and $\tilde{p}_{ij} := \lfloor \frac{p_{ij}}{K} \rfloor$, for some K to be defined later. Then the problem is solved to optimality by AlgQBT with the scaled profit values yielding an optimal solution set $\tilde{X}$. Generally, this set will be different from the solution set X^* which optimizes the original instance with a solution value of z^*. The set $\tilde{X}$ is taken as an approximate solution with solution value z^A obtained for the original profits. Clearly $z^A \leq z^*$. Then one gets the following chain of inequalities, where $E(X) \subseteq E$ denotes the edges induced by a vertex set $X \subseteq V$.

$$z^A = \sum_{j \in \tilde{X}} p_j + \sum_{(i,j) \in E(\tilde{X})} p_{ij} \geq \sum_{j \in \tilde{X}} K \lfloor \frac{p_j}{K} \rfloor + \sum_{(i,j) \in E(\tilde{X})} K \lfloor \frac{p_{ij}}{K} \rfloor$$

$$\geq \sum_{j \in X^*} (p_j - K) + \sum_{(i,j) \in E(X^*)} (p_{ij} - K) = z^* - (|X^*| + |E(X^*)|)\, K$$

[1] Keil and Brecht (1991) proved that DkS is $\mathcal{NP}$-hard on planar graphs when the selected k vertex subgraph has to be connected.

To bound the relative error of the approximation algorithm by a given value ε we get the following inequality:

$$\frac{z^* - z^A}{z^*} \leq \frac{(|X^*| + |E(X^*)|)K}{z^*} \leq \frac{(n+m)K}{z^*} \leq \varepsilon$$

Define the largest coefficient of the objective function as $p_{\max} := \max\{\max\{p_j \mid j = 1, \ldots, n\}, \max\{p_{ij} \mid (i,j) \in E\}\}$. Clearly, $z^* \geq p_{\max}$, since each single item and each pair of items $(i,j) \in E$ is a feasible solution. Choosing $K := \varepsilon \frac{p_{\max}}{n+m}$ trivially satisfies the required condition.

Furthermore, in the running time and space complexity of AlgQBT the trivial upper bound P can be replaced for the scaled instance in the following way: the optimal solution value $\tilde{z}$ of the scaled problem instance is bounded by

$$\tilde{z} \leq (n+m)\,\tilde{p}_{\max} \leq (n+m)\frac{p_{\max}}{K} = \frac{(n+m)^2}{\varepsilon}\,. \tag{4}$$

Therefore, in the running time bound for the FPTAS derived from algorithm AlgQBT one can replace the factor P by $\frac{(n+m)^2}{\varepsilon}$ for the scaled instances.

The number of edges m can be bounded by the following well-known fact (see e.g. Rose (1974)).

Proposition 1. *For a graph $G = (V, E)$ of treewidth at most k, the number of edges can be bounded by $|E| \leq k\,|V| - \frac{1}{2}\,k(k+1)$.*

Thus, the upper bound on P given in (4) is in $O(\frac{(kn)^2}{\varepsilon})$ and the complexity of the FPTAS can be stated as follows.

Theorem 3. *There is an FPTAS for QKP on* graphs of treewidth k *requiring running time $O(2^k k^4 \frac{n^5}{\varepsilon^2})$ and $O(2^k k^2 \frac{n^3}{\varepsilon})$ space.*

3 PTAS for QKP on Certain Graph Classes

An important graph class for which QKP was not considered yet is the class of planar graphs. It is well known that this class can be defined by forbidding the $K_{3,3}$ and the K_5 as a minor. In this section we will show that planar graphs admit a PTAS for QKP. More generally, by applying a structural result of Demaine et al. (2005) we will show that a PTAS exists for QKP on all graph classes defined by a fixed excluded minor H. By Lovász (2006) a graph H is a minor of G if H can be obtained by successively applying the following three operations on G: deleting isolated vertices, deleting edges and contracting edges. Moreover a class of graphs $\mathcal{C}$ is called H-minor free if the graph H is not a minor of any of the graphs of $\mathcal{C}$. Demaine et al. (2005) showed the following decomposition theorem, from which we can obtain a PTAS for QKP under the same scenario.

Theorem 4. *Demaine et al. (2005, Theorem 3.1) For a fixed graph H, there is a constant c_H such that, for any integer $k \geq 2$ and for every H-minor-free* graph

G, the vertices of G can be partitioned into k sets such that any $k-1$ of the sets induce a graph of treewidth at most $c_H k$. Furthermore, such a partition can be found in polynomial time.

Theorem 5. *There is a PTAS for QKP on* H-minor-free graphs *for any fixed graph* H.

Proof. We first compute in polynomial time a decomposition of V into k (which will be determined later) disjoint subsets $V_1, \ldots, V_k$ as given by Theorem 4. Each vertex set V_j induces an edge set E_j. For each subset $\ell \in \{1, \ldots, k\}$ we define by $\bar{E}_\ell$ the set of edges between V_ℓ and $V \setminus V_\ell$, i.e. the edges joining V_ℓ with other subsets V_j, $j \neq \ell$. Removing all edges in $\bar{E}_\ell$ from the graph we obtain a graph G'_ℓ consisting of the graph induced by the $k-1$ remaining sets of vertices of the partition and a disconnected part induced by V_ℓ. According to Theorem 4 both of these two parts have bounded treewidth and so has their union G'_ℓ.

For the optimal variable values x_i^* the optimal solution value of QKP on G can be written as

$$z^* = \sum_{i \in V} p_i x_i^* + \sum_{\ell=1}^{k} \sum_{(i,j) \in E_\ell} p_{ij} x_i^* x_j^* + \frac{1}{2} \sum_{\ell=1}^{k} \sum_{(i,j) \in \bar{E}_\ell} p_{ij} x_i^* x_j^*, \tag{5}$$

where the second term sums up all edges within one subset V_ℓ while the third term sums over all edges between V_ℓ and all other subsets. Since G is an undirected graph, every edge appears twice in the latter expression which necessitates the factor $\frac{1}{2}$.

Choosing

$$\ell^* := \arg\min_{\ell=1}^{k} \left\{ \sum_{(i,j) \in \bar{E}_\ell} p_{ij} x_i^* x_j^* \right\} \tag{6}$$

we obtain the set V_{ℓ^*} with the smallest profit contribution of edges between V_{ℓ^*} and all other subsets. By the usual averaging argument we have

$$\sum_{(i,j) \in \bar{E}_{\ell^*}} p_{ij} x_i^* x_j^* \leq \frac{1}{k} \sum_{\ell=1}^{k} \sum_{(i,j) \in \bar{E}_\ell} p_{ij} x_i^* x_j^*. \tag{7}$$

Removing all edges in $\bar{E}_{\ell^*}$ we obtain a graph G'_{ℓ^*} of bounded treewidth (see above). The optimal solution of QKP on this reduced graph G'_{ℓ^*} yields an optimal solution value z_{ℓ^*} which can be bounded by (7)

$$z_{\ell^*} \geq z^* - \sum_{(i,j) \in \bar{E}_{\ell^*}} p_{ij} x_i^* x_j^* \geq z^* - \frac{1}{k} \sum_{\ell=1}^{k} \sum_{(i,j) \in \bar{E}_\ell} p_{ij} x_i^* x_j^* \,.$$

(x_i^* still denotes the optimal solution on the full graph.) Bounding generously with only the third term of (5) we get

$$z_{\ell^*} \geq z^* - \frac{1}{k} \cdot 2\, z^* = \left(1 - \frac{2}{k}\right) z^*.$$

Since we cannot find the optimal solution of QKP on G'_{ℓ^*} in polynomial time, we have to make use of the FPTAS from Theorem 3 to compute a δ-approximation z^A of z_{ℓ^*}. This yields

$$z^A \geq (1-\delta) z_{\ell^*} \geq (1-\delta)(1-\frac{2}{k})\, z^*.$$

For $\delta := \varepsilon^2$ and $k := \lceil \frac{2}{\varepsilon} \rceil + 2$ we get $z^A \geq (1-\varepsilon) z^*$ as required for a PTAS.

Of course, we cannot find ℓ^* without knowing the optimal solution. Instead, we go through all k possible choices of ℓ^* and run the PTAS on each candidate graph G'_ℓ for $\ell = 1, \ldots, k$. Taking the best of these k approximate solution values guarantees a solution value at least as large as z^A. □

Corollary 1. *There is a PTAS for DkS on* H-minor-free graphs *for any fixed graph* H.

Planar graphs are a subclass of $K_{3,3}$-minor free (resp. K_5-minor free) graphs. Thus, Theorem 4, Corollary 1 and therefore our PTAS apply to planar graphs as well.

4 Hardness for 3-Book Embeddings

In this section, we show that QKP is strongly $\mathcal{NP}$-hard on graphs that are 3-book embeddable. This result is interesting since a k-book embedding generalizes the concept of planar graphs (however not characterized by forbidden minors). Furthermore the $\mathcal{NP}$-hardness proofs for densest k subgraph and QKP on special graph classes presented in the literature are based on the maximum clique problem or on closely related variants (cf. Rader Jr. and Woeginger (2002), Corneil and Perl (1984)), whereas our reduction follows a completely different approach.

A k-book consists of k half planes, called pages, whose common intersection is a line, called the spine. A k-book embedding of a graph $G = (V, E)$ is an embedding of G into a k-book such that all vertices are arranged on the spine and each edge $e = (u, v)$ is embedded into a unique page where the intersection of e with the spine consists only of u and v (cf. Chung et al. (1987)). Clearly every graph has a $|E|$-book embedding. For planar graphs it was shown by Yannakakis (1986) that a book of four pages is sufficient for an embedding.

In our hardness proof we will reduce a special variant of $3SAT$ to QKP on 3-book embeddable graphs: Moore and Robson (2001) showed that *Cubic Planar Monotone 1-in-3 SAT* (*CPM-1-3-SAT*) is strongly $\mathcal{NP}$-hard. In this problem n clauses and n variables are given, where each clause contains exactly three variables and each variable occurs in exactly three clauses. Monotone means that this problem does not contain negated literals. Moreover the graph G_{SAT} that represents each variable and clause by a unique vertex and introduces edges between them whenever a variable appears in a clause is planar. A feasible solution to *CPM-1-3-SAT* consists of exactly $\frac{n}{3}$ variables set *TRUE* such that each clause contains exactly one variable set *TRUE*.

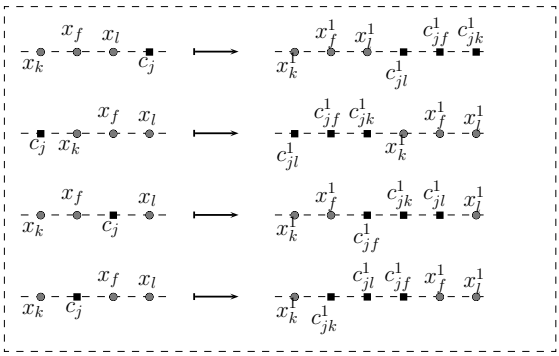

Fig. 1. Handling of position patterns of variables and clauses on the spine

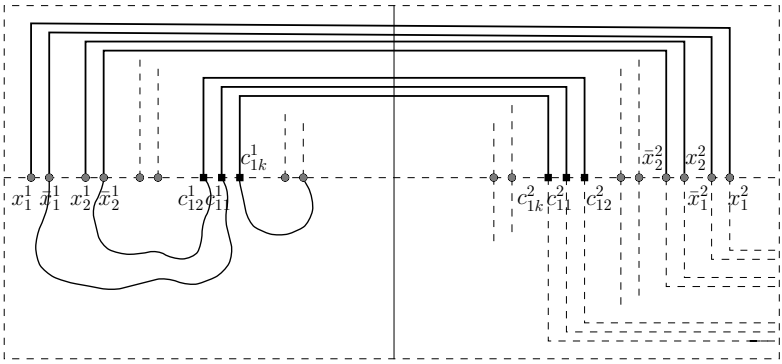

Fig. 2. Duplicating vertices from layer 1 to layer 2

Let I be an instance of *CPM-1-3-SAT* and G_{SAT} its corresponding graph. Kainen and Overbay (2003) showed that any planar graph with girth (shortest cycle) > 3 is a subgraph of a planar Hamiltonian graph. Moreover it is well known that a graph is 2-book embeddable if and only if it is the subgraph of a planar Hamiltonian graph (cf. Bernhart and Kainen (1979)). Therefore G_{SAT}, which is bipartite and thus has no cycle of length 3, has a 2-book embedding.

In the following proof we will transform this 2-book embedding of the *CPM-1-3-SAT* instance I into a QKP instance J defined on a graph G_J that is 3-book embeddable. Note that the transformation will not preserve planarity.

Let G_{SAT} be represented by a 2-book embedding. We first represent the vertices x_i of G_{SAT} by vertices x_i^1 and $\bar{x}_i^1$ in G_J and the c_j vertices of G_{SAT} by three vertices c_{jk}^1, where the index k denotes that variable x_k occurs in clause c_j. We call them vertices of layer 1 and denote the layer by a superscript index. Figure 4 describes how the vertices c_{ik}^1 are arranged with respect to the four

possible position patterns of the variables x_k, x_f and x_l and the clause c_j on the spine of G_{SAT}.

Next we duplicate (in fact mirror) these vertices $n+4$ times. This is done by introducing vertices x_i^ℓ, $\bar{x}_i^\ell$ and c_{jk}^ℓ, $\ell = 2, \ldots, n+5$, on $n+4$ new layers. Furthermore, the following edges are introduced: $(x_i^\ell, x_i^{\ell+1})$, $(\bar{x}_i^\ell, \bar{x}_i^{\ell+1})$, and $(c_{jk}^\ell, c_{jk}^{\ell+1})$ (cf. Figure 4, rectangular edges). Note that after this duplication procedure on layers 1 and $n+5$ two book pages remain unused, whereas on all other layers only one book page remains unused. On one remaining book page of layer 1 we connect all vertices $\bar{x}_i^1$ with the corresponding vertex c_{ji}^1, whenever variable x_i occurs in clause c_j (see Figure 4, curvy edges). On layer $5+i$ we connect x_i^{5+i} to all $\bar{x}_j^{5+i}$ with $j \neq i$.

Table 1. The weights and profits of G_J, all k are from 1 to $5+n$

vertices / edges	profits	weights
x_i^k	n^{24}	n^{24}
$\bar{x}_i^k$	n^{18}	n^{18}
c_{ji}^k	n^{12}	n^{12}
(x_i^k, x_i^{k+1})	n^{21}	
$(\bar{x}_i^k, \bar{x}_i^{k+1})$	n^{15}	
(c_{ij}^k, c_{ij}^{k+1})	n^9	
$(x_i^{5+i}, \bar{x}_j^{5+i})$, $j \neq i$	n^6	
$(\bar{x}_i^1, c_{ji}^1)$	n^3	
(x_i^ℓ, c_{jf}^ℓ), (x_i^ℓ, c_{jl}^ℓ), $\ell = 2, \ldots, 5$	1	

Next we represent the edges of page 1 of G_{SAT} on one remaining page of layer 2 and 3 of G_J. The construction for the case that all edges incident to c_j in G_{SAT} are embedded into one book page is shown in Figure 4, where again all four possible position patterns of the variables x_k, x_f and x_l and the clause c_j on the spine of G_{SAT} are considered. The same construction works for page 2 and layers 4 and 5. Note that the cases where the edges incident to a clause c_j are embedded into both pages of G_{SAT} can be handled by a similar construction: the only difference is that all missing edges on page 1 (resp. page 2) are ignored when constructing G_J. All profits and weights for the vertices and edges of G_J are listed in Table 1.

The proofs of the following Lemmata 1-5 are omitted.

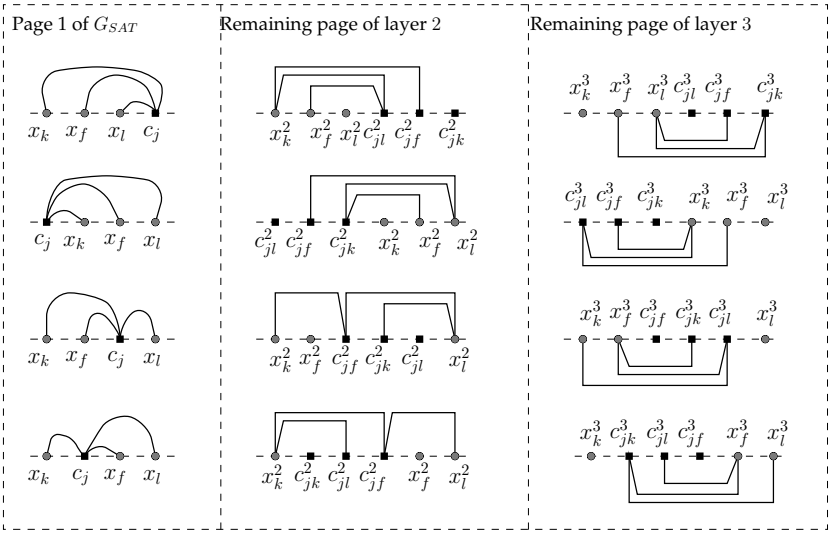

Fig. 3. Representing adjacent vertices of G_{SAT}

Lemma 1. *Let G be a bipartite graph with vertex sets $(A = \{a_1, \ldots, a_n\} \cup B = \{b_1, \ldots, b_n\})$. For every a_i there is an edge to all b_j with $j \neq i$. If a subgraph G' induced by vertex sets $A' \subseteq A$ and $B' \subseteq B$ has $|A'| \cdot |B'|$ edges, then a_i and b_i can not be contained in G' simultaneously for any i.*

Lemma 2. *Let G be a graph that is composed by l disjoint paths of length $k-1$ and $i < l$. Then any subgraph $\bar{G}$ on $k \cdot i$ vertices with the maximum number of induced edges consists of i disjoint paths of length $k-1$.*

In the proof of Theorem 6 will show that an instance I of *CPM-1-3-SAT* has a feasible truth assignment if and only if the corresponding QKP instance J with graph G_J and has a solution set S_J fulfilling the following capacity bound c_J with total profit value at least p_J:

$$c_J := (5+n)\left(\frac{n}{3}n^{24} + \frac{2n}{3}n^{18} + 2nn^{12}\right)$$

$$p_J := (5+n)\left(\frac{n}{3}n^{24} + \frac{2n}{3}n^{18} + 2nn^{12}\right) +$$
$$(4+n)\left(\frac{n}{3}n^{21} + +\frac{2n}{3}n^{15} + 2nn^{9}\right) + \frac{n}{3}\cdot\frac{2n}{3}\cdot n^6 + 2nn^3 + 2n$$

Note that the total number of vertices and edges in G_J is of order n^2. Therefore, any subset of vertices and edges with profits n^k can never reach a total profit of n^{k+3}. This general property will be exploited throughout our construction.

The next lemma determines the number of vertices of every type included in a feasible solution S_J. It also guarantees that each of the $n+5$ layers of G_J has the same structure of included vertices.

Lemma 3. *For any feasible solution S_J of a QKP instance J with profit greater or equal to p_J and weight at most c_J the following structure holds for each layer $k = 1, \ldots, n+5$:*

1. *S_J includes exactly $\frac{n}{3}$ vertices x_i^k from each layer k.*
2. *S_J includes exactly $2\frac{n}{3}$ vertices $\bar{x}_i^k$ from each layer k.*
3. *S_J includes exactly $2n$ vertices c_{ji}^k from each layer k.*

If S_J contains some vertex x_i^k, $\bar{x}_i^k$ or c_{ji}^k of a layer k, then S_j contains x_i^ℓ, $\bar{x}_i^\ell$ or c_{ji}^ℓ for all layers $\ell = 1, \ldots, n+5$.

Lemma 4. *For any feasible solution S_J of a QKP instance J with profit greater or equal to p_J and weight at most c_J the following holds for each layer $k = 1, \ldots, 5+n$:*

$$x_i^k \in S_J \iff \bar{x}_i^k \notin S_J$$

Lemma 5. *For any feasible solution S_J of a QKP instance J with profit greater or equal to p_J and weight at most c_J the following holds for each layer $k = 1, \ldots, 5+n$: For every vertex c_{ji}^k in S_J also $\bar{x}_i^k$ has to be in S_J, whenever variable x_i occurs in clause c_j.*

Theorem 6. *QKP defined on 3-book embeddable graphs is strongly $\mathcal{NP}$-hard.*

Proof. Let I be an instance of *CPM-1-3-SAT* and J the corresponding QKP instance as defined above. We will show that I has a feasible truth assignment if and only if instance J has a feasible solution with objective value at least p_J and weight at most c_J.

In Lemma 3 - 5 we identified the structure of a feasible QKP solution implied by the profit and weight bounds. In the remainder of the proof we can concentrate on edges with profit 1 and ignore all other edges.

" $\Longleftarrow$ ": The profit bound p_J implies that any solution S_J to J contains $2n$ edges of profit 1. By Lemma 3, we know that the same $\frac{n}{3}$ vertices x_i^k (w.r.t. i) are chosen from each layer k.

By the construction of G_J (cf. Figure 4) we know that for each i there are exactly 6 neighbours c_{jf}^k connected to x_i^k over all layers k (in fact these are all found in layers $2, \ldots, 5$) with an edge of profit 1. Hence all these neighbours have to be included in S_J to get a total profit of $2n$.

We can construct a solution to I as follows: whenever x_i^1 is in S_J, we set x_i to *TRUE*. We get that I contains exactly $\frac{n}{3}$ variables set to *TRUE* and that I is a feasible instance: assume that there is a clause c_j that has more than one variable set *TRUE* (denote them x_i and x_f). This means that for some layer $k \in \{2, \ldots, 5\}$ also c_{jf}^k is in S_J, since x_i^k is connected to c_{jf}^k. It follows from Lemma 5 that now also $\bar{x}_f^k$ is included in S_J, in contradiction to Lemma 4.

If there is a clause with no variable set *TRUE* we get by the pigeon-hole principle that one clause must contain more than one variable set *TRUE*, again leading to a contradiction.

" $\Longrightarrow$ ": Let I be a feasible. If x_i is *TRUE*, include x_i^k in S_J for all layers k, otherwise include $\bar{x}_i^k$ and c_{ji}^k in S_J. It is easy to check that S_J is a feasible QKP solution fulfilling the weight and profit bounds c_J and p_J. □

References

Alon, N., Arora, S., Manokaran, R., Moshkovitz, D., Weinstein, O.: Inapproximabilty of Densest k-Subgraph from Average Case Hardness. Technical report (2011)

Bernhart, F., Kainen, P.C.: The book thickness of a graph. Journal of Combinatorial Theory, Series B 27(3), 320–331 (1979)

Chen, D.Z., Fleischer, R., Li, J.: Densest k-subgraph approximation on intersection graphs. In: Jansen, K., Solis-Oba, R. (eds.) WAOA 2010. LNCS, vol. 6534, pp. 83–93. Springer, Heidelberg (2011)

Chung, F.R.K., Leighton, F.T., Rosenberg, A.L.: Embedding graphs in books: A layout problem with applications to vlsi design. SIAM Journal on Algebraic Discrete Methods 8(1), 33–58 (1987)

Corneil, D.G., Perl, Y.: Clustering and domination in perfect graphs. Discrete Applied Mathematics 9(1), 27–39 (1984)

Demaine, E.D., Hajiaghayi, M.T., Kawarabayashi, K.: Algorithmic graph minor theory: Decomposition, approximation, and coloring. In: 46th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2005, pp. 637–646 (2005)

Feige, U.: Relations between average case complexity and approximation complexity. In: STOC, pp. 534–543. ACM (2002)

Feige, U., Peleg, D., Kortsarz, G.: The Dense k-Subgraph Problem. Algorithmica 29(3), 410–421 (2001)

Kainen, P.C., Overbay, S.: Book embeddings of graphs and a theorem of whitney. Technical report (2003)

Keil, J.M., Brecht, T.B.: The complexity of clustering in planar graphs. J. Combinatorial Mathematics and Combinatorial Computing 9, 155–159 (1991)

Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer (2004)

Khot, S.: Ruling Out PTAS for Graph Min-Bisection, Dense k-Subgraph, and Bipartite Clique. SIAM J. Comput. 36(4), 1025–1071 (2006)

Lovász, L.: Graph minor theory. Bulletin of the American Mathematical Society 43(1), 75–86 (2006)

Moore, C., Robson, J.M.: Hard tiling problems with simple tiles. Dscrete & Computational Geometry 26(4), 573–590 (2001)

Overbay, S.: Graphs with small book thickness. Missouri Journal of Mathematical Sciences 19(2), 121–130 (2007)

Pferschy, U., Schauer, J.: The Knapsack Problem with Conflict Graphs. Journal of Graph Algorithms and Applications 13(2), 233–249 (2009)

Pisinger, D.: The quadratic knapsack problem - a survey. Discrete Applied Mathematics 155, 623–648 (2007)

Pisinger, D., Rasmussen, A.B., Sandvik, R.: Solution of large quadratic knapsack problems through aggressive reduction. INFORMS Journal on Computing 19, 280–290 (2007)

Rader Jr., D.J., Woeginger, G.J.: The quadratic 0-1 knapsack problem with series-parallel support. Operations Research Letters 30, 159–166 (2002)

Raghavendra, P., Steurer, D., Tulsiani, M.: Reductions Between Expansion Problems. Electronic Colloquium on Computational Complexity (ECCC) 17, 172 (2010)

Rose, D.J.: On simple characterizations of k-trees. Discrete Mathematics 7, 317–322 (1974)

Yannakakis, M.: Four pages are necessary and sufficient for planar graphs. In: Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing, STOC 1986, New York, NY, USA, pp. 104–108. ACM (1986)

Approximating the Sparsest k-Subgraph in Chordal Graphs*

Rémi Watrigant, Marin Bougeret, and Rodolphe Giroudeau

LIRMM, Université Montpellier 2, France

Abstract. Given a simple undirected graph $G = (V, E)$ and an integer $k < |V|$, the Sparsest k-Subgraph problem asks for a set of k vertices which induces the minimum number of edges. As a generalization of the classical independent set problem, Sparsest k-Subgraph is $\mathcal{NP}$-hard and even not approximable unless $\mathcal{P} = \mathcal{NP}$ in general graphs. Thus, we investigate Sparsest k-Subgraph in graph classes where independent set is polynomial-time solvable, such as subclasses of perfect graphs. Our two main results are the $\mathcal{NP}$-hardness of Sparsest k-Subgraph on chordal graphs, and a greedy 2-approximation algorithm. Finally, we also show how to derive a $PTAS$ for Sparsest k-Subgraph on proper interval graphs.

1 Introduction

1.1 Related Problems

Given a simple undirected graph $G = (V, E)$ and an integer $k < |V|$, the Sparsest k-Subgraph problem asks for a set of k vertices which induces[1] the minimum number of edges. It appears that this problem falls into the family of *cardinality constrained optimization problems*, introduced by [7], and is more precisely a generalization of the so-called independent set problem. This observation immediately implies that Sparsest k-Subgraph is $\mathcal{NP}$-hard and even not approximable in general graphs unless $\mathcal{P} = \mathcal{NP}$, as the optimal value is 0 whenever there is an independent set of size k. Thus, we only consider Sparsest k-Subgraph in graph classes where independent set is polynomial-time solvable. Let us first present some related problems, and then discuss their relation to Sparsest k-Subgraph. Actually, the following three problems can all be considered as *cardinality constrained* versions of other well-known combinatorial optimization problems, namely vertex cover and max clique, both very close to independent set.

In the maximum Quasi-Independent Set (QIS) problem [4] (also called k-edge-in in [10]), we are given a graph G and an integer C, and we ask for a set of vertices S of maximum size inducing at most C edges.

* This work has been funded by grant ANR **2010** BLAN **021902**.

[1] An edge $\{u, v\} \in E$ is said to be induced (resp. covered) by a set S if $u \in S$ and (resp. or) $v \in S$.

C. Kaklamanis and K. Pruhs (Eds.): WAOA 2013, LNCS 8447, pp. 73–84, 2014.

In the MINIMUM PARTIAL VERTEX COVER (PVC) problem [11], we are given a graph G and an integer C, and we ask for a set of vertices S of minimum size which covers[1] at least C edges.

Finally, we can mention the corresponding maximization problem of SPARSEST k-SUBGRAPH, namely DENSEST k-SUBGRAPH, which consists in finding a subset S of exactly k vertices inducing the maximum number of edges.

The decision versions of QIS, PVC, and SPARSEST k-SUBGRAPH are polynomially equivalent. Indeed, QIS could be considered as a dual version of SPARSEST k-SUBGRAPH where the budget (the number of edges in the solution of SPARSEST k-SUBGRAPH) is fixed. PVC and SPARSEST k-SUBGRAPH are also polynomially equivalent as for any S, the number of edges induced by S plus the number of edges covered by $V \backslash S$ equals $|E|$. Then, exact results for DENSEST k-SUBGRAPH on a graph class implies the same result for SPARSEST k-SUBGRAPH on the corresponding complementary class, and conversely. Unlike exact results, approximation algorithms do not transfer directly between any of these problems.

Considering these remarks and previous studies on these problems, Figure 1 presents known results and open problems about SPARSEST k-SUBGRAPH (SkS), DENSEST k-SUBGRAPH (DkS) and PVC in restricted graph classes. In each cell, the first line generally describes the general complexity ($\mathcal{NP}$-hard *versus* Polynomial), whereas other lines present some results concerning approximation or parameterized complexity. We recall that PROPER INTERVAL GRAPHS $\subset$ INTERVAL GRAPHS $\subset$ CHORDAL GRAPHS $\subset$ PERFECT GRAPHS, as well as SPLIT GRAPHS $\subset$ CHORDAL GRAPHS and BIPARTITE GRAPHS, COGRAPHS $\subset$ PERFECT GRAPHS.

Graphs classes	DkS	SkS	PVC
general	$\mathcal{NP}$-h (*c.f.* MAX CLIQUE) $n^{\frac{1}{4}+\epsilon}$-approx. [3]	$\mathcal{NP}$-h, not approx. (*c.f.* INDEP. SET)	$\mathcal{NP}$-h, $W[1]$-h [11] 2-approx.[6] exact $O^*(1, 4^C)$ [13]
chordal	$\mathcal{NP}$-h [8] 3-approx [14]	$\mathcal{NP}$-h [this paper] 2-approx [this paper]	$\mathcal{NP}$-h (*c.f.* SkS)
interval	OPEN, PTAS [15]	OPEN	OPEN
proper interval	OPEN	OPEN $PTAS$ [this paper]	OPEN
bipartite	$\mathcal{NP}$-h [8]	$\mathcal{NP}$-h (*c.f.* PVC)	$\mathcal{NP}$-h [12]
line	OPEN	$\mathcal{P}$ (*c.f.* PVC)	$\mathcal{P}$ [1]
planar	OPEN	$\mathcal{NP}$-h (*c.f.* INDEP. SET)	$\mathcal{NP}$-h (*c.f.* SkS)
cographs, split, bounded treewidth	$\mathcal{P}$ [8]	$\mathcal{P}$ [5]	$\mathcal{P}$ (*c.f.* SkS)
max. degree 2	$\mathcal{P}$ [8]	$\mathcal{P}$ [5]	$\mathcal{P}$ (*c.f.* SkS)
max. degree 3	$\mathcal{NP}$-h [8]	$\mathcal{NP}$-h (*c.f.* INDEP. SET)	$\mathcal{NP}$-h (*c.f.* SkS)

Fig. 1. Main results for DkS, SkS and PVC in some restricted graph classes

1.2 Contributions and Organization of the Paper

According to Figure 1, Densest k-Subgraph was already known to be $\mathcal{NP}$-hard on chordal graphs. However, as the complement of a chordal graph (and in particular the graph used in the reduction of [8]) is a perfect graph and not necessarily a chordal graph, this result only provides the $\mathcal{NP}$-hardness of Sparsest k-Subgraph on perfect graphs.

Thus, our motivation is to study Sparsest k-Subgraph on a classical subclass of perfect graphs. The main results of the paper are the $\mathcal{NP}$-hardness of Sparsest k-Subgraph in chordal graphs (Section 3), and a tight 2-approximation greedy algorithm (Section 2). Finally, we show in Section 4 how the arguments of [15] (which provides a PTAS for DkS in interval graphs) can be adapted to SkS in proper interval graphs. Notice that our $\mathcal{NP}$-hardness result implies the $\mathcal{NP}$-hardness of PVC in chordal graphs, which supplements the recent $\mathcal{NP}$-hardness of [2,12] for PVC in bipartite graphs. Due to space constraints, the proof of Lemma 6 for the NP-hardness in chordal graphs as well as the $PTAS$ in proper interval graphs were omitted. They can be found in the long version of this paper available in [17].

1.3 Notations and Definitions

All graphs studied in this paper are simple and without loop. For the remaining, $G = (V, E)$ will denote the input graph of the problem, and we define as usual $n = |V|$, $m = |E|$.

Chordal graphs are graphs with no induced cycle of length four or more. They may also be defined equivalently in terms of simplicial elimination order [9]. A vertex $v \in V$ is called simplicial if its neighbourhood $N(v)$ is a clique. A *simplicial elimination order* of G is an ordering $v_1, ..., v_n$ of V such that for all $i \in \{1, ..., n\}$, v_i is simplicial in $G[v_i, ..., v_n]$. It is known that a graph G is chordal if and only if it admits a simplicial elimination order. In addition, such an ordering can be found in polynomial time for a chordal graph. Hence, we will suppose in the following that $V = \{v_1, ..., v_n\}$ is sorted according to a simplicial elimination order of G. Similarly, for a subset of vertices $S \subseteq V$, we will denote by $\min(S)$ (resp $\max(S)$) the first (resp. last) vertex of S in the simplicial elimination order chosen for the graph. Finally, since we have a total ordering on the vertices, we will use the notations $x < y$ and $x > y$ for two vertices $x, y \in V$.

Given two sets $S_1, S_2 \subseteq V$, we denote by $cost(S_1)$ the number of edges in the graph induced by vertices of S_1, and $cost(S_1, S_2) = |\{\{v, v'\} \in E, v \in S_1, v' \in S_2\}|$. Given a set $S \subseteq V$ and $x \in V$, we denote by $d(x, S)$ the degree of x in S.

Finally, we refer the reader to the classical literature for definitions of approximation algorithms.

2 2-Approximation in Chordal Graphs

2.1 Idea of the Algorithm

We now present a tight 2-approximation algorithm for chordal graphs. First, notice that any approximation algorithm for Sparsest k-Subgraph must output

a maximum independent set of size k if such a set exists, as in this case the optimal value is 0. Hence, a natural idea for computing a solution to Sparsest k-Subgraph is to choose first a maximum independent set S (this can be done in polynomial time in chordal graphs). If k vertices or more were picked, then the algorithm stops. Otherwise, several ideas may come up.

A first idea would be to remove this independent set from the graph, and iteratively pick another one, until we get k vertices. This approach is the same as the 3-approximation of [14] for Densest k-Subgraph in chordal graphs (computing maximum cliques instead of maximum independent sets). Unfortunately, as shown in Figure 2, this algorithm has an unbounded approximation ratio for Sparsest k-Subgraph even in interval graphs (a subclass of chordal graphs). It still provides a 2-approximation in proper interval graphs [16].

Thus, after picking the first maximum independent set, our idea is to assign weights on remaining vertices according to the size of their neighbourhood in the constructed solution. At each step, the algorithm just picks an independent set (called a *layer*) among the vertices of minimum weight, and then updates the weights of remaining vertices. The algorithm is more formally defined in the next subsection. In the next paragraph, we describe the key idea of the analysis.

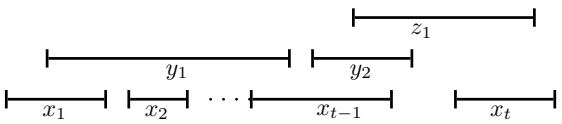

Fig. 2. In this case picking successive independent sets gives an unbounded ratio: for $k = t + 2$ the algorithm will take intervals $\{x_1, ..., x_t, y_1, y_2\}$ of cost t whereas the solution $\{x_1, ..., x_t, y_2, z_1\}$ is of cost 4

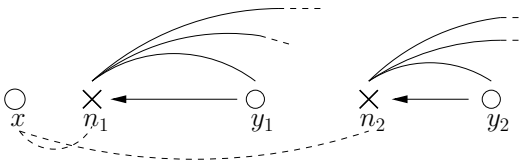

Fig. 3. Idea of the restructuration of a solution S^*. Circles denote vertices of S^*, and crosses denote vertex of L (chosen by the algorithm). When replacing y_1 by n_1 and y_2 by n_2, the degree of x can only increase by one. Indeed, x cannot be connected to n_1 and n_2, as L is an independent set.

The idea of the proof of the 2-approximation ratio is to restructure an optimal solution S^* until we get S (the output of the algorithm), while bounding the cost variation during the restructurations. Let us show what makes this restructuration work for the first layer. Let L be the independent set chosen by the

algorithm at the first step. Roughly speaking, for each $n_j \in L$ which is not in S^*, we restructure S^* by removing y_j, the "first" neighbour of n_j which is after n_j and in S^*, and adding n_j instead. As depicted in Figure 3, we see that the degree of a vertex $x \in S^*$ ($x \notin L$) will increase by at most 1. Concerning future layers, the analysis will become more complex, as we will have to take weights into account.

2.2 Algorithm and Analysis

Presentation of the Algorithm. As described previously, Algorithm 1 picks successively an independent set among the vertices of lower weights. It also updates the weights according to the picked vertices. For technical reasons, the weights are not exactly equal to their degree in the constructed solution. Indeed, when restructuring an optimal solution to match L_i we will see that the degree of almost all "surviving" vertices in the optimal solution increases by at most 1 (this is why we add a "bonus" of -1 in the updated weight Line 13), and even sometimes cannot increase (this is why there is no "bonus" Line 11). This modification will allow us to show that at the end of the algorithm, the value W returned by the algorithm is a lower bound of the optimal value (Lemma 3). We will then show that the real value of the returned solution $cost(S)$ is less than two times W (Lemma 4), and thus is a 2-approximation.

Algorithm 1. A 2-approximation for SPARSEST k-SUBGRAPH in chordal graphs

1: $S \leftarrow \emptyset$, $W \leftarrow 0$, $i \leftarrow 0$, $w_0(x) = 0\ \forall x \in V$
2: **while** $|S| \leq k$ **do**
3: $L_i \leftarrow$ a maximum independent set of the graph induced by $\{x \in V \backslash (L_0 \cup ... \cup L_{i-1}) : w_i(x) = i\}$
4: $S \leftarrow S \cup L_i$ // or the $(k - |S \cup L_i|)$ leftmost vertices of L_i if $|S \cup L_i| > k$
5: $W \leftarrow W + i|L_i|$ // we update the cost computed by the algorithm
6: **for** $x \in V$ **do**
7: **if** $x \in (L_0 \cup ... \cup L_i)$ **then**
8: $w_{i+1}(x) = w_i(x)$
9: **else**
10: **if** $d(x, L_i) = 0$ OR $(d(x, L_i) = 1$ AND $w_i(x) = i)$ **then**
11: $w_{i+1}(x) = w_i(x) + d(x, L_i)$
12: **else**
13: $w_{i+1}(x) = w_i(x) + d(x, L_i) - 1$
14: $i \leftarrow i + 1$
15: $t \leftarrow i - 1$ // L_t is the last "layer" of the algorithm
16: **return** (S, W)

Remark 1. The maximum independent set of line 3 is greedily constructed as follows: pick the first vertex of the simplicial elimination order in the independent set, delete its neighbourhood, and repeat the operation until the graph becomes empty.

Even if we sometimes add -1 when updating the weights, we can observe that for a fixed $x \in V$, its successive weights can be lower bounded as follows:

Lemma 1. *For all $i \in \{0, ..., t\}$, $\forall x \in V \setminus (L_0 \cup \cdots \cup L_i)$, $w_{i+1}(x) \geq i + 1$.*

Proof. Let i and x be as in the statement. Suppose by induction that $w_i(x) \geq i$ (notice that $w_0(x) = 0$). If $w_i(x) \geq i + 1$ then the results follows. Otherwise $w_i(x) = i$, and by construction of the algorithm (Line 11), if $d(x, L_i) \geq 1$, then $w_{i+1}(x) \geq i + 1$. Finally, if $d(x, L_i) = 0$, then x must belong to L_i which contradicts the definition of x. □

Restructuration of Solutions. Let S^* be an optimal solution for the Sparsest k-Subgraph problem in chordal graphs. We will now show that we can modify this solution in order to obtain the output of the algorithm, while bounding the cost variation.

Let us define by induction a sequence $(S_i^*)_{i=-1,0,...,t}$ with $S_{-1}^* = S^*$ and $S_t^* = S$ (the solution returned by the algorithm), such that $S_i^* \subseteq V$ and $|S_i^*| = k$ for all $i = -1...t$. We also assure that $(L_0 \cup ... \cup L_i) \subseteq S_i^*$ for all $i = 0...t$. To that end, given $i \in \{0, ..., t\}$, we show how to restructure the set S_{i-1}^* into a new set S_i^*. Let us first introduce some notations.

We partition the set L_i (defined in the algorithm) into two sets of vertices, whether they belong to S_{i-1}^* or not: $L_i = M_i \cup N_i$, with $M_i = L_i \cap S_{i-1}^*$ (and thus $N_i = L_i \setminus S_{i-1}^*$).

The restructuration consists in adding all vertices of N_i to S_{i-1}^*, and removing a carefully chosen (see Definition 1) subset $D_i \subseteq S_{i-1}^* \setminus (L_0 \cup ... \cup L_i)$ (with $|D_i| = |N_i|$). Then, we will define $S_i^* = (S_{i-1}^* \setminus D_i) \cup N_i$, $R_i = S_{i-1}^* \setminus (D_i \cup L_0 \cup ... \cup L_i)$ and $T_i = M_i \cup D_i$. Figure 4 summarizes the situation.

To bound the cost variation, we show in Lemma 2 that the degree of "surviving" vertices (*i.e.* vertices in R_i) increases by at most one. The next definition shows how to choose properly the set D_i.

Definition 1. *Let $i \in \{0, \ldots, t\}$, and let us suppose we are given a set $S_{i-1}^* \supseteq L_0 \cup ... \cup L_{i-1}$. Let $N_i = \{n_1, ..., n_{p_i}\}$ and suppose that $n_1 < ... < n_{p_i}$ defines an ordering of N_i according to the simplicial elimination order of the graph. For all $j = 1, ..., p_i$ successively, we pick a vertex $y_j \in S_{i-1}^* \setminus (L_0 \cup ... \cup L_i)$ as follows:*

$$y_j = \begin{cases} \min(Q_j) & \text{if } Q_j \neq \emptyset \\ \max(S_{i-1}^* \setminus (L_0 \cup ... \cup L_i \cup \{y_1, ..., y_{j-1}\})) & \text{if } Q_j = \emptyset \end{cases} \quad (1)$$

with $Q_j = \{y \in S_{i-1}^* \setminus (L_0 \cup ... \cup L_i \cup \{y_1, ..., y_{j-1}\})$ such that $n_j < y$, and $\{n_j, y\} \in E\}$ (see Figure 4). Finally, we define $D_i = \{y_j : 1 \leq j \leq p_i\}$.

It is easily seen that $|D_i| = |N_i|$ since all y_j are distinct. Now that D_i is defined, recall that we have $T_i = M_i \cup D_i$, and the "surviving vertices" $R_i = R_{i-1} \setminus T_i$. Let us now upper bound the degree of vertices of R_i.

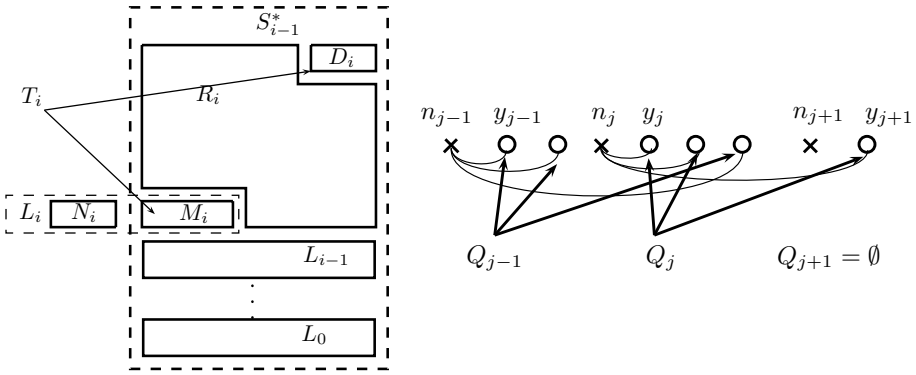

Fig. 4. On the left: description of set S^*_{i-1}. We obtain S^*_i from S^*_{i-1} by removing D_i and adding N_i. Notice that $R_{i-1} = R_i \cup T_i$, and $R_i \cap T_i = \emptyset$. On the right: example of sets Q_j, together with y_j. Circles represent vertices of the considered optimal solution, and crosses represent vertices chosen by the algorithm that are not in the optimal solution. Edges between vertices of the optimal solution have not been drawn for sake of simplicity.

Lemma 2. *Let $R_i = A_i \cup B_i$, with $A_i = \{x \in R_i : d(x, L_i) = 0$ or $(d(x, L_i) = 1$ and $w_i(x) = i)\}$ and $B_i = R_i \backslash A_i$. We have:*

- *if $x \in A_i$, $d(x, L_i) \leq d(x, T_i)$*
- *if $x \in B_i$, $d(x, L_i) \leq d(x, T_i) + 1$*

This immediately implies that for all $x \in R_i$ we have $w_{i+1}(x) \leq d(x, T_i) + w_i(x)$.

Proof. Let us show that if $x \in A_i$, then $d(x, N_i) \leq d(x, D_i)$, and if $x \in B_i$, then $d(x, N_i) \leq d(x, D_i) + 1$. Since $L_i = M_i \cup N_i$ and $T_i = M_i \cup D_i$ (these unions being disjoint), the desired inequalities follow immediatly.

- if $x \in A_i$, then either $d(x, L_i) = 0$, which obviously implies the result, or $d(x, L_i) = 1$ and $w_i(x) = i$. We thus only consider the second case. Here again if $d(x, N_i) = 0$ then the result is straightforward, so let us suppose $d(x, N_i) = 1$, *i.e.* suppose that there exists a vertex of N_i, say n_{j_0}, such that x and n_{j_0} are adjacent. Two cases are possible:
  - First case: $x < n_{j_0}$. Recall that n_{j_0} is the only neighbour of x in L_i. Hence, x is not adjacent to all vertices of L_i that are before n_{j_0} in the simplicial elimination order. In addition, recall that $w_i(x) = i$. Thus, this case cannot happen since by definition of the algorithm, x would have been chosen in L_i instead of n_{j_0}.
  - Second case: $n_{j_0} < x$. It is clear that in this case $Q_{j_0} \neq \emptyset$ (as at least $x \in Q_{j_0}$). As by definition $x \notin D_i$, we have $y_{j_0} < x$. By definition of perfect elimination order, since $\{n_{j_0}, y_{j_0}\} \in E$ and $\{n_{j_0}, x\} \in E$, we must have $\{x, y_{j_0}\} \in E$. Hence $d(x, D_i) = 1$ and the result follows.

– if $x \in B_i$, then let $N_i^- = \{y \in N_i : y < x\}$ and $N_i^+ = N_i \setminus N_i^-$. For all $n_j \in N_i^-$ such that $\{n_j, x\} \in E$, then as previously $Q_j \neq \emptyset$ and by the definition of chordal graphs we have $\{y_j, x\} \in E$ with $y_j \in D_i$. Thus, $d(x, N_i^-) \leq d(x, D_i)$. Finally we claim that $d(x, N_i^+) \leq 1$. Indeed, suppose that there exists $n_{j_1}, n_{j_2} \in N_i^+$ such that $\{x, n_{j_1}\}, \{x, n_{j_2}\} \in E$. By definition of perfect elimination order we must have $\{n_{j_1}, n_{j_2}\} \in E$ which contradicts the definition of L_i which is an independent set. This proves that $d(x, N_i) \leq d(x, D_i) + 1$ □

Let us now define the appropriate ζ function that computes the cost of an intermediate solution S_i^*. For all $i \in \{0, ..., t\}$, let

$$\zeta(S_i^*) = cost(R_i) + \sum_{x \in R_i} w_{i+1}(x) + \sum_{x \in L_0 \cup \cdots \cup L_i} w_{i+1}(x)$$

Notice that $\zeta(S_{-1}^*) = cost(S^*)$ and $\zeta(S_t^*) = \sum_{x \in S} w_t(x) = W$.

Lemma 3. *For all $i \in \{0, ..., t\}$, D_i is such that $\zeta(S_i^*) \leq \zeta(S_{i-1}^*)$.*

Proof. By definition, we have:

$$\begin{aligned}
\zeta(S_i^*) &= cost(R_i) + \textstyle\sum_{x \in R_i} w_{i+1}(x) + \sum_{x \in L_0 \cup \cdots \cup L_i} w_{i+1}(x) \\
&= cost(R_i) + \textstyle\sum_{x \in R_i} w_{i+1}(x) + i|L_i| + \sum_{x \in L_0 \cup \cdots \cup L_{i-1}} w_{i+1}(x) \\
&\leq cost(R_i) + \textstyle\sum_{x \in R_i} (w_i(x) + d(x, T_i)) + i|L_i| + \sum_{x \in L_0 \cup \cdots \cup L_{i-1}} w_i(x) \text{ by Lemma 2}
\end{aligned}$$

In addition, since $R_{i-1} = R_i \cup T_i$ and $|T_i| = |L_i|$, we have:

$$\begin{aligned}
\zeta(S_{i-1}^*) &= cost(R_{i-1}) + \textstyle\sum_{x \in R_{i-1}} w_i(x) + \sum_{x \in L_0 \cup \cdots \cup L_{i-1}} w_i(x) \\
&= cost(R_i) + cost(R_i, T_i) + \textstyle\sum_{x \in R_i} w_i(x) + \sum_{x \in T_i} w_i(x) + \sum_{x \in L_0 \cup \cdots \cup L_{i-1}} w_i(x) \\
&\geq cost(R_i) + cost(R_i, T_i) + \textstyle\sum_{x \in R_i} w_i(x) + i|L_i| + \sum_{x \in L_0 \cup \cdots \cup L_{i-1}} w_i(x)
\end{aligned}$$

which matches the upper bound for $\zeta(S_i^*)$. □

The previous lemma implies that $W = \zeta(S_t^*) \leq \zeta(S_{-1}^*) = cost(S^*)$. Thus, to prove that Algorithm 1 is a 2-approximation we only need the following lemma.

Lemma 4. $cost(S) \leq 2W$.

Proof. Roughly speaking, when creating a layer L_i and updating the cost W, the algorithm adds $i|L_i|$, *i.e.* for all $x \in L_i$ the algorithm only adds i instead of $d(x, L_0 \cup \cdots \cup L_{i-1})$. Thus, we will now prove for any $x \in L_i$, $d(x, L_0 \cup \cdots \cup L_{i-1}) \leq 2i$.

Let x in L_i. For any l, $0 \leq l \leq i$, let $x_l = w_l(x)$ be the weight of x before creating layer L_l. Thus, the successive weights of x is a sequence $(x_0, \ldots, x_i)$ where $x_0 = 0$ and $x_i = i$. Notice that after x is added in L_i its weight will not be changed.

Let us show by induction that for any l, $d(x, L_0 \cup \cdots \cup L_{l-1}) \leq x_l + l$. Let us suppose that the previous statement is true for l and prove it for $l + 1$.

Let $z = d(x, L_l)$. We have $d(x, L_0 \cup \cdots \cup L_l) = d(x, L_0 \cup \cdots \cup L_{l-1}) + z \leq x_l + l + z$. As $x_{l+1} \geq x_l + z - 1$, we get the desired inequality.

Thus, for any $x \in L_i$ we get $d(x, L_0 \cup \cdots \cup L_{i-1}) \leq x_i + i = 2i$, and thus $cost(S) = \sum_{i=1}^{t} \sum_{x \in L_i} d(x, L_0 \cup \cdots \cup L_{i-1}) \leq 2 \sum_{i=1}^{t} i|L_i| = 2W$. □

Theorem 1. *There is a tight polynomial* 2*-approximation algorithm for* SkS *in chordal graphs.*

For the tightness result, consider the instance with $n = 5$, $k = 4$, and edges $\{x_1, x_2\}$, $\{x_2, x_3\}$ and $\{x_4, x_5\}$ (notice that $(x_1, x_2, x_3, x_4, x_5)$ is a simplicial elimination order). The algorithm will first pick x_1, x_3 and x_4. Then, we have $w_1(x_2) = w_1(x_5) = 1$ and the algorithm takes x_2 instead of x_5.

3 $\mathcal{NP}$-Hardness in Chordal Graphs

Main Arguments. The following $\mathcal{NP}$-hardness proof is a reduction from the k-CLIQUE problem in general graphs. Roughly speaking, given an input instance $G = (V, E)$ together with $k \in \mathbb{N}$, we construct the split graph of adjacencies of G, *i.e.* we build a clique on a set A representing the vertices of G, and an independent set F representing the edges of G, connecting A and F with respect to the adjacencies of the graph. Then, we replace each vertex of the independent set (corresponding to an edge $e \in E$) by a gadget F_e represented in Figure 5. Any solution will have to take the same number of vertices among each gadget. The key idea is that there is two ways to take these vertices in a gadget F_e. The first way (choosing X_e and Z_e) encodes that the edge e belongs to the k-clique. It is cheaper than the second way, but is adjacent to the clique A. The second way (choosing X_e and Y_e) encodes that edge e does not belong to the k-clique. It induces more edges, but is not adjacent to the clique A. Thus, as depicted in Figure 5, a k-clique is encoded by *not* picking the corresponding vertices in A, obtaining $\binom{k}{2}$ gadgets of the first type, and $m - \binom{k}{2}$ of the second type. In this way, there is no edge in the solution between any gadget and the clique A. For technical reasons, each vertex of A is duplicated n times.

Gadget. Let us define the gadget F mentioned above. F is composed of three sets X, Y and Z of T vertices each (we will set the value of T later). We define $X = \{x_1, ..., x_T\}, Y = \{y_1, ..., y_T\}$ and $Z = \{z_1, ..., z_T\}$. The set X induces an independent set, while Z induces a clique, and there is a clique of size $(T-1)$ on vertices $\{y_2, ..., y_T\}$. For all $i \in \{1, ..., T\}$, x_i is adjacent to y_i, and y_i is adjacent to all vertices of Z. Such a construction is depicted at the left of Figure 5.

In the following we will force the solution to take $2T$ vertices among each gadget. It is easy to see that the sparsest $2T$-subgraph of F is composed of the sets X and Z, which induces $\binom{T}{2}$ edges. In contrast, notice that choosing X and Y induces $(\binom{T}{2} + 1)$ edges.

Theorem 2. SPARSEST k-SUBGRAPH *remains* $\mathcal{NP}$*-hard in chordal graphs.*

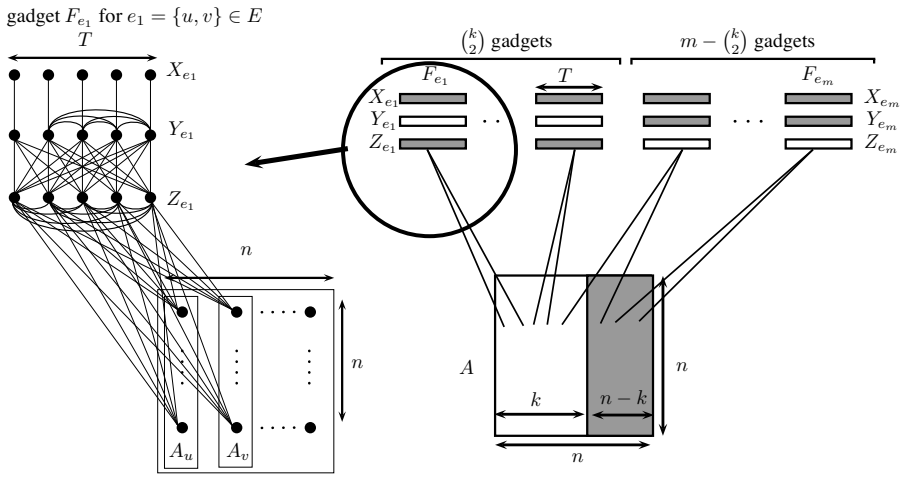

Fig. 5. Schema of the reduction, with an example of a gadget F_{e_1} on the left and its relations to A. Grey rectangles represent vertices of the solution.

Proof. We reduce from the classical k-CLIQUE problem in general graphs. Let $G = (V, E)$ and $k \in \mathbb{N}$. We note $|V| = n$, $V = \{v_1, ..., v_n\}$, $|E| = m$ and $T = n(n-k)$. In the following we will define $G' = (V', E')$ together with $k', C' \in \mathbb{N}$ such that G' is a chordal graphs which can be constructed in polynomial time, and such that G contains a clique of size k if and only if one can find k' vertices in G' which induce C' edges or less.

The Construction. V' is composed of two parts A and F:

- We first define a clique of size n^2 over $A = \{a_i^j : i, j \in \{1, ..., n\}\}$. For each $u \in V$, the "column" $A_u = \{a_u^j : j \in \{1, ..., n\}\}$ represents the vertex u in G.
- For all $e \in E$, we construct a gadget F_e composed of X_e, Y_e and Z_e as defined previously. Let $X_e = \{x_1^e, ..., x_T^e\}$, $Y_e = \{y_1^e, ..., y_T^e\}$ and $Z_e = \{z_1^e, ..., z_T^e\}$. Moreover, for all $e = \{v_p, v_q\} \in E$, all vertices of Z_e are connected to A_p and A_q.
- We define $k' = m2T + T$ and $C' = m\binom{T}{2} + \binom{T}{2} + (m - \binom{k}{2})$.

It is clear that the construction can be carried out in polynomial time. Let us briefly sketch that G' is a chordal graph: for each gadget, X_e, Y_e, Z_e is a simplicial elimination order. Then, the remaining vertices form a clique.

Now we prove that G contains a clique of size k if and only if G' contains k' vertices inducing at most C' edges.

Lemma 5. *G contains a k-clique $\Rightarrow$ G' contains k' vertices inducing at most C' edges.*

Proof. Let us suppose that $K \subseteq V$ is a clique of size k in G. W.l.o.g. we suppose $K = \{v_1, ..., v_k\}$. Moreover, we note $E_0 = \{\{v_p, v_q\} \in E$ such that $v_p, v_q \in K\}$ and $E_1 = \{\{v_p, v_q\} \in E$ such that $v_p \notin K$ or $v_q \notin K\}$. We construct $K' \subseteq V'$ as follows:

- For all $i \in \{(k+1), ..., n\}$ and all $j = \{1, ..., n\}$, we add a_i^j to K'.
- For all $e \in E$, we add all vertices of X_e to K'.
- For all $e \in E_0$, we add all vertices of Z_e to K'.
- For all $e \in E_1$, we add all vertices of Y_e to K'.

One can verify that K' is a set of $k' = 2mT + T$ vertices inducing exactly $C' = \binom{T}{2} + m\binom{T}{2} + (m - \binom{k}{2})$ edges. Indeed, we picked $T = n(n-k)$ vertices from A which is a clique and thus induce $\binom{T}{2}$ edges. Then, for all $e \in E$, we picked $2T$ vertices, which induce $\binom{T}{2}$ edges if $e \in E_0$, and $(\binom{T}{2} + 1)$ edges if $e \in E_1$. Since $|E_0| = \binom{k}{2}$ (and thus $|E_1| = m - \binom{k}{2}$), we have the desired number of edges.

Lemma 6. *G' contains k' vertices inducing at most C' edges $\Rightarrow$ G contains a k-clique.* □

4 Approximation in Proper Interval Graphs

Let us now discuss the status of Sparsest k-Subgraph and Densest k-Subgraph on interval graphs. First, notice that the complexity status ($\mathcal{NP}$-hardness *versus* $\mathcal{P}$) of Sparsest k-Subgraph remains unknown in interval and proper interval graphs. We also recall that this question is a longstanding open problem for DkS, as well as its complexity in planar graphs. Indeed, the former paper [8] proves the $\mathcal{NP}$-hardness of DkS in comparability, chordal graphs, and states the open question of its complexity in planar and (proper) interval graphs. Since then, and despite a lot of effort, no major improvement has been done so far.

As interval graphs are exactly the intersection of chordal graphs and co-comparability graphs, finding out the complexity status of Sparsest k-Subgraph in interval graphs would determine the complexity of Densest k-Subgraph in a subclass of comparability graphs, improving the results of [8]. Finally, as in [15] where the author design a *PTAS* for Densest k-Subgraph on interval graph (despite the unknown complexity status), we are able to show the following theorem.

Theorem 3. *There is a PTAS for* SkS *in proper intervals running in $n^{\mathcal{O}(\frac{1}{\epsilon})}$.*

This result uses the same kind of arguments as in [15]: restructuring an optimal solution in each "block" of consecutive intervals, and using dynamic programing on these restructured blocks.

References

1. Apollonio, N., Sebő, A.: Minconvex factors of prescribed size in graphs. SIAM Journal of Discrete Mathematics 23(3), 1297–1310 (2009)
2. Apollonio, N., Simeone, B.: The maximum vertex coverage problem on bipartite graphs. Preprint (2013)
3. Bhaskara, A., Charikar, M., Chlamtac, E., Feige, U., Vijayaraghavan, A.: Detecting high log-densities: An $\mathcal{O}(n^{1/4})$ approximation for densest k-subgraph. In: Proceedings of the 42nd ACM symposium on Theory of Computing, pp. 201–210. ACM (2010)
4. Bourgeois, N., Giannakos, A., Lucarelli, G., Milis, I., Paschos, V., Pottié, O.: The max quasi-independent set problem. Journal of Combinatorial Optimization 23(1), 94–117 (2012)
5. Broersma, H., Golovach, P.A., Patel, V.: Tight complexity bounds for FPT subgraph problems parameterized by clique-width. In: Marx, D., Rossmanith, P. (eds.) IPEC 2011. LNCS, vol. 7112, pp. 207–218. Springer, Heidelberg (2012)
6. Bshouty, N., Burroughs, L.: Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem. In: Meinel, C., Morvan, M., Krob, D. (eds.) STACS 1998. LNCS, vol. 1373, pp. 298–308. Springer, Heidelberg (1998)
7. Cai, L.: Parameterized complexity of cardinality constrained optimization problems. Computer Journal 51(1), 102–121 (2008)
8. Corneil, D.G., Perl, Y.: Clustering and domination in perfect graphs. Discrete Applied Mathematics 9(1), 27–39 (1984)
9. Fulkerson, D., Gross, O.: Incidence matrices and interval graphs. Pacific J. Math. 15, 835–855 (1965)
10. Goldschmidt, O., Hochbaum, D.S.: k-edge subgraph problems. Discrete Applied Mathematics 74(2), 159–169 (1997)
11. Guo, J., Niedermeier, R., Wernicke, S.: Parameterized complexity of vertex cover variants. Theory of Computing Systems 41(3), 501–520 (2007)
12. Joret, G., Vetta, A.: Reducing the rank of a matroid. CoRR, abs/1211.4853 (2012)
13. Kneis, J., Langer, A., Rossmanith, P.: Improved upper bounds for partial vertex cover. In: Broersma, H., Erlebach, T., Friedetzky, T., Paulusma, D. (eds.) WG 2008. LNCS, vol. 5344, pp. 240–251. Springer, Heidelberg (2008)
14. Liazi, M., Milis, I., Zissimopoulos, V.: A constant approximation algorithm for the densest k-subgraph problem on chordal graphs. Information Processing Letters 108(1), 29–32 (2008)
15. Nonner, T.: PTAS for densest k-subgraph in interval graphs. In: Dehne, F., Iacono, J., Sack, J.-R. (eds.) WADS 2011. LNCS, vol. 6844, pp. 631–641. Springer, Heidelberg (2011)
16. Watrigant, R., Bougeret, M., Giroudeau, R.: The k-sparsest subgraph problem. Technical Report RR-12019, LIRMM (2012)
17. Watrigant, R., Bougeret, M., Giroudeau, R.: Approximating the sparsest k-subgraph in chordal graphs. Technical Report hal-00868188, LIRMM (2013)

Improved Approximation Algorithm for k-Level UFL with Penalties, a Simplistic View on Randomizing the Scaling Parameter

Jaroslaw Byrka[1,⋆,⋆⋆], Shanfei Li[2,⋆], and Bartosz Rybicki[1,⋆,⋆⋆⋆]

[1] Institute of Computer Science, University of Wroclaw, Poland
[2] Delft Institute of Applied Mathematics, TU Delft, The Netherlands
{jby,bry}@ii.uni.wroc.pl, shanfei.li@tudelft.nl

Abstract. The state of the art in approximation algorithms for facility location problems are complicated combinations of various techniques. In particular, the currently best 1.488-approximation algorithm for the uncapacitated facility location (UFL) problem by Shi Li is presented as a result of a non-trivial randomization of a certain scaling parameter in the LP-rounding algorithm by Chudak and Shmoys combined with a primal-dual algorithm of Jain et al. In this paper we first give a simple interpretation of this randomization process in terms of solving an auxiliary (factor revealing) LP. Then, armed with this simple view point, we exercise the randomization on a more complicated algorithm for the k-level version of the problem with penalties in which the planner has the option to pay a penalty instead of connecting chosen clients, which results in an improved approximation algorithm.

1 Introduction

In the uncapacitated facility location (UFL) problem the goal is to open facilities in a subset of given locations and connect each client to an open facility so as to minimize the sum of opening costs and connection costs. In the penalty avoiding (prize collecting) variant of the problem, a fixed penalty can be paid instead of connecting a client.

In the k-level uncapacitated facility location problem with penalties (k-level UFLWP), we are given a set C of clients and a set $F = \bigcup_{t=1}^{k} F_{l_t}$ of facilities (locations to potentially open a facility) in a metric space. Facilities are of k different types (levels), e.g., for $k = 3$ one may think of these facilities as shops, warehouses and factories. Each set F_{l_t} contains all facilities on level t and the sets F_{l_t} are pairwise disjoint. Each client j can either be connected to precisely one facility at each of k levels (via a path), or be rejected in which case the penalty p_j must be paid (p_j can be considered as the loss of profit). To be more precise, for a

⋆ Corresponding authors.
⋆⋆ Supported by FNP HOMING PLUS/2010-1/3 grant and MNiSW grant number N N206 368839, 2010-2013.
⋆⋆⋆ Supported by NCN 2012/07/N/ST6/03068 grant.

C. Kaklamanis and K. Pruhs (Eds.): WAOA 2013, LNCS 8447, pp. 85–96, 2014.

client j to be connected, it must be connected with a path $(j, i_1, i_2, \cdots, i_{k-1}, i_k)$, where i_t is an open facility on level t. The cost of connecting points $i, j \in C \cup F$, is the distance between i and j, denoted by c_{ij}. The cost of opening facility i is f_i ($f_i \geq 0$). The goal is to minimize the sum of the total cost of opening facilities (at all levels), the total connection cost and the total penalty cost. In the uniform version of the problem all penalties are the same, i.e., for any two clients $j_1, j_2 \in C$ we have $p_{j_1} = p_{j_2}$.

1.1 Related Work

If $p_j, j \in C$ are big enough, k-level UFLWP is the k-level UFL problem, for which Krishnaswamy and Sviridenko [14] showed 1.61-hardness of approximation for general k and 1.539-hardness for $k = 2$. Actually, even for $k = 1$ Guha and Khuller [12] showed that the approximation ratio is at least 1.463, unless $NP \subseteq DTIME(n^{\log \log n})$. The current best known approximation ratio for this simplest case $k = 1$ is 1.488 by Li [15].

For 2-level UFL problem Shmoys, Tardos, and Aardal [17] gave the first constant factor approximation algorithm by extending the algorithm for 1-level and obtaining an approximation ratio 3.16. Subsequently, Aardal, Chudak, and Shmoys [1] used randomized rounding to get the first algorithm for general k, which had approximation ratio of 3. Ageev, Ye and Zhang [2] gave a combinatorial 3.27-approximation algorithm for general k by reducing the k-level directly into 1-level problem. By recursive reduction, i.e., reducing k-level to $k-1$ level, they obtained an improved 2.43-approximation for $k = 2$ and 2.85 for $k = 3$. Later, this was improved by Zhang [21], who combined the maximization version of 1-level UFL problem and dual-fitting to get a 1.77-approximation algorithm for $k = 2$, and a 2.53-approximation for $k = 3$. Byrka and Aardal [4] improved the ratio for $k = 3$ to 2.492. For $k > 2$ the ratio was recently improved by Byrka and Rybicki [7] to 2.02 for $k = 3$, 2.14 for $k = 4$, and the ratio converges to 3 when $k \to +\infty$.

UFL with penalties was first introduced by Charikar et al. [8], who gave a 3-approximation algorithm based on a primal-dual method. Later, Jain et al. [13] indicated that their greedy algorithm for UFL could be adapted to UFLWP with the approximation ratio 2. Xu and Xu [19,20] proposed a 2.736-approximation algorithm based on LP-rounding and a combinatorial 1.853-approximation algorithm by combining local search with primal-dual. Later, Geunes et al. [11] presented an algorithmic framework which can extend any LP-based α-approximation algorithm for UFL to get an $(1 - e^{-1/\alpha})^{-1}$-approximation algorithm for UFL with penalties. As a result, they gave a 2.056-approximation algorithm for this problem. Recently, Li et al. [16] extended the LP-rounding algorithm by Byrka and Aardal [4] and the analysis by Li [15] to UFLWP to give the currently best 1.5148-approximation algorithm.

For multi-level UFLWP, Asadi et al. [3] presented an LP-rounding based 4-approximation algorithm by converting the LP-based algorithm for UFLWP by Xu and Xu [19] to k-level. To the best of our knowledge, this is the only algorithm for multi-level UFLWP in the literature.

1.2 Our Contribution

We first show that algorithms whose performance can be analysed with a linear function of certain instance parameters, like the Chudak and Shmoys algorithm [9] for UFL, can be easily combined and analysed with a natural factor revealing LP. This simplifies the argument of Shi Li [15] for his 1.488-approximation algorithm for UFL since an explicit distribution for the parameters obtained by a linear program is not necessary in our factor revealing LP.

With this tool one can easily randomize the scaling factor in LP-rounding algorithms for various variants of the UFL problem. We demonstrate this by randomizing the algorithm for k-level UFLWP. For k-level UFL we can get the same approximation ratios as for k-level UFLWP by setting $p_j = +\infty, j \in C$.

Note that the previously best ratio is 4 for k-level UFLWP ($k \geq 2$) [3] and 1.5148 for $k = 1$ [16]. The following table shows how much we improve the approximation ratios of our algorithm for $k = 1, \ldots, 10$ by involving randomization of the scaling factor. Irrespective of the way in which we choose γ, deterministically or randomly, approximation ratio converges to three.

Table 1. Comparison of ratios

k	1	2	3	4	5	6	7	8	9	10
no randomization of γ	1.58	1.85	2.02	2.14	2.24	2.31	2.37	2.42	2.46	2.50
with randomization of γ	1.52	1.79	1.97	2.09	2.19	2.27	2.33	2.39	2.43	2.47

2 Simple Version of Li's Argument

Consider the following standard LP relaxation of UFL.

$$min \sum_{i \in F} \sum_{j \in C} c_{ij} x_{ij} + \sum_{i \in F} y_i f_i \tag{1}$$

$$\sum_{i \in F} x_{ij} = 1 \qquad \forall_{j \in C} \tag{2}$$

$$y_i - x_{ij} \geq 0 \qquad \forall_{i \in F, j \in C} \tag{3}$$

$$x_{ij}, y_i \geq 0 \qquad \forall_{i \in F, j \in C} \tag{4}$$

Chudak and Shmoys [9] gave a randomized rounding algorithm for UFL based on this relaxation. Later Byrka and Aardal [4] considered a variant of this algorithm where the facility opening variables were initially scaled up by a factor of γ. They showed that for $\gamma \geq \gamma_0 \approx 1.67$ the algorithm returns a solution with cost at most γ times the fractional facility opening cost plus $1 + 2e^{-\gamma}$ times the fractional connection cost. This algorithm, when combined with the (1.11, 1.78)-approximation algorithm of Jain, Mahdian and Saberi [13] (JMS algorithm for short), is easily a 1.5-approximation algorithm for UFL. More recently, Li [15]

showed that by randomly choosing the scaling parameter γ from an certain probability distribution one obtains an improved 1.488-approximation algorithm. A natural question is what improvement this technique gives in the k-level variant.

In what follows we present our simple interpretation and sketch the analysis of the randomization by Li. We argue that a certain factor revealing LP provides a valid upper bound on the obtained approximation ratio. The appropriate probability distribution for the scaling parameter (engineered and discussed in detail in [15]) may in fact be directly read from the dual of our LP. While we do not claim to get any deeper understanding of the randomization process itself, the simpler formalism we propose is important for us to apply randomization to a more complicated algorithm for k-level UFL, which we describe next.

2.1 Notation

Let F_j denote the set of facilities which client $j \in C$ is fractionally connected to, i.e., facilities i with $x_{ij} > 0$ in the optimal LP solution. Since for uncapacitated facility location problems one can split facilities before rounding, to simplify the presentation, we will assume that F_j contains lots of facilities with very small fractional opening y_i. This will enable splitting F_j into subsets of desired total fractional opening.

Definition 1 (definition 15 from [15]). *Given an UFL instance and its optimal fractional solution (x^*, y^*), the characteristic function $h_j : [0,1] \longmapsto R$ of a client $j \in C$ is the following. Let $i_1, i_2, \cdots, i_m$ denote the facilities in F_j, in a non-decreasing order of distances to j. Then $h_j(p) = d(i_t, j)$, where t is the minimum number such that $\sum_{s=1}^{t} y^*_{i_s} \geq p$. Furthermore, define $h(p) = \sum_{j \in C} h_j(p)$ as the characteristic function for the entire fractional solution.*

Definition 2. *Volume of set $F' \subseteq F$, denoted by $vol(F')$ is the sum of facility openings in that set, i.e., $vol(F') = \sum_{i \in F'} y^*_i$.*

For $l = 1, 2 \ldots, n$ define $\gamma_l = 1 + 2 \cdot \frac{n-l}{n}$, which will form the support for the probability distribution of the scaling parameter γ. Suppose that all facilities are sorted in an order of non-decreasing distances from client $j \in C$. Scale up all y^* variables by γ_l and divide the set of facilities F_j into two disjoint subsets: the close facilities of client j, $F_j^{C_l}$, such that $vol(F_j^{C_l}) = 1$; and the distant facilities $F_j^{D_l} = F_j \setminus F_j^{C_l}$. Note that $vol(F_j^{D_l}) = \gamma_l - 1$. Observe that $\frac{1}{\gamma_k} < \frac{1}{\gamma_l} \Rightarrow F_j^{C_k} \subset F_j^{C_l} \wedge F_j^{C_l} \setminus F_j^{C_k} \neq \emptyset$. We now split F_j into disjoint subsets F_j^l. Define $F_j^{C_0} = \emptyset$ and $F_j^l = F_j^{C_l} \setminus F_j^{C_{l-1}}$, where $l = 1, 2 \ldots, n$. The average distance from j to facilities in F_j^l is $c_l(j) = \int_{1/\gamma_{l-1}}^{1/\gamma_l} h_j(p)\, dp$ for $l > 1$ and $\int_0^{1/\gamma_1} h_j(p)\, dp$ for $l = 1$. Note that $c_l(j) \leq c_{l+1}(j)$ and $D^l_{max}(j) \leq c_{l+1}(j)$, where $D^l_{max}(j) = max_{i \in F_j^l} c_{ij}$.

Since the studied algorithm with the scaling parameter $\gamma = \gamma_k$ opens each facility i with probability $\gamma_k \cdot y^*_i$, and there is no positive correlation between facility opening in different locations, the probability that at least one facility is open from the set F_j^l is at least $1 - e^{-\gamma_k \cdot vol(F_j^l)}$.

Crucial to the analysis is the length of a connection via the cluster center j' for client j when no facility in F_j is open. Consider the algorithm with a fixed scaling factor $\gamma = \gamma_k$, an arbitrary client j and its cluster center j'. Li gave the following upper bound on the expected distance from j to an open facility around its cluster center j'.

Lemma 1 (Lemma 14 from [15]). *If no facility in F_j is opened, the expected distance to the open facility around j' is at most $\gamma_k D_{av}(j) + (3-\gamma_k)D^k_{max}(j)$, where $D_{av}(j) = \sum_{i \in F_j} c_{ij} x^*_{ij}$.*

Corollary 1. *If $\gamma = \gamma_k$, then the expected connection cost of client j is at most*

$$E[C_j] \leq \sum_{l=1}^{n} c_l(j) \cdot p_l + (1 - e^{-\gamma_k}) \cdot (\gamma_k D_{av}(j) + (3-\gamma_k)D^k_{max}(j))$$

where p_l is the probability of the following event: no facility is opened in distance at most $D^{l-1}_{max}(j)$ and at least one facility is opened in F^l_j.

2.2 Factor Revealing LP

Consider running once the JMS algorithm and the Chudak and Shmoys algorithm multiple times, one for each choice of the value for the scaling parameter $\gamma = \gamma_l = 1 + 2 \cdot \frac{n-l}{n}, l = 1, 2 \ldots n$. Observe that the following LP captures the expected approximation factor of the best among the obtained solutions, where $p^k_1 = 1 - e^{-\frac{\gamma_k}{\gamma_1}}$ and $p^k_l = e^{-\frac{\gamma_k}{\gamma_{l-1}}} - e^{-\frac{\gamma_k}{\gamma_l}}$ for all $l > 1$. Goal of the below LP is to construct the worst case instance of distances c_l.

$$max\ T \tag{5}$$

$$\gamma_k f + \sum_{l=1}^{n} c_l \cdot p^k_l + (1 - e^{-\gamma_k})(\gamma_k c + (3-\gamma_k)c_{l+1}) \geq T \quad \forall_{k<n} \tag{6}$$

$$1.11f + 1.78c \geq T \tag{7}$$

$$\frac{1}{\gamma_1} \cdot c_1 + \sum_{i=2}^{n} (\frac{1}{\gamma_i} - \frac{1}{\gamma_{i-1}}) \cdot c_i = c \tag{8}$$

$$0 \leq c_i \leq c_{i+1} \leq 1 \quad \forall_{i<n} \tag{9}$$

$$f + c = 1 \tag{10}$$

$$f, c \geq 0 \tag{11}$$

The variables of this program encode certain measurements of the function $h(p)$ defined for an optimal fractional solution. Intuitively, these are average distances between a client and a group of facilities, summed up for all the clients. The program models the freedom of the adversary in selecting cost profile $h(p)$ to maximize the cost of the best of the considered algorithms. Variables f and c model the facility opening and client connection cost in the fractional solution. Inequality (6) correspond to LP-rounding algorithms with different choices

of the scaling parameter γ. Note that $D_{av}(j) = c$ and $D^l_{max} \leq c_{l+1}(j)$ holds for each client, that fact, with corollary (1), justifies inequality (6). Inequality (7) corresponds to the JMS algorithm [13], and equality (8) encodes the total connection cost .

Interestingly, the choice of the best algorithm here is not better in expectation than a certain random choice between the algorithms. To see this, consider the dual of the above LP. In the dual, the variables corresponding to the primal constraints (6) and (7) simply encode the probabilities for choosing a particular algorithm. Our computational experiments with the above LP confirmed the correctness of the analysis of Li [15]. Additionally, from the primal program with distances we obtained the worst case profile $h(p)$ for the state of the art collection of algorithms considered (see Fig. 1 and Fig. 2 respectively for a plot of this tight profile and the distributions of the scaling factor for k-level UFL on different number of levels).

3 Reduction from k-Level UFL with Uniform Penalties to k-Level UFL

The difficulty of k-level UFLWP lies in the extra choice of each client, that is, the penalty. We will explain how to overcome the penalties by converting the instance of UFLWP to an appropriate instance of UFL. We first consider the easy case of uniform penalties.

Lemma 2. *Each instance of UFL with uniform penalties can be modified to an appropriate UFL instance.*

Proof. We can treat the penalty of client $j \in C$ as a facility at distance p_j to client j with opening cost zero. The distance from client j to the penalty-facility of client j' is equal to $c_{j,j'} + p_{j'}$. Note that $p_{j'} = p_j$. We can run any algorithm for UFL on the modified instance as described above. If in the obtained solution client j is connected with the penalty-facility of client j', we can switch j to its penalty-facility without increasing the cost of the solution. □

Lemma 2 implies that for k-level uncapacitated facility location with uniform penalties we have the following approximation ratios. Algorithms for $k = 1$ and 2 are described in [15] and [21], for $k > 2$ are described in this article.

k	1	2	3	4	5	6	7	8	9	10
ratio	1.488	1.77	1.97	2.09	2.19	2.27	2.33	2.39	2.43	2.47

Note that the reduction above does not work for the non-uniform case, because then the distance from client j to the penalty-facility of client j' could be smaller than p_j. Nevertheless we will show that LP-rounding algorithms in this paper can be easily extended to the non-uniform penalty variant.

4 Extended LP Formulation

For non-uniform case, our algorithm is based on rounding a solution to the extended LP-relaxation of the problem. This extended LP may either be seen as the standard LP on a modified graph as described in [7], or originate from the k-th level of the Sherali Adams hierarchy, or explicitly be written in terms of paths on the original instance. Here we use the explicit construction. Note that in the optimal solution to k-level UFLWP each facility is connected to at most one facility on the higher level. We will impose this structure on the fractional solution by creating multiple copies of the original facility, one for each path across the higher levels of facilities.

To describe the linear program we have to give a few definitions. Let P_C be the set of paths which start in a client and end in a facility on level k. Let P_t be the set of paths which start on level t and end on the highest level k, i.e., in a root of some tree. By P we denote the set of all paths, i.e., $P = P_C \cup \bigcup_{t=1}^{k} P_t$. The cost of the path denoted by c_p depends on the kind of path. If $p = (j, i_1, i_2, \cdots, i_k) \in P_C$, then $c_p = c_{i_1 j} + c_{i_2 i_1} + \cdots + c_{i_k, i_{k-1}}$. If $p = (i_t, i_{t+1}, \cdots, i_k) \in P_t$, then $c_p = f_{i_t}$.

$$min \sum_{p \in P} x_p c_p + \sum_{j \in C} g_j p_j \tag{12}$$

$$\sum_{p \in P_C : j \in p} x_p + g_j \geq 1 \qquad \forall_{j \in C} \tag{13}$$

$$x_{(i_{t+1}, i_{t+2}, \ldots i_k)} - x_{(i_t, i_{t+1}, \ldots i_k)} \geq 0 \qquad \forall_{p=(i_t, i_{t+1}, \ldots i_k) \in P_t, t<k} \tag{14}$$

$$x_q - \sum_{p=(j, \ldots i_t, i_{t+1} \ldots i_k) \in P_C} x_p \geq 0 \qquad \forall_{j \in C}, \forall_{q=(i_t, i_{t+1}, \ldots i_k) \in P \setminus P_C} \tag{15}$$

$$x_p \geq 0 \qquad \forall_{p \in P} \tag{16}$$

$$g_j \geq 0 \qquad \forall_{j \in C} \tag{17}$$

The natural interpretation of the above LP is as follows. Inequality (13) states that each client is assigned to at least one path or is rejected. Inequality (14) encodes that opening a lower level facility implies opening its unique higher level facility. The most complicated inequality (15) for a client $j \in C$ and a facility $i_t \in F_{l_t}$, imposes that the opening of i_t must be at least the total usage of it by the client j. Let (x^*, g^*) be an optimal solution to the above LP.

5 Algorithm for k-Level UFL with Penalties

The approximation algorithm A presented below is parameterized by γ_l.

1: formulate and solve the extended LP (12)-(17) to get an optimal solution (x^*, g^*);
2: scale up facility opening and client rejecting variables by γ_l, then recompute values of x_p^* for $p \in P_C$ to obtain a minimum cost solution $(\bar{x}, \bar{g})$;

3: divide clients into two groups $C_{\gamma_l} = \{j \in C | \gamma_l \cdot (1 - g_j^*) \geq 1\}$ and $\bar{C}_{\gamma_l} = C \setminus C_{\gamma_l}$;
4: cluster clients in C_{γ_l};
5: round facility opening (tree by tree);
6: connect each client j with a closest open connection path unless rejecting it is a cheaper option.

Our final algorithm is as follows: run algorithm $A(\gamma_l)$ for each $l = 1, 2 \ldots, n-1$ and select a solution with the smallest cost.

Clustering is based on rules described in [9] which is generalized in [7] for k-level instances. Rounding on a tree was also used in [7]. Nevertheless, for completeness we give a brief description of step 4 and 5 in the following subsections. From now on we are considering only scaled up instance $(\bar{x}, \bar{g})$.

5.1 Close and Distant Facilities

For any client $j \in C_\gamma$, let P^j be the set of top-level facilities fractionally serving j in $(\bar{x}, \bar{g})$. As discussed in Section 6.1, WLOG the fractional connectivity of j to a set of facilities may be assumed to be the fractional opening of these facilities. Sort facilities $i_1, i_2, \ldots i_m$ from P^j by non-decreasing distance from client $j \in C_\gamma$, and select the smallest subset of P^j with volume one - this is the set of close facilities P_c^j, the rest of facilities from P^j are distant facilities P_d^j. By $D_{av}^C(j), D_{av}^D(j)$ and $D_{av}(j)$ we denote the average distances from j to close, distant and all facilities in set P^j respectively. Moreover by $D_{max}^C(j)$ we denote the maximal distance from j to a close facility. Formal definitions are as follows:

$$D_{av}^C(j) = \frac{\sum_{p \in P_c^j} c_p \bar{x}_p}{\sum_{p \in P_c^j} \bar{x}_p} = \sum_{p \in P_c^j} c_p \bar{x}_p; \quad D_{av}^D(j) = \frac{\sum_{p \in P_d^j} c_p \bar{x}_p}{\sum_{p \in P_d^j} \bar{x}_p} = \frac{\sum_{p \in P_d^j} c_p \bar{x}_p}{\gamma(1 - g_j^*) - 1}.$$

Using the similar arguments as in [5] we can define $\rho_j = \frac{D_{av}(j) - D_{av}^C(j)}{D_{av}(j)}$ and express $D_{av}^C(j)$ and $D_{av}^D(j)$ using ρ_j.

$$D_{av}^C(j) = (1 - \rho_j) D_{av}(j); \quad D_{av}^D(j) = (1 + \frac{\rho_j}{\gamma(1 - g_j^*) - 1}) D_{av}(j).$$

5.2 Clustering

Two clients $j_1, j_2 \in C_\gamma$ are called neighbors if $P_c^{j_1} \cap P_c^{j_2} \neq \emptyset$.

1: **while** there is an unclustered client in C_γ **do**
2: select unclustered client $j \in C_\gamma$ that minimizes $D_{av}^C(j) + D_{max}^C(j)$,
3: form a new cluster containing j and all its unclustered neighbors from C_γ,
4: call j the center of the new cluster;
5: **end while**

The above clustering procedure (just like in [9]) partitions all clients into groups called clusters. Such partition has two important properties. First: there are no two neighbors from C_γ which are (both) centers of clusters. Second: distance from any client in cluster to his cluster center is not too big.

5.3 Randomized Facility Opening

Consider an arbitrary cluster center j. Since LP solutions have a form of a forest, we only need to focus on rounding single tree serving j. For clarity, within this rounding procedure we will refer to facilities as vertices (of a tree), and use x_v to denote the fractional opening of vertex (facility) v and y_v to denote the extent in which the cluster center j uses v in $(\bar{x}, \bar{g})$, i.e, $y_v = \sum_{p \in P^j : v \in p} \bar{x}_p$. Note that $x_v \geq y_v$ for each v and $x_v \leq x_{father(v)}$ if v is not the root of a tree.

The main idea is to open exactly one path for cluster center j but keep the probability of opening of each vertex v equal to x_v in the randomized procedure. In [7] we gave a token-passing-based adaptation of the procedure by Garg Konjevod and Ravi [10], that stores the output in $\hat{x}$ and $\hat{y}$, and has exactly the desired properties.

Lemma 3. *$E[\hat{x}_v] = x_v$ and $E[\hat{y}_v] = y_v$ for all $v \in V$.*

It is essential that the probability of opening at least one path in a set $B_j \subseteq \{p \in P_C \mid j \in p\}$ can be lower bounded by a certain function $F_k(x)$, where x is the total flow from client j to all paths in B_j and k is the number of levels in the considered instance. It can be shown that $F_1(x) \geq 1 - e^x$ and the following lemma (from [7]) hold. For more details see [7].

Lemma 4. *Inequality $F_k(x) \geq 1 - e^{(c-1)x}$ implies $F_{k+1}(x) \geq 1 - e^{(e^{c-1}-1)x}$.*

6 Analysis

The high level idea is that we can consider the instance of k-level UFLWP as a corresponding instance of k-level UFL by showing that the worst case approximation ratio is for clients in set C_γ and we can treat the penalty of client $j \in C_\gamma$ as a "penalty-facility" in our analysis. That is, we can overcome penalties by solving an equivalent k-level UFL without penalties.

6.1 Complete Solution and "One-Level" Description

It is standard in uncapacitated location problems to split facilities to obtain a so called *complete* solution, where no facility is used less than it is open by a client (see [18] for details). For our algorithm, to keep the forest structure of the fractional solution, we must slice the whole trees instead of splitting individual facilities to obtain the following.

Lemma 5. *Each solution of our linear program for k-level UFLWP can be transformed to an equivalent complete solution.*

The proof is standard (see [6]).

For the clarity of the following analysis we will use a "one-level" description of the instance and fractional solution despite its k-level structure. Because the

number of levels will have influence only on the probabilities of opening particular paths in our algorithm.

Consider set S_j of paths which start in client j and end in the root of a single tree T. Instead of thinking about all paths from set S_j separately we can now treat them as one path p_T whose fractional opening is $x_{p_T} = \sum_{p \in S_j} \bar{x}_p$ and (expected) cost is $c_{p_T} = \frac{\sum_{p \in S_j} c_p \bar{x}_p}{x_{p_T}}$. Observe that our distance function c_{p_T} satisfy the triangle inequality. From now on we will think only about clients and facilities (on level k) and (unique) paths between them. Accordingly, we will now encode the fractional solution as $(\bar{x}, \bar{y}, \bar{g})$, to denote the fractional connectivity, opening and penalty components.

6.2 Penalty Discussion

Lemma 6. $\forall_{\gamma>1, 1 \geq g_j^* \geq 0}\ D^C_{max}(j) \leq \gamma(1-g_j^*)D_{av}(j) + (3-\gamma(1-g_j^*))D^C_{max}(j)$.

Lemma 7. *The worst case approximation ratio is for clients from set C_γ.*

Proof of lemmas 6 and 7 in full version of the paper, see [6].

Lemma 8. *For clients $j \in C_\gamma$ we can treat its penalty as a facility.*

Proof. If j is a cluster center, j will have at least one (real) facility open in its set of close facilities. Thus, its connection and penalty cost are independent of the value of g_j^*. If j is not a cluster center and we pretend its penalty as a facility, no other client j' will consider to use this fake facility. Because j' only looks at facilities fractionally serving him, and the facilities which serve the center of the cluster containing j'. □

6.3 Approximation Ratio

A single algorithm $A(\gamma)$ has expected facility opening cost $E[F] \leq \gamma \cdot F^*$ and expected connection and penalty cost $E[C+P] \leq max\{3-2 \cdot F_k(\gamma), \frac{2-F_k(\gamma)-F_k(1)}{1-\frac{1}{\gamma}}\} \cdot (C^* + P^*)$ (see [6] for a detailed proof). To obtain an improved approximation ratio we run algorithm A for several values of γ and select the cheapest solution. The following LP gives an upper bound on the approximation ratio.

$$max\ T \tag{18}$$

$$\gamma_i f + \sum_{l=1}^{n} c_l \cdot p_l^i + (1-e^{-\gamma_i})(\gamma_i c + (3-\gamma_i)c_{i+1}) \geq T \ \ \forall_{i<n} \tag{19}$$

$$\frac{1}{\gamma_1} \cdot c_1 + \sum_{i=2}^{n} (\frac{1}{\gamma_i} - \frac{1}{\gamma_{i-1}}) \cdot c_i = c \tag{20}$$

$$0 \leq c_i \leq c_{i+1} \leq 1 \ \ \forall_{i<n} \tag{21}$$

$$f + c = 1 \tag{22}$$

$$f, c \geq 0 \tag{23}$$

Since the number of levels has influence on connection probabilities, the values of p_l^i need to be defined more carefully than for UFL. In particular, for $l = 1$ we now have $p_1^i = 1 - F_k(\frac{\gamma_i}{\gamma_1})$ and $p_l^i = F_k(\frac{\gamma_i}{\gamma_{l-1}}) - F_k(\frac{\gamma_i}{\gamma_l})$ for $l > 1$.

The Table 1 summarizes the obtained ratios for a single algorithm (run with the best choice of γ for particular k) and for a group of algorithms.

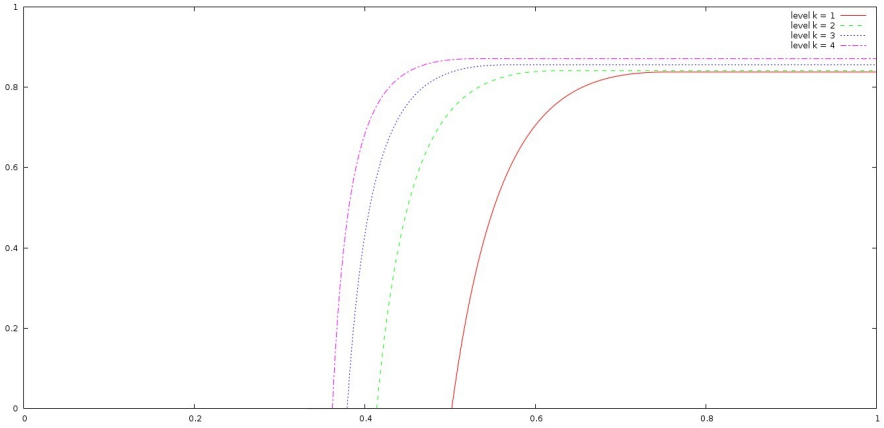

Fig. 1. Worst case profiles of $h(p)$ (i.e., distances to facilities) for $k = 1, 2, 3, 4$ obtained from solution of the LP in section 6.3. X-axis is volume of a considered set and y-axis represents distance to the farthest facility in that set. Values of function $h(p)$ are in one-to-one correspondence with values of c_i in LP from section 6.3.

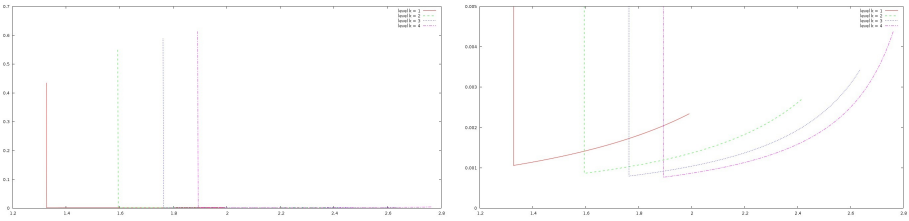

Fig. 2. Probabilities of using a particular γ in a randomized alg. (from the dual LP) for $k = 1, 2, 3, 4$. Left figure: general view; Right figure: close-up on small probabilities.

References

1. Aardal, K., Chudak, F., Shmoys, D.: A 3-Approximation Algorithm for the k-Level Uncapacitated Facility Location Problem. Inf. Process. Lett. 72(5-6), 161–167 (1999)
2. Ageev, A., Ye, Y., Zhang, J.: Improved Combinatorial Approximation Algorithms for the k-Level Facility Location Problem. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 145–156. Springer, Heidelberg (2003)

3. Asadi, M., Niknafs, A., Ghodsi, M.: An Approximation Algorithm for the k-level Uncapacitated Facility Location Problem with Penalties. In: Sarbazi-Azad, H., Parhami, B., Miremadi, S.-G., Hessabi, S. (eds.) CSICC 2008. CCIS, vol. 6, pp. 41–49. Springer, Heidelberg (2009)
4. Byrka, J., Aardal, K.: An Optimal Bifactor Approximation Algorithm for the Metric Uncapacitated Facility Location Problem. SIAM J. Comput. 39(6), 2212–2231 (2010)
5. Byrka, J., Ghodsi, M., Srinivasan, A.: LP-rounding algorithms for facility-location problems. CoRR abs/1007.3611 (2010)
6. Byrka, J., Li, S., Rybicki, B.: Improved approximation algorithm for k-level UFL with penalties, a simplistic view on randomizing the scaling parameter (full version, arxiv.org)
7. Byrka, J., Rybicki, B.: Improved LP-Rounding Approximation Algorithm for k-level Uncapacitated Facility Location. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part I. LNCS, vol. 7391, pp. 157–169. Springer, Heidelberg (2012)
8. Charikar, M., Khuller, S., Mount, D., Narasimhan, G.: Algorithms for facility location problems with outliers. In: SODA, pp. 642–651 (2001)
9. Chudak, F., Shmoys, D.: Improved Approximation Algorithms for the Uncapacitated Facility Location Problem. SIAM J. Comput. 33(1), 1–25 (2003)
10. Garg, N., Konjevod, G., Ravi, R.: A polylogarithmic approximation algorithm for the group Steiner tree problem. In: SODA, pp. 253–259 (1998)
11. Geunes, J., Levi, R., Romeijn, H., Shmoys, D.: Approximation algorithms for supply chain planning and logistics problems with market choice. Math. Program. 130(1), 85–106 (2011)
12. Guha, S., Khuller, S.: Greedy Strikes Back: Improved Facility Location Algorithms. J. Algorithms 31(1), 228–248 (1999)
13. Jain, K., Mahdian, M., Markakis, E., Saberi, A., Vazirani, V.: Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. J. ACM 50(6), 795–824 (2003)
14. Krishnaswamy, R., Sviridenko, M.: Inapproximability of the multi-level uncapacitated facility location problem. In: SODA 2012, pp. 718–734 (2012)
15. Li, S.: A 1.488 Approximation Algorithm for the Uncapacitated Facility Location Problem. Inf. Comput. 222, 45–58 (2013)
16. Li, Y., Du, D., Xiu, N., Xu, D.: Improved approximation algorithms for the facility location problems with linear/submodular penalty. In: Du, D.-Z., Zhang, G. (eds.) COCOON 2013. LNCS, vol. 7936, pp. 292–303. Springer, Heidelberg (2013)
17. Shmoys, D., Tardos, E., Aardal, K.: Approximation algorithms for facility location problems (extended abstract). In: STOC 1997, pp. 265–274 (1997)
18. Sviridenko, M.I.: An Improved Approximation Algorithm for the Metric Uncapacitated Facility Location Problem. In: Cook, W.J., Schulz, A.S. (eds.) IPCO 2002. LNCS, vol. 2337, pp. 240–257. Springer, Heidelberg (2002)
19. Xu, G., Xu, J.: An LP rounding algorithm for approximating uncapacitated facility location problem with penalties. Inf. Process. Lett. 94(3), 119–123 (2005)
20. Xu, G., Xu, J.: An improved approximation algorithm for uncapacitated facility location problem with penalties. J. Comb. Optim. 17(4), 424–436 (2009)
21. Zhang, J.: Approximating the two-level facility location problem via a quasi-greedy approach. Math. Program. 108(1), 159–176 (2006)

Inapproximability Results for Graph Convexity Parameters

Erika M.M. Coelho[1], Mitre C. Dourado[2], and Rudini M. Sampaio[3]

[1] Universidade Federal de Goiás, Goiânia, Brazil
erikamorais@inf.ufg.br
[2] Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil
mitre@dcc.ufrj.br
[3] Universidade Federal do Ceará, Fortaleza, Brazil
rudini@ufc.br

Abstract. In this paper, we prove several inapproximability results on the P_3-convexity and the geodetic convexity on graphs. We prove that determining the P_3-hull number and the geodetic hull number are APX-hard problems. We prove that the Carathéodory number, the Radon number and the convexity number of both convexities are $O(n^{1-\varepsilon})$-inapproximable in polynomial time for every $\varepsilon > 0$, unless P=NP. We also prove that the interval numbers of both convexities are $W[2]$-hard and $O(\log n)$-inapproximable in polynomial time, unless P=NP. Moreover, these results hold for bipartite graphs in the P_3-convexity.

Keywords: P_3-convexity, geodetic convexity, APX-hardness, inapproximability results, hull number, Carathéodory number, Radon number, convexity number, interval number.

1 Introduction

Convexity spaces form a classical topic, studied in some different branches of mathematics. The study of convexities applied to graphs has started later, about 50 years ago. Then the convexity parameters motivated the definition of some graph parameters, whose study has been one of the central issues in graph convexities. In particular, complexity aspects related to the computation of these parameters has been the main goal of various recent papers.

Let G be a simple finite graph, with vertex set $V(G)$, a *graph convexity* on $V(G)$ is a collection $\mathcal{C}$ of subsets of $V(G)$ such that

- $\emptyset, V(G) \in \mathcal{C}$ and
- $\mathcal{C}$ is closed under intersections.

The subsets $C \in \mathcal{C}$ are called *convex sets*. The *convex hull* of a subset $S \subset V(G)$ is the smallest set $hull_{\mathcal{C}}(S)$ in $\mathcal{C}$ containing S. If $hull(S) = V(G)$, we say that S is a *hull set*.

Next, we describe some graph parameters related to a graph convexity. The *hull number* $hn(G)$ of G is the size of a minimum hull set. The *convexity number* $cx(G)$ is the size of the maximum convex set distinct from $V(G)$.

C. Kaklamanis and K. Pruhs (Eds.): WAOA 2013, LNCS 8447, pp. 97–107, 2014.

The *Carathéodory number* $cth(G)$ is the smallest integer c such that for every set S and every vertex $u \in hull(S)$, there is a set $F \subseteq S$ with $|F| \leq c$ and $u \in hull(F)$. A set S of vertices of G is a *Carathéodory set* of $\mathcal{C}$ if the set $\partial hull(S)$ defined as $hull(S) \setminus \bigcup_{u \in S} hull(S \setminus \{u\})$ is not empty. This notion allows an alternative definition of the *Carathéodory number* of $\mathcal{C}$ as the largest cardinality of a Carathéodory set of $\mathcal{C}$.

The *Radon number* $rd(G)$ is the minimum k such that every subset V' of $V(G)$ of size at least k has a Radon partition, which is a partition (V_1', V_2') such that $hull(V_1') \cap hull(V_2') \neq \emptyset$. Alternatively, $rd(G)$ is the size of a maximum anti-Radon set plus one, where a set is anti-Radon if it has no Radon partition.

Clearly, the computation of these parameters for a graph would depend on the particular convexity being considered. Several well known graph convexities $\mathcal{C}$ are defined using some set $\mathcal{P}$ of paths of the underlying graph G. In this case, a subset S of vertices of G is convex, that is, belongs to $\mathcal{C}$, if for every path P in $\mathcal{P}$ whose endvertices belong to S also every vertex of P belongs to S. When $\mathcal{P}$ is the set of all shortest paths in G, this leads to the *geodetic convexity* [7,15,16,22,24]. The *monophonic convexity* is defined by considering as $\mathcal{P}$ the set of all induced paths of G [17,20,23]. The set of all paths of G leads to the *all path convexity* [12]. If $\mathcal{P}$ is the set of all induced paths of length ≥ 3 leads to the m^3 *convexity* [13]. Similarly, if $\mathcal{P}$ is the set of all triangle paths in G, then $\mathcal{C}$ is the *triangle path convexity* [11]. When $\mathcal{P}$ is the set of all paths of length two we have the P_3 *convexity*. The P_3 convexity was first considered for directed graphs [21,25]. For undirected graphs, the P_3 convexity was studied in [5,8,9].

The P_3*-interval* of a set $S \subseteq V(G)$ is S plus every vertex outside S belonging to some path $\mathcal{P}$ between two pairs of vertices in S. The *interval number* $in(G)$ of a graph G is the minimum cardinality of a set $S \subseteq V(G)$ such that $I(S) = V(G)$.

Regarding the P_3-convexity, it was proved that the following parameters are NP-hard: hull number, Radon number, convexity number, interval number and Carathéodory number [5,6,8,10]. Regarding the geodetic convexity, it was proved that the same parameters are also NP-hard [1,14,15,18].

In this paper, we improve these results by proving the following for the P_3-convexity and the geodetic convexity:

- The hull number is APX-hard;
- The Carathéodory number, the Radon number and the convexity number are $O(n^{1-\varepsilon})$-inapproximable in polynomial time for any $\varepsilon > 0$, unless P=NP;
- The interval number is $O(\log n)$-inapproximable in polynomial time, unless P=NP.

Moreover, these results hold for bipartite graphs in the P_3-convexity.

In the following sections, we follow the terminology in [3] for the terms: *AP-reduction*, *L-reduction* and *performance ratio* (which are also defined in the appendix). We also use the following notation: given an optimization problem P, let $opt_P(I)$ denote the optimal solution value for some instance I of P and, for a solution S of I, let $val_P(I, S)$ denote the associated value.

Finally, in this work we present only the sketch of the proofs due to limitation of the number of pages.

2 The P_3-hull Number Is APX-hard

In this section, we prove the following theorem.

Theorem 1. *The P_3-hull number is APX-hard even on bipartite graphs.*

Proof (Sketch of the proof). We obtain an L-reduction from Max-2-Sat-3, which is MaxSat with the following restrictions: (a) every clause has at most 2 literals and, (b) for every variable x_i, there are at most three clauses containing either x_i or $\overline{x_i}$. It is known that Max-2-Sat-3 is APX-Complete [3].

Given a Max-2-Sat-3 instance with k variables and m clauses, we construct a bipartite graph G with at most $22k + 8m$ vertices. Figure 1 shows an example with three clauses $C_1 = (x_1 \vee x_2)$, $C_2 = (x_1 \vee \overline{x_2})$ and $C_3 = (\overline{x_1} \vee x_2)$. The variable gadgets (indicated by the dashed squares), the clause gadgets (indicated by C_i) and their connections (indicated by the dashed edges) can be seen in this example.

Every gadget of a variable x_i has vertices $y_i, \overline{y_i}, z_i, \overline{z_i}$. If x_i (resp. $\overline{x_i}$) is in only one clause, introduce a new vertex on G and connect it to y_i (resp. $\overline{y_i}$). Notice that G has $\gamma(\epsilon) + 3m$ vertices of degree 1, where $\gamma(\epsilon) \leq 7$.

Observe that each dashed square represents a co-convex set (that is, all vertices except the vertices inside the dashed square form a convex set). Consequently, every hull set must contain at least one vertex of each variable gadget.

It is possible to prove that, for every hull set S', we can obtain a *good hull set* S with $|S| \leq |S'|$, where a *good hull set* has exactly one vertex in each dashed square (variable gadget) which is either z_i or $\overline{z_i}$, and at most one vertex in each clause gadget. By the definition, a good hull set has exactly $\gamma(\epsilon)+k+3m+X(S)$ vertices, where $X(S)$ is the number of vertices of S inside a clause gadget.

Given a good hull set S, we can obtain an assignment $g(S)$ for the variables (let x_i be true, if S contains z_i, and let x_i be false, otherwise). It is possible to prove that the assignment $g(S)$ satisfies $m - X(S)$ clauses.

By similar arguments, it is possible to prove that, for any assignment satisfying ℓ clauses, we can obtain a good hull set S with $|S| = \gamma(\epsilon) + k + 3m + (m - \ell)$ vertices. If ℓ_{opt} is the optimum value, then it is known that $\ell_{opt} \geq m/2$. This will imply that there exists a good hull set S_{opt} with $|S_{opt}| \leq 35\ell_{opt}$ vertices, since $k \leq 2m$.

Moreover, $|S|-|S_{opt}| = (\gamma(\epsilon)+k+3m+X(S))-(\gamma(\epsilon)+k+3m+(m-\ell_{opt})) = \ell_{opt}-(m-X(S))$. Recall that $g(S_{opt})$ and $g(S)$ satisfy ℓ_{opt} and $m-X(S)$ clauses, respectively. With this, we conclude that this reduction is an L-reduction with $\alpha = 35$ and $\beta = 1$.

Open Problem: Is the P_3-hull number in APX? That is, there exists a polynomial time r-approximation algorithm for some fixed r?

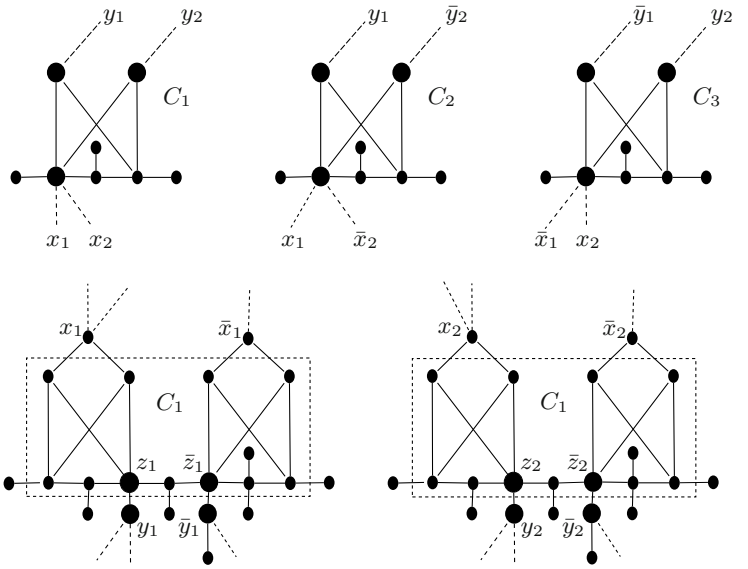

Fig. 1. Reduction of MAX-2-SAT-3 to the P_3-hull number

3 Inapproximability of the P_3-Radon Number and the P_3-convexity Number

In this section, we prove the following theorem.

Theorem 2. *The P_3-convexity number and the P_3-Radon number are $O(n^{1-\varepsilon})$-inapproximable in polynomial time even on bipartite graphs, unless P=NP.*

Proof (Sketch of the proof). We obtain an AP-reduction from the Set Packing Problem. Given a family $\mathcal{S} = \{S_1, \ldots, S_m\}$ of finite sets, the objective is to determine the size of a largest set packing of $\mathcal{S}$, which is a family of mutually disjoint sets of $\mathcal{S}$. We will consider instances $\mathcal{S}$ such that $opt(\mathcal{S}) \geq 3$. Given an instance of Set Packing, we construct a bipartite graph G. Figure 2 shows an example of the construction, where $S_1 = \{a, b, c\}$, $S_2 = \{b, c, d, f, g\}$, $S_3 = \{d, e, f\}$, $S_4 = \{c, e, f, g\}$ and $S_5 = \{g, h, i\}$.

Given a set packing of $\mathcal{S}$, it is possible to prove that the vertices of G associated to the subsets in the packing and the auxiliary vertex W form a P_3-convex set in G (since the neighborhood of these vertices has no intersection). In Figure 2, the P_3-convex set is $C = \{S_1, S_3, S_5, W\}$.

On the other way, given a P_3-convex set C of G with at least 3 vertices, it is possible to prove that all vertices of $C \setminus \{W\}$ are associated with subsets without intersection between them, forming a set packing of $\mathcal{S}$. Let $g(\mathcal{S}, C)$ be the sets of $\mathcal{S}$ associated to the vertices in $C \setminus \{W\}$.

Given an instance $\mathcal{S}$ of Set Packing, let G be the bipartite graph obtained by the reduction above. Given a P_3-convex set C, let $\mathcal{S}' = g(\mathcal{S}, C)$. To simplify the notation, let P_1 and P_2 be the Maximum Set Packing Problem and the Maximum P_3-convexity problem, respectively. Then, from the two paragraphs above, $opt_{P_1}(\mathcal{S}) = opt_{P_2}(G) - 1$ and $val_{P_1}(\mathcal{S}, \mathcal{S}') = val_{P_2}(G, C) - 1$.

With this, we can conclude that this is an AP-reduction with $\gamma = 2$.

It is possible to prove that, given an anti-Radon set R' with at least 3 vertices, we can obtain a P_3-convex set R with $|R| = |R'|$. Then the result for the P_3-Radon number follows.

It is known that, unless P=NP, there can be no polynomial time algorithm that approximates the maximum clique to within a factor better than $O(n^{1-\varepsilon})$, for any $\varepsilon > 0$ [27]. Since Set-Packing is as hard to approximate as the Maximum Clique Problem [4], we are done.

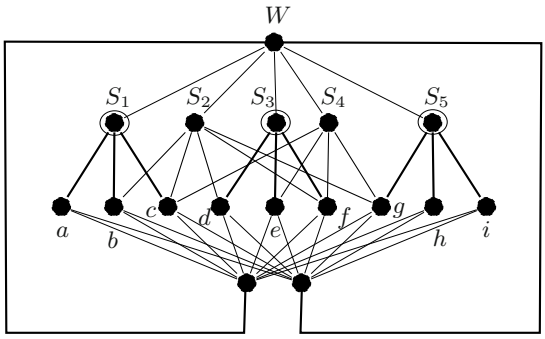

Fig. 2. Reduction of Set Packing to the P_3-convexity number

4 Inapproximability of the P_3-interval Number

In this section, we prove the following theorem.

Theorem 3. *The P_3-interval number is $O(\log n)$-inapproximable in polynomial time, even on bipartite graphs, unless P=NP.*

Proof (Sketch of the proof). We obtain an AP-reduction from the Set Cover Problem. Given a set $U = \{u_1, \ldots, u_n\}$ and family $\mathcal{S} = \{S_1, \ldots, S_m\}$ of subsets of U, the objective is to determine the size of a minimum set cover of U, which is a subfamily of subsets in $\mathcal{S}$ whose union contains all elements in U. We will consider instances $\mathcal{S}$ such that $opt(\mathcal{S}) \geq 5$. Given an instance $\mathcal{S}$ of Set Cover, we construct a bipartite graph G with vertex set $U \cup \mathcal{S} \cup \{x_1, x_2, x_3, x_4, y_1, y_2, y_3\}$, where each x_i and each y_j is a new vertex. Figure 3 shows an example of the reduction, where $U = \{a, b, c, d, e\}$ and $\mathcal{S} = \{S_1, S_,S_3, S_4\}$, where $S_1 = \{a, b, c\}$, $S_2 = \{a, b\}$, $S_3 = \{c, d\}$ and $S_4 = \{c, d, e\}$.

Given a set cover of U, it is possible to prove that the vertices of G associated to the subsets in the cover and the auxiliary vertices x_1, x_2, x_3, x_4 form a P_3-interval set in G (since all vertices of $\mathcal{S}$ are adjacent to x_3 and x_4, and all vertices of U are adjacent to x_2 and to a vertex in the cover). In Figure 3, the P_3-interval set is $C = \{S_1, S_4, x_1, x_2, x_3, x_4\}$.

On the other way, given a P_3-interval set C' of G with at least 5 vertices, it is possible to prove that we can easily obtain a *good* P_3-interval set C with $|C| \leq |C'|$, which is a set C that contains $\{x_1, x_2, x_3, x_4\}$ and $C \setminus \{x_1, x_2, x_3, x_4\} \subseteq \mathcal{S}$. Let $g(\mathcal{S}, C)$ be the sets of $\mathcal{S}$ associated to the vertices in $C \setminus \{x_1, x_2, x_3, x_4\}$.

Given an instance $\mathcal{S}$ of Set Cover, let G be the bipartite graph obtained by the reduction above. Given a good P_3-interval set C, let $\mathcal{S}' = g(\mathcal{S}, C)$. To simplify the notation, let P_1 and P_2 be the Minimum Set Cover Problem and the Minimum P_3-interval problem, respectively. Then, from the two paragraphs above, $opt_{P_1}(\mathcal{S}) = opt_{P_2}(G) - 4$ and $val_{P_1}(\mathcal{S}, \mathcal{S}') = val_{P_2}(G, C) - 4$.

With this, we can conclude that this is an AP-reduction with $\gamma = 2$. Under the assumption that $P \neq NP$, Raz and Safra [26] showed that Set Cover is $O(\log n)$-inapproximable in polynomial time. This holds even for instances in which the family $\mathcal{S}$ has size polynomial in $|U|$.

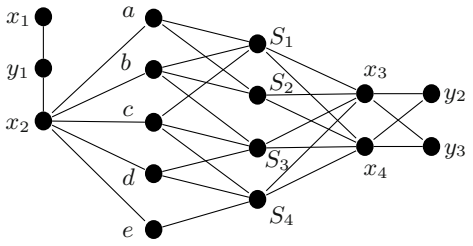

Fig. 3. Reduction of Set Cover to the P_3-interval number

Now we turn to the parameterized complexity of the P_3-interval number. The parameterized P_3-interval number problem asks whether the corresponding instance of the P_3-interval number problem with a given parameter p have a P_3-interval set with p vertices. To show such results, we shall consider the *Parameterized Set Cover* (PSC) problem. In this decision problem, the instance consists of a pair $(U, \mathcal{S})$ as defined above, and a parameter p (a positive integer). The question is whether there is a set cover of U with cardinality p. Given an instance of PSC (Parameterized Set Cover) $((U, \mathcal{S}), p)$, we construct an instance $(G, p + 4)$ of the Parameterized P_3-interval number problem, exactly as in the reduction in the proof of Theorem 3. This defines an FPT-reduction from PSC to the Parameterized P_3-interval number problem. Hence, using the fact the PSC is $W[2]$-complete [19], we may state the following result.

Theorem 4. *The Parameterized P_3-interval number problem is $W[2]$-hard.*

5 Inapproximability of the P_3-Carathéodory Number

In this section, we prove that the P_3-Carathéodory number is $O(n^{1-\varepsilon})$-inapproximable in polynomial time for any $\varepsilon > 0$, even on bipartite graphs, unless P=NP. For this, we will make a AP-reduction from the problem MAX3SAT-INTERVAL to the problem P_3-CARATHÉODORY NUMBER, defined below:

Problem 5 *(*MAX3SAT-INTERVAL*) Given a 3-SAT instance $(C_1, C_2, \ldots C_m)$, to obtain the largest k such that the clauses $(C_i, C_{i+1}, \ldots, C_{i+k-1})$ are satisfied, except for at most one clause.*

Problem 6 *(P_3-*CARATHÉODORY NUMBER*) Given a G, to determine the largest k such that G has a Carathéodory set of order k.*

Theorem 7. *There is no $O(n^{1-\varepsilon})$-approximable polinomial algorithm for* MAX-3SAT-INTERVAL*, for any $\varepsilon > 0$, unless P=NP.*

Proof. By contradiction, suppose that there exists a $(cn^{1-\varepsilon})$-approximable polynomial algorithm A, where $c > 0$ and $\varepsilon > 0$ are constants and n is the number of clauses. Given an instance $\phi = (C_1, C_2, \ldots, C_m)$ of 3-SAT, generate a sequence with $(2cm)^{\frac{1}{\varepsilon}}$ clauses, as follows:

$$\Psi = (C_1, C_1, C_2, C_2, \ldots, C_m, C_m, C_1, C_1, C_2, C_2, \ldots, C_m, C_m, \ldots).$$

Consider Ψ as an instance of MAX3SAT-INTERVAL. If ϕ is satisfied, then is possible to satisfy all $(2cm)^{\frac{1}{\varepsilon}}$ clauses of Ψ. Otherwise, is possible to satisfy less then $2m$ consecutive clauses of Ψ.

In the first case, the algorithm A obtain a solution with $(2cm)^{\frac{1}{\varepsilon}}/(factor)$ consecutive clauses, except for at most one clause, where factor=$cn^{1-\varepsilon}$ and n is the number of clauses. So, $n = (2cm)^{\frac{1}{\varepsilon}}$ and A obtain $2m$ consecutive clauses, except for at most one clause.

In the second case, it is easy to see that the algorithm A achieves less than $2m$ consecutive clauses, except for at most one clause. This is a contradiction, unless P=NP, therefore the algorithm would decide the 3-SAT problem.

Theorem 8. MAX3SAT-INTERVAL $\leq_{AP}$ P_3-CARATHÉODORY NUMBER.

Proof (Sketch of the proof). We obtain an AP-reduction from MAX3SAT-INTERVAL. Let a given MAX3SAT-interval instance $\mathcal{I} = (C_1, C_2, \ldots, C_m)$. We construct a bipartite graph G as follows:

- Add a caterpillar graph F generated as follows:
  - Add the path $P_{2m+5} = u_0, v_0, u_1, v_1, \ldots, u_m, v_m, u_{m+1}, v_{m+1}, z$.
  - Add a leaf l_i for every vertex v_i of the P_{2m+5} path, for $i = 0, 1, \ldots, m+1$.
- For every clause C_i of $\mathcal{I}$, add three vertices $y_{i,1}$, $y_{i,2}$, and $y_{i,3}$ and two further vertices w_i and w'_i. Bijectively associate the three vertices $y_{i,1}$, $y_{i,2}$, and $y_{i,3}$ to the three literals in C_i. Let $Y_i = \{y_{i,1}, y_{i,2}, y_{i,3}\}$. Add all possible 9 edges between the 3 vertices in Y_i and the 3 vertices in $\{u_i, w_i, w'_i\}$. Let $Y = \bigcup_{i=1}^{m} Y_i$ and $W_1 = \{w_i \mid 1 \leq i \leq m\}$.

- Add three new vertices $y_{m+1,1}$, $y_{m+1,2}$, and $y_{m+1,3}$ and two further vertices w_{m+1} and w'_{m+1}. Let $Y_{m+1} = \{y_{m+1,1}, y_{m+2,2}, y_{m+3,3}\}$. Add all possible 9 edges between the 3 vertices in Y_{m+1} and the 3 vertices in $\{u_{m+1}, w_{m+1}, w'_{m+1}\}$.
- Let $Y = \bigcup_{i=1}^{m+1} Y_i$ and $W_1 = \{w_i \mid 1 \leq i \leq m+1\}$.
- For every two vertices $y_{i,j}$ and $y_{m+1,j'}$, where $i = 1, 2, \ldots, m$, add 2 vertices w and w' and add all possible 4 edges between the 2 vertices $y_{i,j}, y_{m+1,j'}$ and the 2 vertices in $\{w, w'\}$. Denote the set of all these vertices by W_2.
- For every two vertices $y_{i,j}$ and $y_{i',j'}$ with $i \neq i'$ such that $y_{i,j}$ and $y_{i',j'}$ are associated to opposite literals x and $\bar{x}$, add 2 vertices w and w' and add all possible 4 edges between the 2 vertices $y_{i,j}, y_{i',j'}$ and the 2 vertices in $\{w, w'\}$. Denote the set of all these vertices by W_3. Let $W = W_1 \cup W_2 \cup W_3$.
- Add two vertices a and b adjacent to all vertices of W.

This completes the construction of G. See Figure 4 for an illustration. Consider the following bipartition of the vertices of G. The bipartition (B_1, B_2) shows that G is a bipartite graph, where $B_1 = \{u_0, u_1, \ldots, u_{m+1}, l_0, l_1, \ldots, l_{m+1}, z\} \cup W$ and $B_2 = \{v_0, v_1, \ldots, v_{m+1}, a, b\} \cup Y$. Furthermore, the vertices in $V(F) \cup Y$ induce a caterpillar C in G whose set of leaves is $Y \cup \{u_0, z, l_0, \ldots, l_m\}$.

It is possible to prove that G contains a Carathéodory set of order k if and only if there exists a truth assignment for $\mathcal{I}$ such that there is a sequence of $\lceil \frac{k-3}{2} \rceil$ clauses with at most one clause not satisfied.

Corollary 1. *For any $\varepsilon > 0$, the P_3-*Carathéodory number *is $O(n^{1-\varepsilon})$-inapproximable, even on bipartite graphs, unless $P = NP$.*

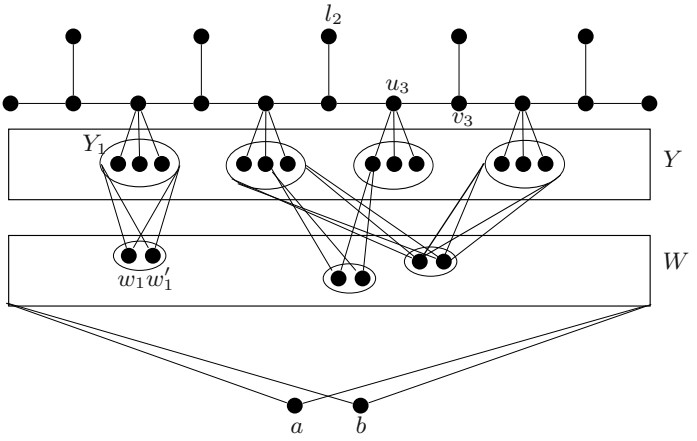

Fig. 4. An illustration of the construction of G for a MAX3SAT-interval instance with 3 clauses, where the second literal of the second clause is the negation of the first literal of the third clause. Note that not all vertices are shown.

6 Inapproximability of Geodetic Convexity Parameters

Given graphs G_1 and G_2, the disjoint union $G_1 \cup G_2$ is the graph obtained from the union of the vertex sets and the edge sets, and the join $G_1 + G_2$ is the graph obtained from $G_1 \cup G_2$ including all edges between G_1 and G_2. The following theorem in [2] shows an important relation between the geodetic convexity and the P_3-convexity. Let K_m be a clique with m vertices.

Theorem 9. *Let G_1 be a triangle free graph with at least three vertices. Then*

$$\begin{aligned}
&(i)\ hn_{gd}(G_1 + K_m) &&= hn_{P_3}(G_1),\\
&(ii)\ in_{gd}(G_1 + K_m) &&= in_{P_3}(G_1),\\
&(iii)\ cx_{gd}(G_1 + K_m) &&= cx_{P_3}(G_1) + m,\\
&(iv)\ cth_{gd}(G_1 + K_m) &&= \max\{cth_{P_3}(G_1), 2\}.
\end{aligned}$$

This theorem implies directly the following inapproximability results on the geodetic convexity.

Corollary 2. *Concerning the geodetic convexity, we have the following inapproximability results:*

- *The hull number is APX-hard;*
- *The Carathéodory number and the convexity number are $O(n^{1-\varepsilon})$-inapproximable in polynomial time for any $\varepsilon > 0$, unless P=NP;*
- *The interval number is $W[2]$-hard and $O(\log n)$-inapproximable in polynomial time, unless P=NP.*

The following theorem proves the inapproximability of the geodetic Radon number.

Theorem 10. *For every $\varepsilon > 0$, approximating the geodetic Radon number to within a factor $n^{1-\varepsilon}$ is NP-hard.*

Proof (Sketch of the proof). We obtain an AP-reduction from the Maximum Clique problem to the Maximum Anti-Radon Set problem, which is the problem of determining the size of a maximum anti-Radon set of a given graph plus one.

Let a graph G be an input instance of Maximum Clique. Let $G' = f(G)$ be the graph such that $V(G') = V(G) \cup \{x, y\}$, where x and y are new vertices, and $E(G') = E(G) \cup \{vx, vy :\ v \in V(G)\}$.

Given a feasible solution R of G' (that is, R is an anti-Radon set of G'), let $C = g(G, R) = R \setminus \{x, y\}$. Notice that, if R has two non-adjacent vertices $\{u, w\}$, then the partition $(\{u, w\}, R \setminus \{u, w\})$ of R is a Radon partition, since $x, y \in hull(\{u, w\})$ and $V(G') \subseteq hull(\{x, y\})$. Consequently, R is a clique of G' and we can assume that R contains either x or y. Thus C is a clique of G. Moreover, $|C| = |R| - 1$. Recall that $val_{AntiRadon}(G, R) = |R| + 1$.

Furthermore, since every clique of G is an anti-Radon set of G', this implies that $\omega(G) \leq rd(G') - 2 \leq 2rd(G')$. Moreover, $\omega(G) - |C| = (rd(G') - 2) - (|R| - 1) = rd(G') - (|R| + 1)$. With this, we can conclude that (f, g) is an AP-reduction with $\gamma = 2$.

7 Appendix: Approximation Preserving Reductions

Given an optimization problem P, let $opt_P(I)$ denote the optimal solution value for some instance I of P and, for a solution S of I, let $val_P(I,S)$ denote the associated value. Given an instance I of P and a solution S of I, the *performance ratio* $\mathcal{R}_P(I,S)$ is defined by

$$\mathcal{R}_P(I,S) \;=\; \max\left\{\frac{opt_P(I)}{val_P(I,S)}, \frac{val_P(I,S)}{opt_P(I)}\right\}.$$

Given a constant $r \geq 1$, an *r-approximation algorithm* for P is an algorithm that, applied to any instance I of P, runs in time polynomial in the size of I and produces a solution S such that $\mathcal{R}(I,S) \leq r$. If such an algorithm exists, P belongs to APX.

A reduction from P_1 to P_2 consists of a pair (f,g) of polynomial-time computable functions such that, for any instance I of P_1, (a) $f(I)$ is an instance of P_2, and (b) $g(I,S)$ is a feasible solution of I, for any feasible solution S for $f(I)$.

We say that P_1 is AP-reducible to P_2 (denoted by $P_1 \leq_{AP} P_2$) if there exists a 3-tuple (f,g,γ), where (f,g) is a reduction from P_1 to P_2 and γ is a positive constant such that, if $\mathcal{R}_{P_2}(f(I),S) \leq r$, then $\mathcal{R}_{P_1}(I,g(S)) \leq 1+\gamma(r-1)$ for each instance I of P_1 and for every feasible solution S for $f(I)$.

We say that a problem P is *APX-hard* if $Q \leq_{AP} P$ for any APX problem Q. Roughly speaking, the existence of a $r-$approximation algorithm for P for any $r > 1$ would imply the existence of a $s-$approximation algorithm for any $s > 1$ for all problems in APX. A problem is *APX-complete* if it is APX and APX-hard.

The type of reduction used most frequently to prove APX-hardness is the L-reduction [3]. We say that P_1 is L-reducible to P_2 (denoted by $P_1 \leq_L P_2$) if there exists a 4-tuple (f,g,α,β), where (f,g) is a reduction from P_1 to P_2 and α and β are positive constants, such that, for any instance I of P_1: (a) $|opt_{P_2}(f(I))| \leq \alpha|opt_{P_1}(I)|$, and (b)$|opt_{P_1}(I) - val_{P_1}(I,g(I,S))| \leq \beta|opt_{P_2}(f(I)) - val_{P_2}(f(I),S)|$ for any feasible solution S for $f(I)$.

It is known [3] that, if $P_1 \leq_L P_2$ and P_1 is APX, then $P_1 \leq_{AP} P_2$.

References

1. Araújo, J., Campos, V., Giroire, F., Sampaio, L., Soares, R.: On the hull number of some graph classes. Elec. Notes Disc. Math. 38, 49–55 (2011)
2. Araújo, R.T., Sampaio, R.M., Szwarcfiter, J.L.: The convexity of induced paths of order three. Electronic Notes in Discrete Mathematics (2013) (to appear)
3. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Combinatorial Optimization Problems and Their Approximability Properties. Springe, Berlin (1999)
4. Ausiello, G., D'Atri, A., Protasi, M.: Structure preserving reductions among convex optimization problems. J. Comput. System Sci. 21, 136–153 (1980)
5. Barbosa, R.M., Coelho, E.M.M., Dourado, M.C., Rautenbach, D., Szwarcfiter, J.L.: On the Carathéodory number for the convexity of paths of order three. SIAM J. Discrete Math. 26, 929–939 (2012)

6. Dourado, M.C., Rautenbach, D., dos Santos, V.F., Schäfer, P.M., Szwarcfiter, J.L., Toman, A.: Algorithmic and structural aspects of the P_3-Radon number. Ann. Oper. Res. 206, 75–91 (2013)
7. Cáceres, J., Hernando, C., Mora, M., Pelayo, I.M., Puertas, M.L., Seara, C.: On geodetic sets formed by boundary vertices. Discrete Math. 306, 188–198 (2006)
8. Centeno, C., Dourado, M., Penso, L., Rautenbach, D., Szwarcfiter, J.L.: Irreversible conversion of graphs. Theoretical Computer Science 412, 3693–3700 (2011)
9. Centeno, C.C., Dantas, S., Dourado, M.C., Rautenbach, D., Szwarcfiter, J.L.: Convex Partitions of Graphs induced by Paths of Order Three. Discrete Mathematics and Theoretical Computer Science 12, 175–184 (2010)
10. Centeno, C.C., Dourado, Szwarcfiter, J.L.: On the convexity of Paths of length two in undirected graphs. Elect. Notes in Disc. Math. 32, 11–18 (2009)
11. Changat, M., Mathew, J.: On triangle path convexity in graphs. Discrete Mathematics 206, 91–95 (1999)
12. Changat, M., Klavžar, S., Mulder, H.M.: The all-paths transit function of a graph. Czech. Math. J. 51 (126), 439–448 (2001)
13. Deagan, F.F., Nicolai, F., Bransdstädt, A.: Convexity and HHD-graphs. SIAM J. Discrete Mathematics 12, 119–135 (1999)
14. Dourado, M.C., Protti, F., Rautenbach, D., Szwarcfiter, J.L.: On the convexity number of graphs. Graphs and Combinatorics 28, 333–345 (2012)
15. Dourado, M., Protti, F., Rautenbach, D., Szwarcfiter, J.L.: Some remarks on the geodetic number of a graph. Discrete Mathematics 310, 832–837 (2012)
16. Dourado, M.C., Protti, F., Rautenbach, D., Szwarcfiter, J.L.: On the hull number of triangle-free graphs. SIAM J. Discrete Math. 23, 2163–2172 (2010)
17. Dourado, M.C., Protti, F., Szwarcfiter, J.L.: Complexity results related to monophonic convexity. Discrete Appl. Math. 158, 1269–1274 (2010)
18. Dourado, M.C., Rautenbach, D., dos Santos, V.F., Schäfer, P.M., Szwarcfiter, J.L.: On the Carathéodory number of interval and graph convexities (to appear)
19. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, New York (1999)
20. Duchet, P.: Convex sets in graphs II: Minimal path convexity. J. Combin. Theory, Ser. B 44, 307–316 (1988)
21. Erdős, P., Fried, E., Hajnal, A., Milner, E.C.: Some remarks on simple tournaments. Algebra Univers. 2, 238–245 (1972)
22. Everett, M.G., Seidman, S.B.: The hull number of a graph. Discrete Math. 57, 217–223 (1985)
23. Farber, M., Jamison, R.E.: Convexity in graphs and hypergraphs. SIAM J. Algebraic Discrete Methods 7, 433–444 (1986)
24. Farber, M., Jamison, R.E.: On local convexity in graphs. Discrete Math. 66, 231–247 (1987)
25. Parker, D.B., Westhoff, R.F., Wolf, M.J.: On two-path convexity in multipartite tournaments. European J. Combin. 29, 641–651 (2008)
26. Raz, R., Safra, S.: A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In: Proc. of the 29th Annual ACM Symposium on Theory of Computing, pp. 475–484 (1987)
27. Zuckerman, D.: Linear degree extractors and the inapproximability of max clique and chromatic number. In: Proc. 38th ACM Symp. Theory of Computing (STOC 2006), pp. 681–690 (2006)

Continuum Armed Bandit Problem of Few Variables in High Dimensions

Hemant Tyagi and Bernd Gärtner

Institute of Theoretical Computer Science,
ETH Zürich (ETHZ), CH-8092 Zürich, Switzerland
{htyagi,gaertner}@inf.ethz.ch

Abstract. We consider the stochastic and adversarial settings of continuum armed bandits where the arms are indexed by $[0,1]^d$. The reward functions $r : [0,1]^d \to \mathbb{R}$ are assumed to *intrinsically* depend on at most k coordinate variables implying $r(x_1,\ldots,x_d) = g(x_{i_1},\ldots,x_{i_k})$ for distinct and unknown $i_1,\ldots,i_k \in \{1,\ldots,d\}$ and some locally Hölder continuous $g : [0,1]^k \to \mathbb{R}$ with exponent $\alpha \in (0,1]$. Firstly, assuming $(i_1,\ldots,i_k)$ to be fixed across time, we propose a simple modification of the CAB1 algorithm where we construct the discrete set of sampling points to obtain a bound of $O(n^{\frac{\alpha+k}{2\alpha+k}}(\log n)^{\frac{\alpha}{2\alpha+k}}C(k,d))$ on the regret, with $C(k,d)$ depending at most polynomially in k and sub-logarithmically in d. The construction is based on creating partitions of $\{1,\ldots,d\}$ into k disjoint subsets and is probabilistic, hence our result holds with high probability. Secondly we extend our results to also handle the more general case where $(i_1,\ldots,i_k)$ can change over time and derive regret bounds for the same.

Keywords: Bandit problems, continuum armed bandits, functions of few variables, online optimization.

1 Introduction

In online decision making problems, a player is required to play a strategy, chosen from a given set of strategies S, over a period of n trials or rounds. Each strategy has a reward associated with it specified by a reward function $r : S \to \mathbb{R}$ which typically changes across time in a manner unknown to the player. The aim of the player is to choose the strategies in a manner so as to minimize the *regret* defined as the difference between the total expected reward of the best fixed strategy (not varying with time) and the expected reward of the sequence of strategies played by the player. If the regret after n rounds is sub-linear in n, this implies as $n \to \infty$ that the per-round expected reward of the player asymptotically approaches that of the best strategy. There are many applications of online decision making problems such as routing [1,2], wireless networks [3], online auction mechanisms [4,5], statistics (sequential design of experiments [6]) and economics (pricing [7]), to name a few. An important type of online decision making problem is the *multi-armed bandit* problem, where the

C. Kaklamanis and K. Pruhs (Eds.): WAOA 2013, LNCS 8447, pp. 108–119, 2014.

player only receives the reward associated with the strategy that was played in the round[1]. These problems have been studied extensively when the strategy set S is finite and optimal regret bounds are known within a constant factor [8,9,6]. On the other hand, the setting in which S is infinite has been an area of recent attention due to its practical significance. Such problems are referred to as continuum armed bandit problems and are the focus of this paper. Usually S is considered to be a compact subset of a metric space such as $\mathbb{R}^d$. Some applications of these problems are in: (i) online auction mechanism design [4,5] where the set of feasible prices is representable as an interval and, (ii) online oblivious routing [2] where S is a flow polytope.

For a d-dimensional strategy space it is well known that any multi-armed bandit algorithm will incur worst-case regret of $\Omega(2^d)$ (see [10]). To circumvent this curse of dimensionality, additional assumptions are made on the structure of the reward functions such as linearity (see for example [11]) or convexity (see for example [10]). For these classes of reward functions the regret is typically polynomial in d and sub-linear in n. We consider the setting where the reward function $r : [0,1]^d \to \mathbb{R}$ depends on an unknown subset of k *active* coordinate variables implying $r(x_1, \ldots, x_d) = g(x_{i_1}, \ldots, x_{i_k})$. The environment is allowed to sample the underlying function g either in an i.i.d manner from some fixed underlying distribution (stochastic) or arbitrarily (adversarial). To the best of our knowledge, such a structure for reward functions has not been considered in the bandit setting previously. On the other hand, there has been significant effort in other fields to develop tractable algorithms for approximating such types of functions (cf. [12,13] and references within). Our contribution is therefore to combine ideas from different communities and apply them to the bandit setting.

The continuum armed bandit problem was first introduced in [14] for $d = 1$ in the stochastic setting where a regret bound of $o(n^{(2\alpha+1)/(3\alpha+1)+\eta})$ for any $\eta > 0$ was shown for Hölder continuous[2] reward functions with exponent $\alpha \in (0, 1]$. In [5] a lower bound of $\Omega(n^{1/2})$ was proven for this problem. This was then improved upon in [10] where upper and lower bounds of $O(n^{\frac{\alpha+1}{2\alpha+1}}(\log n)^{\frac{\alpha}{2\alpha+1}})$ and $\Omega(n^{\frac{\alpha+1}{2\alpha+1}})$ were derived for both stochastic and adversarial settings. [15] considered a class of reward functions with additional smoothness properties and derived a regret bound of $O(n^{1/2})$ which is also optimal. In [16] the case $d = 1$ was treated, with the reward function assumed to only satisfy a local Hölder condition around the maximum $\mathbf{x}^*$ with exponent $\alpha \in (0, \infty)$. Under these assumptions the authors considered a modification of Kleinberg's CAB1 algorithm [10] and achieved a regret bound of $O(n^{\frac{1+\alpha-\alpha\beta}{1+2\alpha-\alpha\beta}}(\log n)^{\frac{\alpha}{1+2\alpha-\alpha\beta}})$ for some known $0 < \beta < 1$. In [17,18] the authors studied a very general setting in which S forms a metric space, with the reward function assumed to satisfy a Lipschitz condition with respect to this metric and derived close to optimal regret bounds.

[1] Another type is the *best expert problem*, where the entire reward function is revealed to the player at the end of each round.

[2] A function $r : S \to \mathbb{R}$ is Hölder continuous if $|r(\mathbf{x}) - r(\mathbf{y})| \leq L \parallel \mathbf{x} - \mathbf{y} \parallel^{\alpha}$ for constants $L > 0$, $\alpha \in (0, 1]$ and any $\mathbf{x}, \mathbf{y} \in S$.

Our contributions are twofold. Firstly, we prove that when $(i_1, \ldots, i_k)$ is fixed across time but unknown to the player, then a simple modification of the `CAB1` algorithm can be used to achieve a regret bound [3] of $O(n^{\frac{\alpha+k}{2\alpha+k}}(\log n)^{\frac{\alpha}{2\alpha+k}}C(k,d))$ where $\alpha \in (0,1]$ denotes the exponent of Hölder continuity of the reward functions. The factor $C(k,d) = O(\text{poly}(k) * o(\log d))$ captures the uncertainty of not knowing the k active coordinates. The modification is in the manner of discretization of $[0,1]^d$ for which we consider a probabilistic construction based on creating *partitions* of $\{1,\ldots,d\}$ into k disjoint subsets. The above bound holds for both the stochastic (underlying g is sampled in an i.i.d manner) and the adversarial (underlying g chosen arbitrarily at each round) models. Secondly, we extend our results to handle the more general setting where an adversary chooses some sequence of k-tuples $(\mathbf{i}_t)_{t=1}^n = (i_{1,t}, \ldots, i_{k,t})_{t=1}^n$ before the start of plays. For this setting we derive a regret bound of $O(n^{\frac{\alpha+k}{2\alpha+k}}(\log n)^{\frac{\alpha}{2\alpha+k}}H[(\mathbf{i}_t)_{t=1}^n]C(k,d))$ where $H[(\mathbf{i}_t)_{t=1}^n]$ denotes the "hardness" [4] of the sequence $(\mathbf{i}_t)_{t=1}^n$. Furthermore, in case $H[(\mathbf{i}_t)_{t=1}^n] \leq S$ for some $S > 0$ known to player, the regret bound then improves to $O(n^{\frac{\alpha+k}{2\alpha+k}}(\log n)^{\frac{\alpha}{2\alpha+k}}S^{\frac{\alpha}{2\alpha+k}}C(k,d))$.

The rest of the paper is organized as follows. In Section 2 we define the problem statement formally and outline our main results. In Section 3 we present an analysis for the setting when the active k coordinates are fixed across time, including the construction of the discrete strategy sets. In Section 4 we consider the setting where the active k coordinates change across time. Finally in Section 5 we summarize our results and provide directions for future work.

2 Problem Setup and Main Results

The compact set of strategies $S = [0,1]^d \subset \mathbb{R}^d$ is available to the player. At each time step $t = 1, \ldots, n$, a reward function $r_t : S \to \mathbb{R}$ is chosen by the environment. Upon playing a strategy $\mathbf{x}_t \in [0,1]^d$, the player receives the reward $r_t(\mathbf{x}_t)$ at time step t. For some $k \leq d$, we assume each r_t to depend on a fixed but unknown subset of k variables implying $r_t(x_1, \ldots, x_d) = g_t(x_{i_1}, \ldots, x_{i_k})$ where $(i_1, \ldots, i_k)$ is a k-tuple with distinct integers $i_j \in \{1, \ldots, d\}$ and $g_t : [0,1]^k \to \mathbb{R}$. For simplicity of notation, we denote the set of such k-tuples of the set $\{1, \ldots, d\}$ by $\mathcal{T}_k^d$ and the ℓ_2 norm by $\|\cdot\|$. We assume that k is known to the player, however it suffices to know a bound for k as well. The second assumption that we make is on the smoothness property of the reward functions.

Definition 1. *A function $f : [0,1]^k \to \mathbb{R}$ is locally uniformly Hölder continuous with constant $0 \leq L < \infty$, exponent $0 < \alpha \leq 1$, and restriction $\delta > 0$ if we have for all $\mathbf{u}, \mathbf{u}' \in [0,1]^k$ with $\| \mathbf{u} - \mathbf{u}' \| \leq \delta$ that $|f(\mathbf{u}) - f(\mathbf{u}')| \leq L \| \mathbf{u} - \mathbf{u}' \|^\alpha$. We denote the class of such functions f as $\mathcal{C}(\alpha, L, \delta, k)$.*

The function class defined in Definition 1 was also considered in [14,10] and is a generalization of Lipschitz continuity (obtained for $\alpha = 1$). We now define the

[3] See Remark 1 in Section 3 for discussion on how the $\log n$ term can be removed.
[4] See Definition 3 in Section 4.

two models that we analyze in this paper. These models describe how the reward functions g_t are generated at each time step t.

Stochastic model. The reward functions g_t are considered to be i.i.d samples from some fixed but unknown probability distribution over functions $g : [0,1]^k \to \mathbb{R}$. We define the expectation of the reward function as $\bar{g}(\mathbf{u}) = \mathbb{E}[g(\mathbf{u})]$ where $\mathbf{u} \in [0,1]^k$. We require $\bar{g}$ to belong to $\mathcal{C}(\alpha, L, \delta, k)$ and note that the individual samples g_t need not necessarily be Hölder continuous. The optimal strategy $\mathbf{x}^*$ is then defined as follows.

$$\mathbf{x}^* := \text{argmax}_{\mathbf{x}\in[0,1]^d} \mathbb{E}[r(\mathbf{x})] = \text{argmax}_{\mathbf{x}\in[0,1]^d} \bar{g}(x_{i_1}, \ldots, x_{i_k}). \tag{1}$$

Adversarial model. The reward functions $g_t : [0,1]^k \to [0,1]$ are a fixed sequence of functions in $\mathcal{C}(\alpha, L, \delta, k)$ chosen arbitrarily by an *oblivious* adversary i.e., an adversary oblivious to the actions of the player. The optimal strategy $\mathbf{x}^*$ is then defined as follows.

$$\mathbf{x}^* := \text{argmax}_{\mathbf{x}\in[0,1]^d} \sum_{t=1}^{n} r_t(\mathbf{x}) = \text{argmax}_{\mathbf{x}\in[0,1]^d} \sum_{t=1}^{n} g_t(x_{i_1}, \ldots, x_{i_k}). \tag{2}$$

Given the above models we measure the performance of a player over n rounds in terms of the *regret* defined as

$$R(n) := \sum_{t=1}^{n} \mathbb{E}\left[r_t(\mathbf{x}^*) - r_t(\mathbf{x}_t)\right] = \sum_{t=1}^{n} \mathbb{E}\left[g_t(\mathbf{x}^*_{i_1}, \ldots, \mathbf{x}^*_{i_k}) - g_t(\mathbf{x}^{(t)}_{i_1}, \ldots, \mathbf{x}^{(t)}_{i_k})\right]. \tag{3}$$

In (3) the expectation is defined over the random choices of g_t for the stochastic model and the random choice of the strategy $\mathbf{x}_t$ at each time t by the player.

Main Results. The main results of our work are as follows. Firstly, assuming that the k-tuple $(i_1, \ldots, i_k) \in \mathcal{T}_k^d$ is chosen once at the beginning of play and kept fixed thereafter, we provide in the form of Theorem 1 a bound on the regret which is $O(n^{\frac{\alpha+k}{2\alpha+k}}(\log n)^{\frac{\alpha}{2\alpha+k}} C(k,d))$ where $C(k,d) = O(\text{poly}(k) * o(\log d))$. This bound holds for both the stochastic and adversarial models and is *almost optimal*. To see this, we note that [19] showed a precise exponential lower bound of $\Omega(n^{\frac{d+1}{d+2}})$ after $n = \Omega(2^d)$ plays for stochastic continuum armed bandits with d-variate Lipschitz continuous reward functions defined over $[0,1]^d$. In our setting though, the reward functions depend on an unknown subset of k coordinate variables hence any algorithm after $n = \Omega(2^k)$ plays would incur worst case regret of $\Omega(n^{\frac{k+1}{k+2}})$ which is still mild if $k \ll d$. We see that our upper bound matches this lower bound for the case of Lipschitz continuous reward functions ($\alpha = 1$) up to a mild factor of $(\log n)^{\frac{1}{2+k}} C(k,d)$. We also note that the $o(\log d)$ factor in (4) accounts for the uncertainty in not knowing which k coordinates are active from $\{1, \ldots, d\}$.

Theorem 1. *Given that the k-tuple* $(i_1, \ldots, i_k) \in \mathcal{T}_k^d$ *is kept fixed across time but unknown to the player, the algorithm* `CAB(d,k)` *incurs a regret of*

$$O\left(n^{\frac{\alpha+k}{2\alpha+k}}(\log n)^{\frac{\alpha}{2\alpha+k}} k^{\frac{\alpha(k+6)}{2(2\alpha+k)}} e^{\frac{k\alpha}{2\alpha+k}} (\log d)^{\frac{\alpha}{2\alpha+k}}\right) \tag{4}$$

after n rounds of play with high probability for both the stochastic and adversarial models.

The above result is proven in Section 3 along with a description of the `CAB(d,k)` algorithm which achieves this bound. The main idea here is to discretize $[0,1]^d$ by first constructing a family of partitions $\mathcal{A}$ of $\{1, \ldots, d\}$ with each partition consisting of k disjoint subsets. The construction is probabilistic and the resulting $\mathcal{A}$ satisfies an important property (with high probability) namely the *Partition Assumption* as described in Section 3. In particular we have that $|\mathcal{A}|$ is $O(ke^k \log d)$ resulting in a total of $M^k|\mathcal{A}|$ sampling points for some integer $M > 0$. This discrete strategy set is then used with a finite armed bandit algorithm such as UCB-1 [9] for the stochastic setting and Exp3 [8] for the adversarial setting, to achieve the regret bound of Theorem 1.

Secondly we extend our results to the setting where $(i_1, \ldots, i_k)$ can change over time. Considering that an oblivious adversary chooses arbitrarily before the start of plays a sequence of k tuples $(\mathbf{i}_t)_{t=1}^n = (i_{1,t}, \ldots, i_{k,t})_{t=1}^n$ of *hardness* $H[(\mathbf{i}_t)_{t=1}^n] \leq S$ (see Definition 3 in Section 4) with $S > 0$ known to the player, we show how Algorithm `CAB(d,k)` can be adapted to this setting to achieve a regret bound of $O\left(n^{\frac{\alpha+k}{2\alpha+k}}(\log n)^{\frac{\alpha}{2\alpha+k}} S^{\frac{\alpha}{2\alpha+k}} C(k,d)\right)$. Hardness of a sequence is defined as the number of adjacent elements with different values. In case the player has no knowledge of S, the regret bound then changes to $O(n^{\frac{\alpha+k}{2\alpha+k}}(\log n)^{\frac{\alpha}{2\alpha+k}} H[(\mathbf{i}_t)_{t=1}^n] C(k,d))$. Although our bound becomes trivial when $H[(\mathbf{i}_t)_{t=1}^n]$ is close to n (as one would expect), we can still achieve sub-linear regret when $H[(\mathbf{i}_t)_{t=1}^n]$ is small relative to n. We again consider a discretization of the space $[0,1]^d$ constructed using the family of partitions $\mathcal{A}$ mentioned earlier. The difference lies in now using the Exp3.S algorithm [20] on the discrete strategy set, which in contrast to the Exp3 algorithm is designed to control regret against arbitrary sequences. This is described in Section 4.

3 Analysis When k Active Coordinates Are Fixed across Time

We begin with the setting where the set of active k coordinates is fixed across time. The very core of our analysis involves the usage of a specific family of partitions $\mathcal{A}$ of $\{1, \ldots, d\}$ where each $\mathbf{A} \in \mathcal{A}$ consists of k disjoint subsets $(A_1, \ldots, A_k)$. In particular we require $\mathcal{A}$ to satisfy an important property namely the *partition assumption* defined below.

Definition 2. *A family of partitions* $\mathcal{A}$ *of* $\{1, \ldots, d\}$ *into k disjoint subsets is said to satisfy the partition assumption if for any k distinct integers* $i_1, i_2, \ldots, i_k \in \{1, \ldots, d\}$*, there exists a partition* $\mathbf{A} = (A_1, \ldots, A_k)$ *in* $\mathcal{A}$ *such that each set in* $\mathbf{A}$ *contains exactly one of* $i_1, i_2, \ldots, i_k$*.*

The above definition is known as *perfect hashing* in theoretical computer science and is widely used such as in finding juntas [21]. There exists a fairly simple probabilistic method using which one can construct $\mathcal{A}$ consisting of $O(ke^k \log d)$ partitions satisfying the partition assumption property with high probability (see for example[5], Section 5 in [12]). For our purposes, we consider the aforementioned probabilistic construction. However, there also exist deterministic constructions resulting in larger family sizes such as the one proposed in [22] where a family of size $O(k^{O(\log k)} e^k \log d)$ is constructed deterministically in time $poly(d, k)$. For details we refer the reader to the full version of this paper [23].

Constructing Strategy set $\mathcal{P}_M$ Using $\mathcal{A}$. Suppose we are given a family of partitions $\mathcal{A}$ satisfying the partition assumption. Then using $\mathcal{A}$ we construct the discrete set of strategies $\mathcal{P}_M \in [0,1]^d$ for some fixed integer $M > 0$ as follows.

$$\mathcal{P}_M := \left\{ \frac{1}{M} \sum_{j=1}^{k} \alpha_j \chi_{\mathbf{A}_j} ; \alpha_j \in \{1, \ldots, M\}, (\mathbf{A}_1, \ldots, \mathbf{A}_k) \in \mathcal{A} \right\} \subset [0,1]^d \quad (5)$$

Note that a strategy $\mathbf{x} = \frac{1}{M} \sum_{j=1}^{k} \alpha_j \chi_{\mathbf{A}_j}$ has coordinate value $\frac{1}{M}\alpha_j$ at each of the coordinate indices in A_j. Therefore we see that for each partition $\mathbf{A} \in \mathcal{A}$ we have M^k strategies implying a total of $M^k|\mathcal{A}|$ strategies in $\mathcal{P}_M$.

Algorithm 1. Algorithm CAB(d, k)

$T = 1$
Construct family of partitions $\mathcal{A}$ satisfying partition assumption
while $T \leq n$ **do**
  $M = \left\lceil \left(k^{\frac{\alpha-3}{2}} e^{-\frac{k}{2}} (\log d)^{-\frac{1}{2}} \sqrt{\frac{T}{\log T}} \right)^{2/(2\alpha+k)} \right\rceil$
  – Create $\mathcal{P}_M$ using $\mathcal{A}$
  – Initialize **MAB** with $\mathcal{P}_M$
  for $t = T, \ldots, \min(2T-1, n)$ **do**
    – get $\mathbf{x}_t$ from **MAB**
    – Play $\mathbf{x}_t$ and get $r_t(\mathbf{x}_t)$
    – Feed $r_t(\mathbf{x}_t)$ back to **MAB**
  end for
  $T = 2T$
end while

Projection Property. An important property of the strategy set $\mathcal{P}_M$ is the following. Given any k-tuple of distinct indices $(i_1, \ldots, i_k)$ with $i_j \in \{1, \ldots, d\}$ and any integers $1 \leq n_1, \ldots, n_k \leq M$, there is a strategy $\mathbf{x} \in \mathcal{P}_M$ such that

$$(x_{i_1}, \ldots, x_{i_k}) = \left(\frac{n_1}{M}, \ldots, \frac{n_k}{M} \right).$$

[5] In [12] the authors consider the significantly different *function approximation* problem as opposed to our setting of online optimization.

To see this, one can simply take a partition $\mathbf{A} = (A_1, \ldots, A_k)$ from $\mathcal{A}$ such that each i_j is in a different set A_j for $j = 1, \ldots, k$. Then setting appropriate $\alpha_j = n_j$ when $i_j \in A_j$ we get that coordinate i_j of $\mathbf{x}$ has the value n_j/M.

Upper Bound on Regret. We now describe our Algorithm `CAB(d,k)` and provide bounds on its regret. Note that the outer loop is a standard doubling trick which is used as the player has no knowledge of the time horizon n. Observe that before the start of the inner loop of duration T, the player constructs the finite strategy set $\mathcal{P}_M$, where M increases progressively with T. Within the inner loop, the problem reduces to a finite armed bandit problem. The **MAB** routine can be any standard multi-armed bandit algorithm such as UCB-1 (stochastic model) or Exp3 (adversarial model). The main idea is that for increasing values of M, we would have for any $\mathbf{x}^*$ and any $(i_1, \ldots, i_k)$ the existence of an arbitrarily close point to $(x^*_{i_1}, \ldots, x^*_{i_k})$ in $\mathcal{P}_M$. This follows from the projection property of $\mathcal{P}_M$. Coupled with the Hölder continuity of the reward functions this then ensures that the **MAB** routine progressively plays strategies closer and closer to $\mathbf{x}^*$ leading to a bound on regret. The algorithm is motivated by the `CAB1` algorithm [10], however unlike the equi-spaced sampling done in `CAB1` we consider a probabilistic construction of the discrete set of sampling points based on partitions of $\{1, \ldots, d\}$. Now for the stochastic setting, we make the following assumption on the distribution from which the random samples g are generated.

Assumption 1. *We assume that there exist constants $\zeta, s_0 > 0$ so that*

$$\mathbb{E}[e^{s(g(\mathbf{u})-\bar{g}(\mathbf{u}))}] \leq e^{\frac{1}{2}\zeta^2 s^2} \quad \forall s \in [-s_0, s_0], \mathbf{u} \in [0,1]^k.$$

The above assumption was considered in [10] for the case $d = 1$ and allows us to consider reward functions g_t whose range is not bounded. Note that the mean reward $\bar{g}$ is assumed to be Hölder continuous and is therefore bounded. We now present in the following lemma the regret bound incurred within an inner loop of duration T.

Lemma 1. *Given that $(i_1, \ldots, i_k)$ is fixed across time then if the strategy set $\mathcal{P}_M$ is used with (i) the UCB-1 algorithm for the stochastic setting or, (ii) the Exp3 algorithm for the adversarial setting, we have for the choice $M = \left\lceil \left(k^{\frac{\alpha-3}{2}} e^{-\frac{k}{2}} (\log d)^{-\frac{1}{2}} \sqrt{\frac{T}{\log T}} \right)^{\frac{2}{2\alpha+k}} \right\rceil$ that the regret incurred by the player after T rounds is given by $R(T) = O\left(T^{\frac{\alpha+k}{2\alpha+k}} (\log T)^{\frac{\alpha}{2\alpha+k}} k^{\frac{\alpha(k+6)}{2(2\alpha+k)}} e^{\frac{k\alpha}{2\alpha+k}} (\log d)^{\frac{\alpha}{2\alpha+k}} \right)$.*

Proof. For some $\mathbf{x}' \in \mathcal{P}_M$ we can split $R(T)$ into $R_1(T) + R_2(T)$ where:

$$R_1(T) = \sum_{t=1}^{T} \mathbb{E}[g_t(x^*_{i_1}, \ldots, x^*_{i_k}) - g_t(x'_{i_1}, \ldots, x'_{i_k})], \tag{6}$$

$$R_2(T) = \sum_{t=1}^{T} \mathbb{E}[g_t(x'_{i_1}, \ldots, x'_{i_k}) - g_t(x^{(t)}_{i_1}, \ldots, x^{(t)}_{i_k})]. \tag{7}$$

For the k tuple $(i_1, \ldots, i_k) \in \mathcal{T}_k^d$, there exists $\mathbf{x}' \in \mathcal{P}_M$ with $x'_{i_1} = \frac{\alpha_1}{M}, \ldots, x'_{i_k} = \frac{\alpha_k}{M}$ where $\alpha_1, \ldots, \alpha_k$ are such that $|\alpha_j/M - x^*_{i_j}| < (1/M)$. This follows from the projection property of $\mathcal{A}$. On account of the Hölder continuity of reward functions we then have that

$$\mathbb{E}[g_t(x^*_{i_1}, \ldots, x^*_{i_k}) - g_t(x'_{i_1}, \ldots, x'_{i_k})] < L\left(\left(\frac{1}{M}\right)^2 k\right)^{\alpha/2}.$$

In other words, $R_1(T) = O(Tk^{\alpha/2}M^{-\alpha})$. In order to bound $R_2(T)$, we note that the problem has reduced to a $|\mathcal{P}_M|$-armed bandit problem. Specifically we note from (7) that we are comparing against a suboptimal strategy $\mathbf{x}'$ instead of the optimal one in $\mathcal{P}_M$. Hence $R_2(T)$ can be bounded by using existing bounds for finite-armed bandit problems. Now for the stochastic setting we can employ the UCB-1 algorithm [9] and play at each t a strategy $\mathbf{x}_t \in \mathcal{P}_M$. In particular, on account of Assumption 1, it can be shown that $R_2(T) = O(\sqrt{|\mathcal{P}_M| T \log T})$ (Theorem 3.1, [10]). For the adversarial setting we can employ the Exp3 algorithm [8] so that $R_2(T) = O(\sqrt{|\mathcal{P}_M| T \log |\mathcal{P}_M|})$. Combining the bounds for $R_1(T)$ and $R_2(T)$ and recalling that $|\mathcal{P}_M| = O(M^k k e^k \log d)$ we obtain:

$$R(T) = O(TM^{-\alpha}k^{\alpha/2} + \sqrt{M^k k e^k \log d \; T \log T}) \text{ (stochastic) and,} \tag{8}$$

$$R(T) = O(TM^{-\alpha}k^{\alpha/2} + \sqrt{M^k k e^k \log d \; T \log(M^k k e^k \log d)}). \text{ (adversarial)} \tag{9}$$

Plugging $M = \left\lceil \left(k^{\frac{\alpha-3}{2}} e^{-\frac{k}{2}} (\log d)^{-\frac{1}{2}} \sqrt{\frac{T}{\log T}}\right)^{\frac{2}{2\alpha+k}} \right\rceil$ in (8) and (9) we obtain the stated bound on $R(T)$ for the respective models. □

Lastly equipped with the above bound we have that the regret incurred by Algorithm 1 over n plays is given by:

$$\sum_{i=0, T=2^i}^{i=\log n} R(T) = O\left(n^{\frac{\alpha+k}{2\alpha+k}} (\log n)^{\frac{\alpha}{2\alpha+k}} k^{\frac{\alpha(k+6)}{2(2\alpha+k)}} e^{\frac{k\alpha}{2\alpha+k}} (\log d)^{\frac{\alpha}{2\alpha+k}}\right).$$

Remark 1. For the adversarial setting we can use the INF algorithm of [24] as the MAB routine in our algorithm and get rid of the $\log n$ factor from the regret bound. The same holds for the stochastic setting, if the range of the reward functions was restricted to be $[0, 1]$. When the range of the reward functions is $\mathbb{R}$, as is the case in our setting, it seems possible to consider a variant of the MOSS algorithm [24] along with Assumption 2 on the distribution of the reward functions (using proof techniques similar to [25]), to remove the $\log n$ factor from the regret bound.

4 Analysis When k Active Coordinates Change across Time

We now consider a more general *adversarial* setting where the k tuple $(i_1, \ldots, i_k)$ is allowed to change over time. Formally this means that the reward functions $(r_t)_{t=1}^n$ now have the form $r_t(x_1, \ldots, x_d) = g_t(x_{i_{1,t}}, \ldots, x_{i_{k,t}})$ where

$(i_{1,t}, \ldots, i_{k,t})_{t=1}^n$ denotes the sequence of k-tuples chosen by the adversary before the start of plays. However we assume that this sequence of k-tuples is not "hard" meaning that it contains a small number of consecutive pairs (relative to the number of rounds n) with different values. Furthermore, $r_t : [0,1]^d \to [0,1]$ with $g_t : [0,1]^k \to [0,1]$ where $g_t \in \mathcal{C}(\alpha, \delta, L, k)$. We now formally present the definition of hardness of a sequence.

Definition 3. *For any set $\mathcal{B}$ we define the hardness of the sequence $(b_1, \ldots, b_n) \in \mathcal{B}^n$ by:*

$$H[b_1, \ldots, b_n] := 1 + |\{1 \leq l < n : b_l \neq b_{l+1}\}|. \tag{10}$$

The above definition is borrowed from Section 8 in [20] where the authors considered the non-stochastic multi armed bandit problem, and employed the definition to characterize the hardness of a sequence of actions against which the regret of the players actions is measured. In our setting, we consider the sequence of k-tuples chosen by the adversary to be at most S-hard, meaning that $H[(i_{1,t}, \ldots, i_{k,t})_{t=1}^n] \leq S$ for some $S > 0$, and also assume that S is known to the player. We now proceed to show how a slight modification of Algorithm CAB(d,k) can be used to derive a bound on the regret in this setting. Recall that the optimal strategy $\mathbf{x}^* := \text{argmax}_{\mathbf{x} \in [0,1]^d} \sum_{t=1}^n g_t(x_{i_{1,t}}, \ldots, x_{i_{k,t}})$. Since the sequence of k-tuples is S-hard, this in turn implies for any $\mathbf{x}^*$ that $H[(x^*_{i_{1,t}}, \ldots, x^*_{i_{k,t}})_{t=1}^n] \leq S$. Therefore we can now consider this as a setting where the players regret is measured against an S-hard sequence $(x^*_{i_{1,t}}, \ldots, x^*_{i_{k,t}})_{t=1}^n$.

Now the player does not know which k-tuple is chosen at each time t. Hence we again construct the discrete strategy set $\mathcal{P}_M$ (as defined in (5)) using the family of partitions $\mathcal{A}$ of $\{1, \ldots, d\}$. By construction, we will have for any $\mathbf{x} \in [0,1]^d$ and any k-tuple $(i_1, \ldots, i_k)$, the existence of a point $\mathbf{z}$ in $\mathcal{P}_M$ such that $(z_{i_1}, \ldots, z_{i_k})$ approximates $(x_{i_1}, \ldots, x_{i_k})$ arbitrarily well for increasing values of M. Hence, for the optimal sequence $(x^*_{i_{1,t}}, \ldots, x^*_{i_{k,t}})_{t=1}^n$, we have the existence of a sequence of points $(\mathbf{z}^{(t)})_{t=1}^n$ where $\mathbf{z}^{(t)} \in \mathcal{P}_M$ with the following two properties.

1. *S-hardness.* $H[(z^{(t)}_{i_{1,t}}, \ldots, z^{(t)}_{i_{k,t}})_{t=1}^n] \leq S$. This follows easily from the S-hardness of the sequence $(x^*_{i_{1,t}}, \ldots, x^*_{i_{k,t}})_{t=1}^n$ and by choosing for each $(x^*_{i_{1,t}}, \ldots, x^*_{i_{k,t}})$ a corresponding $\mathbf{z}^{(t)} \in \mathcal{P}_M$ such that $\| (x^*_{i_{1,t}}, \ldots, x^*_{i_{k,t}}) - (z^{(t)}_{i_{1,t}}, \ldots, z^{(t)}_{i_{k,t}}) \|$ is minimized.
2. *Approximation property.* $\| (x^*_{i_{1,t}}, \ldots, x^*_{i_{k,t}}) - (z^{(t)}_{i_{1,t}}, \ldots, z^{(t)}_{i_{k,t}}) \| = O(k^{\alpha/2} M^{-\alpha})$. This is easily verifiable via the projection property of the set $\mathcal{P}_M$.

Therefore by employing the Exp3.S algorithm [20] on the strategy set $\mathcal{P}_M$ we reduce the problem to a finite armed adversarial bandit problem where the players regret measured against the S-hard sequence $(z^{(t)}_{i_{1,t}}, \ldots, z^{(t)}_{i_{k,t}})_{t=1}^n$ is bounded from above. The approximation property of this sequence (as explained above) coupled with the Hölder continuity of g_t ensures in turn that the players regret against the original sequence $(x^*_{i_{1,t}}, \ldots, x^*_{i_{k,t}})_{t=1}^n$ is also bounded. With this in

mind we present the following lemma, which formally states a bound on regret after T rounds of play.

Lemma 2. *Given the above setting and assuming that:*

1. *the sequence of k-tuples $(i_{1,t},\dots,i_{k,t})_{t=1}^n$ is at most S-hard and,*
2. *the Exp3.S algorithm is used along with the strategy set $\mathcal{P}_M$,*

we have for the choice $M = \left\lceil \left(k^{\frac{\alpha-3}{2}} e^{-\frac{k}{2}} (S\log d)^{-\frac{1}{2}} \sqrt{\frac{T}{\log T}} \right)^{\frac{2}{2\alpha+k}} \right\rceil$ *that the regret incurred by the player after T rounds is given by:*

$$R(T) = O\left(T^{\frac{\alpha+k}{2\alpha+k}} (\log T)^{\frac{\alpha}{2\alpha+k}} k^{\frac{\alpha(k+6)}{2(2\alpha+k)}} e^{\frac{k\alpha}{2\alpha+k}} (S\log d)^{\frac{\alpha}{2\alpha+k}} \right).$$

Proof. At each time t, for some $\mathbf{z}^{(t)} \in \mathcal{P}_M$ we can split $R(T)$ into $R_1(T)+R_2(T)$ where $R_1(T) = \mathbb{E}[\sum_{t=1}^T g_t(x^*_{i_{1,t}},\dots,x^*_{i_{k,t}}) - g_t(z^{(t)}_{i_{1,t}},\dots,z^{(t)}_{i_{k,t}})]$ and $R_2(T) = \mathbb{E}[\sum_{t=1}^T g_t(z^{(t)}_{i_{1,t}},\dots,z^{(t)}_{i_{k,t}}) - g_t(x^{(t)}_{i_{1,t}},\dots,x^{(t)}_{i_{k,t}})]$.

Let us consider $R_1(T)$ first. As before, from the projection property of $\mathcal{A}$ we have for each $(x^*_{i_{1,t}},\dots,x^*_{i_{k,t}})$, that there exists $\mathbf{z}^{(t)} \in \mathcal{P}_M$ with $z^{(t)}_{i_{1,t}} = \frac{\alpha^{(t)}_1}{M},\dots,z^{(t)}_{i_{k,t}} = \frac{\alpha^{(t)}_k}{M}$ where $\alpha^{(t)}_1,\dots,\alpha^{(t)}_k$ are such that $|\alpha^{(t)}_j/M - x^*_{i_{j,t}}| < (1/M)$ holds for $j = 1,\dots,k$ and each $t = 1,\dots,n$. Therefore from Hölder continuity of g_t we obtain $R_1(T) = O(Tk^{\alpha/2}M^{-\alpha})$. It remains to bound $R_2(T)$. To this end, note that the sequence $(z^{(t)}_{i_{1,t}},\dots,z^{(t)}_{i_{k,t}})_{t=1}^n$ with $\mathbf{z}^{(t)} \in \mathcal{P}_M$ is at most S-hard. Hence the problem has reduced to a $|\mathcal{P}_M|$ armed adversarial bandit problem with a S-hard optimal sequence of plays against which the regret of the player is to be bounded. This is accomplished by using the Exp3.S algorithm of [20] which is designed to control regret against *any* S-hard sequence of plays. In particular from Corollary 8.3 of [20] we have that $R_2(T) = O(\sqrt{S|\mathcal{P}_M|T\log(|\mathcal{P}_M|T)})$. Combining the bounds for $R_1(T)$ and $R_2(T)$ and recalling that $|\mathcal{P}_M| = O(M^k k e^k \log d)$ we obtain the following expression for $R(T)$:

$$R(T) = O(Tk^{\alpha/2}M^{-\alpha} + \sqrt{STM^k k e^k \log d \log(TM^k k e^k \log d)}). \tag{11}$$

Lastly after plugging in the value $M = \left\lceil \left(k^{\frac{\alpha-3}{2}} e^{-\frac{k}{2}} (S\log d)^{-\frac{1}{2}} \sqrt{\frac{T}{\log T}} \right)^{\frac{2}{2\alpha+k}} \right\rceil$ in (11), we obtain the stated bound on $R(T)$. □

By employing Algorithm 1 with **MAB** sub-routine being the Exp3.S algorithm, we have that its regret over n plays is given by

$$\sum_{i=0,T=2^i}^{i=\log n} R(T) = O\left(n^{\frac{\alpha+k}{2\alpha+k}} (\log n)^{\frac{\alpha}{2\alpha+k}} k^{\frac{\alpha(k+6)}{2(2\alpha+k)}} e^{\frac{k\alpha}{2\alpha+k}} (S\log d)^{\frac{\alpha}{2\alpha+k}} \right).$$

Remark 2. In case the player does not know S, then a regret of

$$R(n) = O\left(n^{\frac{\alpha+k}{2\alpha+k}}(\log n)^{\frac{\alpha}{2\alpha+k}} k^{\frac{\alpha(k+6)}{2(2\alpha+k)}} e^{\frac{k\alpha}{2\alpha+k}} (\log d)^{\frac{\alpha}{2\alpha+k}} H[(\mathbf{i}_t)_{t=1}^n]\right)$$

would be incurred by Algorithm 1 with the **MAB** routine being the Exp3.S algorithm and for the choice $M = \left\lceil \left(k^{\frac{\alpha-3}{2}} e^{-\frac{k}{2}} (\log d)^{-\frac{1}{2}} \sqrt{\frac{T}{\log T}}\right)^{\frac{2}{2\alpha+k}} \right\rceil$. Here $\mathbf{i}_t$ is shorthand notation for $(i_{1,t}, \ldots, i_{k,t})$. This can be verified easily along the lines of the proof of Lemma 2 by noting that on account of Corollary 8.2 of [20], we have $R_2(T) = O(H[(\mathbf{i}_t)_{t=1}^n]\sqrt{|\mathcal{P}_M| T \log(|\mathcal{P}_M| T)})$.

5 Concluding Remarks

In this work we considered continuum armed bandit problems for the stochastic and adversarial settings where the reward function $r : [0,1]^d \to \mathbb{R}$ depends at each time step on only k out of the d coordinate variables. We proposed an algorithm and proved regret bounds, both for the setting when the the active k coordinates remain fixed across time and also for the more general scenario when they can change over time. There are several interesting lines of future work. Firstly for the case when $(i_1, \ldots, i_k)$ is fixed across time it would be interesting to investigate whether the dependence of regret on k and dimension d achieved by our algorithm, is optimal or not. Secondly, for the case when $(i_1, \ldots, i_k)$ can also change with time, it would be interesting to derive lower bounds on regret to know what the optimal dependence on the hardness of the sequence of k tuples is.

Acknowledgments. The authors thank Sebastian Stich for the helpful discussions and comments on the manuscript and Fabrizio Grandoni for making us aware of deterministic constructions of perfect hash functions. The project CG Learning acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 255827.

References

1. Awerbuch, B., Kleinberg, R.: Near-optimal adaptive routing: Shortest paths and geometric generalizations. In: Proceedings of ACM Symposium on Theory of Computing (2004)
2. Bansal, N., Blum, A., Chawla, S., Meyerson, A.: Online oblivious routing. In: Proceedings of ACM Symposium in Parallelism in Algorithms and Architectures, pp. 44–49 (2003)
3. Monteleoni, C., Jaakkola, T.: Online learning of non-stationary sequences. In: Advances in Neural Information Processing Systems (2003)
4. Blum, A., Kumar, V., Rudra, A., Wu, F.: Online learning in online auctions. In: Proceedings of 14th Symp. on Discrete Alg., pp. 202–204 (2003)
5. Kleinberg, R., Leighton, T.: The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In: Proceedings of Foundations of Computer Science, pp. 594–605 (2003)

6. Lai, T.L., Robbins, H.: Asymptotically efficient adaptive allocations rules. Proceedings of Adv. in Appl. Math. 6, 4–22 (1985)
7. Rothschild, M.: A two-armed bandit theory of market pricing. Journal of Economic Theory 9, 185–202 (1974)
8. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: Gambling in a rigged casino: The adversarial multi-armed bandit problem. In: Proceedings of 36th Annual Symposium on Foundations of Computer Science, pp. 322–331 (1995)
9. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Mach. Learn. 47(2-3), 235–256 (2002)
10. Kleinberg, R.: Nearly tight bounds for the continuum-armed bandit problem. In: 18th Advances in Neural Information Processing Systems (2004)
11. Abernethy, J., Hazan, E., Rakhlin, A.: Competing in the dark: An efficient algorithm for bandit linear optimization. In: Proceedings of the 21st Annual Conference on Learning Theory, COLT 2008 (2008)
12. DeVore, R., Petrova, G., Wojtaszczyk, P.: Approximation of functions of few variables in high dimensions. Constr. Approx. 33, 125–143 (2011)
13. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. 15, 1373–1396 (2003)
14. Agrawal, R.: The continuum-armed bandit problem. SIAM J. Control and Optimization 33, 1926–1951 (1995)
15. Cope, E.W.: Regret and convergence bounds for a class of continuum-armed bandit problems. IEEE Transactions on Automatic Control 54, 1243–1253 (2009)
16. Auer, P., Ortner, R., Szepesvari, C.: Improved rates for the stochastic continuum-armed bandit problem. In: Proceedings of 20th Conference on Learning Theory (COLT), pp. 454–468 (2007)
17. Kleinberg, R., Slivkins, A., Upfal, E.: Multi-armed bandits in metric spaces. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC 2008, pp. 681–690 (2008)
18. Bubeck, S., Munos, R., Stoltz, G., Szepesvari, C.: X-armed bandits. Journal of Machine Learning Research (JMLR) 12, 1587–1627 (2011)
19. Bubeck, S., Stoltz, G., Yu, J.Y.: Lipschitz bandits without the Lipschitz constant. In: Kivinen, J., Szepesvári, C., Ukkonen, E., Zeugmann, T. (eds.) ALT 2011. LNCS (LNAI), vol. 6925, pp. 144–158. Springer, Heidelberg (2011)
20. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The nonstochastic multi-armed bandit problem. SIAM J. Comput. 32(1), 48–77 (2003)
21. Mossel, E., O'Donnell, R., Servedio, R.: Learning juntas. In: Proceedings of the thirty-fifth Annual ACM Symposium on Theory of Computing, STOC 2009, pp. 206–212. ACM (2003)
22. Naor, M., Schulman, L.J., Srinivasan, A.: Splitters and near-optimal derandomization. In: Proceedings of the 36th Annual Symposium on Foundations of Computer Science, pp. 182–191 (1995)
23. Tyagi, H., Gärtner, B.: Continuum armed bandit problem of few variables in high dimensions. CoRR, abs/1304.5793 (2013)
24. Audibert, J.-Y., Bubeck, S.: Regret bounds and minimax policies under partial monitoring. Journal of Machine Learning Research 11, 2635–2686 (2010)
25. Kleinberg, R.D.: Online Decision Problems with Large Strategy Sets. PhD thesis. MIT, Boston (2005)

Approximability of Connected Factors*

Kamiel Cornelissen[1], Ruben Hoeksma[1], Bodo Manthey[1],
N.S. Narayanaswamy[2], and C.S. Rahul[2]

[1] University of Twente, Enschede, The Netherlands
{k.cornelissen,r.p.hoeksma,b.manthey}@utwente.nl
[2] Indian Institute of Technology Madras, Chennai, India
{swamy,rahulcs}@cse.iitm.ac.in

Abstract. Finding a d-regular spanning subgraph (or d-factor) of a graph is easy by Tutte's reduction to the matching problem. By the same reduction, it is easy to find a minimal or maximal d-factor of a graph. However, if we require that the d-factor is connected, these problems become NP-hard – finding a minimal connected 2-factor is just the traveling salesman problem (TSP).

Given a complete graph with edge weights that satisfy the triangle inequality, we consider the problem of finding a minimal connected d-factor. We give a 3-approximation for all d and improve this to an $(r+1)$-approximation for even d, where r is the approximation ratio of the TSP. This yields a 2.5-approximation for even d. The same algorithm yields an $(r+1)$-approximation for the directed version of the problem, where r is the approximation ratio of the asymmetric TSP. We also show that none of these minimization problems can be approximated better than the corresponding TSP.

Finally, for the decision problem of deciding whether a given graph contains a connected d-factor, we extend known hardness results.

1 Introduction

The traveling salesman problem (Min-TSP) is one of the basic combinatorial optimization problems: given a complete graph $G = (V, E)$ with edge weights that satisfy the triangle inequality, the goal is to find a Hamiltonian cycle of minimum total weight. Phrased differently, we are looking for a subgraph of G of minimum weight that is 2-regular, connected, and spanning. While Min-TSP is NP-hard [11, ND22], omitting the requirement that the subgraph must be connected makes the problem polynomial-time solvable [18, 24]. In general, d-regular, spanning subgraphs (also called d-factors) of minimum weight can be found in polynomial time using Tutte's reduction [18, 24] to the matching problem. Cheah and Corneil [6] have shown that deciding whether a given graph $G = (V, E)$ has a d-regular connected spanning subgraph is NP-complete for every $d \geq 2$, where $d = 2$ is just the Hamiltonian cycle problem [11, GT37]. Thus, finding a connected d-factor of minimum weight is also NP-hard for all

* A full version with all proofs is available at http://arxiv.org/abs/1310.2387

C. Kaklamanis and K. Pruhs (Eds.): WAOA 2013, LNCS 8447, pp. 120–131, 2014.

d. While one might think at first glance that the problem cannot become easier for larger d, finding (minimum-weight) connected d-factors is easy for $d \geq n/2$, where $n = |V|$, as in this case any d-factor is already connected. This poses the question for which values of d (as a function of n) the problem becomes tractable. In this paper, we analyze the complexity and approximability of the problem of finding a d-factor of minimum weight.

1.1 Problem Definitions and Preliminaries

In the following, n is always the number of vertices. To which graph n refers will be clear from the context.

All problems defined below deal with undirected graphs, unless stated otherwise. For any d, d-RCS is the following decision problem: Given an arbitrary undirected graph G, does G have a connected d-factor? Here, d can be a constant, but also a function of the number n of vertices of the input graph G. 2-RCS is just the Hamiltonian cycle problem.

Just as Min-TSP is the optimization variant of 2-RCS, we consider the optimization variant of d-RCS, which we call Min-d-RCS: As an instance, we are given an undirected complete graph $G = (V, E)$ and non-negative edge weights w that satisfy the triangle inequality, i.e., $w(\{x, z\}) \leq w(\{x, y\}) + w(\{y, z\})$ for every $x, y, z \in V$. The goal of Min-d-RCS is to find a connected d-factor of G of minimum weight. Min-2-RCS is just Min-TSP.

A *bridge edge* of a graph is an edge whose removal increases the number of components of the graph. A graph G is called *2-edge connected* if G is connected and does not contain bridge edges. For even d, any connected d-factor is also 2-edge-connected, i.e., does not contain bridge edges. This is not true for odd d. If we require 2-edge-connectedness also for odd d, we obtain the problem Min-d-R2CS, which is defined as Min-d-RCS, but asks for a 2-edge-connected d-factor. For consistency, Min-d-R2CS is also defined for even d, although it is then exactly the same problem as Min-d-RCS.

Finally, we also consider the asymmetric variant of the problem: given a directed complete graph $G = (V, E)$, find a spanning connected subgraph of G that is d-regular. Here, d-regular means that every vertex has indegree d and outdegree d. We denote the corresponding minimization problem by Min-d-ARCS. Min-1-ARCS is just the asymmetric TSP (Min-ATSP).

Max-d-RCS and Max-d-ARCS are the maximization variants of Min-d-RCS and Min-d-ARCS, respectively. For Max-d-RCS and Max-d-ARCS we do not require that the edge weights satisfy the triangle inequality. In the same way as for the minimization variants, Max-2-RCS is the maximum TSP (Max-TSP) and Max-1-ARCS is the maximum ATSP (Max-ATSP).

If the graph and its edge weights are clear from the context, we abuse notation by also denoting by d-RCS a minimum-weight connected d-factor, by d-R2CS a minimum-weight 2-edge-connected d-factor, and by d-ARCS a minimum-weight connected d-regular subgraph of a directed graph.

In the same way, let d-F denote a minimum-weight d-factor (no connectedness required) of a graph and let d-AF denote a minimum-weight d-factor of a directed

graph. Let MST denote a minimum-weight spanning tree, and let TSP and ATSP denote minimum-weight (asymmetric) TSP tours. We have 2-RCS = TSP and 1-ARCS = ATSP. Furthermore, 2-F is the undirected cycle cover problem and 1-AF is the directed cycle cover problem.

We note that d-factors do not exist for all combinations of d and n. If both n and d are odd, then no n-vertex graph possesses a d-factor. For all other combinations of n and d with $d \leq n-1$, there exist d-factors in n-vertex graphs, at least in the complete graph.

In the following, K_n denotes the undirected complete graph on n vertices. A vertex v of a graph G is called a *cut vertex* if removing v increases the number of components of G.

1.2 Previous Results

Requiring connectedness in addition to some other combinatorial property has already been studied for dominating sets [13] and vertex cover [8]. For problems such as minimum s-t vertex separator, which are known to be solvable in polynomial time, the connectedness condition makes it NP-hard, and recent results have studied the parameterized complexity of finding a connected s-t vertex separator [19]. Also finding connected graphs with given degree sequences that are allowed to be violated only slightly has been well-studied [5, 23].

As far as we are aware, so far only the maximization variant Max-d-RCS of the connected factor problem has been considered for $d \geq 3$. Baburin, Gimadi, and Serdyukov proved that Max-d-RCS can be approximated within a factor of $1-\frac{2}{d\cdot(d+1)}$ [2,12]. A slightly better approximation ratio can be achieved if the edge weights are required to satisfy the triangle inequality [3]. Baburin and Gimadi also considered approximating both Max-d-RCS and Min-d-RCS (both without triangle inequality) for random instances [3,4]. For $d = 2$, we inherit the approximation results for Min-TSP of 3/2 [26, Section 2.4] and Max-TSP of 7/9 [20]. For $d = 1$, we inherit the $O(\log n / \log \log n)$-approximation for Min-ATSP [1] and 2/3 for Max-ATSP [14]. As far as we know, no further polynomial-time approximation algorithms with worst-case guarantees are known for Min-d-RCS. Like for Min-TSP [26, Section 2.4], the triangle inequality is crucial for approximating Min-d-RCS and Min-d-ARCS – otherwise, no polynomial-time approximation algorithm is possible, unless P = NP. Baburin and Gimadi [2, 3] claimed that Max-d-RCS is APX-hard because it generalizes Max-TSP. However, this is only true if we consider d as part of the input, as then $d = 2$ corresponds to Max-TSP.

1.3 Our Results

Table 1 shows an overview of previous results and our results. Our main contributions are a 3-approximation algorithm for Min-d-RCS for any d and a 2.5-approximation algorithm for Min-d-RCS for even d (Section 3). The latter is in fact an $(r + 1)$-approximation algorithm for Min-d-RCS, where r is the factor within which Min-TSP can be approximated. This result can be extended

Table 1. Overview of the complexity and approximability of finding (optimal) connected d-factors. We left out that all optimization variants are polynomial-time solvable for $d \geq n/2$ and APX-hard according to Sections 4.1 and 4.2. Here, r is the approximation ratio of Min-TSP or Min-ATSP.

problem	*result*	*reference*
d-RCS	in P for $d \geq \frac{n}{2} - 1$	trivial for $d \geq n/2$, Section 5.2
	NP-complete for constant d	Cheah and Corneil [6]
	and d of any growth rate up to $O(n^{1-\varepsilon})$	Section 4.2
Min-d-RCS	$(r+1)$-approximation for even d	Section 3.2
	3-approximation for odd d	Section 3.1
	2-approximation for $d \geq n/3$	Section 5.1
	no better approximable than Min-TSP	Section 4.1
Min-d-R2CS	3-approximation	Section 3.1
	no better approximable than Min-TSP	Section 4.1
Min-d-ARCS	$(r+1)$-approximation	Section 3.2
	no better approximable than Min-ATSP	Section 4.1
Max-d-RCS	$(1 - \frac{2}{d \cdot (d+1)})$-approximation	Baburin and Gimadi [2]
Max-d-ARCS	$(1 - \frac{1}{d \cdot (d+1)})$-approximation	Section 5.3

to Min-d-ARCS, where r is now the approximation ratio of Min-ATSP. Our approximation algorithms, in particular for the maximization variants, are in the spirit of the classical approximation algorithm of Fisher et al. [10] for Max-TSP: compute a non-connected structure, and then remove and add edges to make it connected.

As lower bounds, we prove that Min-d-RCS and Min-d-ARCS cannot be approximated better than Min-TSP and Min-ATSP, respectively (Section 4). In particular, this implies the APX-hardness of the problems.

We prove some structural properties of connected d-factors and their relation to TSP, MST, and d-factors without connectedness requirement (Section 2). Some of these properties are needed for the approximation algorithms and some might be interesting in their own right or were initially counterintuitive to us.

Our algorithms work for all values of d, even when d is part of the input. The hardness results are extended to the case where d grows with n. In Section 5, we improve our approximation guarantee for $d \geq n/3$, prove that $(\frac{n}{2} - 1)$-RCS $\in$ P, and generalize Baburin and Gimadi's algorithm [2] to directed instances.

2 Structural Properties

In the following two lemmas, we make statements about the relationship between the weights of optimal solutions of the different minimization problems. We call an inequality $A \leq c \cdot B$ *tight* if, for every $\varepsilon > 0$, replacing c by $c - \varepsilon$ does not yield a valid statement for all instances.

Lemma 2.1 (Undirected Comparison)

1. $w(\mathsf{MST}) \leq w(d\text{-}\mathsf{RCS}) \leq w(d\text{-}\mathsf{R2CS})$ *for all* d *and all undirected instances, and this is tight.*
2. $w(d\text{-}\mathsf{F}) \leq w(d\text{-}\mathsf{RCS})$ *for all* d *and all undirected instances, and this is tight.*
3. $w(d\text{-}\mathsf{R2CS}) \leq 3 \cdot w(d\text{-}\mathsf{RCS})$ *for all odd* d *and all undirected instances, and this is tight for all odd* d*.*
4. $w(\mathsf{TSP}) \leq w(d\text{-}\mathsf{RCS})$ *for all even* d *and all undirected instances, and this is tight.*
5. $w(\mathsf{TSP}) \leq 2 \cdot w(d\text{-}\mathsf{RCS})$ *for all odd* d *and all undirected instances, and this is tight for all odd* d*.*
6. $w(\mathsf{TSP}) \leq \frac{4}{3} \cdot w(3\text{-}\mathsf{R2CS})$ *for all undirected instances, and this is tight.*
7. *For all odd* d*, there are instances with* $w(\mathsf{TSP}) \geq (\frac{4}{3} - o(1)) \cdot w(d\text{-}\mathsf{R2CS})$*.*
8. $w((d-2)\text{-}\mathsf{F}) \leq \frac{d-2}{d} \cdot w(d\text{-}\mathsf{F})$ *and* $w((d-2)\text{-}\mathsf{RCS}) \leq w(d\text{-}\mathsf{RCS})$ *for all even* $d \geq 4$ *and all undirected instances, and both inequalities are tight.*
9. *Monotonicity does not hold for odd* d*: for every odd* $d \geq 5$*, there exist instances with* $w((d-2)\text{-}\mathsf{RCS}) \geq \frac{d+2}{d} \cdot w(d\text{-}\mathsf{RCS})$*.*

Lemma 2.2 (Directed Comparison)

1. $w(d\text{-}\mathsf{AF}) \leq w(d\text{-}\mathsf{ARCS})$ *for all* d *and all directed instances, and this is tight.*
2. $w(\mathsf{ATSP}) \leq w(d\text{-}\mathsf{ARCS})$ *for all* d *and all directed instances, and this is tight.*
3. $w((d-1)\text{-}\mathsf{AF}) \leq \frac{d-1}{d} \cdot w(d\text{-}\mathsf{AF})$ *and* $w((d-1)\text{-}\mathsf{ARCS}) \leq w(d\text{-}\mathsf{ARCS})$ *for all* $d \geq 2$ *and all directed instances, and both inequalities are tight.*

3 Approximation Algorithms

3.1 3-Approximation for Min-d-RCS and Min-d-R2CS

The 3-approximation that we present in this section works for all d, odd or even. It also works for d growing as a function of n. An interesting feature of this algorithm, and possibly an indication that a better approximation ratio is possible for Min-d-RCS, is that the same algorithm provides an approximation ratio of 3 for both Min-d-RCS and Min-d-R2CS. In fact, we compute a 2-edge-connected d-regular graph that weighs at most three times the weight of the optimal connected d-regular graph.

First we make some preparatory observations on 2-edge-connectedness. Given a connected graph $G = (V, E)$, we can create a tree $T(G)$ as follows: We have a vertex for every maximal subgraph of G that is 2-edge-connected (called a 2-edge-connected component), and two such vertices are connected if the corresponding components are connected in G. In this case, they are connected by a bridge edge. Now consider a leaf of tree $T(G)$ and its corresponding 2-edge-connected component C. Since C is a leaf in $T(G)$, it is only incident to a single bridge edge e in G. Now assume that G is d-regular with $d \geq 3$ odd (for $d = 2$, any connected graph is also 2-edge-connected). Let u be the vertex of C that is incident to e. Then u must be incident to $d - 1$ other vertices in C. Thus, C has at least d

input : undirected complete graph $G = (V, E)$, edge weights w, $d \geq 2$
output: 2-edge-connected d-factor R of G
1 compute a minimum-weight d-factor d-F of G;
2 $k \leftarrow k(d\text{-F})$
3 $Q \leftarrow \{e_1, \ldots, e_k\}$ with $e_i = e_i(d\text{-F}) = \{u_i, v_i\}$
4 compute MST of G;
5 duplicate each edge of MST and take shortcuts to obtain a Hamiltonian cycle H
6 take shortcuts to obtain from H a Hamiltonian cycle H' through $\{u_1, \ldots, u_k\}$, assume w.l.o.g. that H' traverses the vertices in the order $u_1, \ldots, u_k, u_1$
7 obtain R from d-F by adding the edges $\{u_i, v_{i+1}\}$ (with $k + 1 = 1$) and removing Q

Algorithm 1: 3-approximation for Min-d-RCS and Min-d-R2CS

vertices. Since the $d - 1$ neighbors of u are not incident to bridge edges, they must be adjacent to other vertices in C. Since G is d-regular, C has at least $d+1$ vertices and more than $d^2/2 > d$ edges. Therefore, there exists an edge e' in C that is not incident to u, i.e., e' does not share an endpoint with a bridge edge.

If G is not connected, we have exactly the same properties with "tree" replaced by "forest".

To simplify notation in the algorithm, let $k = k(G)$ denote the number of 2-edge-connected components of G that are leaves in the forest described above, and let $L_1(G), \ldots, L_k(G)$ denote the 2-edge-connected components of a graph G that correspond to leaves in the tree described above. For such an $L_i(G)$, let $e_i(G)$ denote an edge that is not adjacent to a bridge edge in G. The choice of $e_i(G)$ is arbitrary.

We prove that Algorithm 1 is a 3-approximation for both Min-d-RCS and Min-d-R2CS by a series of lemmas. Since the set of vertices is fixed, we sometimes identify graphs with their edge set. In particular, R denotes both the connected d-factor that we compute and its edge set.

Lemma 3.1. *Assume that R is computed as in Algorithm 1. Then R is a d-regular spanning subgraph of G.*

Lemma 3.2. *Assume that R is computed as in Algorithm 1. Then R is 2-edge-connected.*

Lemma 3.3. *Assume that R is computed as in Algorithm 1. Then $w(R) \leq 3 \cdot w(d\text{-RCS}) \leq 3 \cdot w(d\text{-R2CS})$.*

The following theorem is an immediate consequence of the lemmas above.

Theorem 3.4. *For all d, Algorithm 1 is a polynomial-time 3-approximation for* Min-d-RCS *and* Min-d-R2CS*. This includes the case that d is a function of n.*

Remark 3.5. If we are only interested in a 3-approximation for Min-d-RCS and not for Min-d-R2CS, then we can simplify Algorithm 1 a bit: we only pick one non-bridge edge for each component and not for every 2-edge-connected component.

The rest of the algorithm and its analysis remain the same. However, this does not seem to improve the worst-case approximation ratio.

Remark 3.6. The analysis is tight in the following sense: By Lemma 2.1(3), a minimum-weight 2-edge-connected d-factor can be three times as heavy as a minimum-weight connected d-factor. Thus, any algorithm that outputs a 2-edge-connected d-factor cannot achieve an approximation ratio better than 3. Furthermore, since $w(\mathsf{MST}) \leq w(d\text{-}\mathsf{R2CS})$ and $w(d\text{-}\mathsf{F}) \leq w(d\text{-}\mathsf{R2CS})$ are tight (Lemma 2.1(1) and (2)), the analysis is essentially tight. If we only require connectedness and not 2-edge-connectedness, we see that the analysis cannot be improved since $w(\mathsf{TSP}) \leq 2w(d\text{-}\mathsf{RCS})$ and $w(d\text{-}\mathsf{F}) \leq w(d\text{-}\mathsf{RCS})$ are tight.

However, it is reasonable to assume that not all these inequalities can be tight at the same time and, in addition, taking shortcuts in the duplicated MST to obtain a TSP tour through $u_1, \ldots, u_k$ does not yield any improvement. Therefore, it might be possible to improve the analysis and show that Algorithm 1 achieves a better approximation ratio than 3.

Remark 3.7. Lines 4 and 5 of Algorithm 1 are in fact the double-tree heuristic for Min-TSP [26, Section 2.4]. One might be tempted to construct a better tour using Christofides' algorithm [26, Section 2.4], which achieves a ratio of 3/2 instead of only 2. However, in the analysis we compare the optimal solution for Min-d-RCS to the MST, and we know that $w(\mathsf{MST}) \leq w(d\text{-}\mathsf{RCS}) \leq w(d\text{-}\mathsf{R2CS})$. If we use Christofides' algorithm directly, we have to compare a TSP tour to the minimum-weight connected d-factor. In particular for odd d, we have that for some instances $w(\mathsf{TSP}) \geq (\frac{4}{3} - o(1)) \cdot w(d\text{-}\mathsf{R2CS}) \geq (\frac{4}{3} - o(1)) \cdot w(d\text{-}\mathsf{RCS})$ (Lemma 2.1(7)). Even if this is the true bound – as it is for $d = 3$ (Lemma 2.1(6)) –, the TSP tour constructed contributes with a factor 3/2 times 4/3, which equals 2, to the approximation ratio, which is no improvement.

3.2 $(r + 1)$-Approximation

In this section, we give an $(r+1)$-approximation for Min-d-RCS for even values of d and Min-d-ARCS for all values of d. Here, r is the ratio within which Min-TSP (for Min-d-RCS) or Min-ATSP (for Min-d-ARCS) can be approximated. This means that we currently have $r = 3/2$ for the symmetric case by Christofides' algorithm [26, Section 2.4] and, for the asymmetric case, we have either $r = O(\log n / \log \log n)$ if we use the randomized algorithm by Asadpour et al. [1] or $r = \frac{2}{3} \cdot \log_2 n$ if we use Feige and Singh's deterministic algorithm [9]. Although the algorithm is a simple modification of Algorithm 1, we summarize it as Algorithm 2 for completeness.

Theorem 3.8. *If* Min-TSP *can be approximated in polynomial time within a factor of r, then Algorithm 2 is a polynomial-time $(r + 1)$-approximation for* Min-d-RCS *for all even d.*

If Min-ATSP *can be approximated in polynomial time within a factor of r, then Algorithm 2 is a polynomial-time $(r + 1)$-approximation for* Min-d-ARCS *for all d.*

The results still hold if d is part of the input.

input : undirected or directed complete graph $G = (V, E)$, edge weights w, d
output: connected d-factor R of G
1 compute a minimum-weight d-factor C of G
2 let $C_1, \ldots, C_k$ be the connected components of C, and let $e_i = (u_i, v_i)$ be any edge of C_i
3 compute a TSP tour H using an approximation algorithm with ratio r
4 take shortcuts to obtain from H a TSP tour H' through $\{u_1, \ldots, u_k\}$, assume w.l.o.g. that H' traverses the vertices in the order $u_1, \ldots, u_k, u_1$
5 obtain R from C by adding the edges (u_i, v_{i+1}) (with $k + 1 = 1$) and removing $e_1, \ldots, e_k$

Algorithm 2: $(r + 1)$-approximation for Min-d-RCS for even d and Min-d-ARCS

4 Hardness Results

4.1 TSP-Inapproximability

In this section, we prove that Min-d-RCS cannot be approximated better than Min-TSP.

Theorem 4.1. *For every $d \geq 2$, if* Min-d-RCS *can be approximated in polynomial time within a factor of r, then* Min-TSP *can be approximated in polynomial time within a factor of r.*

The same construction as in the proof of Theorem 4.1 yields the same result for Min-d-R2CS. A similar construction yields the same result for Min-d-ARCS.

Corollary 4.2. *For every $d \geq 2$, if* Min-d-R2CS *can be approximated in polynomial time within a factor of r, then* Min-TSP *can be approximated in polynomial time within a factor of r.*

Corollary 4.3. *For every $d \geq 2$, if* Min-d-ARCS *can be approximated in polynomial time within a factor of r, then* Min-ATSP *can be approximated in polynomial time within a factor of r.*

Min-TSP, Min-ATSP, Max-TSP, and Max-ATSP are APX-hard [22]. Furthermore, the reduction from Min-TSP to Min-d-RCS is in fact an L-reduction [21] (see also Shmoys and Williamson [26, Section 16.2]). This proves the APX-hardness of Min-d-RCS for all d. The reductions from Min-TSP to Min-d-R2CS and from Min-ATSP to Min-d-ARCS work in the same way. Furthermore, by reducing from Max-TSP and Max-ATSP in a similar way (here, the edges between the copies of a vertex have high weight), we obtain APX-hardness for Max-d-RCS and Max-d-ARCS as well.

Corollary 4.4. *For every fixed $d \geq 2$, the problems* Min-d-RCS, Min-d-R2CS, *and* Max-d-RCS *are* APX*-complete. For every fixed $d \geq 1$,* Min-d-ARCS *and* Max-d-ARCS *are* APX*-complete.*

4.2 Hardness for Growing d

In this section, we generalize the NP-hardness proof for d-RCS by Cheah and Corneil [6] to the case that d grows with n. Furthermore, we extend Theorem 4.1 and Corollaries 4.2 and 4.3 and the APX-hardness of the minimization variants (Corollary 4.4) to growing d. The APX-hardness of Max-d-RCS and Max-d-ARCS does not transfer to growing d – both can be approximated within a factor of $1 - O(1/d^2)$, which is $1 - o(1)$ for growing d.

Let us consider Cheah and Corneil's [6, Section 3.2] reduction from 2-RCS, i.e., the Hamiltonian cycle problem, to d-RCS. Crucial for their reduction is the notion of the d-expansion of a vertex v, which is obtained as follows:

1. We construct a gadget G_{d+1} by removing a matching of size $\lceil \frac{d}{2} \rceil - 1$ from a complete graph on $d+1$ vertices.
2. We connect each vertex whose degree has been decreased by one to v.

The reduction itself takes a graph G for which we want to test if $G \in$ 2-RCS and maps it to a graph $R_d(G)$ as follows: For even d, $R_d(G)$ is the graph obtained by performing a d-expansion for every vertex of G. For odd d, the graph $R_d(G)$ is obtained by doing the following for each vertex v of G: add vertices $u_1, u_2, \ldots, u_{d-2}$; connect v to $u_1, \ldots, u_{d-2}$; perform a d-expansion on $u_1, \ldots, u_{d-2}$. We have $G \in$ 2-RCS if and only if $R_d(G) \in d$-RCS.

We note that $R_d(G)$ has $(d+2) \cdot n$ vertices for even d and $\Theta(d^2 n)$ vertices for odd d and can easily be constructed in polynomial time since $d < n$.

Theorem 4.5. *For every fixed $\varepsilon > 0$, there is a function $f = \Theta(n^{1-\varepsilon})$ that maps to even integers such that f-RCS is* NP*-hard.*

For every fixed $\varepsilon > 0$, there is a function $f = \Theta(n^{\frac{1}{2}-\varepsilon})$ that maps to odd integers such that f-RCS is NP*-hard.*

In the same way as the NP-completeness, the inapproximability can be transferred. The reduction creates graphs of size $(d+1) \cdot n$. The construction is the same as in Section 4.1, and the proof follows the line of the proof of Theorem 4.5. Here, however, we do not have to distinguish between odd and even d for the symmetric variant, as the reduction in Section 4.1 is the same for both cases.

Theorem 4.6. *For every fixed $\varepsilon > 0$, there is a function $f = \Theta(n^{1-\varepsilon})$ such that* Min-f-RCS *and* Min-f-R2CS *are* APX*-hard and cannot be approximated better than* Min-TSP*.*

For every fixed $\varepsilon > 0$, there is a function $f = \Theta(n^{1-\varepsilon})$ such that Min-f-ARCS *is* APX*-hard and cannot be approximated better than* Min-ATSP*.*

5 Further Algorithms

5.1 2-Approximation for $d \geq n/3$

If $d \geq n/3$, then we easily get a better approximation algorithm for Min-d-R2CS and Min-d-RCS. In this case, d-F consists either of a single component – then

we are done – or of two components C_1 and C_2 with $C_i = (V_i, E_i)$. In the latter case, we proceed as follows: first, find the lightest edge $e = \{u, v\}$ with $u \in V_1$ and $v \in V_2$. Second, choose any edges $\{u, u'\} \in E_1$ and $\{v, v'\} \in E_2$. Third, remove $\{u, u'\}$ and $\{v, v'\}$ and add $\{u, v\}$ and $\{u', v'\}$. The increase in weight is at most $2 \cdot w(\{u, v\})$ by the triangle inequality.

The resulting graph is clearly d-regular. It is connected since C_1 and C_2 are 2-edge-connected: they both consist of at most $\frac{2n}{3} - 1$ vertices and are d-regular with $d \geq n/3$. Thus, they are even Hamiltonian by Dirac's theorem [25]. Furthermore, any connected d-regular graph must have at least two edges connecting V_1 and V_2: If d is even, then this follows by 2-edge-connectedness. If d is odd, then $|V_1|$ and $|V_2|$ are even and, thus, an even number of edges must leave either of them. Thus, $w(\{u, v\}) \leq \frac{1}{2} \cdot w(d\text{-RCS})$. Since we add at most $2 \cdot w(\{u, v\})$ and also have $w(d\text{-F}) \leq w(d\text{-RCS})$, we obtain the following theorem.

Theorem 5.1. *For $d \geq n/3$, there is a polynomial-time 2-approximation for* Min-d-RCS.

5.2 Decision Problem for $d = \lceil \frac{n}{2} \rceil - 1$

For $d \geq n/2$, any d-factor is immediately connected and also the minimization variant can be solved efficiently. In this section, we slightly extend this to the case of $d \geq \frac{n}{2} - 1$.

We assume that the input graph G is connected. To show that the case $d = \lceil \frac{n}{2} \rceil - 1$ is in P, we compute a d-factor. If none exists or we obtain a connected d-factor, then we are done. Otherwise, we have a d-factor consisting of two components C_1 and C_2 which are both cliques of size $n/2$. If G contains a cut vertex, say, $u \in C_1$, then this is the only vertex with neighbors in C_2. In this case, G does not contain a connected d-factor. If G does not contain a cut vertex, there are two disjoint edges $e = \{u, v\}$, $e' = \{u', v'\}$ with $u, u' \in C_1$ and $v, v' \in C_2$. Adding e and e' and removing $\{u, u'\}$ and $\{v, v'\}$ yields a connected d-factor.

Theorem 5.2. *d-RCS is in P for every d with $d \geq \frac{n}{2} - 1$.*

5.3 Approximating Max-d-ARCS

The approximation algorithm for Max-d-RCS [2] can easily be adapted to work for Max-d-ARCS: We compute a directed d-factor of maximum weight. Any component consists of at least $d + 1$ vertices, thus at least $d \cdot (d + 1)$ arcs. We remove the lightest arc of every component and connect the resulting (still at least weakly connected) components arbitrarily to obtain a connected d-factor. Since we have removed at most a $\frac{1}{d \cdot (d+1)}$-fraction of the weight, we obtain the following result.

Theorem 5.3. *For every d,* Max-d-ARCS *can be approximated within a factor of $1 - \frac{1}{d \cdot (d+1)}$.*

6 Open Problems

An obvious open problem is to improve the approximation ratios. Apart from this, let us mention two open problems: First, is it possible to achieve constant factor approximations for minimum-weight k-edge-connected or k-vertex-connected d-regular graphs? Without the regularity requirement, the problem of computing minimum-weight k-edge-connected graphs can be approximated within a factor of 2 [17] and the problem of computing minimum-weight k-vertex-connected graphs can be approximated within a factor of $2 + 2 \cdot \frac{k-1}{n}$ for metric instances [15] and still within a factor of $O(\log k)$ if the instances are not required to satisfy the triangle inequality [7]. We refer to Khuller and Raghavachari [16] for a concise survey.

Second, we have seen that $(\lceil \frac{n}{2} \rceil - 1)$-RCS $\in$ P, but we do not know if Min-$(\lceil \frac{n}{2} \rceil - 1)$-RCS can be solved in polynomial time as well. In addition, we conjecture that also $(\lceil \frac{n}{2} \rceil - k)$-RCS is in P for any constant k.

References

1. Asadpour, A., Goemans, M.X., Madry, A., Gharan, S.O., Saberi, A.: An $O(\log n / \log \log n)$-approximation algorithm for the asymmetric traveling salesman problem. In: Proc. of the 21st Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA), pp. 379–389. SIAM (2010)
2. Baburin, A.E., Gimadi, E.K.: Approximation algorithms for finding a maximum-weight spanning connected subgraph with given vertex degrees. In: Operations Research Proceedings 2004, pp. 343–351 (2005)
3. Baburin, A.E., Gimadi, E.K.: Polynomial algorithms for some hard problems of finding connected spanning subgraphs of extreme total edge weight. In: Operations Research Proceedings 2006, pp. 215–220 (2007)
4. Baburin, A.E., Gimadi, E.K.: An approximation algorithm for finding a d-regular spanning connected subgraph of maximum weight in a complete graph with random weights of edges. Journal of Applied and Industrial Mathematics 2(2), 155–166 (2008)
5. Chan, Y.H., Fung, W.S., Lau, L.C., Yung, C.K.: Degree bounded network design with metric costs. SIAM Journal on Computing 40(4), 953–980 (2011)
6. Cheah, F., Corneil, D.G.: The complexity of regular subgraph recognition. Discrete Applied Mathematics 27(1-2), 59–68 (1990)
7. Cheriyan, J., Vempala, S., Vetta, A.: An approximation algorithm for the minimum-cost k-vertex connected subgraph. SIAM Journal on Computing 32(4), 1050–1055 (2003)
8. Escoffier, B., Gourvès, L., Monnot, J.: Complexity and approximation results for the connected vertex cover problem in graphs and hypergraphs. Journal of Discrete Algorithms 8(1), 36–49 (2010)
9. Feige, U., Singh, M.: Improved approximation ratios for traveling salesperson tours and paths in directed graphs. In: Charikar, M., Jansen, K., Reingold, O., Rolim, J.D.P. (eds.) APPROX and RANDOM 2007. LNCS, vol. 4627, pp. 104–118. Springer, Heidelberg (2007)
10. Fisher, M.L., Nemhauser, G.L., Wolsey, L.A.: An analysis of approximation for finding a maximum weight Hamiltonian cycle. Operations Research 27(4), 799–809 (1979)

11. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company (1979)
12. Gimadi, E.K., Serdyukov, A.I.: A problem of finding the maximal spanning connected subgraph with given vertex degrees. In: Operations Research Proceedings 2000, pp. 55–59. Springer (2001)
13. Guha, S., Khuller, S.: Improved methods for approximating node weighted steiner trees and connected dominating sets. Information and Computation 150(1), 57–74 (1999)
14. Kaplan, H., Lewenstein, M., Shafrir, N., Sviridenko, M.I.: Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. Journal of the ACM 52(4), 602–626 (2005)
15. Khuller, S., Raghavachari, B.: Improved approximation algorithms for uniform connectivity problems. Journal of Algorithms 21(2), 434–450 (1996)
16. Khuller, S., Raghavachari, B.: Graph connectivity. In: Kao, M.Y. (ed.) Encyclopedia of Algorithms. Springer (2008)
17. Khuller, S., Vishkin, U.: Biconnectivity approximations and graph carvings. Journal of the ACM 41(2), 214–235 (1994)
18. Lovász, L., Plummer, M.D.: Matching Theory. North-Holland Mathematics Studies, vol. 121. Elsevier (1986)
19. Marx, D., O'sullivan, B., Razgon, I.: Finding small separators in linear time via treewidth reduction. ACM Transactions on Algorithms 9(4), 30:1–30:35 (2013)
20. Paluch, K., Mucha, M., Mądry, A.: A 7/9 approximation algorithm for the maximum traveling salesman problem. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX and RANDOM 2009. LNCS, vol. 5687, pp. 298–311. Springer, Heidelberg (2009)
21. Papadimitriou, C.H., Yannakakis, M.: Optimization, approximation, and complexity classes. Journal of Computer and System Sciences 43(3), 425–440 (1991)
22. Papadimitriou, C.H., Yannakakis, M.: The traveling salesman problem with distances one and two. Mathematics of Operations Research 18(1), 1–11 (1993)
23. Singh, M., Lau, L.C.: Approximating minimum bounded degree spanning trees to within one of optimal. In: Proc. of the 39th Ann. Int. Symp. on Theory of Computing (STOC), pp. 661–670. ACM (2007)
24. Tutte., W.T.: A short proof of the factor theorem for finite graphs. Canadian Journal of Mathematics 6, 347–352 (1954), http://dx.doi.org/10.4153/CJM-1954-033-3
25. West, D.B.: Introduction to Graph Theory. Prentice-Hall (2001)
26. Williamson, D.P., Shmoys, D.B.: The Design of Approximation Algorithms. Cambridge University Press (2011)

Reordering Buffer Management with Advice

Anna Adamaszek[1,⋆], Marc P. Renault[2], Adi Rosén[3], and Rob van Stee[4,⋆⋆]

[1] Max-Planck-Institut für Informatik, Saarbrücken, Germany
anna@mpi-inf.mpg.de
[2] Université Paris Diderot - Paris 7 and UPMC, France
mrenault@liafa.univ-paris-diderot.fr
[3] CNRS and Université Paris Diderot - Paris 7, France
adiro@liafa.univ-paris-diderot.fr
[4] University of Leicester, Leicester, United Kingdom
rob.vanstee@le.ac.uk

Abstract. In the reordering buffer management problem, a sequence of colored items arrives at a service station to be processed. Each color change between two consecutively processed items generates cost. A reordering buffer of capacity k items can be used to preprocess the input sequence in order to decrease the number of color changes. The goal is to find a scheduling strategy that, using the reordering buffer, minimizes the number of color changes in the given sequence of items.

We consider the problem in the setting of online computation with advice. In this model, the color of an item becomes known only at the time when the item enters the reordering buffer. Additionally, together with each item entering the buffer, we get a fixed number of advice bits, which can be seen as information about the future or as information about an optimal solution (or an approximation thereof) for the whole input sequence. We show that for any $\varepsilon > 0$ there is a $(1+\varepsilon)$-competitive algorithm for the problem which uses only a constant (depending on ε) number of advice bits per input item.

We complement the above result by presenting a lower bound of $\Omega(\log k)$ bits of advice per request for an optimal deterministic algorithm.

1 Introduction

In the *reordering buffer management problem*, a sequence of colored items arrives at a service station for processing. At any time, the service station is configured to process items of a certain color c. Changing the configuration of the service station, i.e., preparing the station to service items of a different color, generates cost. The total cost of processing a sequence of items is equal to the number of color changes in the sequence. In order to reduce the processing cost, the service station is equipped with a *reordering buffer* that has capacity to hold k items. At each point in time, the buffer contains the first k items of the input sequence

⋆ Supported by the Alexander von Humboldt Foundation.
⋆⋆ Work performed while the author was at the Max-Planck-Institut für Informatik, Saarbrücken, Germany.

C. Kaklamanis and K. Pruhs (Eds.): WAOA 2013, LNCS 8447, pp. 132–143, 2014.

that have not yet been processed, and one arbitrary item has to be chosen to be extracted from the buffer and processed by the service station. If the color of the chosen item differs from the color of the previously processed item, the configuration of the service station has to be changed, generating cost. The goal is to find a scheduling strategy that, using the reordering buffer, preprocesses the input sequence to minimize the number of color changes. In the *online setting*, which is considered in this paper, the color of an item becomes known only at the time when the item enters the reordering buffer, i.e., we do not know the whole input sequence in advance.

This framework has many applications in various areas, such as production engineering, storage systems, network optimization and computer graphics (see [3,6,14,17,18] for more details). One simple example is a paint shop of a car manufacturing plant, where switching colors between two consecutively painted cars generates costs and delays due to the necessary cleaning and set-up. Therefore, some paint shops are equipped with a reordering buffer to preprocess the sequence of incoming cars (see [14]).

In this paper, we study the problem in the setting of *online computation with advice* [11]. This setting has received much attention in recent years as it relaxes the traditional online setting of no information about the future, and allows the online algorithm to get some partial information about future requests or about the structure of an optimal (or near-optimal) solution. It also allows one to quantify the amount of information about the future available to the online algorithm and to study the interplay between the amount of information and the attainable competitive ratio. Informally (See Section 2 for a formal definition), in this setting, the online algorithm receives with each request some b bits of *advice* which are the value of a function, defined by the algorithm, of the whole input sequence (including the future). In this manner the online algorithm receives some information about the future. Note that we view the advice as given by an oracle, i.e., we do not consider the issue if the advice can be computed efficiently when knowing the whole input sequence.

Related work. The reordering buffer management problem has been introduced by Räcke et al. [18], and has been extensively studied. In the online setting, the best known results are a deterministic $O(\sqrt{\log k})$-competitive algorithm by Adamaszek et al. [1], and a randomized $O(\log \log k)$-competitive algorithm by Avigdor-Elgrabli and Rabani [5]. To complement this, there are (nearly) matching lower bounds of $\Omega(\sqrt{\log k / \log \log k})$ and $\Omega(\log \log k)$ on the competitive ratio of any online deterministic and randomized algorithms, respectively [1]. In the offline setting, i.e., when the whole input sequence is known in advance, the best known result is a constant factor approximation by Avigdor-Elgrabli and Rabani [4], while the problem is known to be NP-hard [9,2].

More general versions of the problem have been studied, where the context switching cost for switching from an item of color c to an item of color c' depends on c' (e.g., [13,1]), or on both c and c' [12].

The model of online computation with advice considered in the present paper has been introduced by Emek et al. [11]. In that paper, the authors give tight

bounds of $\Theta(\log n/b)$ on the competitive ratio of deterministic and randomized online algorithms with advice for metrical task systems, where n is the number of states of the system and b is the number of advice bits per request. They also give a deterministic online algorithm with advice for the k-server problem which is $k^{O(\frac{1}{b})}$-competitive, where $\Theta(1) \leq b < \log k$. This has been first improved by Böckenhauer et al. [7], and subsequently by Renault and Rosén [19] to $\left\lceil \frac{\lceil \log k \rceil}{b-2} \right\rceil$.

Böckenhauer et al. [8] introduced a somewhat similar model for online algorithms with advice, where the advice is a single tape of bits instead of being given separately for each request. This allows an algorithm to read a different number of bits of advice per request, but it also requires that the online algorithm knows how many bits of advice to read with each request. Thus, the two models are, in general, incomparable. Several results have been given in this related model [10,8,15,16,7]. For example, in [10,8], the authors explore the number of bits of advice required for deterministic and randomized paging algorithms, algorithms for the DiffServ problem, algorithms for a special case of the job shop scheduling problem, and algorithms for the disjoint path allocation problem, to be 1-competitive.

Our Contribution. We give an online algorithm with advice for the reordering buffer management problem that, for any $\varepsilon > 0$, achieves a competitive ratio of $1 + \varepsilon$, using only $O(\log(1/\varepsilon))$ advice bits per request. For any input sequence, we show how to construct the advice, based on an optimal solution for the given sequence, which allows us to obtain a good competitive ratio. The overview of the construction is as follows. The advice bits for each element of a color c encode how the algorithm should handle all "adjacent" elements of color c, i.e., if the algorithm should keep all elements of color c in the buffer until more items of color c arrive, output them at once, or output them, but only after a certain waiting period. The idea is that the order of the colors with the latter property is not contained in the advice of single elements (as that would require too many bits of advice), but it is encoded among all advice bits of the elements of the given color. The key obstacle is that with a small number of advice bits per item we cannot encode the exact order in which the colors should be output, in particular when a small number of elements of some color has to wait for a long time in the buffer. To deal with this problem, we modify the optimal solution by selecting some elements which will be removed earlier from the buffer. That frees additional space in the buffer, which allows us to keep some other elements longer in the buffer, until one more item of their color arrives, after which we can output these elements immediately. That significantly shortens the list of colors which have to be output "soon, but not yet". After this operation, we can encode the desired position in the list for all but a small fraction of elements (which also must be removed earlier from the buffer). We upper bound the increase in the cost of the generated output sequence by charging the additional color changes to the color changes of the optimal solution.

We complement the above result by showing that in order for an online algorithm to have optimal performance, the number of bits of advice must depend

on k. More precisely, $\Omega(\log k)$ bits are required. This lower bound applies even if all the advice bits are given to the algorithm before the sequence starts, and it matches (up to a constant factor) the trivial upper bound, where the advice indicates which color switch has to be performed at each step.

2 The Model

We use the definition of *deterministic online algorithms with advice* as presented in [11]. An online algorithm is defined as a request-answer game that consists of a request set R, a sequence of finite nonempty answer sets $A_1, A_2, \ldots$, and a sequence of cost functions $\text{cost}_n : R^n \times A_1 \times A_2 \times \cdots \times A_n \to \mathbb{R}^+ \cup \{\infty\}$ for $n = 1, 2, \ldots$. In addition, there is an advice space U of size 2^b, where $b \geq 0$ is the number of bits of advice provided to the algorithm with each request. With each request, the online algorithm receives some advice that is defined by a function, $u_i : R^* \to U$, where i is the request index, that is applied to the whole request sequence R^*, including future requests. A deterministic online algorithm with advice can, thus, be represented as a sequence of pairs (g_i, u_i), where g_i is the function defining the action of the online algorithm at step i and is defined $g_i : R^i \times U^i \to A_i$ for $i = 1, 2, \ldots$. The action taken by the online algorithm after receiving request r_i is therefore a function of the first i requests, $r_1, \ldots, r_i$, and the advice received so far, $u_1(\sigma), \ldots, u_i(\sigma)$.

We use the standard definitions of competitive analysis. We say that an algorithm is *α-competitive* if, for every finite request sequence σ, $|\text{ALG}(\sigma)| \leq \alpha \cdot |\text{OPT}(\sigma)| + \zeta$, where ζ is a constant which does not depend on the request sequence σ, $|\text{ALG}(\sigma)|$ denotes the cost of the solution generated by ALG for σ, and $|\text{OPT}(\sigma)|$ is the cost of an optimal solution for σ.

3 Structure of Advice and the Algorithm

The advice for each input element e consists of two parts: the *type* t_e and the *value* v_e. The type of an element e describes how e, or rather a whole collection of elements of the same color as e, should be handled by the algorithm ALG, i.e., if ALG should keep them in the buffer until more elements of the same color arrive, if ALG should output them at once, or output them, but only after a certain waiting period. There are four possible types of an element, i.e., $t_e \in \{\text{WAIT}, \text{LIST}, \text{READY}, \text{COMPLETE}\}$. The value v_e of an element e is used to encode an order, according to which some input elements will be output. For each element e we have $v_e \in \{0, \ldots, D-1\}$, where D is a constant depending on ε and will be fixed later.

The advice sequence will be constructed based on an optimal solution OPT for the input sequence.

Time. In this paper, we use the following notion of time. In each time step $1, \ldots, n$, one element arrives and is stored in the buffer. In each time step $k, \ldots, k+n-1$, one element is removed from the buffer, after the element which arrives at that time (if any) has been stored.

Color Blocks. The elements which have the same color are partitioned into *color blocks*. A *complete color block* is a maximal set of elements of one color ending with an element of type COMPLETE, and containing no other elements of type COMPLETE. The advice sequence will be constructed in such a way that the types of the consecutive elements from one complete color block always form an expression

$$(\text{WAIT})^* \left((\text{LIST})^* \cup (\text{READY})^*\right) \text{COMPLETE},$$

i.e., first there is an arbitrary number (possibly zero) of elements of type WAIT, then an arbitrary number of elements of type LIST or an arbitrary number of type READY, and at the end exactly one element of type COMPLETE.

At a time t, a *color block* is that part of a complete color block which is contained in the buffer (some elements of the complete color block may have been already served, and some may not have been read yet). The notion of a color block here differs from the standard definition of a color block for the reordering buffer management problem, where it denoted the set of all elements of one color contained in the buffer, which were then output consecutively from the buffer. We now can have more than one color block of the same color in the buffer. The advice sequence will be constructed in such a way, that the algorithm ALG will always output all elements of each complete color block consecutively.

We define the *type* t_B *of a color block* B in the following way. If there is an element of type COMPLETE in B, then the type of B is COMPLETE. If all the elements of B have type WAIT, B has type WAIT. Otherwise, the type of B is LIST or READY, depending on whether B contains elements of type LIST or of type READY.

The values v_e of all elements of a color block B together encode a *value of a color block* $v_B \in \{0, \ldots, D^{|B|} - 1\}$. The value of a color block will be encoded only for color blocks containing elements of type LIST, and it will define the order in which these color blocks will be output by the algorithm ALG.

Waiting List. The algorithm ALG maintains a *waiting list* of color blocks of type LIST. This list contains only (a subset of) those color blocks that have the property that some elements of their color are read into the buffer of OPT while OPT is serving it. At the beginning of the algorithm, the waiting list is empty. Whenever a color block B of type LIST appears (i.e. the first element of type LIST for some color block is read from the input), it is inserted into the waiting list. The *initial* position of the block on the waiting list, i.e., the position where the block is inserted, is defined by the value v_B of the block. Value 0 denotes the head of the list. Note that when a new color block is inserted into the waiting list, the position of other blocks on the list can change.

The value v_B of each color block is read from the advice of its elements. However, some short color blocks may not contain sufficiently many elements to encode their required position v_B. We will check for this case and ensure that such blocks are never stored in the waiting list. Instead, we serve parts of them immediately and keep only part in the buffer.

The Algorithm. Starting at time k, the algorithm ALG chooses a color block to be output in the following way.

1. If ALG has a complete color block in the buffer, it chooses the oldest of them (i.e., the block which first became complete is chosen).
2. Otherwise, if ALG has a color block of type READY in the buffer, it chooses the oldest of them (i.e., the block which first obtained type READY is chosen).
3. Otherwise ALG chooses a color block of type LIST which is at the head of the waiting list. The color block is then removed from the waiting list.

The algorithm then outputs consecutively all elements of the chosen color block, one element at each time step. Notice that when ALG starts outputting a color block, possibly some elements of the block have not yet been read from the input. When ALG gets such elements from the input sequence, it appends them to the color block and outputs them without making any color change.

The complete construction of the advice sequence is given in Section 4. In Section 5 we show that the construction of the advice sequence guarantees that ALG can always choose a color block to be output (i.e., there is always a block of type COMPLETE, READY or LIST in the buffer of ALG), and that ALG can always output all elements of a complete color block without any color change in between (Theorem 1). That means in particular that the cost of the algorithm is upper bounded by the number of color blocks in the advice sequence.

4 Constructing the Advice Sequence

Overview. The advice sequence $A_\varepsilon(\sigma)$ is constructed *offline*, based on an *optimal* solution OPT for the instance σ of the reordering buffer management problem. The idea of the construction is as follows. We initially assign each input element e type t_e based on the way OPT handles e. Let c be the color of e. If OPT makes a color change right after outputting e, e is assigned type COMPLETE. Otherwise, if the whole color block containing e is kept in the buffer of OPT until the next element of color c is read from the input sequence, e gets type WAIT. Otherwise, e gets type LIST. This ensures the following invariant.

Invariant 1. *For each color block, its elements are output by* OPT *in a single block (i.e., without making a color change).*

For each color block B which contains elements of type LIST, we want to assign a value v_B, which is the initial position of the block on the waiting list. The blocks in the waiting list are ordered according to the order in which they are output by OPT. For each color block, if the advice included the exact position of the block in the waiting list based on the output sequence of opt, the algorithm ALG would output an optimal solution. However, for some short color blocks, we cannot encode their position on the waiting list. To deal with this problem, we will modify the advice data so as to decrease the number of blocks which are in the waiting list at any time. This modification increases the number of

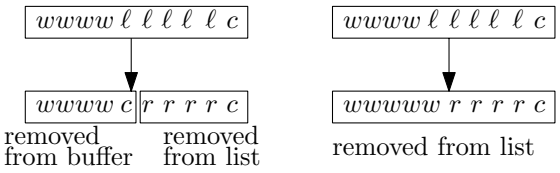

Fig. 1. The procedures Split(B) (on the left) and Postpone(B) (on the right). None of the resulting blocks remain on the waiting list. The letters w, l, r, c are used to denote elements of types WAIT, LIST, READY and COMPLETE, respectively. The frames represent color blocks.

color blocks (i.e., the cost of ALG becomes larger than the cost of OPT), and introduces elements of type READY. Blocks containing elements of type READY are not inserted into the waiting list. We now give an overview of the procedures used to modify the advice data.

Procedure Remove(B) Remove the block B from the waiting list. For each block B' that was inserted into the waiting list at a time later than B, and at a position (in the list) behind B, decrease $v_{B'}$ by one.

Procedure Split(B) Run Remove(B). Reassign type COMPLETE to the first element of block B which had type LIST, and reassign type READY to the remaining elements of B which had type LIST assigned (see Figure 1, left). Note that possibly some of these elements have not been read from the input yet.

The block B has been split into two blocks, both ending with an element of type COMPLETE. The elements of the first block (called the *early* block) are removed from the buffer. The second block is called a *late* block. This block is kept in the buffer (but not on the waiting list).

Procedure Postpone(B) Run Remove(B). Reassign type WAIT to the first element of B which originally had type LIST assigned, and reassign type READY to the remaining elements of B which had type LIST assigned (see Figure 1, right). Possibly some of these elements have not been read from the input yet. Blocks processed in this way are called *postponed blocks*.

The procedure Split(B) makes ALG evict the elements from the first block of B (i.e., the early block) earlier than they were evicted by OPT, generating free space in the buffer of ALG. This allows some blocks to stay in the buffer until the next element of the block is read from the input, and only then to be output by ALG. These are the blocks for which we run procedure Postpone(B). We will specify later which blocks will be treated in this way. A block B that is removed from the waiting list by these procedures will never be inserted into the waiting list of ALG. This is the reason we update the values $v_{B'}$ of other blocks on the waiting list when applying procedure Remove(B). We have the following observation.

Observation 1. *The procedures Split(B) and Postpone(B) maintain Invariant 1.*

We are now ready to describe the details of the construction of the advice sequence $A_\varepsilon(\sigma)$. The input elements from the sequence σ are processed one by one, using a buffer of size k (which we call an *advice buffer*) and a waiting list. Notice that at this point we are only creating an advice sequence, that is, assigning type t_e and value v_e for each input element e, and not creating an output sequence. In particular, "removing elements from the buffer" does not mean "appending elements to the output sequence".

Processing an Input Element. Whenever there is empty space in the buffer, we read the next element e from the input sequence σ. We proceed as follows. First, we assign the initial type of the element e as described above, based on the way OPT handles e. Then, if e is the first element in its block which is of type LIST, i.e., its color block changes type to LIST, we insert the color block of e at the appropriate position, based on the optimal output sequence OPT, into the waiting list. After this operation, if the waiting list has too many blocks of length similar to the one added, we remove all such blocks from the waiting list by applying procedure Split(B) to some of the blocks, and procedure Postpone(B) to the remaining blocks of the given length. We use the following definition.

Definition 1. *The* class *of a block B on the waiting list is $\lfloor \log |B| \rfloor$, where $|B|$ is the number of items of this block up to and including the first element of type* LIST.

We now give the detailed description of actions performed for a newly read element e of color c. Here C is another constant which will depend on ε.

1. (*Type assignment*) We assign the element e type t_e based on the way OPT handles e (see Figure 2).
   (a) If OPT makes a color change right after outputting e, e is assigned type COMPLETE.
   (b) Else, if e has already been assigned type READY due to some previous Split or Postpone operation (i.e., before e has been read into the buffer), it keeps the type READY.
   (c) Else, if the whole color block containing e is kept in the buffer of OPT until the next element element of color c is read from the input sequence, e gets type WAIT.
   (d) Else, e gets type LIST.

   Note that the type of the element e can be modified later by applying procedure Split(B) or Postpone(B) to the color block B of e.
2. (*Insertion into the waiting list*) If e is the *first* element of its color block of type LIST, insert the color block B of e into the waiting list at position v_B (derived from the solution OPT). Let i be the class of block B. Count the number n_i of color blocks of class i which are currently in the waiting list.

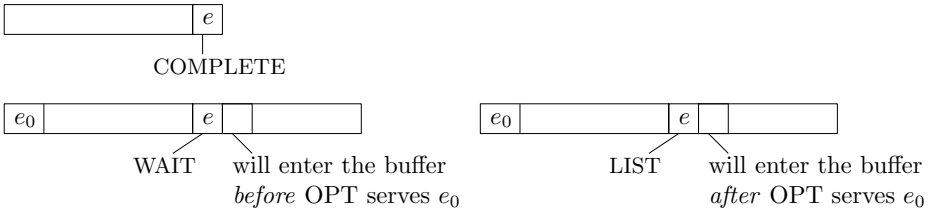

Fig. 2. Possible initial types of elements (excluding the special case of READY). The type LIST indicates that OPT starts serving this color block before all elements have been read into the buffer; the remaining elements enter the buffer while the color block is being served (and while other elements of the color block are being removed).

If $n_i = C$, let B_1 and B_2 be the two last color blocks of class i in the waiting list. Run Split(B_1) and Split(B_2). For each other block B_0 of class i on the waiting list, run Postpone(B_0). By performing these operations, we ensure that ALG will never insert these C blocks into the waiting list.

Note that in step 2 some blocks may be split, with the resulting early blocks being removed from the buffer.

Whenever the advice buffer becomes full, and there are no elements of the current color block in the buffer, we choose a new color block B to be processed (i.e., removed from the buffer), according to the rules of ALG. The following lemma shows that the rules of ALG can always be successfully applied.

Lemma 1. *While creating the advice sequence, if blocks are removed from the buffer, according to the rules of* ALG *and Step 2 above, then starting from time k there is always an element of type different than* WAIT *in the buffer.*

Let B be the color block chosen from the full advice buffer according to the rules of ALG. If B is not on the waiting list, we simply remove it from the buffer. If B is on the waiting list, we have to consider two cases.

1. The value v_B is smaller than $D^{|B|}$, i.e., it can be encoded using $|B|$ values v_e of the elements of the block. We set appropriately the values v_e, and we remove B from the buffer and from the waiting list. (Note that this time we do not apply the procedure Remove(B), i.e., we do not modify any values $v_{B'}$ of blocks B'.)

   Only blocks processed in this way will ever be inserted by ALG into the waiting list.
2. The value v_B is at least $D^{|B|}$. Run Split(B). In this case, only the early block obtained from B is removed from the buffer.

Notice that, as above, in determining the length of B in Step 1, we count only the elements up to and including the first element of type LIST. The reason is that these are the only elements of B that have been inserted into the buffer before B has to be inserted into the waiting list, i.e., before the value of B has to be computed.

Note also that the initial value of v_B might decrease (due to calls to the procedure Remove when B is already in the waiting list) before B is processed by ALG. Specifically, it does not matter if v_B was initially too high, as long as it drops below $D^{|B|}$ before the block is chosen to be processed.

5 Analysis of ALG

Lemma 1 leaves open the possibility that ALG may be able to serve only *part* of a block. This could happen if not all elements of the (complete) block enter the buffer while ALG is serving its color. In such a case, ALG would have to return to this color several times, and its cost would be higher than the number of color blocks given by the advice sequence $A_\varepsilon(\sigma)$. At the beginning of this section, we will show that this does not happen, and ALG always outputs complete color blocks. Then, we will bound the competitive ratio of ALG.

First, we present two technical lemmas.

Lemma 2. *In the advice sequence $A_\varepsilon(\sigma)$, types of the consecutive elements from one complete color block always form an expression*

$$(\text{WAIT})^* \left((\text{LIST})^* \cup (\text{READY})^*\right) \text{COMPLETE}.$$

Lemma 3. *Let $c_1, \ldots, c_\ell$ be a collection of consecutive elements of color c, output by* OPT *with no color change in between, and let t be the time when* OPT *outputs the first element c_1 of the collection. If* ALG *outputs c_1 at some time $t' \geq t$, and it has not output any c_i before time t', it can output all elements $c_1, \ldots, c_\ell$ with no color change in between.*

We can now consider all types of blocks which ALG starts outputting before they become complete, and show that OPT cannot keep the elements of such blocks longer in the buffer. Applying Lemma 3 gives us the following results.

Lemma 4. *When the algorithm* ALG *starts outputting a color block, it can finish it with no color changes.*

Theorem 1. *The algorithm* ALG *always finds a color block to be output, and outputs only complete color blocks.*

In the remaining part of this section, we will bound the competitive ratio of ALG, if ALG uses the advice sequence $A_\varepsilon(\sigma)$. For this, we will need an upper bound on the cost of ALG, as well as a lower bound on the cost of OPT. From the construction of the advice sequence, we know the following.

Observation 2. *The cost of* ALG *is upper bounded by the number of color blocks given by the advice sequence $A_\varepsilon(\sigma)$.*

Observation 3. *The cost of* OPT *is lower bounded by the number of color blocks other than the late blocks given by the advice sequence $A_\varepsilon(\sigma)$.*

To bound the competitive ratio, it is enough to bound the number of late color blocks in $A_\varepsilon(\sigma)$, compared to the number of color blocks of other types. Late blocks can be generated in two ways: while inserting new elements into the waiting list, and while processing a color block at the head of the waiting list. We call the sets of late blocks generated during these two operations by LATE_I and LATE_P, respectively.

Lemma 5. *We have* $|\text{LATE}_I| \leq \frac{2}{C}(|\text{POSTPONED}| + |\text{EARLY}|)$, *where* POSTPONED *and* EARLY *denote the sets of postponed and early blocks for* $A_\varepsilon(\sigma)$.

Lemma 6. *We have* $|\text{LATE}_P| \leq \frac{C}{D-1}|\text{LISTED}|$, *where* LISTED *denotes the set of color blocks which contain elements of type* LIST.

Combining the two Lemmas above and setting $C = \lceil 2/\varepsilon \rceil$ and $D = \lceil C/\varepsilon \rceil + 1$ gives us the main theorem of the paper.

Theorem 2. *For any* $\varepsilon > 0$, *there is a* $(1+\varepsilon)$*-competitive algorithm for the reordering buffer management problem which uses only* $O(\log(1/\varepsilon))$ *advice bits per input item.*

6 Optimality Lower Bound

In this section, we show that a deterministic algorithm requires $\Omega(\log k)$ bits of advice per request in order to be optimal. Throughout this section, we assume without loss of generality that an algorithm will only perform a color switch when there are no more items of the current color in the buffer, and the buffer is either full or contains the last request. Let k be the size of the buffer, and let

$$\sigma = \big\langle c_1, c_2 \ldots, c_k, \pi_1(1), c_{k+1}, \pi_1(2), c_{k+2} \ldots, \pi_1(k), c_{2k}, \pi_2(1), \pi_2(2), \ldots, \pi_2(k) \big\rangle,$$

where π_1 is a permutation of the colors $c_1, \ldots, c_k$, and π_2 is a permutation of the colors $c_{k+1}, \ldots, c_{2k}$. Note that $|\sigma| = 4k$.

As long as the buffer of an optimal algorithm remains full, the algorithm must switch to the color c_i when the next element waiting to enter the buffer has color c_i. In particular, the first k consecutive colors output by the algorithm must be $\pi_1(1), \ldots, \pi_1(k)$. More formally,

Lemma 7. *Given* σ, *let* S *be the sequence of colors* $\pi_1(1), \ldots, \pi_1(k)$. *Let* $\mathcal{C}$ *be the set of all the possible optimal sequences of color switches for* σ. *The* k*-prefix of all the sequences in* $\mathcal{C}$ *is* S.

Using Lemma 7, we can prove the main theorem of this section.

Theorem 3. *At least* $\frac{\log k}{8}$ *bits of advice per request are required for a deterministic algorithm with advice to be optimal.*

References

1. Adamaszek, A., Czumaj, A., Englert, M., Räcke, H.: Almost tight bounds for reordering buffer management. In: Fortnow, L., Vadhan, S.P. (eds.) STOC, pp. 607–616. ACM (2011)
2. Asahiro, Y., Kawahara, K., Miyano, E.: NP-hardness of the sorting buffer problem on the uniform metric. Discrete Applied Mathematics 160(10-11), 1453–1464 (2012)
3. Avigdor-Elgrabli, N., Rabani, Y.: An improved competitive algorithm for reordering buffer management. In: Charikar, M. (ed.) SODA, pp. 13–21. SIAM (2010)
4. Avigdor-Elgrabli, N., Rabani, Y.: A constant factor approximation algorithm for reordering buffer management. In: Khanna, S. (ed.) SODA, pp. 973–984. SIAM (2013)
5. Avigdor-Elgrabli, N., Rabani, Y.: An optimal randomized online algorithm for reordering buffer management. CoRR abs/1303.3386 (2013)
6. Blandford, D.K., Blelloch, G.E.: Index compression through document reordering. In: DCC, pp. 342–351. IEEE Computer Society (2002)
7. Böckenhauer, H.J., Komm, D., Královic, R., Královic, R.: On the advice complexity of the k-server problem. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 207–218. Springer, Heidelberg (2011)
8. Böckenhauer, H.-J., Komm, D., Královič, R., Královič, R., Mömke, T.: On the advice complexity of online problems. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 331–340. Springer, Heidelberg (2009)
9. Chan, H.-L., Megow, N., Sitters, R., Stee, R.v.: A note on sorting buffers offline. Theor. Comput. Sci. 423, 11–18 (2012)
10. Dobrev, S., Královič, R., Pardubská, D.: How much information about the future is needed? In: Geffert, V., Karhumäki, J., Bertoni, A., Preneel, B., Návrat, P., Bieliková, M. (eds.) SOFSEM 2008. LNCS, vol. 4910, pp. 247–258. Springer, Heidelberg (2008)
11. Emek, Y., Fraigniaud, P., Korman, A., Rosén, A.: Online computation with advice. Theor. Comput. Sci. 412(24), 2642–2656 (2011)
12. Englert, M., Räcke, H., Westermann, M.: Reordering buffers for general metric spaces. Theory of Computing 6(1), 27–46 (2010)
13. Englert, M., Westermann, M.: Reordering buffer management for non-uniform cost models. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 627–638. Springer, Heidelberg (2005)
14. Gutenschwager, K., Spiekermann, S., Voß, S.: A sequential ordering problem in automotive paint shops. Internat. J. Production Research 42(9), 1865–1878 (2004)
15. Hromkovič, J., Královič, R., Královič, R.: Information complexity of online problems. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 24–36. Springer, Heidelberg (2010)
16. Komm, D., Královic, R.: Advice complexity and barely random algorithms. RAIRO - Theor. Inf. and Applic. 45(2), 249–267 (2011)
17. Krokowski, J., Räcke, H., Sohler, C., Westermann, M.: Reducing state changes with a pipeline buffer. In: Girod, B., Magnor, M.A., Seidel, H.P. (eds.) VMV, p. 217. Aka GmbH (2004)
18. Räcke, H., Sohler, C., Westermann, M.: Online scheduling for sorting buffers. In: Möhring, R., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 820–832. Springer, Heidelberg (2002)
19. Renault, M.P., Rosén, A.: On online algorithms with advice for the k-server problem. Theory of Computing Systems, 1–19 (2012)

Online Knapsack Revisited*

Marek Cygan[1] and Łukasz Jeż[2]

[1] Institute of Informatics, University of Warsaw, Poland
cygan@mimuw.edu.pl
[2] Inst. of Computer Science, University of Wrocław (PL),
and Dept. of Computer, Control, and Management Engineering,
Sapienza University of Rome (IT)
lje@cs.uni.wroc.pl

Abstract. We investigate the online variant of the *Multiple Knapsack* problem: an algorithm is to pack items, of arbitrary sizes and profits, in k knapsacks (bins) without exceeding the capacity of any bin. We study two objective functions: the sum and the maximum of profits over all bins. Both have been studied before in restricted variants of our problem: the sum in *Dual Bin Packing* [1], and the maximum in *Removable Knapsack* [7,8]. Following these, we study two variants, depending on whether the algorithm is allowed to remove (forever) items from its bins or not, and two special cases where the profit of an item is a function of its size, in addition to the general setting.

We study both deterministic and randomized algorithms; for the latter, we consider both the oblivious and the adaptive adversary model. We classify each variant as either admitting $O(1)$-competitive algorithms or not. We develop simple $O(1)$-competitive algorithms for some cases of the max-objective variant believed to be infeasible because only 1-bin deterministic algorithms were considered for them before.

1 Introduction

Knapsack Problems form a fundamental class of problems in computer science, and entire books [13,10] are dedicated to them. We focus on an online variant of the *Multiple Knapsack* problem, in which k knapsacks (bins) of integer *capacities* C_i and a set of items of arbitrary integer *sizes* and *profits* are given, and the goal is to find k disjoint subsets of items, one per bin, that fit in their bins such that the total profit of all k sets is maximized.

In the online variant that we study, we relax the assumption that all numbers are integers, and allow real numbers instead. On the other hand, we restrict the problem significantly by assuming that *all bins have the same capacity*. This assumption is motivated by the fact that without any restrictions of this kind, we would have to take into account instances where all but one bin have capacities so small that the can accommodate no item. Clearly, such instances are at least as hard as those with only a single bin, for which there are many impossibility

* Most of this work was carried out while authors were at IDSIA, University of Lugano.

C. Kaklamanis and K. Pruhs (Eds.): WAOA 2013, LNCS 8447, pp. 144–155, 2014.

results. With all bins having the same capacity, we normalize it and the item sizes so that the capacity of each bin is 1.

The online algorithm is given the items one after another, at which time it learns the item's size and profit. When an item is presented, the algorithm is required to immediately accept or reject it. In the former cases, it also has to choose a bin to put the item into. The algorithm is not allowed to move the items between the bins at any time but it is allowed to remove any item from any bin at any time in the *removable* variant. An item removed from a bin is treated as if it was never placed there, i.e., it cannot be brought back to the bin. In the *non-removable* variant, an algorithm is not allowed to remove items from bins: once put in a bin, an item remains there forever.

In both variants, we consider two objective functions: the sum and the maximum profit over all bins, denoted *sum*-objective and *max*-objective respectively; the profit of the bin is the sum of profit of all items currently in it. Note that both objective functions coincide for $k = 1$; in such case we do not state which one we are considering. Moreover, we remark that under the *max*-objective increasing k cannot increase the competitive ratio, cf. Section 2.1.

The study of online variants of the knapsack problem is motivated by the richness of its applications and the need of efficiently solving very large instances, for which data may only be accessible at bunches from external storage [12]. Thus the potential for improving the competitive ratio at little computational cost by removals, randomization or additional bins motivates their study.

We investigate general instances and, following existing literature, some restricted classes. When each item in the instance has profit equal to its size, we call the instance or case *proportional*; in previous studies such instances have been called either uniform or unweighted [7,8]. When each item in the instance has the same profit, irrespective of the size, we call the instance or case *unit*.

1.1 Previous and Related Results

***Max*-objective.** No *deterministic* algorithm for 1-bin non-removable problem (the classic online knapsack problem) is $O(1)$-competitive [11,12], even in the proportional and unit case. For the 1-bin proportional case a tight bound of 2 is known for randomized algorithms in the oblivious adversary model [4]. However, that work focuses on the advice complexity rather than randomized algorithms, and prior studies focused on deterministic algorithms for the removable variant.

Even in the removable variant, there is no $O(1)$-competitive 1-bin deterministic algorithm [7,8], so Iwama et al. [7] focused on deterministic algorithms for its proportional variant. They obtained a tight bound of $\phi \approx 1.618$ for the 1-bin variant, and also a tight bound of (approximately) 1.3815 for the k-bin variant where $k \geq 2$; we note that their optimal algorithm for the latter case uses only two bins even if more are allowed.

Iwama et al. [8] also studied the effect of resource augmentation for 1-bin settings, i.e., letting a deterministic algorithm use a bin of capacity $C > 1$ while comparing its profit with the optimum profit attained with a bin of capacity 1. These results are incomparable with ours: on the one hand, a single bin of

capacity C is superior to C bins of capacity 1 each, on the other, C need not be an integer; in particular, in all the variants where such large bin helps, $C \leq 2$ suffices to achieve ratio 1.

Partially fractional online knapsack is a related problem, which differs from our setting only by allowing the algorithm to accept any fraction of any item; the general case of its removable variant has been studied for 1 bin of capacity $C \geq 1$ (i.e., with resource augmentation) [14].

***Sum*-objective.** The *sum*-objective has been studied for the *Dual Bin Packing* problem [1], which coincides with the unit case of our non-removable variant. Since there is no $O(1)$-competitive algorithm for this setting, those studies focused on the so called *accommodating sequences*, in which Opt can fit all the items in its k bins or, in a generalization, $c \cdot k$ bins for some constant c.

Moreover, as the problems that we consider are fundamental and our algorithmic solutions simple, similar algorithms have appeared in various contexts, for example in ad auctions [2,6].

1.2 Our Results

We study both objective functions, *sum*-objective and *max*-objective, for both variants, removable and non-removable, for both general and restricted instance classes. Furthermore, we study both deterministic and randomized algorithms, distinguishing between the "standard" oblivious and the less common adaptive adversary model for the latter; see the article that distinguished the models [3] or the textbook on online algorithms [5] for definition of the adaptive (online) adversary. Finally, we consider different numbers of bins. We classify all these settings into "feasible", which admit $O(1)$-competitive algorithms, and "infeasible", which do not, by giving upper bounds for all feasible settings and lower bounds for all settings, or pointing to any previously known bounds.

All relevant results are summarized in Table 1 (and its caption; some are not listed in the table). Here we only note a few features. The proportional case, studied by Iwama et al. for one and multiple bins [7] with removals, is feasible even without removals if $k > 1$ or randomization is allowed. The general case of the removable variant exhibits similar properties.

We remark that we obtain simple $O(1)$-competitive randomized algorithms for some 1-bin settings that are infeasible for deterministic algorithms. This is a rare phenomenon, exhibited only by a single other *natural* problem we are aware of. Namely, online throughput maximization on a single machine [9], where the constant hidden in $O(1)$ is large and the analysis quite involved.

Due to space constraints, some of the proofs are omitted.

2 Preliminaries

For an item i, we denote its profit by p_i and its size by s_i. In case of sequences of items, say $i_1, i_2, \ldots, i_n$, we shall abbreviate p_{i_j} and s_{i_j} to p_j and s_j.

Table 1. Summary of previous and our results. Column 'var' describes the variant (R: removable, NR: non-removable), 'obj' the objective (while the meaning of max and $\sum$ is clear, note that it implicitly regards $k > 1$; 1 as 'obj' means the single bin setting, in which both objectives coincide), in the next column 'gen.' and 'prop.' stand for general and proportional case respectively. In 'max' rows, all upper bounds are attained for $k = 2$ whereas all lower bounds hold for all k. For this reason, Theorem 5, which gives a lower bound matching Theorem 3 for $k = 2$ does not appear in the table. The table does not present our results on the adaptive adversary model (Theorems 14 and 15), which show that for $k = 1$ randomization does not help against such adversary. The results without a reference are either folklore or follow from other entries.

var	obj	case	deterministic lower	deterministic upper	randomized (oblivious) lower	randomized (oblivious) upper
R	1	gen.	∞	–	$\frac{e+1}{e} \approx 1.37$ [Thm. 7]	2 [Sec. 3.1]
		prop.	$\phi \approx 1.62$	$\phi \approx 1.62$	1.25 [Thm. 8]	$\phi \approx 1.62$
		unit	1	1	1	1
	max	gen.	$\phi \approx 1.62$ [Thm. 6]	2 [Thm. 3]	1	2
		prop.	≈ 1.38 [7]	≈ 1.38 [7]	1	≈ 1.38
		unit	1	1	1	1
	$\sum$	gen.	$\frac{7}{6} \approx 1.17$	$3 + O(\frac{1}{k})$ [Thm. 2]	$\frac{7}{6} \approx 1.17$	3 [Thm. 2]
		prop.	$\frac{8}{7} \approx 1.14$	1.5 [15]	$\frac{8}{7} \approx 1.14$ [Thm. 4]	2
		unit	$\frac{7}{6} \approx 1.17$	$3 + O(\frac{1}{k})$	$\frac{7}{6} \approx 1.17$ [1]	3
NR	1	gen.	∞	–	∞	–
		prop.	∞	–	2 [4]	2 [4]
		unit	∞	–	∞	–
	max	gen.	∞	–	∞	–
		prop.	2	2 [4]	2 [Thm. 12]	2
		unit	∞	–	∞ [Thm. 11]	–
	$\sum$	gen.	∞	–	∞	–
		prop.	$1 + \ln 2 \approx 1.69$	2 [Thm. 10]	$1 + \ln 2$ [Thm. 13]	2
		unit	∞	–	∞ [Thm. 11]	–

For any positive integer k, the set of bins available to a k-bin algorithm is $\mathcal{B}_k = \{B_1, B_2, \ldots, B_k\}$; if $k = 1$ we shall denote the sole bin by B rather than B_1. Given a set of items X, we shall denote the total profit and the total size of the items in X by $p(X)$ and $s(X)$, i.e., $p(X) = \sum_{e \in X} p_e$ and $s(X) = \sum_{e \in X} s_e$ respectively. In most of our analyses, the set X will be the set of all items in a bin; in such case we shall simply write $p(B)$ and $s(B)$ for a bin B, ignoring the particular instance and step in notation, since they will be clear from the context. We extend this notation further to sets of bins: if $\mathcal{B}$ is a subset of the set of all bins, then $p(\mathcal{B}) = \sum_{B \in \mathcal{B}} p(B)$ denotes the total profit of all the items in the bins of $\mathcal{B}$ and similarly $s(\mathcal{B}) = \sum_{B \in \mathcal{B}} s(B)$ denotes their total size. For an algorithm Alg, we will use Alg to denote both the algorithm and its profit; in particular this applies to the offline optimum, denoted by Opt. Unless otherwise stated, all the algorithms are presented assuming there are k bins available.

2.1 Max-objective and Different Numbers of Bins

Note that in the *max*-objective problem, the optimum offline solution can use only a single bin. Therefore, in the oblivious adversary model, an R-competitive k-bin algorithm (which may be randomized) for any variant of the problem remains R-competitive when $\ell > k$ bins are allowed, by ignoring the extra bins. On the other hand, running an arbitrary k_1-bin algorithm and k_2-bin algorithm in parallel does not result in a $(k_1 + k_2)$-bin algorithm, since the former two algorithms might *conflict* by accepting the very same item.

However, if $k < \ell$, one can obtain many k-bin algorithms from a given ℓ-bin algorithm through projections: let ALG be an ℓ-bin algorithm Then, for any $\mathcal{B} \subset \mathcal{B}_\ell$, let $\pi(\text{ALG}, \mathcal{B})$ be a $|\mathcal{B}|$-bin algorithm defined as follows: for every bin $B \in \mathcal{B}$, $\pi(\text{ALG}, \mathcal{B})$ simulates ALG on B, ignoring the items that ALG does not place in B. For the oblivious adversary model, it is easy to see that if ALG is an R-competitive ℓ-bin algorithm, then choosing $\mathcal{B}'$ of cardinality k from its bins uniformly at random yields an $R \cdot \ell/k$-competitive barely random algorithm.

If $k = 1$, using ℓ single-bin algorithms is less restrictive, since they are allowed to conflict. In particular, if ALG_1 and ALG_2 are (deterministic) 1-bin algorithms such that at any point $\text{ALG}_1 + \text{ALG}_2 \geq \text{OPT}$, then a barely random 1-bin algorithm that simulates one of them chosen uniformly at random, is 2-competitive.

3 Removable Variants

3.1 Upper Bounds

Let us discuss restricted instances first. For *max*-objective, optimum bounds for proportional case were given by Iwama et al. [7], whereas the unit case is trivial: a deterministic 1-bin algorithm that tentatively puts all new items in its bin and then repeatedly removes the largest items while the bin overflows is 1-competitive. We do not consider restricted instances for *sum*-objective explicitly.

We focus on general instances, for both objective functions. We develop a $\left(3 + O(\frac{k \mod 3}{k})\right)$-competitive deterministic algorithm for *sum*-objective, whose barely random variant is 3-competitive irrespective of k. Then we consider the *max*-objective and note that for $k = 2$ bins the same algorithm is 2-competitive (and optimal). Moreover, while aforementioned randomized algorithm is 3-competitive for all k, we note that for $k = 1$ its competitive ratio can be improved to 2 by altering the probability distribution.

To state our algorithm, let us classify items as follows: an item is *small* if its size is at most $1/2$, otherwise it is *large*. Essentially, we are going to use two well known algorithms, extended to many bins, to handle each type of items.

Algorithm GREEDY: When a new item e is issued, while there is not enough space in any bin to put e there, remove the item $e' \in \{e\} \cup \bigcup_{i=1,\ldots,k} B_i$ that minimizes $p(e')/s(e')$ from its bin; when that item is e itself, which has no bin, stop. If e was not removed, put in the bin that can now fit it.

Algorithm PGREEDY: Maintain k most profitable items, one per bin, as follows. When a new item e is issued, put it an empty bin if there is one. Otherwise, if $p(e) > \min_{i=1,\ldots,k} p(B_i)$, replace the minimum-profit item with e.

Lemma 1. *Let $e_1, e_2, \ldots, e_n$ be a sequence of items such that $p_1/s_1 \geq p_2/s_2 \geq \ldots \geq p_n/s_n$. Given these items* in any order*,* GREEDY *will have $e_1, e_2, \ldots, e_{n_0}$ in its bins, where either $n_0 = n$ or $s(B_i) > 1 - \max_{1 \leq j \leq n} s_j$ for $i = 1, \ldots, k$. Finally, for any set of items X from the sequence,*

$$p(X) \leq \frac{p(\mathcal{B}_k)}{s(\mathcal{B}_k)} \cdot \max\{s(\mathcal{B}_k),\ s(X)\} \ .$$

Our algorithm is a combination of the two above.

Algorithm MULTIGREEDY: Use PGREEDY on $\lceil k/3 \rceil$ bins for large items and GREEDY on the remaining $\lfloor 2k/3 \rfloor$ bins for small items.

Theorem 2. MULTIGREEDY *is $R(k)$-competitive for the* sum*-objective, where*

$$R(k) = \begin{cases} 3 & \text{if } k \equiv 0 \\ 3 + \frac{3}{k-1} & \text{if } k \equiv 1 \\ 3 + \frac{3}{2k-1} & \text{if } k \equiv 2 \end{cases} \pmod 3 \ .$$

A barely random variant of the algorithm is 3-competitive for all k.

Proof. We focus on MULTIGREEDY first, introducing and analyzing the random variant afterwards. Let L (L^*) and S (S^*) denote ALG's (OPT's) profit for large and small items, respectively. Note that the following invariant is clearly maintained: the MULTIGREEDY algorithm has the $\lceil k/3 \rceil$ most profitable large items. Moreover, by Lemma 1, at each step MULTIGREEDY either has all the small items, or all of its $\lfloor 2k/3 \rfloor$ bins dedicated to small items are at least half-full.

To bound the competitive ratio, we observe that $(L^* + S^*)/(L + S) \leq \max(L^*/L, S^*/S)$, hence we focus on large and small items separately. OPT can have at most k large items, so $L^*/L \leq k/\lceil k/3 \rceil$. As for the small items, Lemma 1 implies $S^*/S \leq 2k/\lfloor 2k/3 \rfloor$.

The bound on MULTIGREEDY's competitive ratio follows by determining which of the two upper bounds is larger for each possible remainder. The barely random algorithm works like MULTIGREEDY, but it randomly partitions the bins into "small" and "large" in the very beginning. By previous analysis, it is easy to observe that its competitive ratio is no larger than the maximum of $L^*/\mathbb{E}[L]$ and $S^*/\mathbb{E}[S]$, and that this number is at most 3 if the expected number of "large" and "small" bins is 1/3 and 2/3 respectively. □

Theorem 3. MULTIGREEDY *with* 2 *bins is* 2*-competitive for the* max*-objective.*

Proof. We refine the proof of Theorem 2. Note that for $k = 2$ MULTIGREEDY uses one bin for large items and one bin for small items. On the other hand, OPT only uses a single bin. If OPT does not use any large item in its solution, then 2-competitiveness of MULTIGREEDY follows from the proof of Theorem 2.

Otherwise, OPT uses a single large item, say x. Then MULTIGREEDY has some large item y in its "large" bin, and $p_y \geq p_x$. Furthermore, the total size of OPT's small items is at most $1 - s_x < 1/2$. With this in mind, it follows again from the proof of Theorem 2 that $S \geq S^*$. Hence, $L + S \geq L^* + S^* = \text{OPT}$, and consequently $\text{MULTIGREEDY} = \max\{L, S\} \geq \text{OPT}/2$.

By Theorem 2, a barely random 1-bin variant of MultiGreedy is 3-competitive. But a trivial 2-approximation algorithm for *Knapsack* gives rise to a 2-competitive barely random 1-bin algorithm: toss a fair coin and, depending on the result, simulate either 1-bin Greedy or 1-bin PGreedy.

3.2 Lower Bounds

Note that proving a $k/(k-1)$ lower bound for deterministic algorithms with the *sum*-objective is straightforward. First issue k items of size 1 and profit 1, and then keep issuing items of size ϵ^2 and profit ϵ. Eventually, the gain for those items alone becomes huge, so at some point Alg has to remove one of large items. At that point the ratio is at least $k/(k-1+\epsilon)$.

Moreover, it follows from known results on *dual bin packing* [1] that even for *randomized* algorithms for the unit case there is a lower bound of 7/6. Adapting that construction yields a slightly weaker bound for proportional case.

Theorem 4. *For any k and the* sum*-objective, no randomized k-bin algorithm has competitive ratio smaller than $\frac{8}{7}$ in the proportional case.*

Now we turn our attention to lower bounds for the *max*-objective.

Theorem 5. *No deterministic 2-bin algorithm has competitive ratio smaller than* 2 *for the* max*-objective.*

Proof. The adversary's strategy involves two interwoven sequences of items. To define them, let us fix arbitrarily small $\epsilon > 0$. Then the sequences are

small: each item has size ϵ and profit $\sqrt{\epsilon}$.
large: the i-th ($i \geq 0$) item has size $1 - \epsilon \cdot (1 - 2^{-i})$ and profit $1 - i \cdot \sqrt{\epsilon}$.

Whenever we say that a small (large) item is issued, we mean the successive item from the respective sequence. Initially a small item and a large item are released, in any order. Observe that no large item can be placed together with a small item in one bin, and consequently (wlog) Alg puts the large item in one bin and the small one in the other. From this point, the adversary's strategy is as follows: if Alg has a small item in one of its bins, issue a large item, otherwise issue a small item.

Note that by this strategy Alg always has (wlog) a large item in one bin, and either another large item or a *single* small one in the other bin; we denote these two states of Alg by LL and LS respectively.

We claim that Alg's ratio is at least $2/(1+2\sqrt{\epsilon})$, which tends to 2 as ϵ tends to zero. Assume, for contradiction, it is not so. Then Alg behaves in such a way that no more than $2/\sqrt{\epsilon}$ small items are issued in total, since then the total profit of all the small items would be at least 2, whereas Alg's profit is at most 1. Therefore, eventually Alg loops in LS state, since a small item is issued every time Alg is in LL state.

Once Alg loops in LS state, large items (with slowly decreasing weights) are released in each step, and eventually their profits drop below 0.5. Since Opt ≥ 1

due to the first large item and ALG is 2-competitive, there is a first step when it does not replace its large item with the new one while in LS state. Denote the large item that ALG keeps by l and the next large one it forfeits by l', and note that $p_{l'} = p_l - \sqrt{\epsilon} \geq \frac{1}{2}$, and $s_{l'} < s_l$. Right after l', its "complement" l'' is released, with $s_{l''} = 1 - s_{l'}$ and $p_{l''} = p_{l'}$, and the sequence ends. ALG cannot put l'' together with l, so even if it puts l'' together with a small item in the other bin, its profit is at most p_l, whereas OPT's is at least $2p_{l'}$, so the ratio is at least $2/(1 + 2\sqrt{\epsilon})$. This is the final contradiction. □

Notice that Theorem 5 matches the upper bound of Theorem 3. However, the former applies to $k = 2$ only. When more than two bins are available, we can only prove the following weaker lower bound.

Theorem 6. *For every $k \geq 3$, no deterministic k-bin algorithm has competitive ratio smaller than $\phi = (1 + \sqrt{5})/2 \approx 1.618$ for the* max*-objective.*

Proof. Assume, for the sake of contradiction, that ALG is R-competitive for $R = \phi - \epsilon$ where $\epsilon > 0$. Consider the following instance. Initially, two items of size $1/2$ are issued, one of profit 1 and the other of profit ϕ. Since $1 + \phi = \phi^2$ and $R < \phi$, ALG puts both these items into a single bin, say B_1.

Afterwards, a sequence of "small" items of size ϵ^2 and profit ϵ is issued until one of the following happens: either the total profit of all the small items issued so far exceeds ϕ^2 or ALG removes an item (wlog of profit 1) from B_1.

In the first case, ALG's profits for each of its bins is no larger than $\phi^2 + \epsilon$, while the optimum solution uses the large item of profit ϕ and all the small items for a total profit of at least $\phi + \phi^2 = \phi^3$, which gives ratio larger than $R = \phi - \epsilon$, a contradiction.

Thus we focus on the other case: suppose that the total profit of all the small items released by the moment when ALG removes the item from B_1 is x. At that moment the sequence ends, and ALG's profit for any bin other than B_1 is at most x, while the one for B_1 is at most $\phi + \epsilon$. On the other hand, the optimum solution uses the large item of profit ϕ and, depending on the value of x, either the other large item (of profit 1) or all the small items, whose total profit is x. Hence in this case the ratio is $(\phi + \max\{1, x\}) / \max\{\phi + \epsilon, x\}$, which is larger than $R = \phi - \epsilon$ for all $x \leq \phi^2$. □

Theorem 7. *No randomized 1-bin algorithm has competitive ratio smaller than $\frac{e+1}{e} \approx 1.3678$ in the oblivious adversary model.*

Proof. We employ Yao's principle. Fix a large integer n. The set of sequences that we consider are all the prefixes of length larger than 1 of the following sequence of items: an item of size and profit 1, followed by n items of size $1/(n + 1)$ and profit $1/n$ each, followed by an item of size $1/(n + 1)$ and profit 1.

Observe that as the first item's size is 1 and the total size of all the remaining items is no larger than 1, every deterministic algorithm (wlog) behaves as one the following canonical algorithms. ALG_k keeps the first item (of size and profit 1) in the bin until it sees the k-th small item, i.e., one of size $1/(n+1)$, at which

point it removes the large item from its bin and starts collecting the small items. Since, given the chance, wlog an algorithm replaces the item of profit 1 and size 1 with the one of same profit but size $1/(n+1)$, we have that $1 \leq k \leq n+1$.

With only $n+1$ algorithms to consider, we establish the probability distribution over the $n+1$ instances (recall that the first small item appears in all of them) in such a way that all these algorithms have the same expected profit. Note that this profit is 1, since $\textsc{Alg}_{n+1}$ always holds a single item of profit 1. Let p_i $(1 \leq n+1)$ denote the probability of the i-th instance, or, in other words, the probability that the instance ends after the i-th small item.

We fix $\{p_i\}$ as follows

$$p_i = \begin{cases} \frac{1}{n}\left(1-\frac{1}{n}\right)^{i-1}, & \text{for } i \leq n \\ \left(1-\frac{1}{n}\right)^{n}, & \text{for } i = n+1 \end{cases} \tag{1}$$

It is straightforward to observe that this is a probability distribution.

As for the expected gains of the algorithms, note that

$$\textsc{Alg}_k = \sum_{i<k} p_i \cdot 1 + \sum_{i=k}^{n} p_i \cdot \frac{i-k+1}{n} + p_{n+1} \cdot \left(\frac{n-k+1}{n} + 1\right) \ ,$$

so that for $k \leq n$, the following holds

$$\textsc{Alg}_{k+1} - \textsc{Alg}_k = p_k - \sum_{i=k}^{n+1} p_i \cdot \frac{1}{n} = \left(1-\frac{1}{n}\right) \cdot p_k - \frac{1}{n} \cdot \sum_{i=k+1}^{n+1} p_i \ . \tag{2}$$

From (1) it follows that (2) is zero, as

$$\frac{1}{n} \cdot \sum_{i=k+1}^{n+1} p_i = \frac{1}{n} \cdot \left(\left(1-\frac{1}{n}\right)^{n} + \frac{1}{n}\left(1-\frac{1}{n}\right)^{k} \cdot \sum_{i=0}^{n-k-1}\left(1-\frac{1}{n}\right)^{i}\right) = \frac{1}{n} \cdot \left(1-\frac{1}{n}\right)^{k} \ ,$$

which is exactly $(1-1/n)p_k$.

As every deterministic algorithm's expected gain is at most one, to prove the theorem we only need to lower bound the expected optimum profit. Notice that $\textsc{Opt} = 1$ unless the last item is issued, in which case $\textsc{Opt} = 2$. Thus

$$\mathbb{E}\left[\textsc{Opt}\right] = 1 + p_{n+1} = 1 + \left(1-\frac{1}{n}\right)^{n} \ ,$$

which tends to $1 + 1/e$ from below as n tends to infinity. □

Theorem 8. *No randomized 1-bin algorithm for the proportional case has competitive ratio smaller than* 1.25 *in the oblivious adversary model.*

4 Non-removable Variants

It is well known that no deterministic 1-bin algorithm has constant competitive ratio in the non-removable variant [11,12], even for proportional or unit case.

This can be seen by considering (prefixes of) an instance with only two items for the proportional case: one of arbitrarily small size $\epsilon > 0$, followed by another of size 1. For the unit case, one needs to look at (prefixes of) an instance where an item of size 1 is followed by $1/\epsilon$ items of arbitrarily small size ϵ.

We demonstrate that there is a significant difference in these special cases once more than one bin or random bits are available to the algorithm: for each objective function, either of these two advantages allows for an optimum ratio of 2 in the proportional case, but even combined these two advantages are not sufficient to attain constant ratio in the unit case.

4.1 Upper Bounds (Proportional Case)

For the proportional variant, we note that for $k \geq 2$ bins the (deterministic) algorithm FirstFit is 2-competitive for both *max*-objective and *sum*-objective, and that it gives rise to a 2-competitive 1-bin randomized algorithm.

Algorithm FirstFit: For each item, put it in the first (the one with lowest number) bin where it fits, ignoring the item if it does not fit in any bin.

Lemma 9. *If $k \geq 2$ bins, then at any time* FirstFit *either has all the items in its bins, or for each $1 \leq i < j \leq k$ we have $p(B_i) + p(B_j) > 1$.*

Lemma 9, whose proof is straightforward, immediately implies the following.

Theorem 10. FirstFit *is* 2*-competitive for $k \geq 2$ bins for the* sum*-objective.*

Lemma 9 also implies that FirstFit with 2 bins is 2-competitive for the *max*-objective, and simulating one of $\pi(\text{FirstFit}, \{B_1\})$ and $\pi(\text{FirstFit}, \{B_2\})$ chosen uniformly at random constitutes a 2-competitive barely random algorithm [4].

4.2 Lower Bounds

We provide lower bounds for both objective functions, for both the unit case and the proportional case. For the former case, we prove that for every k, no randomized k-bin algorithm is $O(1)$-competitive for either objective function, whereas for the latter case our results are subtler. For the *max*-objective, we prove a lower bound of 2 for *randomized* algorithms with *any* number of bins, proving the optimality of FirstFit and its barely random single-bin variant. For *sum*-objective we only prove a lower bound of $1 + \ln 2 \approx 1.693$, leaving a gap.

Theorem 11. *For every k, no randomized k-bin algorithm is $O(1)$-competitive for either* max*-objective or* sum*-objective in the unit case.*

Theorem 12. *For every k, no randomized k-bin algorithm has competitive ratio smaller than* 2 *for* max*-objective in the proportional case.*

Proof. We use Yao's principle. We consider instances that defined as follows. Fix an arbitrarily large integer n and an arbitrarily small $\epsilon > 0$. For each integer i,

$1 \leq i \leq n$, let $s_i = 1/2 + \epsilon/2^i$. The instance I_ℓ, $1 \leq \ell \leq n$, consists of items of sizes $s_1, s_2, \ldots, s_\ell$, in this order, followed by a single item of size $1 - s_\ell$. Note that in such an instance every item except the last one has size strictly greater than $1/2$, requiring a separate bin, and the last item fits together with the last item of size larger than $1/2$. Therefore, a deterministic k-bin algorithm for such set of instances, whatever the probability distribution over them, can be (wlog) identified by a set of k items, among the potential n items larger than $1/2$, that it is going to put to its k bins given the chance. Note that such an algorithm can have two items in some of the bins iff one of those k items is the last item of size larger than $1/2$, and otherwise it gains at most $1/2 + \epsilon$ from any bin. By adopting the uniform probability distribution over the n instances, we make the probability of the former at most k/n. Thus with n tending to infinity and ϵ tending to zero, in the limit the gain of any deterministic algorithm is at most $1/2$ per bin. Clearly, the optimum solution has profit 1 for one of the bins. □

We note that the result of Theorem 12 was already known for $k = 1$ [4] but, unlike ours, the proof technique of [4] does not extend to larger k.

Theorem 13. *For every k, no randomized k-bin algorithm has competitive ratio smaller than $1 + \ln 2 \approx 1.693$ for* sum*-objective in the proportional case.*

5 A Note on Adaptive Adversary

We note that the observations of Section 2.1 need not hold in the adaptive adversary model. Namely, as such an adversary solves an instance on-line, he may benefit from the presence of additional bins even under the *max*-objective. Thus it is possible that the optimum competitive ratio in such setting is not a non-increasing function of k. However, for $k = 1$ we establish tight bounds for randomized algorithms in the adaptive adversary model by proving that they cannot perform better than deterministic algorithms.

Theorem 14. *In the adaptive adversary model, no randomized 1-bin algorithm is $O(1)$-competitive for the general case of the removable variant.*

Theorem 15. *In the adaptive adversary model, no randomized 1-bin algorithm for the proportional case of the removable variant has competitive ratio smaller than $\phi = (1 + \sqrt{5})/2 \approx 1.618$.*

Theorem 16. *In the adaptive adversary model, no randomized 1-bin algorithm is $O(1)$-competitive for the proportional case of the non-removable variant.*

6 Conclusion and Open Problems

A single gap remains in the non-removable variant, for the *sum*-objective in the proportional case. The gaps for this objective in the removable variant are also significant, both in the general and the two special cases that we studied.

There are many gaps to be bridged in the removable variant under the *max*-objective as well. Of those, the question whether more than 2 bins or randomization permit ratios smaller than 2 in the general case seems particularly interesting. A related direction of interest is relating the power of barely random algorithms to unrestricted randomized algorithms.

Acknowledgements. We thank Monaldo Mastrolilli for suggesting the study of knapsack problems, Fabrizio Grandoni for suggesting the study of the sum objective, Yann Disser and Jiří Sgall for helpful discussions, and anonymous referees for their comments.

Łukasz Jeż was partially supported by Hasler Foundation, Grant 11099, MNiSW grant N N206 368839, 2010-2013, EU ERC project 259515 PAAl, and FNP Start scholarship. Marek Cygan was partially supported by NCN grant N N206 567940.

References

1. Azar, Y., Boyar, J., Favrholdt, L.M., Larsen, K.S., Nielsen, M.N., Epstein, L.: Fair versus unrestricted bin packing. Algorithmica 34(2), 181–196 (2002)
2. Azar, Y., Khaitsin, E.: Prompt mechanism for ad placement over time. In: Persiano, G. (ed.) SAGT 2011. LNCS, vol. 6982, pp. 19–30. Springer, Heidelberg (2011)
3. Ben-David, S., Borodin, A., Karp, R.M., Tardos, G., Wigderson, A.: On the power of randomization in online algorithms. Algorithmica 11(1), 2–14 (1994)
4. Böckenhauer, H.-J., Komm, D., Královič, R., Rossmanith, P.: On the advice complexity of the knapsack problem. In: Fernández-Baca, D. (ed.) LATIN 2012. LNCS, vol. 7256, pp. 61–72. Springer, Heidelberg (2012)
5. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press (1998)
6. Chekuri, C., Gamzu, I.: Truthful mechanisms via greedy iterative packing. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX and RANDOM 2009. LNCS, vol. 5687, pp. 56–69. Springer, Heidelberg (2009)
7. Iwama, K., Taketomi, S.: Removable online knapsack problems. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 293–305. Springer, Heidelberg (2002)
8. Iwama, K., Zhang, G.: Online knapsack with resource augmentation. Information Processing Letters 110(22), 1016–1020 (2010)
9. Kalyanasundaram, B., Pruhs, K.: Maximizing job completions online. J. of Algorithms 49(1), 63–85 (2003)
10. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack problems. Springer (2004)
11. Lueker, G.S.: Average-case analysis of off-line and on-line knapsack problems. J. Algorithms 29(2), 277–305 (1998)
12. Marchetti-Spaccamela, A., Vercellis, C.: Stochastic on-line knapsack problems. Math. Program. 68, 73–104 (1995)
13. Martello, S., Toth, P.: Knapsack problems. John Wiley & Sons (1990)
14. Noga, J., Sarbua, V.: An online partially fractional knapsack problem. In: Proc. of the 8th Int. Symp. on Parallel Architectures, Algorithms, and Networks (ISPAN), pp. 108–112 (2005)
15. Sgall, J.: Private communication (2013)

Counting Approximately-Shortest Paths in Directed Acyclic Graphs

Matúš Mihalák, Rastislav Šrámek, and Peter Widmayer

Institute of Theoretical Computer Science, ETH Zurich, Zurich, Switzerland
{mmihalak,rsramek,widmayer}@inf.ethz.ch

Abstract. Given a directed acyclic graph with positive edge-weights, two vertices s and t, and a threshold-weight L, we present a fully-polynomial time approximation-scheme for the problem of counting the s-t paths of length at most L. We extend the algorithm for the case of two (or more) instances of the same problem. That is, given two graphs that have the same vertices and edges and differ only in edge-weights, and given two threshold-weights L_1 and L_2, we show how to approximately count the s-t paths that have length at most L_1 in the first graph and length not much larger than L_2 in the second graph. We believe that our algorithms should find application in counting approximate solutions of related optimization problems, where finding an (optimum) solution can be reduced to the computation of a shortest path in a purpose-built auxiliary graph.

1 Introduction

Systematic generation and enumeration of combinatorial objects (such as graphs, set systems, and many more) has been a topic of extensive study in the field of combinatorial algorithms for decades [10]. Counting of combinatorial objects has been investigated at least as thoroughly, even leading to their own computational complexity class #P, defined in Valiant's seminal paper [15]. A counting problem usually asks for the number of solutions to a given combinatorial problem, such as the number of perfect matchings in a bipartite graph. In combinatorial optimization, the number of optimum solutions can sometimes be computed by a modification of an algorithm for finding a single optimum solution. For instance, for shortest s-t paths in graphs with positive edge weights, Dijkstra's algorithm easily admits such a modification. The problem we discuss in this paper has a more general flavor: We aim at counting the number of approximate solutions, in the sense of solutions whose objective value is within a given threshold from optimum. For shortest s-t paths, it is not obvious how to count the number of paths within, say, 10% from optimum. A related problem of enumerating feasible solutions makes a step in this direction: If we can enumerate solutions in order of decreasing quality, starting from an optimum solution, we have a way to count approximate solutions. Even though for some problems there are known enumeration algorithms that return the next feasible solution in the sequence of solutions within only polynomial extra time (called "polynomial delay"), this

C. Kaklamanis and K. Pruhs (Eds.): WAOA 2013, LNCS 8447, pp. 156–167, 2014.

approach will usually not be satisfactory in our setting. The reason is that the number of approximate solutions can be exponential, and counting by enumerating then takes exponential time, while our interest is only in the count itself.

In this paper we propose a way to count approximate solutions for the shortest s-t path problem in directed acyclic graphs (DAGs) in polynomial time, but the count that we get is only approximate, even though we come as close to the exact count as we wish (technically, we propose an FPTAS). We also show that exact counting for our problem is #P-hard, thus (together with the FPTAS) fully settling its complexity. We achieve our result by a modification of a conceptually interesting dynamic program for all feasible solutions for the knapsack problem [14]. Our motivation for studying our counting problem comes from a new approach [2] to cope with uncertainty in optimization problems. There, we not only need to count the number of approximate solutions for a given problem instance, but we also need to count the number of solutions that are approximate (within a given approximation ratio) for two problem instances at the same time. For the case of shortest s-t paths, this means that we are given two input graphs that are structurally identical, but are allowed to differ in their edge weights. We now want to count the number of s-t paths that are within, say, 10% from optimum in both input graphs at the same time. For this problem we propose both a pseudo-polynomial algorithm and an algorithm that calculates an approximate solution for a potentially slightly different threshold in fully polynomial time. Our hope is that our study paves the way for approximately counting approximate solutions for other optimization problems, such as minimum spanning trees.

The rest of the paper is organized as follows. We outline possible implications of our result in Section 1.1. We show in Section 1.2 that our problem is #P-complete. We present the algorithms in Section 2, and conclude the paper in Section 3.

1.1 Dynamic Programming as Shortest-Path Computation in DAGs

The concept of computing a shortest s-t path in a directed acyclic graph has a large number of applications in many areas of algorithmics. This is partly due to the fact that dynamic programming algorithms in which the inductive step consists of searching for a maximum or a minimum among some functions of previously-computed values can be viewed as the problem of looking for the shortest or longest path in a directed acyclic graph.[1]

In many problems that admit a dynamic programming solution we are interested not only in the single optimum, but also in other approximately optimal solutions. For instance, if we single out the context of analysis of biological data, de novo peptide sequencing [4,11], sequence alignment [13], or Viterbi decoding of HMMs [3,5] all use dynamic programming to find a shortest path in some implicit graph. Due to the nature of the data in these applications, producing a single solution is often insufficient and enumerating all solutions close to the optimum is necessary. Our contribution, therefore, provides a faster solution than

[1] Note that due to the lack of cycles, the problems of looking for shortest and longest paths on DAGs are computationally identical.

explicit enumeration for the problems where counting of approximate solutions is required [13]. Counting and sampling from close-to-optimum solutions is the key-element of the recent optimization method with uncertain input data of Buhmann et al. [2]. Our work thus makes a step towards practical algorithms in this context.

1.2 Counting Approximate Solutions Is #P-complete

The problem of counting the number of all self-avoiding s-t walks in a directed (or undirected) graph is known to be #P-complete [16]. The proof makes use of graphs containing cycles, thus it cannot be used to show the hardness of the problem of counting approximate shortest paths on a directed *acyclic* graph. In fact, we can easily count all s-t paths in a directed acyclic graph in time proportional to the number of edges, if we traverse the graph vertices sorted in topological order and add up the number of paths arriving to each vertex from its predecessors. The difficulty thus lies in the addition of edge-weights and the requirement to count s-t paths of length at most L. In the following, we show that this problem is #P-complete, by a reduction from the NP-complete *partition problem*. Given a set of positive integers $S = \{s_1, \ldots, s_n\}$, the partition problem asks for a partition of S into sets S_1 and S_2 such that the sums of numbers in both sets are equal.

Given an instance $S = \{s_1, \ldots, s_n\}$ of the partition problem, we construct a graph with $n+1$ vertices $v_1, \ldots, v_{n+1}$ as follows. We consider the elements of S in an arbitrary order $s_1, \ldots, s_n$. Then, for every $i < n$, the graph will contain two parallel edges between vertices v_i and v_{i+1} with lengths s_i and $-s_i$, respectively. Then every path from v_1 to v_{n+1} corresponds to one partition of S to subsets S_1 and S_2. If, between two consecutive vertices v_i and v_{i+1}, the edge with length s_i is chosen, s_i will belong to the set S_1. If the chosen edge has length $-s_i$, the element s_i will belong to the set S_2. The length of the v_1-v_{n+1} path then corresponds to the difference between the sums of elements in S_1 and in S_2 and the number of paths of length 0 is then equal to the number of optimal solutions of the partition problem.

If we had an algorithm that can count the number of v_1-v_{n+1} paths of length at most -1 and the number of v_1-v_{n+1} paths of length at most 0, the difference between these two numbers is the number of paths of length exactly 0 and thus the number of solutions to the partition problem.

Since the partition problem is reducible from the #P-complete knapsack problem [6] and its own reduction as well as ours is parsimonious [9], the problem of counting all s-t paths of length at most L is #P-complete. Note that the existence of parallel edges is not necessary for the reduction; we could bisect each parallel edge creating an auxiliary vertex to form a graph of the same functionality but without parallel edges. Also, observe that the use of negative edge-weights is not necessary; we can add to every edge-weight a very large number M (say, the maximum number in S), and then ask whether there exists a path of length nM. Thus, we have shown the following.

Theorem 1. *Let G be a directed acyclic graph with integer edge-weights, and L be an integer. The problem of counting all s-t paths of length at most L is #P-complete, even if all edge-weights are non-negative.*

2 Approximation Algorithms

In this section we present an FPTAS for our counting problem. That is, we present an algorithm that when given a directed acyclic graph G on n vertices, two dedicated vertices s and t, a weight-threshold L, and a constant $\varepsilon > 0$, computes a $(1+\varepsilon)$-approximation of the total number of s-t paths of length at most L, and which runs in time polynomial in both n and $\frac{1}{\varepsilon}$.

Let us note why the most immediate attempt to solve the problem directly does not work. We could try to calculate the number of paths from s to each vertex i that are shorter than all possible thresholds L. We can do this incrementally by calculating the paths for vertices sorted in topological order and for each new vertex combining the paths that arrived from previously computed vertices. We can then pick some polynomially large subset of the thresholds L and round all distances down to the nearest one in the subset. While we would end up with an algorithm of polynomial run-time, it would not constitute a FPTAS, since we would exactly count the number of paths that are no longer than some length L' which does not differ much from our desired maximum length L, instead of approximately counting the number of solutions that are shorter than the exact length L.

We first show a recurrence that can be used to exactly count the number of s-t paths of length at most L. Evaluating the recurrence takes exponential time, but we will later show how to group partial solutions together in such way that we trade accuracy for the number of recursive calls. We adapt the approach of Štefankovič et al. [14], which they used to approximate the number of all feasible solutions to the knapsack problem.

Let G be a directed acyclic graph with n vertices. We will label the vertices $v_1, \ldots, v_n$ in such order that there is no path from v_i to v_j unless $i < j$, i.e., $v_1, \ldots, v_n$ defines a topological ordering. We suppose that $v_1 = s$ and $v_n = t$, otherwise the graph can be pruned by discarding all vertices that appear before s and after t in the topological order, since no path from s to t ever visits these.

Now, for a given L, instead of asking for the number of s-t paths that have length at most L, we indirectly ask: for a given value a, what is the smallest threshold L' such that there are at least a paths from s to t of length at most L'? Let $\tau(v_i, a)$ denote the minimum length L' such that there are at least a paths from v_1 to v_i of length at most L'. To find the number of s-t paths of length at most L using this function τ, we simply search for the largest a such that $\tau(v_n, a) \leq L$, and return it as the output. In particular, if the length of the shortest s-t path is OPT (which can be computed in polynomial time), we can find, for any $\rho > 1$, the number of ρ-approximate s-t paths by setting $L := \rho OPT$.

For a concrete vertex v_i with in-degree d_i, let us denote its d_i neighbors that precede it in the topological order by $p_1, \ldots, p_{d_i}$ and let us denote the corresponding incoming edge lengths by $l_1, \ldots, l_{d_i}$. For simplicity, we usually drop the index i when it is clear from the context and just write d, $p_1, \ldots, p_d$ and $l_1, \ldots, l_d$. Now, $\tau(v_i, a)$ can be expressed by the following recurrence

$$\begin{aligned}
&\tau(v_1, 0) = -\infty \\
&\tau(v_1, a) = 0, \forall a : 0 < a \leq 1 \\
&\tau(v_1, a) = \infty, \forall a : a > 1 \\
&\tau(v_i, a) = \min_{\substack{\alpha_1, \ldots, \alpha_d \\ \sum \alpha_j = 1}} \max_s (\tau(p_s, \alpha_s a) + l_s).
\end{aligned}$$

Intuitively, the a paths starting at v_1 and arriving at v_i must split in some way among incoming edges. The values α_j define such split. We look for a set of $\alpha_1, \ldots, \alpha_d$ that minimizes the maximum allowed path length needed such that the incoming paths can be distributed according to α_j, $j = 1, \ldots, d$. Note that while the values of $\alpha_i a$ do not have to be integer, $\tau(v_i, \alpha_i a)$ is equal to $\tau(v_i, \lceil \alpha_i a \rceil)$. Moreover, when evaluating the recursion, it is enough to search for values α_i such that each of the values $\alpha_1 a, \ldots, \alpha_d a$ is an integer.

Calculating τ using the given recurrence will not result in a polynomial time algorithm since we might need to consider an exponential number of values for a, namely 2^{n-2} on a DAG with a maximal number of edges.[2] To overcome this, we will consider only a polynomial number of possible values for a, and always round down to the closest previously considered one in the recursive evaluation. If we are looking for an algorithm that counts with $1 + \varepsilon$ precision, the ratio between two successive considered values of a must be at most $1 + \varepsilon$.

For this purpose, we introduce a new function τ'. In order to achieve precision of $1 + \varepsilon$, we will only consider values of τ' for minimum path numbers in the form of q^k for all positive integers k such that $q^k < 2^{n-2}$, where $q = \sqrt[n+1]{1 + \varepsilon}$. The values of τ' for other numbers of paths will be undefined. The function τ' is defined by the recurrence

$$\begin{aligned}
&\tau'(v_1, 0) = -\infty \\
&\tau'(v_1, a) = 0, \forall a : 0 < a \leq 1 \\
&\tau'(v_1, a) = \infty, \forall a : a > 1 \\
&\tau'(v_i, q^j) = \min_{\substack{\alpha_1, \ldots, \alpha_d \\ \sum \alpha_j = 1}} \max_s (\tau'(p_s, q^{\lfloor j + \log_q \alpha_s \rfloor}) + l_s). \qquad (1)
\end{aligned}$$

To give a meaning to the expression $q^{\lfloor j + \log_q \alpha_i \rfloor}$ when $\alpha_i = 0$, we define it to be equal to 0, which is consistent with its limit when α_i goes to 0. We now show that the rounding does not make the values of τ' too different from the values of τ.

[2] To see this, observe that in a topologically sorted graph G, any subset of $V \setminus \{s, t\}$ gives a unique candidate for an s-t path.

Lemma 1. *Let $1 \leq i$ and $i \leq j$. Then*

$$\tau(v_i, q^{j-i}) \leq \tau'(v_i, q^j) \leq \tau(v_i, q^j). \tag{2}$$

Proof. We first prove the first inequality, proceeding by induction on i. The base case holds since $\tau(v_1, a) \leq \tau'(v_1, b)$ for any $a \leq b$. Suppose now that the first inequality of (2) holds for every p, $p < i$. Then, for every $0 \leq \alpha < 1$,

$$\begin{aligned} &\tau'(p, q^{\lfloor j+\log_q \alpha \rfloor}) \geq \tau(p, q^{\lfloor j+\log_q \alpha \rfloor - p}) \\ &\geq \tau(p, q^{j-p-1+\log_q \alpha}) \geq \tau(p, \alpha q^{j-i}). \end{aligned}$$

Thus, since every predecessor of v_i is earlier in the vertex ordering, we can use the obtained inequality to get the claimed bound

$$\begin{aligned} \tau'(v_i, q^j) &= \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max_s \tau'(p_s, q^{\lfloor j+\log_q \alpha_s \rfloor}) + l_s \\ &\geq \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max_s \tau(p_s, \alpha_s q^{j-i}) + l_s = \tau(v_i, q^{j-i}). \end{aligned}$$

The other inequality $\tau'(v_i, q^j) \leq \tau(v_i, q^j)$ follows by a simpler induction on i. The base case holds since $\tau(v_1, x) = \tau'(v_1, x)$ for all x. Assume now that the second part of (2) holds for all $p < i$. Then

$$\tau'(p, q^{\lfloor j+\log_q \alpha_i \rfloor}) \leq \tau(p, q^{\lfloor j+\log_q \alpha_i \rfloor}) \leq \tau(p, \alpha_i q^j).$$

We can now use the recursive definition to obtain the claimed inequality $\tau'(v_i, q^j) \leq \tau(v_i, q^j)$:

$$\begin{aligned} \tau'(v_i, q^j) &= \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max_s \tau'(p_s, q^{\lfloor j+\log_q \alpha_s \rfloor}) + l_s \\ &\leq \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max_s \tau(p_s, \alpha_s q^j) + l_s = \tau(v_i, q^j). \end{aligned}$$

□

We can now use $\tau'(v_n, q^k)$ to obtain a $(1+\varepsilon)$-approximation for the counting problem. Basically, for any L, we show that for the largest integer k such that $\tau'(v_n, q^k) \leq L < \tau'(v_n, q^{k+1})$, the value q^k will be no more than $(1+\varepsilon)^{\pm 1}$ away from the optimum.

Lemma 2. *Given L, let k be such that $\tau'(v_n, q^k) \leq L < \tau'(v_n, q^{k+1})$ and a be such that $\tau(v_n, a) \leq L < \tau(v_n, a+1)$. Then $(1+\varepsilon)^{-1} \leq \frac{a}{q^k} \leq 1+\varepsilon$.*

Proof. Using Lemma 1 twice, we get $\tau(v_n, q^{k-n}) \leq \tau'(v_n, q^k) \leq L < \tau'(v_n, q^{k+1}) \leq \tau(v_n, q^{k+1})$. As $\tau(v_n, q^{k-n})$ is at most L, and a is largest such that $\tau(v_n, a) \leq L$, and τ is monotonous in its second parameter, it must be that $q^{k-n} \leq a$. Similarly, $\tau(v_n, q^{k+1})$ is larger than L, so by monotonicity $a \leq q^{k+1}$. Thus both a and q^k must lie between q^{k-n} and q^{k+1} and their ratio can be at most $q^{k+1-(k-n)} = q^{n+1} = 1 + \varepsilon$ and at least $q^{k-(k+1)} = (1+\varepsilon)^{-1/(n+1)} > (1+\varepsilon)^{-1}$. □

We now show that computing the values of $\tau'(v_i, q^k)$ can be done in time polynomial in n and $\frac{1}{\varepsilon}$. This then, together with Lemma 2, gives an FPTAS for the counting problem.

Theorem 2. *For any L, any edge-weighted directed acyclic graph G, and any vertices s, t, there is an FPTAS that counts the number of all s-t paths in G of length at most L in time $O(mn^3\varepsilon^{-1}\log n)$.*

Proof. Recall that a directed acyclic graph on n vertices has at most 2^{n-2} s-t paths. The values of a in τ therefore span at most $\{1, 2, \ldots, 2^{n-2}\}$, and the values of q^k in τ' span at most $\{1, q, q^2, \ldots, q^s\}$, where

$$s := \log_q(2^{n-2}) = \frac{(n-2)}{\log_2 q} = \frac{(n-2)(n+1)}{\log_2(1+\varepsilon)} = O(n^2\varepsilon^{-1}).$$

Thus, we evaluate function τ' for at most $ns = O(n^3\varepsilon^{-1})$ different parameter pairs.

To show that the evaluation of τ' can be done in polynomial time, we need to show that we can efficiently find $\alpha_1, \ldots, \alpha_d$ that minimize Expression (1). Fortunately, $\tau'(v_i, q^k)$ is monotonous with increasing k, we can thus apply a greedy approach. Given v_i, we will evaluate $\tau'(v_i, q^k)$ for all possible values of q^k in one run. Instead of looking for the tuple $\alpha_1, \ldots, \alpha_d$ such that $\sum \alpha_i = 1$ we will consider an integer tuple $k_1, \ldots, k_d$ that minimizes $\max_i \tau'(p_i, q^{k_i})$ restricted by $\sum q^{k_i} > q^{k-1}$. We start with all k_i equal to 0 and always increase by one the k_i that minimizes $\tau'(p_i, q^{k_i+1}) + l_i$. Whenever the sum of all q^{k_i} gets larger than some value q^{k-1}, we store the current maximum of $\tau'(p_i, q^{k_i}) + l_i$ as the value $\tau'(v_i, q^k)$. We terminate once $\sum_i q^{k_i}$ reaches 2^{n-2}. It can be shown that such approach calculates the same values of τ' as searching through ratios α_i. As we can increase each k_i at most s times, we make at most ds steps, each of which involves choosing a minimum from d values and replacing it with a new value. The latter can be done in time $O(\log d) \subseteq O(\log n)$, for instance by keeping the values $\tau'(v_i, q^{k_i+1}) + l_i$ in a heap. The sum of the d's for all considered vertices is equal to the number of edges m. The update of $\sum_i q^{k_i}$, calculation of q^{k+1} from q^k, and comparison with the maximum number of paths can all be done in $O(\log(2^n)) = O(n)$ time if we choose q in the form $1 + 2^{-t}$ in order to be able to implement multiplication by q by a sequence of bit-shifts and a single addition. The resulting bit-time complexity is thus $O(mn^3\varepsilon^{-1}\log n)$. □

We note that processing the dynamic programming table for all path numbers in one go would to improve the time complexity of the original Knapsack FPTAS [14] by a factor of $O(\log(n))$.

2.1 Counting Solutions of Given Lengths in Multiple Instances

In this section we consider the problem of counting solutions that are approximately-optimum for two given instances at the same time. The two instances differ in edge lengths, but share the same topology, effectively forming a bi-criteria instance. Formally, given two directed acyclic graphs G_1 and G_2, differing only in edge-weights, given two vertices s and t, and given two threshold values L_1 and L_2, we are interested in the number of the s-t paths that have at the same time length at most L_1 in G_1 and length at most L_2 in G_2.

To solve this algorithmic problem, we cannot directly apply the approach for the single-instance case (by defining τ to be a pair of path lengths, one for each of the two instances), as we now have two lengths per edge and it is unclear how to suitably define a maximum over pairs in Equation (1). In fact, we can show that we cannot construct a FPTAS for the two instance scenario, or indeed any approximation algorithm.

Theorem 3. *Let G_1 and G_2 be two directed acyclic graphs with the same sets of vertices and edges, but possibly different edge-weights, let s and t be two vertices in them, let L_1 and L_2 be two length thresholds. The existence of an algorithm that in time polynomial in number of vertices n computes any finite approximation of the number of paths from s to t that are shorter than L_1 if measured in the graph G_1 and shorter than L_2 if measured in the graph G_2, implies that $P = NP$.*

Proof. We show this by reducing the decision version of the knapsack problem to the aforementioned problem. Let us have a knapsack instance with n items with weights $w_1, \ldots, w_n$ and prices $p_1, \ldots, p_n$. Given a total weight limit W and a price limit P we want to know if we can select a set of items such that the total weight is at most W and the total price is at least P. The corresponding DAG will have $n + 1$ vertices $v_0, \ldots, v_n$, with two edges between all successive vertices v_k and v_{k+1} that will correspond to the action of taking or not taking the $k + 1$-st element into the knapsack. The first edge between v_k and v_{k+1} will have length w_{k+1} in the graph G_1 and length $\frac{2P}{n+1} - p_{k+1}$ in the graph G_2, the second edge will have length 0 in the graph G_1 and $\frac{2P}{n+1}$ in the graph G_2. We can now ask for the number of paths from v_0 to v_n that are shorter than W in the graph G_1 and shorter than P in the graph G_2. If we had an algorithm that gives us a number that differs from this number by any real and finite multiplicative ratio c, we could determine whether the original knapsack problem had at least one solution since the ratio between 1 and 0 is not a real number. □

This proof is perhaps surprising due to the fact that Gopalan et al. [7] showed a FPTAS that counts the number of solutions of multi-criteria knapsack instances. This shows that while knapsack is a special version of our problem, it is in fact less complex due to the common assumption that the item values are non-negative.

While we cannot obtain a $(1 + \varepsilon)$-approximation of the number of s-t paths that have length at most L_1 in the first instance, and at the same time length at most L_2 in the second instance, we will adopt the techniques for FPTAS in a

single instance, and show a polynomial-time algorithm that provides heuristics for good estimates of s-t paths that have length at most $(1+\delta)L_1$ in the first instance, and at the same time length at most L_2 in the second instance. We will only consider the case where L_1 is positive.

To do so, we define a function τ_2 similar in spirit to τ that uses a maximum path-length L_1 in the form of a "budget" as a parameter of τ_2. Formally, $\tau_2(v_i, a, L_1)$ is the smallest length L_2 such that there are at least a v_1-v_i paths, each of length at most L_1 with respect to the edge lengths in the first instance, and of length at most L_2 with respect to the edge length in the second instance. Similarly to τ, we can express τ_2 recursively using the following notation. Let v_i be a vertex of in-degree d, and let $p_1, \ldots, p_d$ be the neighbors of v_i preceding it in the topological order. The edge-length of the incoming edge (p_j, v_i), $j = 1, \ldots, d_i$, is l_j in the first instance, and l'_j in the second instance. Then, τ_2 satisfies the following recursion:

$$\begin{aligned}
&\tau_2(v_1, 0, x) = -\infty, \forall x \in \mathbb{R}^+ \\
&\tau_2(v_1, a, x) = 0, \forall a : 0 < a \leq 1, \forall x \in \mathbb{R}^+ \\
&\tau_2(v_1, a, x) = \infty, \forall a : a > 1, \forall x \in \mathbb{R}^+ \\
&\tau_2(v_i, a, L_1) = \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max_s \tau_2(p_s, \alpha_s a, L_1 - l_s) + l'_s
\end{aligned}$$

If we wanted to use τ_2 to directly use to solve our counting problem, the function τ_2 would have to be evaluated not only for an exponential number of path counts a, but also for possibly exponential number of values of L_1. To end up with polynomial runtime, we thus need to consider only a polynomial number of values for both parameters of τ_2. For this purpose, we will introduce a function τ'_2 that does this by considering only path lengths in the form of r^k, where $r = \sqrt[n]{1+\delta}$, and path numbers a in the form of q^j, where $q = \sqrt[n]{1+\varepsilon}$, for positive ε and δ. Function τ'_2 is defined by the following recurrence:

$$\begin{aligned}
&\tau'_2(v_1, 0, x) = -\infty, \forall x \in \mathbb{R}^+ \\
&\tau'_2(v_1, a, x) = 0, \forall a : 0 < a \leq 1, \forall x \in \mathbb{R}^+ \\
&\tau'_2(v_1, a, x) = \infty, \forall a : a > 1, \forall x \in \mathbb{R}^+ \\
&\tau'_2(v_i, q^j, r^k) = \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max_s \tau'_2(p_s, q^{\lfloor j + \log_q \alpha_s \rfloor}, r^{\lfloor \log_r (r^k - l_s) \rfloor}) + l'_s
\end{aligned}$$

Similarly to the case of one instance only, one can show that τ'_2 approximates τ_2 well, this time in two variables.

Lemma 3. *Let $0 \leq i$, $i \leq j$, and $i \leq k$. Then*

$$\tau_2(v_i, q^{j-i}, r^k) \leq \tau'_2(v_i, q^j, r^k) \leq \tau_2(v_i, q^j, r^{k-i}). \tag{3}$$

The proof is conceptually similar to the proof of Lemma 1 and is stated in full in the full version of this paper [12].

Using Lemma 3, we can show that τ_2' provides enough information to compute an approximation of τ_2. However, we cannot get a $(1+\varepsilon)$ approximation to the optimal value as in Lemma 2, because we need to round the value of L_1 to a power of r in order for it to be legal parameter of τ_2' and we further round it during the evaluation of τ_2'. We will therefore relate the result of τ_2' to the results of τ_2 we would have gotten if we considered the value of L_1 when rounded up towards the nearest number that can be represented as r^k for integer k and the value r^{k-n}. Due to the choice of r, the ratio of these two values is $1+\delta$.

Lemma 4. *Let k be such that $\tau_2'(v_n, q^k, r^{\lceil \log_r L_1 \rceil}) \leq L_2 < \tau_2'(v_n, q^{k+1}, r^{\lceil \log_r L_1 \rceil})$, a be such that $\tau_2(v_n, a, r^{\lceil \log_r L_1 \rceil - n}) \leq L_2 < \tau_2(v_n, a+1, r^{\lceil \log_r L_1 \rceil - n})$, and b be largest such that $\tau_2(v_n, b, r^{\lceil \log_r L_1 \rceil}) \leq L_2 < \tau_2(v_n, b+1, r^{\lceil \log_r L_1 \rceil})$. Then $a \leq b$, $\frac{a}{q^k} \leq 1+\varepsilon$, and $\frac{q^k}{b} \leq 1+\varepsilon$.*

Proof. The statement that $a \leq b$ follows from the definition of a and b: decreasing the limit on the path length in the first instance from $r^{\lceil \log_r L_1 \rceil}$ to $r^{\lceil \log_r L_1 \rceil - n}$ cannot increase the number of possible paths. By applying Lemma 3 twice, we get

$$\tau_2(v_n, q^{k-n}, r^{\lceil \log_r L_1 \rceil}) \leq \tau_2'(v_n, q^k, r^{\lceil \log_r L_1 \rceil}) \leq L_2, \tag{4}$$

and

$$L_2 < \tau_2'(v_n, q^{k+1}, r^{\lceil \log_r L_1 \rceil}) \leq \tau_2(v_n, q^{k+1}, r^{\lceil \log_r L_1 \rceil - n}). \tag{5}$$

From the definition of a and (5) we can conclude $a \leq q^{k+1}$. This implies that $\frac{a}{q^k} \leq q \leq 1+\varepsilon$, due to our choice of q. Similarly, from the definition of b and (4) we get $b \geq q^{k-n}$ and thus $\frac{q^k}{b} \leq q^n \leq 1+\varepsilon$. □

Lemma 4 shows that the computed number of s-t paths q^k cannot be larger than b by more than a factor of $1+\varepsilon$, nor can it be smaller than a by a factor larger than $1+\varepsilon$. Furthemore, with the aforementioned choice of r as $\sqrt[n]{1+\delta}$, the difference between the rounded up value of L_1 which is $r^{\lceil \log_r L_1 \rceil}$ and the rounded down value which is $r^{\lceil \log_r L_1 \rceil - n}$ is $(1+\delta)$. We can now state the overall running time of the approach. Compared to the function τ' we need to evaluate τ_2' for $\lceil \log_r L_1 \rceil = O(n\delta^{-1} \log L_1)$ values of r^l, in addition to the values of v_i and q^k. Otherwise the arguments are identical to the proof of Theorem 2. Note that $\log L_1$ is by definition in $O(n)$, but we list it explicitly since it can be much smaller in practice.

Lemma 5. *Given path-lengths L_1 and L_2 for two given instances G_1 and G_2 of a graph with n edges and m vertices, there is an algorithm that finds k satisfying $\tau_2'(v_n, q^k, r^{\lceil \log_r L_1 \rceil}) \leq L < \tau_2'(v_n, q^{k+1}, r^{\lceil \log_r L_1 \rceil})$ in time $O(mn^3\varepsilon^{-1}\delta^{-1} \log n \log L_1)$.*

Putting together Lemma 4 and Lemma 5 we can state the overall result:

Theorem 4. *For any L_1, L_2, any edge-weighted directed acyclic graphs on the same topology G_1 and G_2, and any two of their vertices s, t, there exists a length L_2' satisfying $(1+\delta)^{-1}L_2 \leq L_2' \leq L_2$ and an FPTAS for counting the number of paths from s to t no longer than L_1 when evaluated on the graph G_1 and no longer than L_2' when evaluated on the graph G_2 in the time $O(mn^4\varepsilon^{-1}\delta^{-1} \log n \log L_1)$.*

It is easy to see that we can extend the approach to count paths that approximate m instances at the same time by adding "budgets" $L_1, \ldots, L_{m-1}$ for the desired maximal lengths of paths in instances $1, 2, \ldots, m-1$. The time complexity would again increase, for every additional instance with threshold L_i by $O(n\delta^{-1} \log L_i)$.

Pseudo-Polynomial Algorithm for Two Instances. If the discrepancy between a and b as defined in Lemma 4 is too large and all edges have integer lengths, we can consider all possible lengths in the first instance, instead of rounding to values in the form of r^k. If implemented in a similar way to the previously discussed two-instance approximation algorithm, this pseudo-polynomial algorithm would run in the time $O(mn^3\varepsilon^{-1}L_1 \log n)$.

3 Concluding Remarks

We have shown that there is an efficient algorithm to approximate the number of approximately shortest paths in a directed acyclic graph. This problem is implicitly or explicitly present as an algorithmic tool in algorithmic solutions to a large number of different computational problems, not limited to the evaluation of solutions achieved by dynamic programming which we noted in Section 1.1.

Our result allows us, for instance, to approximately count only the small (or large) terms of a polynomial $p(x) = \sum_i a_i x^i$, $a_i \geq 0$, represented as a product $\prod_j p_j(x)$ of polynomially many polynomial factors $p_j(x)$, where each $p_j(x) = \sum_k b_k x^k$ has polynomially many terms, and where every $b_k \geq 0$. This is especially interesting if the full expansion of $p(x)$ has exponentially many terms. This may be a powerful tool, if extended to the case of both negative and positive b_k, enabling the counting of approximate solutions for problems with known generating polynomials of solutions by weight. For instance, counting of large graph matchings [8] or short spanning trees [1] can be done via generating polynomials (which, in general, have exponentially many terms). This direction is our primary future work.

We have also showed that our algorithm can be extended, given threshold weights $L_1, \ldots, L_m$, and polynomially many graphs $G_1, \ldots, G_m$, to count s-t paths that have, at the same time, length at most L_1 in G_1 and at most $(1+\delta)L_i$ in G_i, $i = 2, \ldots, m$. In the case when $m = 2$, this algorithm is necessary for application of the aforementioned robust optimization method [2] to the various mentioned optimization problems.

Acknowledgements. We thank Octavian Ganea and anonymous reviewers for their suggestions and comments. The work has been partially supported by the Swiss National Science Foundation under grant no. 200021_138117/1, and by the EU FP7/2007-2013, under the grant agreement no. 288094 (project eCOMPASS).

References

1. Broder, A.Z., Mayr, E.W.: Counting minimum weight spanning trees. J. Algorithms 24, 171–176 (1997)
2. Buhmann, J.M., Mihalák, M., Šrámek, R., Widmayer, P.: Robust optimization in the presence of uncertainty. In: Proc. 4th Conference on Innovations in Theoretical Computer Sciencei (ITCS), pp. 505–514. ACM, New York (2013)
3. Burge, C., Karlin, S.: Prediction of complete gene structures in human genomic DNA. Journal of Molecular Biology 268(1), 78–94 (1997)
4. Chen, T., Kao, M.Y., Tepel, M., Rush, J., Church, G.M.: A dynamic programming approach to de novo peptide sequencing via tandem mass spectrometry. Journal of Computational Biology 8(3), 325–337 (2001)
5. Durbin, R., Eddy, S.R., Krogh, A., Mitchison, G.: Biological sequence analysis: Probabilistic models of proteins and nucleic acids. Cambridge university press (1998)
6. Dyer, M., Frieze, A., Kannan, R., Kapoor, A., Perkovic, L., Vazirani, U.: A mildly exponential time algorithm for approximating the number of solutions to a multidimensional knapsack problem. Combinatorics, Probability and Computing 2(3), 271–284 (1993)
7. Gopalan, P., Klivans, A., Meka, R., Štefankovič, D., Vempala, S., Vigoda, E.: An FPTAS for # knapsack and related counting problems. In: Proc. 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 817–826 (2011)
8. Jerrum, M.: Two-dimensional monomer-dimer systems are computationally intractable. Journal of Statistical Physics 48(1-2), 121–134 (1987)
9. Karp, R.M.: Reducibility among combinatorial problems. Springer (1972)
10. Kreher, D.L., Stinson, D.R.: Combinatorial Algorithms: Generation, Enumeration, and Search (1998)
11. Lu, B., Chen, T.: A suboptimal algorithm for de novo peptide sequencing via tandem mass spectrometry. Journal of Computational Biology 10(1), 1–12 (2003)
12. Mihalák, M., Šrámek, R., Widmayer, P.: Counting approximately-shortest paths in directed acyclic graphs. arXiv preprint arXiv:1304.6707 (2013)
13. Naor, D., Brutlag, D.: On suboptimal alignments of biological sequences. In: Apostolico, A., Crochemore, M., Galil, Z., Manber, U. (eds.) CPM 1993. LNCS, vol. 684, pp. 179–196. Springer, Heidelberg (1993)
14. Štefankovič, D., Vempala, S., Vigoda, E.: A deterministic polynomial-time approximation scheme for counting knapsack solutions. SIAM Journal on Computing 41(2), 356–366 (2012)
15. Valiant, L.G.: The complexity of computing the permanent. Theoretical Computer Science 8(2), 189–201 (1979)
16. Valiant, L.G.: The complexity of enumeration and reliability problems. SIAM J. Comput. 8(3), 410–421 (1979)

Author Index