Daniël Paulusma
Bernard Ries (Eds.)

LNCS 14093

Graph-Theoretic Concepts in Computer Science

49th International Workshop, WG 2023
Fribourg, Switzerland, June 28–30, 2023
Revised Selected Papers

Springer

Lecture Notes in Computer Science 14093

Advanced Research in Computing and Software Science

Subline of Lecture Notes in Computer Science

More information about this series at https://link.springer.com/bookseries/558

Daniël Paulusma · Bernard Ries
Editors

Graph-Theoretic Concepts in Computer Science

49th International Workshop, WG 2023
Fribourg, Switzerland, June 28–30, 2023
Revised Selected Papers

Springer

Editors
Daniël Paulusma
Durham University
Durham, UK

Bernard Ries
University of Fribourg
Fribourg, Switzerland

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-031-43379-5 ISBN 978-3-031-43380-1 (eBook)
https://doi.org/10.1007/978-3-031-43380-1

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Paper in this product is recyclable.

Preface

This volume contains the 33 papers presented at the 49th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2023). The conference was held in Fribourg, Switzerland from 28 June 2023 to 30 June 2023. About 55 participants from all over the world attended the conference in person, and about 30 participants registered for the conference on-line.

WG has a long-standing tradition. Since 1975, WG has taken place 25 times in Germany, five times in The Netherlands, three times in France, twice in Austria, the Czech Republic, and the UK, and once in Greece, Israel, Italy, Norway, Poland, Slovakia, Spain and Turkey. This was the second time the conference was held in Switzerland.

WG aims to merge theory and practice by demonstrating how concepts from graph theory can be applied to various areas in computer science, or by extracting new graph-theoretic problems from applications. The conference is well balanced with respect to established researchers and junior scientists.

We received 116 submissions, sixteen of which were withdrawn before entering the (single-blind) review process. The Program Committee provided three independent reviews for each submission. After a careful discussion, the Program Committee accepted 33 papers, which yields an acceptance ratio of 33%. Due to the high competition and a limited schedule, there were papers that could not be accepted although they deserved to be.

The program included three inspired invited talks by Flavia Bonomo (University of Buenos Aires, Argentina) on “Generalized List Matrix Partition Problems on Chordal Graphs, Parameterized by Leafage”, Eunjung Kim (LAMSADE, Paris-Dauphine University, France) on “Twin-width, Graph Classes and a Bit of Logic” and Nicolas Trotignon, (CNRS, École normale supérieure de Lyon, France) on “Triangle-free Graphs of Large Chromatic Number”.

The third WG Test of Time Award, given for a highly influential paper presented at a previous WG conference, was given to Alistair Sinclair and Mark Jerrum for their paper “Approximate Counting, Uniform Generation and Rapidly Mixing Markov Chains” from WG 1987. An excellent fourth invited talk on “35 Years of Counting, Sampling and Mixing” was given by Mark Jerrum (Queen Mary University of London, UK).

The WG 2023 Best Paper award was given to Paul Jungeblut, Samuel Schneider and Torsten Ueckerdt for their paper “Cops and Robber – When Capturing is not Surrounding”. The WG 2023 Best Student Paper Award was given to Falko Hegerfeld for his paper “Tight Algorithms for Connectivity Problems Parameterized by Modular-Treewidth”, co-authored by Stefan Kratsch. Both awards were sponsored by Springer-Verlag.

We would like to thank the following organisations for their financial support (in alphabetical order): the Canton of Fribourg, the City of Fribourg, Springer-Verlag, the

Swiss National Science Foundation (SNSF) and the University of Fribourg. We would also like to thank all the authors, the members of the Program Committee and additional reviewers, the speakers, session chairs, student helpers and all other participants for their contribution towards making the conference a successful event. In particular, we thank Felicia Lucke for all her excellent work on the Organising Committee.

July 2023

Daniël Paulusma
Bernard Ries

Organization

Program Committee

Nick Brettell	Victoria University of Wellington, New Zealand
Yixin Cao	Hong Kong Polytechnic University, China
Clément Dallard	Université d'Orléans, France
Vida Dujmović	McGill University, Canada
David Eppstein	University of California, Irvine, USA
Bruno Escoffier	Sorbonne University, France
Carl Feghali	École Normale Supérieure de Lyon, France
Esther Galby	Hamburg University of Technology, Germany
Danny Hermelin	Ben-Gurion University of the Negev, Israel
Yusuke Kobayashi	Kyoto University, Japan
Stephen Kobourov	University of Arizona, USA
Daniel Král'	Masaryk University, Czech Republic
Michael Lampis	Paris Dauphine University, France
Paloma Lima	IT University of Copenhagen, Denmark
Amer Mouawad	American University of Beirut, Lebanon
Andrea Munaro	University of Parma, Italy
Daniel Paulusma (Chair)	Durham University, UK
Irena Penev	Charles University, Czech Republic
Bernard Ries (Chair)	University of Fribourg, Switzerland
Laura Sanitá	Bocconi University, Italy
Roohani Sharma	Max Planck Institute for Informatics, Germany
Uéverton Souza	Fluminense Federal University, Brazil
Maya Stein	University of Chile, Chile
Csaba Tóth	California State University Northridge, USA
Virginia Vassilevska Williams	Massachusetts Institute of Technology, USA
Shira Zerbib	Iowa State University, USA

Additional Reviewers

Aboulker, Pierre
Abu-Khzam, Faisal
Ackerman, Eyal
Agrawal, Akanksha
Ahmed, Abu Reyan
Ahn, Jungho
Aigner-Horev, Elad
Akhoondian Amiri, Saeed
Akmal, Shyan
Alecu, Bogdan
Atminas, Aistis
Bampis, Evripidis
Bartier, Valentin
Barát, János
Baste, Julien
Belmonte, Rémy

Benedek, Márton
Bentz, Cédric
Bhore, Sujoy
Bonomo, Flavia
Bouquet, Valentin
Boyar, Joan
Bradshaw, Peter
Brandenburg, Franz
Broersma, Hajo
Burgess, Andrea
Cameron, Kathie
Casel, Katrin
Chakraborty, Dibyayan
Chan, Timothy M.
Chaplick, Steven
Chen, Yong
Chepoi, Victor
Chitnis, Rajesh
Choi, Ilkyoo
Curticapean, Radu
Dalirrooyfard, Mina
Dey, Sanjana
Dibek, Cemil
Dourado, Mitre
Dreier, Jan
Duron, Julien
Dvořák, Pavel
Dósa, György
Eiben, Eduard
El Sabeh, Remy
Faria, Luerbio
Fernau, Henning
Figueiredo, Celina
Focke, Jacob
Friggstad, Zachary
Gajarský, Jakub
Garlet Milani, Marcelo
Gaspers, Serge
Geniet, Colin
Ghahremani, Mani
Ghosh, Anirban
Giroudeau, Rodolphe
Golovach, Petr
Gonzalez, Carolina
Haslegrave, John
Hatzel, Meike
Heeger, Klaus
Huang, Shenwei
Huroyan, Vahan
Huszár, Kristóf
Hörsch, Florian
Itzhaki, Yuval
Jacob, Dalu
Jaffke, Lars
Jain, Pallavi
Jana, Satyabrata
Jin, Ce
Kanj, Iyad
Kaul, Matthias
Kellerhals, Leon
Kloks, Ton
Knop, Dušan
Koana, Tomohiro
Korchemna, Viktoriia
Kratochvil, Jan
Krenn, Mario
Krithika, R.
Kryven, Myroslav
Kumar, Ravi
Kwon, O-Joung
Křišťan, Jan Matyáš
La, Hoang
Lagoutte, Aurélie
Lendl, Stefan
Li, Shaohua
Li, Yanjia
Lidicky, Bernard
Liedloff, Mathieu
Lin, Bingkai
Liu, Yunlong
Lucke, Felicia
Maaz, Stephanie
Madaras, Tomas
Maia, Ana Karolinna
Majewski, Konrad
Mann, Felix
Martin, Barnaby
Masařík, Tomáš
Mathew, Rogers
Mathialagan, Surya
Mazzoleni, María Pía
McGinnis, Daniel

Melissinos, Nikolaos
Miyazaki, Shuichi
Mnich, Matthias
Molter, Hendrik
Moore, Benjamin
Mütze, Torsten
Narayanaswamy, N. S.
Newman, Alantha
Nguyen, Tung
Nisse, Nicolas
Okamoto, Yoshio
Okrasa, Karolina
Oliveira, Fabiano
Oostveen, Jelle
Oum, Sang-il
Paesani, Giacomo
Panolan, Fahad
Picouleau, Christophe
Pierron, Théo
Pollner, Tristan
Rajendran, Goutham
Razgon, Igor
Rong, Guozhen
Roy, Sanjukta
Schindl, David
Schirneck, Martin
Schlotter, Ildikó
Sen, Sagnik
Serna, Maria
Siebertz, Sebastian
Sikora, Florian
Simonov, Kirill
Sivaraman, Vaidy
Smith, Siani
Sokołowski, Marek
Sonmez Turan, Meltem
Spence, Richard
Stamoulis, Giannos
Štorgel, Kenny
Sylvester, John
Takazawa, Kenjiro
Tale, Prafullkumar
Telle, Jan Arne
Thilikos, Dimitrios
Tong, Weitian
Tsur, Dekel
Turcotte, Jèrèmie
Ueckerdt, Torsten
van der Poel, Andrew
van Iersel, Leo
van Leeuwen, Erik Jan
Vasilakis, Manolis
Vaxès, Yann
Vialette, Stéphane
Vu, Tung Anh
Weller, Mathias
Wiederrecht, Sebastian
Wulms, Jules
Xefteris, Michalis
Xu, Yinzhan
Yang, Shizhou
Yang, Yongjie
Zerovnik, Janez

Contents

Proportionally Fair Matching with Multiple Groups

Sayan Bandyapadhyay[1], Fedor V. Fomin[2], Tanmay Inamdar[2](✉), and Kirill Simonov[3]

[1] Portland State University, Portland, USA
sayanb@pdx.edu
[2] University of Bergen, Bergen, Norway
{Fedor.Fomin,Tanmay.Inamdar}@uib.no
[3] Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
Kirill.Simonov@hpi.de

Abstract. We study matching problems with the notion of proportional fairness. Proportional fairness is one of the most popular notions of group fairness where every group is represented up to an extent proportional to the final selection size. Matching with proportional fairness or more commonly, proportionally fair matching, was introduced in [Chierichetti et al., AISTATS, 2019]. In this problem, we are given a graph G whose edges are colored with colors from a set C. The task is for given $0 \leq \alpha \leq \beta \leq 1$, to find a maximum (α, β)-balanced matching M in G, that is a matching where for every color $c \in C$ the number of edges in M of color c is between $\alpha|M|$ and $\beta|M|$. Chierichetti et al. initiated the study of this problem with two colors and in the context of bipartite graphs only. However, in many practical applications, the number of colors—although often a small constant—is larger than two. In this work, we make the first step towards understanding the computational complexity of proportionally fair matching with more than two colors. We design exact and approximation algorithms achieving reasonable guarantees on the quality of the matching as well as on the time complexity, and our algorithms work in general graphs. Our algorithms are also supported by suitable hardness bounds.

Keywords: Matching · Fairness · Parameterized Algorithms

1 Introduction

In this paper, we consider the proportionally fair matching problem in general graphs. Matching is one of the most fundamental notions in graph theory whose

Most of this work was done when all four authors were affiliated with the University of Bergen, Norway. The research leading to these results has received funding from the Research Council of Norway via the project BWCA (grant no. 314528), and the European Research Council (ERC) via grant LOPPRE, reference 819416.

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 1–15, 2023.
https://doi.org/10.1007/978-3-031-43380-1_1

study can be traced back to the classical theorems of Kőnig [33] and Hall [24]. The first chapter of the book of Lovász and Plummer [37] devoted to matching contains a nice historical overview of the development of the matching problem. The problems of finding a maximum size or a perfect matching are the classical algorithmic problems; an incomplete list of references covering the history of algorithmic improvements on these problems is [18,25,40,41,44], see also the book of Schrijver [47] for a historical overview of matching algorithms. Matchings appear naturally in various applications, e.g., kidney transplant matching [46] or numerous assignment problems like assigning products to customers [45]; students to schools [34]; reviewers to manuscripts [8]; and workers to firms [1]. There are scores of works that study fair versions of matchings [10,20,21,26,28,31,50]. Among these distinct notions of matchings, our work is most relevant to the work on (α, β)-balanced matching of Chierichetti et al. [10]. The notion of (α, β)-balanced matching was formulated in [10] by bringing *proportional* fairness and maximum cardinality matching together. Proportional fairness is based on the concept of disparate impact [19], which in the context of matching is defined as follows. A matching is (α, β)-balanced or proportionally fair if the ratio between the number of edges from each group (a color) and the size of the matching is at least α and at most β.

As a motivating example of proportionally fair matching, consider the product recommendation problem in e-commerce. With the advancement of digital marketing and advertising, nowadays companies are interested in more fine-tuned approaches that help them reach the target groups of customers. These groups may be representative of certain underlying demographic categorizations based on gender, age group, geographic location etc. Thus, the number of groups is often a small constant. In particular, in this contemporary setting, one is interested in finding assignments that involve customers from all target groups and have a balanced impact on all these groups. This assignment problem can be modeled as the proportionally fair matching problem between customers and products. In a realistic situation, one might need to assign many products to a customer and many customers to a product. This can be achieved by computing multiple matchings in an iterative manner while removing the edges from the input graph that are already matched.

In a seminal work, Chierichetti et al. [10] obtained a polynomial-time 3/2-approximation for the size of the matching, when the input graph is bipartite and the number of groups is 2. However, in many real-world situations, like in the above example, it is natural to assume that the number of target groups is more than 2. Unfortunately, the algorithm of [10] strongly exploits the fact that the number of groups $\ell = 2$. It is not clear how to adapt or extend their algorithm when we have more than two groups. The only known algorithm for $\ell > 2$ groups is an $n^{O(\ell)}$-time randomized exact algorithm [10,14], which also works for general graphs. The running time of this algorithm has a "bad" exponential dependence on the number of groups, i.e., the running time is not a *fixed* polynomial in n. Thus, this algorithm quickly becomes impractical if ℓ grows. Our research on proportionally fair matching is driven by the following question. *Do there exist* *efficient algorithms with guaranteed performance for proportionally fair matching when the number of groups ℓ is more than two?*

1.1 Our Results and Contributions

In this work, we obtain several results on the Proportionally Fair Matching problem in general graphs with any arbitrary ℓ number of groups.

– First, we show that the problem is extremely hard for any general ℓ number of groups, in the sense that it is not possible to obtain any approximation algorithm in $2^{o(\ell)}n^{O(1)}$ time even on path graphs, unless the Exponential Time Hypothesis (ETH) [27] is false.

– To complement our hardness result, we design a $\frac{1}{4\ell}$-approximation algorithm that runs in $2^{O(\ell)}n^{O(1)}$ time. Our algorithm might violate the lower (α) and upper (β) bounds by at most a multiplicative factor of $(1 + \frac{4\ell}{|\text{OPT}|}$ if $|\text{OPT}|$ is more than $4\ell^2$, where OPT is any optimum solution. Thus, the violation factor is at most $1 + \frac{1}{\ell}$, and tends to 1 with asymptotic values of $|\text{OPT}|$.

– We also consider a restricted case of the problem, referred to as the β-limited case in [10], where we only have the upper bound, i.e., no edges might be present from some groups. In this case, we could improve the approximation factor to $\frac{1}{2\ell}$ and running time to polynomial.

– Lastly, we show that the parameterized version of the problem where one seeks for a proportionally fair matching of size k, can be solved exactly in $2^{O(k)}n^{O(1)}$ time. Thus the problem is fixed-parameter tractable parameterized by k.

All of our algorithms are based on simple schemes. Our approximation algorithms use an iterative peeling scheme that in each iteration, extracts a *rainbow* matching containing at most one edge from every group. The exact algorithm is based on a non-trivial application of the celebrated color-coding scheme [2]. These algorithms appear in Sects. 3, 4, and 5, respectively. The hardness proof is given in Sect. 6.

1.2 Related Work

In recent years, researchers have introduced and studied several different notions of fairness, e.g., disparate impact [19], statistical parity [29,51], individual fairness [15] and group fairness [16]. Kleinberg et al. [32] formulated three notions of fairness and showed that it is theoretically impossible to satisfy them simultaneously. See also [11,12] for similar exposures.

The notion of proportional fairness with multiple protected groups is widely studied in the literature, which is based on disparate impact [19]. Bei et al. [4] studied the *proportional candidate selection problem*, where the goal is to select a subset of candidates with various attributes from a given set while satisfying certain proportional fairness constraints. Goel et al. [22] considered the problem of learning non-discriminatory and proportionally fair classifiers and proposed the *weighted sum of logs* technique. Proportional fairness has also been considered in the context of Federated learning [53]. Additionally, proportional fairness has been studied in the context of numerous optimization problems including voting [17], scheduling [30,38], kidney exchange [5,48], and Traveling Salesman Problem [43].

Several different fair matching problems have been studied in the literature. Huang et al. [26] and Boehmer et al. [7] studied fair b-matching, where matching preferences for each vertex are given as ranks, and the goal is to avoid assigning vertices to the highly ranked vertices as much as possible (see [7,26] for a formal definition). Fair-by-design-matching is studied in [21], where instead of a single matching, a probability distribution over all feasible matchings is computed which guarantees individual fairness. See also [28,31].

Apart from the fair versions of matchings, many constrained versions are also studied [6,49]. [49] studied the Bounded Color Matching (BCM) problem where edges are colored and from each color class, only a given number of edges can be chosen. BCM is a special case of 3-set packing and, hence, admits a 3/4-approximation [49]. We note that the β-limited case of Proportionally Fair Matching is a special case of BCM and, thus, a 3/4-approximation follows in this case where the upper bound might be violated by 3/4 factor. One should compare this factor with our violation factor, which asymptotically tends to 1.

Finally, we note that when the input graph is bipartite, a matching has a natural interpretation as an intersection of two matroids. Matroid intersection has a rich literature containing work on both exact [36,39] and approximation algorithms [35]. However, these algorithms do not consider fairness constraints.

2 Preliminaries

For an integer $\ell \geq 1$, let $[\ell] := \{1, 2, \ldots, \ell\}$. Consider any undirected n-vertex graph $G = (V, E)$ such that the edges in E are colored by colors in $C = \{1, \ldots, \ell\}$. The function $\chi : E \to C$ describes the color assignment. For each color $c \in C$, let E_c be the set of edges colored by the color c, i.e., $E_c = \chi^{-1}(c)$. A subset $E' \subseteq E$ is a *matching* in G if no two edges in E' share a common vertex.

Definition 1. ***(α, β)-balanced matching.*** *Given $0 \leq \alpha \leq \beta \leq 1$, a matching $M \subseteq E$ is called (α, β)-balanced if for each color $c \in C$, we have that $\alpha \leq \dfrac{|M \cap E_c|}{|M|} \leq \beta$.*

Thus a matching is (α, β)-balanced if it contains at least α and at most β fraction of edges from every color. In the Proportionally Fair Matching problem, the goal is to find a maximum-sized (α, β)-balanced matching. In the restricted β-limited case of the problem, $\alpha = 0$, i.e., we only have the upper bound.

For $\gamma \leq 1$ and $\Delta \geq 1$, a (γ, Δ)-approximation algorithm for Proportionally Fair Matching computes a matching of size at least $\gamma \cdot |\mathrm{OPT}|$, where every color appears in at least α/Δ fraction of the edges and in at most $\beta \cdot \Delta$ fraction. OPT is an optimum (α, β)-balanced matching.

A matching is called a *rainbow matching* if all of its edges have distinct colors. We will need the following result due to Gupta et al. [23].

Theorem 1 (Theorem 2 in [23]). *For some integer $k > 0$, suppose there is a rainbow matching in G of size k. There is a $2^k \cdot n^{O(1)}$ time algorithm that computes a rainbow matching of size k.*

3 A $(\frac{1}{4\ell}, 1 + \frac{4\ell}{|OPT|})$-Approximation for Proportionally Fair Matching

In this section, we design an approximation algorithm for Proportionally Fair Matching. Let OPT be an optimum (α, β)-balanced matching (which, we assume, exists), $\text{OPT}_c = \text{OPT} \cap E_c$. We design two algorithms: one for the case when $\alpha > 0$ and the other for the complementary β-limited case. In this section, we slightly abuse the notation, and use OPT (resp. OPT_c for some color $c \in C$) to refer to $|\text{OPT}|$ (resp. $|\text{OPT}_c|$). The intended meaning should be clear from the context; however we will disambiguate in case there is a possibility of confusion.

First, we consider the $\alpha > 0$ case. Immediately, we have the following observation.

Observation 1. *For any color $c \in C$,* OPT *contains at least one edge of color c and, hence, G contains a rainbow matching of size ℓ.*

Our algorithm runs in *rounds*. In the following, we define a round. The input in each round is a subgraph $G' = (V', E')$ of G.

Round. Initially $M = \emptyset$. For every color $1 \leq c \leq \ell$, do the following in an iterative manner. If there is no edge of color c in G', go to the next color or terminate and return (G', M) if $c = \ell$. Otherwise, pick any edge e of color c from G' and add e to the already computed matching M. Remove all the edges (including e) from G' that share a common vertex with e. Repeat the process for the next color with the current (or updated) graph G' or terminate and return (G', M) if $c = \ell$. Thus in each round, we find a rainbow matching in a greedy manner.

Next, we describe our algorithm. The most challenging part of our algorithm is to ensure that the final matching computed is (α, β)-balanced modulo a small factor, i.e., we need to ensure both the lower and the upper bounds are within a small factor for each color. Note that just the above greedy way of picking edges might not even ensure that at least one edge is selected from each color. We use the algorithm of [23] in the beginning to overcome this barrier. However, the rest of our algorithm is extremely simple.

The Algorithm. We assume that we know the size of OPT. We describe later how to remove this assumption. Apply the algorithm in Theorem 1 on G to compute a rainbow matching M' of size ℓ. If $\text{OPT} \leq 4\ell^2$, return $M := M'$ as the solution and terminate. Otherwise, remove all the edges of M' and the edges adjacent to them from G to obtain the graph G_0. Initialize M to M'. Greedily pick matched edges in rounds using the **Round** procedure and add them to M until exactly $\lceil \mathrm{OPT}/(4\ell) \rceil$ edges are picked in total. In particular, the graph G_0 is the input to the 1-st round and G_1 is the output graph of the 1-st round. G_1 is the input to the 2-nd round and G_2 is the output graph of the 2-nd round, and so on. Note that it might be the case that the last round is not completed fully if the size of M is reached to $\lceil \mathrm{OPT}/(4\ell) \rceil$ before the completion of the round.

Note that the above algorithm is oblivious to α and β in the sense that it never uses these values. Nevertheless, we prove that the computed matching is (α, β)-balanced modulo a small factor. Now we analyze our algorithm.

3.1 The Analysis

Let $M_c = M \cap E_c$. Also, let c^* be a color $c \in C$ such that $|\mathrm{OPT}_c|$ is the minimum at $c = c^*$. The proof of the following observation is fairly straightforward and can be found in the full version [3].

Observation 2. *$\alpha \leq 1/\ell \leq \beta$.*

First we consider the case when $\mathrm{OPT} \leq 4\ell^2$. In this case the returned matching M is a rainbow matching of size exactly ℓ. The existence of such a matching follows by Observation 1. Thus, we immediately obtain a 4ℓ-approximation. As $|M_c|/|M| = 1/\ell$ in this case, by Observation 2, $\alpha \leq |M_c|/|M| \leq \beta$. Thus we obtain the desired result. In the rest of the proof, we analyze the case when $\mathrm{OPT} > 4\ell^2$. We start with the following lemma.

Lemma 1. *The algorithm successfully computes a matching of size exactly $\lceil \mathrm{OPT}/(4\ell) \rceil$. Moreover, for each color c with $\mathrm{OPT}_c > 4\ell$ and round $i \in [1, \lceil \mathrm{OPT}_c/(4\ell) \rceil - 1]$, G_{i-1} contains an edge of color c.*

Proof. Note that by Observation 1, the algorithm in Theorem 1 successfully computes a rainbow matching M' of size ℓ. Now consider any color c such that $\mathrm{OPT}_c \leq 4\ell$. For such a color, M already contains at least $1 \geq \lceil \mathrm{OPT}_c/(4\ell) \rceil$ edge. Now consider any other color c with $|\mathrm{OPT}_c| > 4\ell$. Consider the rainbow matching M' computed in the beginning. As $|M'| = \ell$, the edges of M' can be adjacent to at most 2ℓ edges from OPT, since it is a matching. In particular, the edges of M' can be adjacent to at most 2ℓ edges from the set OPT_c. Hence, G_0 contains at least $\mathrm{OPT}_c - 2\ell$ edges of the set OPT_c. Now consider the execution of round $i \geq 1$. At most ℓ edges are chosen in this round. Hence, these edges can be adjacent to at most 2ℓ edges of OPT_c. It follows that at most 2ℓ fewer edges of the set OPT_c are contained in G_i compared to G_{i-1}. As G_0 has at least $\mathrm{OPT}_c - 2\ell$ edges from the set OPT_c of color c and $\mathrm{OPT}_c > 4\ell$, for each of the first $\lceil (\mathrm{OPT}_c - 2\ell)/(2\ell) \rceil = \lceil \mathrm{OPT}_c/(2\ell) \rceil - 1$ rounds, the algorithm will be able to pick an edge of color c. Thus from such a color c with $\mathrm{OPT}_c > 4\ell$, it can safely pick at least $\lceil \mathrm{OPT}_c/(2\ell) \rceil \geq \lceil \mathrm{OPT}_c/(4\ell) \rceil$ edges in total. Now, as $\mathrm{OPT} = \sum_c \mathrm{OPT}_c$, $\sum_{c \in C} \lceil \mathrm{OPT}_c/(4\ell) \rceil \geq \lceil \mathrm{OPT}/(4\ell) \rceil$. It follows that the algorithm can pick at least $\lceil \mathrm{OPT}/(4\ell) \rceil$ edges. As we stop the algorithm as soon as the size of M reaches to $\lceil \mathrm{OPT}/(4\ell) \rceil$, the lemma follows.

Note that the claimed approximation factor trivially follows from the above lemma. Next, we show that M is (α, β)-balanced modulo a small factor that asymptotically tends to 1 with the size of OPT.

Lemma 2. *For each color $c \in C$, $|M_c| \geq |\mathrm{OPT}_{c^*}|/(4\ell)$.*

Proof. If $\mathrm{OPT}_{c^*} \leq 4\ell$, $|M_c| \geq 1 \geq \mathrm{OPT}_{c^*}/(4\ell)$. So, assume that $\mathrm{OPT}_{c^*} > 4\ell$. Now suppose $|M_c| < \mathrm{OPT}_{c^*}/(4\ell)$ for some c. By Lemma 1, for each of the first $\lceil \mathrm{OPT}_c/(4\ell) \rceil - 1 \geq \lceil \mathrm{OPT}_{c^*}/(4\ell) \rceil - 1$ rounds, G_{i-1} contains an edge of color c. It follows that the algorithm was forcibly terminated in some round $i \leq (\mathrm{OPT}_{c^*}/(4\ell)) - 1$. Thus, the number of edges chosen from each color $c' \neq c$ is at most $\mathrm{OPT}_{c^*}/(4\ell)$. Hence,

$$|M| = \sum_{c' \neq c} |M_{c'}| + |M_c| < (\ell - 1) \cdot (\mathrm{OPT}_{c^*}/(4\ell)) + (\mathrm{OPT}_{c^*}/(4\ell)) \leq \lceil \mathrm{OPT}/(4\ell) \rceil.$$

This contradicts Lemma 1, which states that we select exactly $\lceil \mathrm{OPT}/(4\ell) \rceil$ edges.

Corollary 1. *For each color $c \in C$, $(|M_c|/|M|) \geq \frac{\alpha}{1+\frac{4\ell}{\mathrm{OPT}}}$.*

Proof. By Lemma 2, $|M_c| \geq \mathrm{OPT}_{c^*}/(4\ell)$.

$$\frac{|M_c|}{|M|} \geq \frac{(\mathrm{OPT}_{c^*}/(4\ell))}{\lceil \mathrm{OPT}/(4\ell) \rceil} \geq \frac{(\mathrm{OPT}_{c^*}/(4\ell))}{(\mathrm{OPT}/(4\ell)) + 1} = \frac{(\mathrm{OPT}_{c^*})/(\mathrm{OPT})}{1 + \frac{4\ell}{OPT}} \geq \frac{\alpha}{1 + \frac{4\ell}{OPT}}.$$

The last inequality follows as OPT satisfies the lower bound for all colors.

Now we turn to proving the upper bound. Let $\alpha^* = \mathrm{OPT}_{c^*}/\mathrm{OPT}$.

Lemma 3. *For each color $c \in C$, $|M_c| \leq \frac{\beta}{\alpha^*} \cdot (\mathrm{OPT}_{c^*}/(4\ell)) + 1$.*

Proof. Suppose for some $c \in C$, $|M_c| > \frac{\beta}{\alpha^*} \cdot (\mathrm{OPT}_{c^*}/(4\ell)) + 1$. Then the number of rounds is strictly greater than $\frac{\beta}{\alpha^*} \cdot (\mathrm{OPT}_{c^*}/(4\ell))$. Now, for any c', $\mathrm{OPT}_{c'} \geq \alpha^* \cdot \mathrm{OPT}$ and $\mathrm{OPT}_{c'} \leq \beta \cdot \mathrm{OPT}$. Thus, by the definition of α^*, $\frac{\beta}{\alpha^*} \cdot \mathrm{OPT}_{c^*} \geq \mathrm{OPT}_{c'}$. It follows that, for each c', the number of rounds is strictly greater than $\mathrm{OPT}_{c'}/(4\ell)$. Hence, for each $c' \in C$, more than $(\mathrm{OPT}_{c'}/(4\ell)) + 1$ edges have been chosen. Thus, the total number of edges chosen is strictly larger than

$$\sum_{c' \in C} ((\mathrm{OPT}_{c'}/(4\ell)) + 1) \geq \lceil \mathrm{OPT}/(4\ell) \rceil.$$

This contradicts Lemma 1, which states that we select exactly $\lceil \mathrm{OPT}/(4\ell) \rceil$ edges.

Corollary 2. *For each color $c \in C$, $(|M_c|/|M|) \leq \beta \cdot (1 + \frac{4\ell}{\mathrm{OPT}})$.*

Proof. By Lemma 3,

$$\begin{aligned}\frac{|M_c|}{|M|} \leq \frac{(\beta/\alpha^*) \cdot (\mathrm{OPT}_{c^*}/(4\ell)) + 1}{\lceil \mathrm{OPT}/(4\ell) \rceil} &\leq \frac{(\beta/\alpha^*) \cdot (\mathrm{OPT}_{c^*}/(4\ell)) + (\beta/\alpha^*)}{\mathrm{OPT}/(4\ell)} \\ &= \frac{\beta}{\alpha^*} \cdot \frac{\mathrm{OPT}_{c^*}}{\mathrm{OPT}} \cdot \left(1 + \frac{4\ell}{\mathrm{OPT}}\right) \\ &= \frac{\beta}{\alpha^*} \cdot \alpha^* \left(1 + \frac{4\ell}{\mathrm{OPT}}\right) = \beta \cdot \left(1 + \frac{4\ell}{\mathrm{OPT}}\right).\end{aligned}$$

The second inequality follows, as $\alpha^* \leq \beta$ or $\beta/\alpha^* \geq 1$.

Now let us remove the assumption that we know the size of an optimal solution. Note that $\ell \leq \text{OPT} \leq n$. We probe all values between ℓ and n, and for each such value T run our algorithm. For each matching M returned by the algorithm, we check whether M is $(\frac{\alpha}{(1+4\ell/T)}, \beta \cdot (1 + \frac{4\ell}{T}))$-balanced. If this is the case, then we keep this solution. Otherwise, we discard the solution. Finally, we select a solution of the largest size among the ones not discarded. By the above analysis, with $T = \text{OPT}$, the matching returned satisfies the desired lower and upper bounds, and has size exactly $\lceil \text{OPT}/(4\ell) \rceil$. Finally, the running time of our algorithm is dominated by $2^\ell n^{O(1)}$ time to compute a rainbow matching algorithm, as stated in Theorem 1.

Theorem 2. *There is a $2^\ell \cdot n^{O(1)}$ time $(\frac{1}{4\ell}, 1 + \frac{4\ell}{\text{OPT}})$-approximation algorithm for* Proportionally Fair Matching *with $\alpha > 0$.*

4 A Polynomial-Time Approximation in the β-Limited Case

In the β-limited case, again we make use of the **Round** procedure. But, the algorithm is slightly different. Most importantly, we do not apply the algorithm in Theorem 1 in the beginning. Thus, our algorithm runs in polynomial time.

The Algorithm. Assume that we know the size of OPT. If $\text{OPT} \leq 2\ell$, we pick any edge and return it as the solution. Otherwise, we just greedily pick matched edges in rounds using the **Round** procedure with the following two cautions. If for a color, at least $\beta \cdot \text{OPT}/(2\ell)$ edges have already been chosen, do not choose any more edge of that color. If at least $\frac{\text{OPT}}{2\ell} - 1$ edges have already been chosen, terminate.

Now we analyze the algorithm. First note that if $\text{OPT} \leq 2\ell$, the returned matching has only one edge. The upper bound is trivially satisfied and also we obtain a 2ℓ-approximation. Henceforth, we assume that $\text{OPT} > 2\ell$. Before showing the correctness and analysis of the approximation factor, we show the upper bound for each color. Again let M be the computed matching and $M_c = M \cap E_c$. The proof of the following lemma can be found in the full version [3].

Lemma 4. *Algorithm always returns a matching of size at least* $(\text{OPT}/2\ell) - 1$.

Assuming this we have the following proposition.

Proposition 1. *For each color $c \in C$, $|M_c|/|M| \leq \beta \cdot (1 + \frac{2\ell}{|\text{OPT}|})$.*

Proof. By Lemma 4 and the threshold put on each color in the algorithm, $\frac{|M_c|}{|M|} \leq \frac{\beta \cdot \text{OPT}/(2\ell)}{(\text{OPT}/(2\ell))-1} \leq \beta \cdot (1 + \frac{2\ell}{\text{OPT}})$ The last inequality follows, as $\text{OPT} > 2\ell$.

Theorem 3. *There is a polynomial time algorithm for* Proportionally Fair Matching *in the β-limited case that returns a matching of size at least* $(\text{OPT}/2\ell) - 1$ *where every color appears in at most* $\beta \cdot (1 + 2\ell/\text{OPT})$ *fraction of the edges.*

5 An Exact Algorithm for Proportionally Fair Matching

Theorem 4. *There is a $2^{O(k)}n^{O(1)}$-time algorithm that either finds a solution of size k for a* Proportionally Fair Matching *instance, or determines that none exists.*

Proof. We present two different algorithms using the well-known technique of color coding: one for the case $\alpha = 0$ (β-limited case), and one for the case $\alpha > 0$.
β-limited case. We aim to reduce the problem to finding a rainbow matching of size k, which we then solve via Theorem 1. The graph G remains the same, however the coloring is going to be different. Namely, for each of the original colors $c \in C$ we color the edges in E_c uniformly and independently at random from a set of k' new colors, where $k' = \lfloor \beta k \rfloor$. Thus, the new instance I' is colored in $\ell \cdot k'$ colors. We use the algorithm of Theorem 1 to find a rainbow matching of size k in the colored graph in I'. Clearly, if a rainbow matching M of size k is found, then the same matching M is a β-limited matching of size k in the original coloring. This holds since by construction for any original color $c \in C$, there are k' new colors in the edge set E_c, and therefore no more than k' edges in $|M \cap E_c|$.

In the other direction, we show that if there exists a β-limited matching M of size k with respect to the original coloring, then with good probability M is a rainbow matching of size k in the new coloring. Assume the original colors c_1, $\ldots$, c_t, for some $1 \leq t \leq \ell$, have non-empty intersection with M, and for each $j \in [t]$ denote $k_j = |M \cap E_{c_j}|$. Observe that $\sum_{j=1}^{t} k_j = k$, and for each $j \in [t]$, $1 \leq k_j \leq k'$. The proof of the following claim can be found in the full version [3].

Claim. There exists some $\delta > 0$ such that for each $j \in [t]$:

$$\mathsf{Pr}\left[M \cap \left(\bigcup_{i=1}^{j} E_{c_i}\right) \text{ is a rainbow matching in } I'\right] \geq \exp\left(-\delta \sum_{i=1}^{j} k_i\right), \qquad (1)$$

Applying (1) with $j = t$, we obtain that M is a rainbow matching with probability at least $2^{-\delta k}$. By repeating the reduction above $2^{O(k)}$ times independently, we achieve that the algorithm succeeds with constant probability.

The case $\alpha > 0$. We observe that in this case, if a matching is fair it necessarily contains at least one edge from each of the groups. Thus, if the number of groups ℓ is greater than k, we immediately conclude there cannot be a fair matching of size k. Otherwise, we guess how the desired k edges are partitioned between the ℓ groups $C = \{c_1, \ldots, c_\ell\}$. That is, we guess the numbers k_j for $j \in [\ell]$ such that $\sum_{j=1}^{\ell} k_j = k$, and $\alpha k \leq k_j \leq \beta k$ for each $j \in [\ell]$. From now on, the algorithm is very similar to the β-limited case. For each group c_j, we color the edges of E_{c_j} from a set of k_j colors uniformly and independently at random, where the colors used for each E_{c_j} are non-overlapping. Now we use the algorithm of Theorem 1 to find a rainbow matching of size k. If there is a rainbow matching M of size k, the same matching is a fair matching of size k for the original instance, since in each E_{c_j} exactly k_j edges are chosen, which is at least αk and at most βk. In the other direction, if there is a fair matching M of size k in the original instance, by (1) the matching M is a rainbow matching in the new instance with probability at least $2^{-\delta k}$. Again, by repeating the coloring subprocess independently $2^{O(k)}$ times, we achieve a constant probability of success. Since there are $2^{O(k)}$ options for partitioning k edges into at most $\ell \leq k$ groups, the running time of the whole algorithm is $2^{O(k)}n^{O(1)}$.

Finally, we note that the coloring part in both cases can be derandomized in the standard fashion by using perfect hash families [42], leading to a completely deterministic algorithm.

6 Hardness of Approximation for Proportionally Fair Matching

In this section, we show an inapproximability result for Proportionally Fair Matching under the Exponential Time Hypothesis (ETH) [27]. ETH states that $2^{\Omega(n)}$ time is needed to solve any generic 3SAT instance with n variables. For our purpose, we need the following restricted version of 3SAT.

3SAT-3
INPUT: Set of clauses $T = \{C_1, \ldots, C_m\}$ in variables $x_1, \ldots, x_n$, each clause being the disjunction of 3 or 2 literals, where a literal is a variable x_i or its negation $\bar{x}_i$. Additionally, each variable appears 3 times.
QUESTION: Is there a truth assignment that simultaneously satisfies all the clauses?

3SAT-3 is known to be NP-hard [52]. We need the following stronger lower bound for 3SAT-3 proved in [13].

Proposition 2 ([13]). *Under ETH, 3SAT-3 cannot be solved in $2^{o(n)}$ time.*

We reduce 3SAT-3 to Proportionally Fair Matching which rules out any approximation for the latter problem in $2^{o(\ell)}n^{O(1)}$ time. Our reduction is as follows. For each clause C_i, we have a color i. Also, we have $n-1$ additional colors $m+1, \ldots, m+n-1$. Thus, the set of colors $C = \{1, \ldots, m+n-1\}$. For each variable x_i, we construct a gadget, which is a 3-path (a path with 3 edges). Note that x_i can either appear twice in its normal form or in its negated form, as it appears 3 times in total. Let C_{i_1}, C_{i_2} and C_{i_3} be the clauses where x_i appears. Also, suppose it appears in C_{i_1} and C_{i_3} in one form, and in C_{i_2} in the other form. We construct a 3-path P_i for x_i where the j-th edge has color i_j for $1 \leq j \leq 3$. Additionally, we construct $n-1$ 3-paths $Q_{i,i+1}$ for $1 \leq i \leq n-1$. All edges of $Q_{i,i+1}$ is of color $m+i$. Finally, we glue together all the paths in the following way to obtain a single path. For each $1 \leq i \leq n-1$, we glue $Q_{i,i+1}$ in between P_i and P_{i+1} by identifying the last vertex of P_i with the first vertex of $Q_{i,i+1}$ and the last vertex of $Q_{i,i+1}$ with the first vertex of P_{i+1}. Thus we obtain a path P with exactly $3(n+n-1) = 6n-3$ edges. Finally, we set $\alpha = \beta = 1/(m+n-1)$.

Lemma 5. *There is a satisfying assignment for the clauses in 3SAT-3 if and only if there is an (α, β)-balanced matching of size at least $m + n - 1$.*

Proof. Suppose there is a satisfying assignment for all the clauses. For each clause C_j, consider a variable, say x_i, that satisfies C_j. Then there is an edge of color j on P_i. Add this edge to a set M. Thus, after this step, M contains exactly one edge of color j for $1 \leq j \leq m$. Also, note that for each path P_i, if the middle edge is chosen, then no other edge from P_i can be chosen. This is true, as the variable x_i can either satisfy the clauses where it appears in its normal form or the clauses where it appears in its negated form, but not both types of clauses. Hence, M is a matching. Finally, for each path $Q_{i,i+1}$, we add its middle edge to M. Note that M still remains a matching. Moreover, M contains exactly one edge of color j for $1 \leq j \leq m + n - 1$. As $\alpha = \beta = 1/(m + n - 1)$, M is an (α, β)-balanced matching.

Now suppose there is an (α, β)-balanced matching M of size at least $m+n-1$. First, we show that $|M| = m+n-1$. Note that if $|M| > m+n-1$, then the only possibility is that $|M| = 2(m+n-1)$, as $\alpha = \beta$ and at most 2 edges of color j can be picked in any matching for $m+1 \leq j \leq m+n-1$. Suppose $|M| = 2(m+n-1)$. Then from each $Q_{i,i+1}$, M contains the first and the third edge. This implies, from each P_t, $1 \leq t \leq n$, we can pick at most one edge. Thus, total number of edges in M is at most $2(n - 1) + n$. It follows that $2m + 2n - 2 \leq 2n - 2 + n$ or $n \geq 2m$. Now, in 3SAT-3 the total number of literals is $3n$ and at most $3m$, as each variable appears 3 times and each clause contains at most 3 literals. This implies $n \leq m$, and we obtain a contradiction. Thus, $|M| = m + n - 1$. Now, consider any P_i. In the first case, the first and third edges of P_i are corresponding to literal x_i and, hence, the middle edge is corresponding to the literal $\bar{x}_i$. If the middle edge is in M, assign 0 to x_i, otherwise, assign 1 to x_i. In the other case, if the middle edge is in M, assign 1 to x_i, otherwise, assign 0 to x_i. We claim that the constructed assignment satisfies all the clauses. Consider any clause C_j. Let $e \in P_i$ be the edge in M of color j for $1 \leq j \leq m$. Note that e can be the middle edge in P_i or not. In any case, if e is corresponding to $\bar{x}_i$, we assigned 0 to x_i, and if e is corresponding to x_i, we assigned 1 to x_i. Thus, in either case, C_j is satisfied. This completes the proof of the lemma. □

Note that for a 3SAT-3 instance the total numbers of literals is $3n$. As each clause contains at least 2 literals, $m \leq 3n/2$. Now, for the instances constructed in the above proof, the number of colors $\ell = m+n-1 \leq 3n/2+n-1 = 5n/2-1$. Thus, the above lemma along with Proposition 2 show that it is not possible to decide whether there is an (α, β)-balanced matching of a given size in time $2^{o(\ell)}n^{O(1)}$. Using this, we also show that even no $2^{o(\ell)}n^{O(1)}$ time approximation algorithm is possible. Suppose there is a $2^{o(\ell)}n^{O(1)}$ time γ-approximation algorithm, where $\gamma < 1$. For our constructed path instances, we apply this algorithm to obtain a matching. Note that the γ-approximate solution M must contain at least one edge of every color, as $\alpha = \beta$. By the proof in the above lemma, $|M|$ is exactly $m + n - 1$. Hence, using this algorithm, we can decide in $2^{o(\ell)}n^{O(1)}$ time whether there is an (α, β)-balanced matching of size $m + n - 1$. But, this is a contradiction, which leads to the following theorem.

Theorem 5. *For any $\gamma > 1$, under ETH, there is no $2^{o(\ell)} n^{O(1)}$ time γ-approxim- ation algorithm for* Proportionally Fair Matching*, even on paths.*

7 Conclusions

In this paper, we study the notion of proportional fairness in the context of matchings in graphs, which has been studied by Chierichetti et al. [9]. We obtained approximation and exact algorithms for the proportionally fair matching problem. We also complement these results by showing hardness results. It would be interesting to obtain a $o(\ell)$- or a true $O(\ell)$-approximation for Proportionally Fair Matching improving our result. As evident from our hardness result, there is a lower bound of $2^{\Omega(\ell)} n^{O(1)}$ on the running time of such an algorithm.

References

1. Ahmadi, S., Ahmed, F., Dickerson, J.P., Fuge, M., Khuller, S.: An algorithm for multi-attribute diverse matching. In: Bessiere, C. (ed.) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, pp. 3–9. ijcai.org (2020)
2. Alon, N., Yuster, R., Zwick, U.: Color-coding. J. ACM (JACM) **42**(4), 844–856 (1995)
3. Bandyapadhyay, S., Fomin, F.V., Inamdar, T., Simonov, K.: Proportionally fair matching with multiple groups. CoRR abs/2301.03862 (2023). https://doi.org/10.48550/arXiv.2301.03862
4. Bei, X., Liu, S., Poon, C.K., Wang, H.: Candidate selections with proportional fairness constraints. In: Seghrouchni, A.E.F., Sukthankar, G., An, B., Yorke-Smith, N. (eds.) Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2020, Auckland, New Zealand, 9–13 May 2020, pp. 150–158. International Foundation for Autonomous Agents and Multiagent Systems (2020). https://doi.org/10.5555/3398761.3398784, https://dl.acm.org/doi/10.5555/3398761.3398784
5. Benedek, M., Biró, P., Kern, W., Paulusma, D.: Computing balanced solutions for large international kidney exchange schemes. In: Faliszewski, P., Mascardi, V., Pelachaud, C., Taylor, M.E. (eds.) 21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022, Auckland, New Zealand, 9–13 May 2022, pp. 82–90. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS) (2022). https://doi.org/10.5555/3535850.3535861, https://www.ifaamas.org/Proceedings/aamas2022/pdfs/p82.pdf
6. Berger, A., Bonifaci, V., Grandoni, F., Schäfer, G.: Budgeted matching and budgeted matroid intersection via the gasoline puzzle. Math. Program. **128**(1–2), 355–372 (2011)
7. Boehmer, N., Koana, T.: The complexity of finding fair many-to-one matchings. In: Bojanczyk, M., Merelli, E., Woodruff, D.P. (eds.) 49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, 4–8 July 2022, Paris, France. LIPIcs, vol. 229, pp. 27:1–27:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.ICALP.2022.27

8. Charlin, L., Zemel, R.: The toronto paper matching system: an automated paper-reviewer assignment system (2013)
9. Chierichetti, F., Kumar, R., Lattanzi, S., Vassilvitskii, S.: Fair clustering through fairlets. In: Advances in Neural Information Processing Systems, pp. 5029–5037 (2017)
10. Chierichetti, F., Kumar, R., Lattanzi, S., Vassilvitskii, S.: Matroids, matchings, and fairness. In: Chaudhuri, K., Sugiyama, M. (eds.) The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16–18 April 2019, Naha, Okinawa, Japan. Proceedings of Machine Learning Research, vol. 89, pp. 2212–2220. PMLR (2019)
11. Chouldechova, A.: Fair prediction with disparate impact: a study of bias in recidivism prediction instruments. Big Data **5**(2), 153–163 (2017)
12. Corbett-Davies, S., Pierson, E., Feller, A., Goel, S., Huq, A.: Algorithmic decision making and the cost of fairness. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017, pp. 797–806. ACM (2017)
13. Cygan, M., Marx, D., Pilipczuk, M., Pilipczuk, M.: Hitting forbidden subgraphs in graphs of bounded treewidth. Inf. Comput. **256**, 62–82 (2017)
14. Czabarka, E., Szekely, L.A., Toroczkai, Z., Walker, S.: An algebraic monte-carlo algorithm for the bipartite partition adjacency matrix realization problem. arXiv preprint arXiv:1708.08242 (2017)
15. Dwork, C., Hardt, M., Pitassi, T., Reingold, O., Zemel, R.: Fairness through awareness. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, pp. 214–226 (2012)
16. Dwork, C., Ilvento, C.: Group fairness under composition. In: Proceedings of the 2018 Conference on Fairness, Accountability, and Transparency (FAT* 2018) (2018)
17. Ebadian, S., Kahng, A., Peters, D., Shah, N.: Optimized distortion and proportional fairness in voting. In: Proceedings of the 23rd ACM Conference on Economics and Computation, pp. 563–600 (2022)
18. Edmonds, J.: Paths, trees, and flowers. Can. J. Math. **17**(3), 449–467 (1965)
19. Feldman, M., Friedler, S.A., Moeller, J., Scheidegger, C., Venkatasubramanian, S.: Certifying and removing disparate impact. In: proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 259–268 (2015)
20. Freeman, R., Micha, E., Shah, N.: Two-sided matching meets fair division. In: Zhou, Z. (ed.) Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19–27 August 2021, pp. 203–209. ijcai.org (2021). https://doi.org/10.24963/ijcai.2021/29
21. García-Soriano, D., Bonchi, F.: Fair-by-design matching. Data Min. Knowl. Disc. **34**(5), 1291–1335 (2020). https://doi.org/10.1007/s10618-020-00675-y
22. Goel, N., Yaghini, M., Faltings, B.: Non-discriminatory machine learning through convex fairness criteria. In: Furman, J., Marchant, G.E., Price, H., Rossi, F. (eds.) Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society, AIES 2018, New Orleans, LA, USA, 02–03 February 2018, p. 116. ACM (2018). https://doi.org/10.1145/3278721.3278722
23. Gupta, S., Roy, S., Saurabh, S., Zehavi, M.: Parameterized algorithms and kernels for rainbow matching. Algorithmica **81**(4), 1684–1698 (2019)
24. Hall, P.: On representatives of subsets. J. London Math. Soc. **10**, 26–30 (1935)
25. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Comput. **2**, 225–231 (1973)

26. Huang, C., Kavitha, T., Mehlhorn, K., Michail, D.: Fair matchings and related problems. Algorithmica **74**(3), 1184–1203 (2016)
27. Impagliazzo, R., Paturi, R.: On the complexity of k-sat. J. Comput. Syst. Sci. **62**(2), 367–375 (2001)
28. Kamada, Y., Kojima, F.: Fair matching under constraints: theory and applications (2020)
29. Kamishima, T., Akaho, S., Sakuma, J.: Fairness-aware learning through regularization approach. In: Spiliopoulou, M., et al. (eds.) Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on, Vancouver, BC, Canada, 11 December 2011, pp. 643–650. IEEE Computer Society (2011)
30. Kesavan, D., Periyathambi, E., Chokkalingam, A.: A proportional fair scheduling strategy using multiobjective gradient-based African buffalo optimization algorithm for effective resource allocation and interference minimization. Int. J. Commun Syst **35**(1), e5003 (2022)
31. Klaus, B., Klijn, F.: Procedurally fair and stable matching. Econ. Theory **27**(2), 431–447 (2006)
32. Kleinberg, J., Mullainathan, S., Raghavan, M.: Inherent trade-offs in the fair determination of risk scores. In: 8th Innovations in Theoretical Computer Science Conference (ITCS 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017)
33. Kőnig, D.: Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. Math. Ann. **77**(4), 453–465 (1916)
34. Kurata, R., Hamada, N., Iwasaki, A., Yokoo, M.: Controlled school choice with soft bounds and overlapping types. J. Artif. Intell. Res. **58**, 153–184 (2017)
35. Linhares, A., Olver, N., Swamy, C., Zenklusen, R.: Approximate multi-matroid intersection via iterative refinement. Math. Program. **183**(1), 397–418 (2020). https://doi.org/10.1007/s10107-020-01524-y
36. Lokshtanov, D., Misra, P., Panolan, F., Saurabh, S.: Deterministic truncation of linear matroids. ACM Trans. Algorithms **14**(2), 14:1-14:20 (2018). https://doi.org/10.1145/3170444
37. Lovász, L., Plummer, M.D.: Matching Theory. AMS (2009)
38. Lu, Y.: The optimization of automated container terminal scheduling based on proportional fair priority. Math. Probl. Eng. 2022 (2022)
39. Marx, D.: A parameterized view on matroid optimization problems. Theoret. Comput. Sci. **410**(44), 4471–4479 (2009)
40. Mądry, A.: Navigating central path with electrical flows: from flows to matchings, and back. In: FOCS 2013, pp. 253–262. IEEE Computer Society (2013)
41. Mucha, M., Sankowski, P.: Maximum matchings via Gaussian elimination. In: FOCS 2004, pp. 248–255. IEEE Computer Society (2004)
42. Naor, M., Schulman, L.J., Srinivasan, A.: Splitters and near-optimal derandomization. In: Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS 1995), pp. 182–191. IEEE (1995)
43. Nguyen, M.H., Baiou, M., Nguyen, V.H., Vo, T.Q.T.: Nash fairness solutions for balanced tsp. In: 10th International Network Optimization Conference (INOC) (2022)
44. Rabin, M.O., Vazirani, V.V.: Maximum matchings in general graphs through randomization. J. Algorithms **10**(4), 557–567 (1989)
45. Ristoski, P., Petrovski, P., Mika, P., Paulheim, H.: A machine learning approach for product matching and categorization. Semant. web **9**(5), 707–728 (2018)
46. Roth, A.E., Sönmez, T., Ünver, M.U.: Efficient kidney exchange: coincidence of wants in markets with compatibility-based preferences. Am. Econ. Rev. **97**(3), 828–851 (2007)

47. Schrijver, A.: Combinatorial Optimization. Polyhedra and Efficiency, vol. A. Springer-Verlag, Berlin (2003)
48. St-Arnaud, W., Carvalho, M., Farnadi, G.: Adaptation, comparison and practical implementation of fairness schemes in kidney exchange programs. arXiv preprint arXiv:2207.00241 (2022)
49. Stamoulis, G.: Approximation algorithms for bounded color matchings via convex decompositions. In: Csuhaj-Varjú, E., Dietzfelbinger, M., Ésik, Z. (eds.) MFCS 2014. LNCS, vol. 8635, pp. 625–636. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44465-8_53
50. Sun, Z., Todo, T., Walsh, T.: Fair pairwise exchange among groups. In: IJCAI, pp. 419–425 (2021)
51. Thanh, B.L., Ruggieri, S., Turini, F.: k-NN as an implementation of situation testing for discrimination discovery and prevention. In: Apté, C., Ghosh, J., Smyth, P. (eds.) Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 21–24 August 2011, pp. 502–510. ACM (2011)
52. Yannakakis, M.: Node- and edge-deletion np-complete problems. In: Lipton, R.J., Burkhard, W.A., Savitch, W.J., Friedman, E.P., Aho, A.V. (eds.) Proceedings of the 10th Annual ACM Symposium on Theory of Computing, 1–3 May 1978, San Diego, California, USA, pp. 253–264. ACM (1978)
53. Zhang, G., Malekmohammadi, S., Chen, X., Yu, Y.: Equality is not equity: Proportional fairness in federated learning. arXiv preprint arXiv:2202.01666 (2022)

Reconstructing Graphs from Connected Triples

Paul Bastide[1](✉), Linda Cook[2], Jeff Erickson[3], Carla Groenland[4],
Marc van Kreveld[4], Isja Mannens[4], and Jordi L. Vermeulen[4]

[1] LaBRI - Bordeaux University, Bordeaux, France
paul.bastide@ens-rennes.fr
[2] Institute for Basic Science, Discrete Math Group, Daejeon, Republic of Korea
linda.cook@ibs.re.kr
[3] University of Illinois, Urbana-Champaign, Champaign, USA
jeffe@illinois.edu
[4] Utrecht University, Utrecht, The Netherlands
{c.e.groenland,m.j.vankreveld,i.m.e.mannens}@uu.nl

Abstract. We introduce a new model of indeterminacy in graphs: instead of specifying all the edges of the graph, the input contains all triples of vertices that form a connected subgraph. In general, different (labelled) graphs may have the same set of connected triples, making unique reconstruction of the original graph from the triples impossible. We identify some families of graphs (including triangle-free graphs) for which all graphs have a different set of connected triples. We also give algorithms that reconstruct a graph from a set of triples, and for testing if this reconstruction is unique. Finally, we study a possible extension of the model in which the subsets of size k that induce a connected graph are given for larger (fixed) values of k.

Keywords: Algorithms · Graph reconstruction · Indeterminacy · Uncertainty · Connected Subgraphs

1 Introduction

Imagine that we get information about a graph, but not its complete structure by a list of edges. Does this information uniquely determine the graph? In this paper we explore the case where the input consists of all triples of vertices whose induced subgraph is connected. In other words, we know for each given triple of vertices that two or three of the possible edges are present, but we do not know which ones. We may be able to deduce the graph fully from all given triples.

LC is supported by the Institute for Basic Science (IBS-R029-C1) and CG by Marie-Skłodowska Curie grant GRAPHCOSY (number 101063180). MvK and JV are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.651.

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 16–29, 2023.
https://doi.org/10.1007/978-3-031-43380-1_2

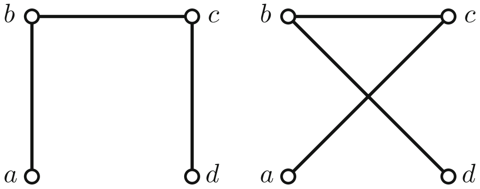


Fig. 1. Two different labelled trees that give the same set of connected triples.

As a simple example, assume we are given the (unordered, labelled) triples *abc*, *bcd*, and *cde*. Then the only (connected) graph that matches this specification by triples is the path a—b—c—d—e. On the other hand, if we are given all possible triples on a set of four vertices a, b, c, d except for *abc*, then there are several graphs possible. We must have the edges *ad*, *bd*, and *cd*, and zero or one of the edges *ab*, *bc*, and *ca*. See Fig. 1 for another example.

This model of indeterminacy of a graph does not use probability and is perhaps the simplest combinatorial model of partial information. Normally a graph is determined by pairs of vertices which are the edges; now we are given triples of vertices with indeterminacy on the edges between them. As such, we believe this model is interesting to study.

As illustrated in our previous example, there are cases where reconstruction of the graph from the set T of triples is unique and there are cases where multiple (labelled) graphs may have the same set of triples. There are also cases where T is not consistent with any graph, such as $T = \{abc, cde\}$. Can we characterize these cases, and what can we say if we have additional information, for example, when we know that we are reconstructing a tree or a triangle-free graph?

1.1 Our Results

After preliminaries in Sect. 2, we provide two relatively straightforward, general algorithms for reconstruction in Sect. 3. One runs in $O(n^3)$ time when the triples use n vertex labels, and the other runs in $O(n \cdot |T|)$ time when there are $|T|$ triples in the input. These algorithms return a graph that is consistent with the given triples, if one exists, and decide on uniqueness.

Then, we give an $O(|T|)$ time algorithm to reconstruct trees on at least five vertices, provided that the unknown graph is known to be a tree, in Sect. 4. In fact, all triangle-free graphs can be reconstructed, provided we know that the unknown graph is triangle-free. We give an algorithm running in expected $O(|T|)$ time for this in Sect. 5. Moreover, we show that 2-connected outerplanar graphs and triangulated planar graphs can be uniquely reconstructed.

In Sect. 6 we study a natural extension of the model where we are given the connected k-sets of a graph for some fixed $k \geq 4$, rather than the connected triples. We show the largest value of k such that each n-vertex tree is distinguished from other trees by its set of connected k-sets is $\lceil n/2 \rceil$. A similar threshold is shown for the random graph. Finally, we show that graphs with girth strictly larger than k, on at least $2k-1$ vertices, can be uniquely reconstructed, among the class of thus graphs, by their collection of connected k-sets.

1.2 Related Work

The problem of graph reconstruction arises naturally in many cases where some unknown graph is observed indirectly. For instance, we may have some (noisy) measurement of the graph structure, or only have access to an oracle that answers specific types of queries. Much previous research has been done for specific cases, such as reconstructing metric graphs from a density function [10], road networks from a set of trajectories [1], graphs using a shortest path or distance oracle [19], labelled graphs from all r-neighbourhoods [25], or reconstructing phylogenetic trees [7]. A lot of research has been devoted to the *graph reconstruction conjecture* [21,29], which states that it is possible to reconstruct any graph on at least three vertices (up to isomorphism) from the multiset of all (unlabeled) subgraphs obtained through the removal of one vertex. This conjecture is open even for planar graphs and triangle-free graphs, but has been proved for outerplanar graphs [15] and maximal planar graphs [22]. We refer the reader to one of the many surveys (e.g. [5,17,23,28]) for further background. Related to our study of the random graph in Sect. 6 is a result from Cameron and Martins [8] from 1993, which implies that for each graph H, with high probability the random graph $G \sim G(n, \frac{1}{2})$ can be reconstructed from the set of (labelled) subsets that induce a copy of H (up to complementation if H is self-complementary).

Many types of uncertainty in graphs have been studied. Fuzzy graphs [27] are a generalisation of fuzzy sets to relations between elements of such sets. In a fuzzy set, membership of an element is not binary, but a value between zero and one. Fuzzy graphs extend this notion to the edges, which now also have a degree of membership in the set of edges. Uncertain graphs are similar to fuzzy graphs in that each edge has a number between zero and one associated with it, although here this number is a probability of the edge existing. Much work has been done on investigating how the usual graph-theoretic concepts can be generalised or extended to fuzzy and uncertain graphs [20,24].

2 Preliminaries

All graphs in this paper are assumed to be connected, finite, and simple. Let G be an unknown graph with n vertices and let T be the set of all triples of vertices that induce a connected subgraph in G. Since the graph is connected, we can recover the vertex set V of G easily from T. We will use $\overline{T}$ to denote the complement of this set T, i.e. $\overline{T}$ is the set of all triples of vertices for which the induced subgraph is not connected. Note that $\left|T \cup \overline{T}\right| = \binom{n}{3} \in \Theta(n^3)$.

Observe that both the presence and absence of a triple gives important information: in the former case, at most one of the three possible edges is absent, whereas in the latter case, at most one of these edges is present.

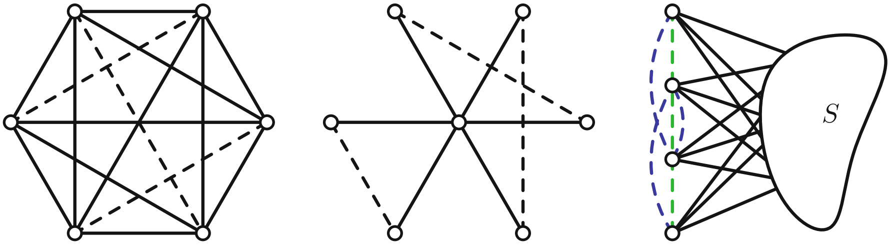


Fig. 2. Three classes of ambiguous triples: a complete graph minus any independent set of edges, a star graph plus any (partial) matching of the leaves, and a path of length four in which all vertices are fully adjacent to some set S. In this last case, we cannot tell the difference between the red and green path. (Color figure online)

It is possible that graphs that are not the same (as labelled graph) or even not isomorphic yield the same set of triples, for example, a path on three vertices and a triangle. We also give examples of larger graphs that cannot be distinguished from their set of connected triples in Fig. 2.

We will make use of (LSD) radix string sorting (as described in e.g. [9]) to sort a collection of t sets of cardinality k in time $\mathcal{O}(tk)$ in several of the algorithms presented in this paper.

3 Algorithm for Finding Consistent Graphs from Triples

Given a set of triples T, we can find a graph G consistent with those triples by solving a 2-SAT formula. The main observation here is that the presence of a triple abc means that at least two of the edges ab, ac and bc must exist, whereas the absence of a triple means at most one of the edges can exist. We can then construct a 2-SAT formula where each variable corresponds to an edge of the graph, and truth represents presence of that edge. For each triple $abc \in T$, we add clauses $(ab \vee ac)$, $(ab \vee bc)$ and $(ac \vee bc)$ to the formula. For each triple $abc \in \overline{T}$, we add clauses $(\neg ab \vee \neg ac)$, $(\neg ab \vee \neg bc)$ and $(\neg ac \vee \neg bc)$. A graph consistent with the set of triples can then be found by solving the resulting 2-SAT formula and taking our set of edges to be the set of true variables in the satisfying assignment. If the formula cannot be satisfied, no graph consistent with T exists.

We can solve the 2-SAT formula in linear time with respect to the length of the formula [2, 11]. We add a constant number of clauses for each element of T and $\overline{T}$, so our formula has length $O(|T \cup \overline{T}|)$. As $|T \cup \overline{T}| = \binom{n}{3}$, this gives us an $O(n^3)$ time algorithm to reconstruct a graph with n vertices. However, we prefer an algorithm that depends on the size of T, instead of also on the size of $\overline{T}$. We can eliminate the dependency on the size of $\overline{T}$ by observing that some clauses can be excluded from the formula because the variables cannot be true.

Lemma 1. *We can find a graph G consistent with T in $O(n \cdot |T|)$ time, or output that no consistent graph exists.*

Proof. The basic observation that allows us to exclude certain clauses from the formula is that if there is no connected triple containing two vertices a and b, the variable ab will always be false. Consequently, if we have a triple $abc \in \overline{T}$ for which at most one of the pairs ab, ac and bc appear in some connected triple, we do not need to include its clauses in the formula, as at least two of the variables will be false, making these clauses necessarily satisfied.

We can construct the formula that excludes these unnecessary clauses in $O(n \cdot |T|)$ time as follows. We build a matrix $M(i,j)$, where $i,j \in V(G)$ with each entry containing a list of all vertices with which i and j appear in a connected triple, i.e. $M(i,j) = \{x \mid ijx \in T\}$. This matrix can be constructed in $O(n^2 + |T|)$ time, by first setting every entry to $\emptyset$ (this takes n^2 time) and then running through T, adding every triple abc to the entries $M(a,b)$, $M(b,c)$ and $M(a,c)$. We also sort each list in linear time using e.g. radix sort. As the total length of all lists is $O(|T|)$, this takes $O(n^2 + |T|)$ time in total.

Using this matrix, we can decide which clauses corresponding to triples from $\overline{T}$ to include as follows. For all pairs of vertices (a,b) that appear in some connected triple (i.e. $M(a,b) \neq \emptyset$), we find all x such that $abx \in \overline{T}$. As $M(a,b)$ is sorted, we can find all x in $O(n)$ time by simply recording the missing elements of the list $M(a,b)$. We then check if $M(a,x)$ and $M(b,x)$ are empty. If either one is not, we include the clause associated with $abx \in \overline{T}$ in our formula. Otherwise, we can safely ignore this clause, as it is necessarily satisfied by the variables for ax and bx being false.

Our algorithm takes $O(n)$-time for each non-empty element of $M(i,j)$, of which there are $O(|T|)$, plus $O(n^2)$ time to traverse the matrix. The total time to construct the formula is $O(n^2 + n \cdot |T|)$. As $|T| \in \Omega(n)$ for connected graphs, this simplifies to $O(n \cdot |T|)$ time. The resulting formula also has $O(n \cdot |T|)$ length, and can be solved in time linear in that length. □

Observe that this is only an improvement on the naive $O(n^3)$ approach if $|T| \in o(n^2)$. We also note that we can test the uniqueness of the reconstruction in the same time using Feder's approach for enumerating 2-SAT solutions [12].

4 Unique Reconstruction of Trees

In this section, we prove the following result.

Theorem 1. *Let T be a set of triples, and let it be known that the underlying graph $G = (V,E)$ is a tree. If $n \geq 5$, then G can be uniquely reconstructed in $O(|T|)$ time.*

Let us briefly examine trees with three or four vertices. A tree with three vertices is always a path and it will always have one triple with all three vertices. We do not know which of the three edges is absent. A tree with four vertices is either a path or a star. The path has two triples and the star has three triples. For the star, the centre is the one vertex that appears in all three triples, and hence the reconstruction is unique. For the path, we will know that the graph is a path, but we will not know in what order the middle two vertices appear (see Fig. 1).

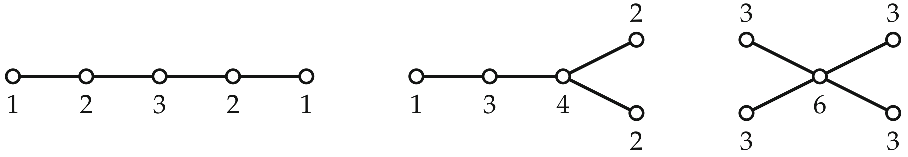


Fig. 3. All trees on five vertices, and the number of triples each vertex occurs in.

Next we consider trees with at least five vertices. We first show that we can recognise all leaves and their neighbours from the triples. In the following, we say that a vertex v *dominates* a vertex u if v appears in all the triples that u appears in. If u is a leaf, then it is dominated by its unique neighbour v. It is possible that u is also dominated by a neighbour of v, but in this case uvw will be the only triple containing u, and v will be dominated by w. Moreover, if $|V| \geq 4$, there exists a triple vwx for some vertex x different from u. This can be used to recognise the leaves as long as $|V| \geq 4$. Moreover, if $|V| \geq 5$, we can also identify the neighbour v of each leaf u as either the unique vertex that dominate u or when u is dominated by two vertices v and w, there exits $x, y \in V$ such that wxy is a triple and w dominates v. We can use this to prove that any tree can be reconstructed from its triples, provided that we know that the result must be a tree and $|V| \geq 5$, since we can iteratively recognise and remove vertices of degree 1, while recording where to 'glue them back at the end' until at most four vertices remain. We can complete the reconstruction via some closer examination of the connected triples in the original tree that contain the remaining vertices.

In order to derive an optimal, $O(|T|)$ time reconstruction algorithm, we will use a further characterisation of vertices of a tree using the triples. The main idea is that we can recognise not only leaves, but also other vertices where we can reduce the tree. If a vertex v has degree 2 in a tree, then there are two nodes w, w' such that every triple with v also contains w or w' (or both). The converse is not true for two reasons: if v is a leaf, it also has the stated property, and if v has degree 3 where at least one neighbour is a leaf, then it has this property as well. This brings us to the following characterisation.

Lemma 2. *A vertex v of a tree G of at least five vertices with triple set T is:*

(i) *a leaf if and only if v is dominated by some vertex w and does not dominate any vertex itself;*

(ii) *if v is not a leaf, then v is (a) a node of degree 2, or (b) a node of degree 3 with at least one leaf neighbour, if and only if there are two nodes w_1, w_2 such that all triples with v also contain w_1 or w_2.*

Moreover, both characterisations can be checked in time $O(|T_v|)$, when the set of triples T_v that include v is given for all $v \in V(G)$.

Proof. A leaf v can necessarily only appear in triples with its adjacent vertex w, as it is not adjacent to any other vertices by definition. A leaf is therefore always dominated by its neighbour w. Since $|V| \geq 5$, v does not dominate any vertex. Conversely, suppose that v is dominated by some vertex w and does not dominate any other vertex. It is straightforward to check that v cannot be dominated if it has three neighbours or if it has two non-leaf neighbours. Since v does not dominate any vertices, it does not have a leaf neighbour. So v must be a leaf itself. This proves (i).

The second characterisation can be seen as follows. If v has degree at least 4, then no w_1, w_2 as in (ii) exist, which is easily verified by looking at the triples with v and its neighbours only. Furthermore, if v has degree 3 and none of its neighbours are leaves, then again there are no such w_1, w_2. On the other hand, in case (a) the two neighbours can be taken as w_1, w_2 and in case (b) w_1, w_2 can be chosen to be two neighbours of v so that the unique neighbour of v that is not in $\{w_1, w_2\}$ is a leaf.

For testing (i), take any triple $vab \in T_v$, and test both a and b separately if they are the sought w. For testing (ii), take any triple $vab \in T_v$. If characterisation (ii) holds, then w_1 must be a or b. We try both as follows: For $w \in \{a, b\}$ we remove all triples with w from T_v. Then $w = w_1$ if and only if all of the remaining triples of T_v all contain some $w_2 \neq v$. We test this by looking at some remaining triple in $vcd \in T_v$ and testing whether either c or d is contained in every other remaining triple. In total, we get four options to test for w_1 and w_2; each option is easily checked in $O(|T_v|)$ time. □

The vertices of V partition into V', V'', and V''', where V' contains the leaves, V'' contains the vertices that are not leaves but satisfy the second condition of the lemma, and $V''' = V \setminus (V' \cup V'')$. Note that more than half of the vertices of G are in $V' \cup V''$. We next turn to the proof of Theorem 1. We will assume that there is a total order on the vertex set of G and that the connected triples uvw are stored in an ordered tuple with $u < v < w$. For each triple uvw in T, we generate vwu and wuv as well. We collect the triples with the same first vertex to generate T_v for all $v \in V$. We will begin by showing how we recognize whether each vertex is in V', V'' or V'''.

Then, for all $v \in V$, we use T_v to test if v is dominated by some vertex u. We can find the vertices that dominate v, in time $O(|T_v|)$, by examining the first triple vwx and noting that only w and x can dominate v. We then check every other triple in T_v for the presence of w and x. By labeling dominated/dominating vertices as we go we can, in time $O(|T|)$, find all leaves as vertices that are dominated by some vertex and don't dominate any vertex themselves, i.e. all vertices that satisfy condition (i) of Lemma 2. We then check T_v for the remaining vertices, to see if condition (ii) is satisfied and find the vertices w_1 and w_2 in a similar fashion. We again start with the first triple vwx and check if there is some triple vyz that does not contain w or x. If not, then $w1 = w$ and $w_2 = x$. Otherwise we have four candidates w, x, y, z for w_1 and w_2 and we check, for every pair, whether there is a triple that contains neither of them. This can be done in $O(|T_v|)$ time.

After this, we remove all triples containing a leaf from T. Let G' be the graph obtained by removing all leaves and incident edges. Then the new triple set is the set of connected triples for this graph G', and we can recover G from G'.

For all vertices in V'' note that they can no longer be vertices of degree 3 in G', but they may have become leaves. We test this and consider the subset $W \subseteq V''$ of vertices that have not become leaves.

The subgraph of G' induced on W consists of a disjoint union of paths. Let $v \in W$. Then v has exactly two neighbours in $V(G')$ and they are the two vertices w_1, w_2 satisfying the second condition of the Lemma 2. As mentioned above, we can find these two vertices w_1, w_2 for each $v \in W$ in time $O(|T_v|)$. In particular, we know all path components of $G'[W]$, as well as the unique vertices in $V(G') \setminus W$ that the endpoints of any such path are adjacent to. Suppose that v_1—v_2—...—v_ℓ. is one of the path components of $G[W]$. Let $x_1, x_2 \in V(G') \setminus W'$ such that x_1 is the other neighbour of v_1, and x_2 is the other neighbour of v_k (in G'). We record the edges x_1v_1 and x_2v_k, as well as the edges and vertices in the path $v_1, \ldots, v_\ell$. Then we replace each triple ux_1v_1 by ux_1x_2 and each triple v_kx_2u by x_1x_2u. Afterwards, we discard all triples that contain any of $v_1, \ldots, v_\ell$. Let G'' be the graph obtained by deleting $v_1, \ldots, v_\ell$ and adding the edge $\{x_1, x_2\}$. The resulting triple set is the triple set for G'', and we can recover G from G''. We repeat this for all path components. Note that G'' is a tree, if and only if G' is a tree and thus we maintain throughout that the stored triple set corresponds to a tree, and that we can reconstruct the original tree G from knowing this tree and the additional information that we record.

Finally, we also remove all leaves in $V'' \setminus W$ by discarding more triples, similar to the first leaf removal. This process takes time linear in $|T|$, and reduces the number of vertices occurring in T to half or less. We recurse the process on the remaining tree until it has size five, at which point we can uniquely identify the structure of the tree by simply looking at the number of triples each vertex occurs in (see Fig. 3). We may not remove all vertices of V' or V'' if the remaining tree would be smaller than five vertices; in that case, we can simply leave some leaf or not contract the paths in W completely. A standard recurrence shows that the total time used is $O(|T|)$. This finishes the proof of Theorem 1.

We note that if the tree contains no leaves that are siblings, then we do not need to know that the graph is a tree for unique reconstruction.

5 Further Reconstructible Graph Classes

In this section, we give larger classes of graphs for which the graphs that are determined by their set of connected triples.

Theorem 2. *There is an algorithm that reconstructs a graph G on $n \geq 5$ vertices that is known to be triangle-free from its set T of triples in deterministic $O(|T|\log(|T|))$ time or randomized $O(|T|)$ expected time.*

Proof. Let T be the given list of connected triples. For every triple abc we create three ordered copies abc, acb, bca. We then sort the list in lexicographical order in $O(|T|)$ time using radix sort. For every potential edge ab that appears as the first two vertices of some triple we test whether it is an edge as follows.

If we find two triples abc and abd for some $c, d \in V(G)$, we search for the triples acd and bcd. If $ab \notin E(G)$, then we must have that $bc, ac, bd, ad \in E(G)$, and thus both acd and bcd are connected. Therefore, if either triple is not in the list, we know that ab is an edge. Otherwise, we find that $\{a, b, c, d\}$ induces a C_4. We then search for another vertex e that appears in a triple with any of a, b, c, d and reconstruct the labeling of the C_4 as follows. Suppose $G[\{a, b, c, d\}]$ induces a C_4 and we try to retrieve the exact order of the vertices in the cycle. Assume w.l.o.g. that a has the remaining vertex e as a neighbour, then since G is triangle-free, e is not adjacent to b and d. This means that bde is known to be disconnected, whereas abe, ade are connected. If e is not a neighbour of c, then ace is not connected. Hence, if any of a, b, c, d has a private neighbour (one not adjacent to other vertices in the cycle), then we get the labeling of our C_4 (and find that e is a private neighbour of a). If e is adjacent to c besides a, then we know the following two vertex sets also induce C_4's: $abce, acde$. We know e is adjacent to two out of $\{a, b, c\}, \{a, d, c\}$ but not to b and d. So we find e is adjacent to a and c and also have found our labeling.

Suppose abc is the unique triple we found in our list that begins with ab. We check for triples starting in ac or bc. Since one of the three vertices involved must be adjacent to some other vertex d, one of these two potential edges must appear in at least two triples. We can then use the previous methods to reconstruct some subgraph containing a, b and c. The result will tell us whether ab is an edge or not.

Note that we can search our list for a specific triple or a triple starting with a specific pair of vertices, in time $O(\log(|T|))$ using binary search. This means that the above checks can be done in $O(\log(|T|))$ time. By handling potential edges in the order in which they appear in the list, we only need to run through the list once, and thus we obtain a runtime of $O(|T| \log(|T|))$.

Using a data structure for "perfect-hashing" like the one described by Fredman, Komlós and Szemerédi [13], we can query the required triples in $O(1)$ time and thus reconstruct the graph in $O(|T|)$ time. Given a list S of n distinct items from a set of m items, [13] describes an algorithm to create a data-structure that can store n items, and allow for membership in S to be queried in constant time for all m and n. The construction of the data-structure is a randomized process that takes $O(n)$ time in expectation. In our case S is the list of triples and our universe is $V(G) \times V(G) \times V(G)$ so the process takes time $O(T)$. □

We also prove the following two results in the full version [3].

Theorem 3. *We can reconstruct any graph on $n \geq 6$ vertices that is known to be 2-connected and outerplanar from its list of connected triples.*

Our approach is similar to the one for trees: we show that we can identify a vertex of degree two, and remove it from the graph by 'merging' it with one of its neighbours.

A *triangulated planar graph*, also called a *maximal planar graph*, is a planar graph where every face (including the outer face) is a triangle.

Theorem 4. *Let T be a set of triples, and let it be known that the underlying graph $G = (V, E)$ is planar and triangulated. Then G can be uniquely reconstructed from T if $n \geq 7$.*

To show this result, we first show that unique reconstruction of such graphs is possible if they do not contain any separating triangles: A *separating triangle* is a triangle in the graph whose removal would result in the graph being disconnected. We then show we can reduce the problem of reconstructing a triangulated planar graph that contains a separating triangle to reconstructing triangulated planar graphs that do *not* contain a separating triangle.

6 Reconstruction from Connected k-Sets

For $k \geq 2$ and a graph $G = (V, E)$, we define the *connected k-sets* of G as the set $\{X \subseteq V \mid |X| = k \text{ and } G[X] \text{ is connected}\}$. We will denote the set of neighbours of a vertex v by $N(v)$.

Observation 1. *For $k' \geq k \geq 2$, the connected k'-sets of a graph are determined by the connected k-sets.*

Indeed, a $(k+1)$-set $X = \{x_1, \ldots, x_{k+1}\} \subseteq V$ induces a connected subgraph of G if and only if for some $y, z \in X$, both $G[X \setminus \{y\}]$ and $G[X \setminus \{z\}]$ are connected.

Given a class $\mathcal{C}$ of graphs, we can consider the function $k(n)$, where for any integer $n \geq 1$, we define $k(n)$ to be the largest integer $k \geq 2$ such that all (labelled) n-vertex graphs in $\mathcal{C}$ have a different collection of connected k-sets. By Observation 1, asking for the largest such k is a sensible question: reconstruction becomes more difficult as k increases. We will always assume that we only have to differentiate the graph from other graphs in the graph class, and remark that often the *recognition problem* (is $G \in \mathcal{C}$?) cannot be solved even from the connected triples.

First, we give an analogue of Theorem 1. The proof is given in the full version [3].

Theorem 5. *If it is known that the input graph is a tree, then the threshold for reconstructing trees is at $\lceil n/2 \rceil$: we can reconstruct an n-vertex tree from the connected k-sets if $k \leq \lceil n/2 \rceil$ and we cannot reconstruct the order of the vertices in an n-vertex path if $k \geq \lceil n/2 \rceil + 1$.*

In the full version [3], using the theorem above, we give examples showing that for every $k \geq 2$, there are infinitely many graphs that are determined by their connected k-sets but not by their connected $(k+1)$-sets.

We next show that a threshold near $n/2$ that we saw above for trees, holds for almost every n-vertex graph. The Erdős-Renyi random graph $G \sim G(n, \frac{1}{2})$ has n vertices and each edge is present with probability $\frac{1}{2}$, independently of the other edges. This yields the uniform distribution over the collection of (labelled) graphs on n vertices. If something holds for the random graph with high probability (that is, with a probability that tends to 1 as $n \to \infty$), then we say that it holds for *almost every graph*. The random graph is also interesting since it is often use to deduce the existence of extremal graphs for many problems. More information can be found in e.g. [4,14,18].

We say an n-vertex graph $G = (V, E)$ is *random-like* if the following three properties hold (with log of base 2).

1. For every vertex $v \in V$,
$$n/2 - 3\sqrt{n \log n} \leq |N(v)| \leq n/2 + 3\sqrt{n \log n}.$$
2. For every pair of distinct vertices $v, w \in V$,
$$n/4 - 3\sqrt{n \log n} \leq |N(v) \cap (V \setminus N(w))| \leq n/4 + 3\sqrt{n \log n}.$$
3. There are no disjoint subsets $A, B \subseteq V$ with $|A|, |B| \geq 2 \log n$ such that there are no edges between a vertex in A and a vertex in B.

Lemma 3. *For $G \sim G(n, \frac{1}{2})$, with high probability G is random-like.*

The claimed properties of the random graph are well-known, nonetheless we added a proof in full version for the convenience of the reader [3].

Theorem 6. *For all sufficiently large n, any n-vertex graph G that is random-like can be reconstructed from the set of connected k-sets for $2 \leq k \leq \frac{1}{2}n - 4\sqrt{n \log n}$ in time $O(n^{k+1})$. On the other hand, $G[S]$ is connected for all subsets S of size at least $\frac{1}{2}n + 4\sqrt{n \log n}$.*

In particular, for almost every graph (combining Lemma 3 and Theorem 6), the connectivity of k-tuples for $k \geq \frac{1}{2}n + 4\sqrt{n \log n}$ gives no information whatsoever, whereas for $k \leq \frac{1}{2}n - 4\sqrt{n \log n}$ it completely determines the graph.

Proof (of Theorem 6). Let K be the set of connected k-sets.

We first prove the second part of the statement. Let $k \geq \frac{1}{2}n + 4\sqrt{n \log n}$ be an integer and let S be a subset of V of size at least k. Consider two vertices $u, v \in S$. We will prove that u and v are in the same connected component of $G[S]$. By the first random-like property, there are at most $\frac{1}{2}n + 3\sqrt{n \log n}$ vertices non-adjacent to u, which implies that u has at least $\sqrt{n \log n} - 1 \geq 2 \log n$ neighbours in S. Note that, for the same reason, this is also true for v. Therefore, we can apply the third random-like property on $A = N(u) \cap S$ and $B = N(v) \cap S$ to ensure that there is an edge between the two sets. We conclude that there must exist a path from u to v.

Let us now prove the first part of the statement. Let $2 \leq k \leq \lfloor \frac{1}{2}n - 4\sqrt{n \log n} \rfloor$. Let $u \in V(G)$. We claim that the set of vertices $V \setminus N[v]$ that are not adjacent or equal to u, is the largest set S such that $G[S]$ is connected and $G[S \cup \{u\}]$ is not. Note that the two conditions directly imply that $S \subseteq V \setminus N[u]$. To prove equality, it is sufficient to prove that $G[V \setminus N[u]]$ is connected. Consider two vertices $v, w \in G[V \setminus N[u]]$. By the second random-like property, the sets $A = N(v) \cap (V \setminus N[u])$ and $B = N(w) \cap (V \setminus N[u])$ have size at least $n/4 - 3\sqrt{n \log n}$. By the third property of random-like, there is therefore an edge between A and B.

This proves that there is a path between v and w using vertices in $V \setminus N[u]$, and so $G[V \setminus N[v]]$ is connected.

For each vertex u, the set of vertices it is not adjacent to can now be found by finding the largest set S such that $G[S]$ is connected and $G[S \cup \{u\}]$ is not. Since any such S is a subset of $V \setminus N[u]$, there is a unique maximal (and unique maximum) such S. We now give the $O(n^{k+1})$ time algorithm for this.

We begin by constructing a data structure like the deterministic version described by Fredman et al. [13], which allows us to query the required k-sets in $O(1)$. This takes deterministic time of $O(|K| \log |K|) = O(n^k k \log n) = O(n^{k+1})$. For a vertex v, we give an algorithm to reconstruct the neighbourhood of v in time $O(n^{k+1})$.

1. We first run over the subsets S of size k until we find one for which $G[S]$ is connected but $G[S \cup \{v\}]$ is not. This can be done in time $O(kn^k)$: if $G[S]$ is connected, then $G[S \cup \{v\}]$ is disconnected if and only if $G[S \setminus \{s\} \cup \{v\}]$ is disconnected for all $s \in S$.
2. For each vertex $w \in V \setminus (S \cup \{v\})$, we check whether there is a subset $U \subseteq S$ of size $k-1$ for which $U \cup \{w\}$ is connected, and whether for each subset $U' \subseteq S$ of size $k-2$, $U' \cup \{w, v\}$ is not connected. If both are true for the vertex w, then $G[S \cup \{w\}]$ is connected and $G[S \cup \{w, v\}]$ is not connected, so we add w to S and repeat this step.
3. If no vertex can be added anymore, we stop and output $V \setminus (S \cup \{v\})$ as the set of neighbours of v.

We repeat step 2 at most n times, and each time we try at most n vertices as potential w and run over subsets of size at most $k-1$. Hence, this part runs in time $O(n^{k+1})$. We repeat the algorithm above n times (once per vertex) in order to reconstruct all edges. □

We prove the following analogue to Theorem 2 in the full version [3].

Theorem 7. *Let $k \geq 4$ be an integer. Every graph on at least $2k-1$ vertices that is known to have no cycles of length at most k is determined by its connected k-sets.*

7 Conclusion

We have presented a new model of uncertainty in graphs, in which we only receive all triples of vertices that form a connected induced subgraph. In a way, this is the simplest model of combinatorial indeterminacy in graphs. We have studied some basic properties of this model, and provided an algorithm for finding a graph consistent with the given indeterminacies. We also proved that trees, triangle-free graphs and various other families of graphs are determined by the connected triples, although we need to know the family the sought graph belongs to. In order to obtain a full characterisation, it is natural to put conditions on the way a triangle may connect to the rest of the graph, for instance, it is not too difficult to recognise that a, b, c induces a triangle if at least two of a, b, c have private neighbour, whereas it is impossible to distinguish whether a, b, c induce a triangle or a path if all three vertices have the same neighbours outside of the triangle. We leave this open for future work.

Similar to what has been done for graph reconstruction (see e.g. [23]), another natural direction is to loosen the objective of reconstruction, and to see if rather than determining the (labelled) graph, we can recover some graph property such as the number of edges or the diameter. A natural question is also how many connected triples are required (when given a 'subcollection', similar to [6,16,26], or when we may perform adaptive queries, as in [19]).

We gave various results in an extension of our model to larger k-sets, including trees and random graphs. There are several other logical extensions to the concept of reconstructing a graph from connected triples. We could define a (k, ℓ)-representation T to contain all k-sets that are connected and contain at least ℓ edges. The definition of connected triples would then be a $(3, 2)$-representation. Note that in this case some vertices may not appear in T, or T might even be empty altogether (e.g. for trees when $\ell \geq k$). Another natural extension would be to specify the edge count for each k-set, but this gives too much information even when $k = n-2$: the existence of any edge $\{u, v\}$ can be determined from the number of edges among vertices in the four sets $V, V \setminus \{u\}, V \setminus \{v\}$ and $V \setminus \{u, v\}$.

Some interesting algorithmic questions remain open as well. In particular, we presented an efficient algorithm to specify whether a collection of connected k-sets, for $k = 3$ uniquely determines a graph, but do not know how to solve this efficiently for larger values of k. Is the following decision problem solvable in polynomial time: given a graph G and an integer k, is G determined by its collection of k-tuples? We note that when k equals 4, membership in coNP is clear (just give another graph with the same connected 4-sets) whereas even NP-membership is unclear.

Finally, a natural question is whether the running time of $O(n \cdot |T|)$ for finding a consistent graph with a set of connected triples (Lemma 1) can be improved to $O(|T|)$ (or expected time $O(|T|)$).

References

1. Ahmed, M., Wenk, C.: Constructing street networks from GPS trajectories. In: Epstein, L., Ferragina, P. (eds.) ESA 2012. LNCS, vol. 7501, pp. 60–71. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33090-2_7
2. Aspvall, B., Plass, M.F., Tarjan, R.E.: A linear-time algorithm for testing the truth of certain quantified Boolean formulas. Inf. Process. Lett. **8**(3), 121–123 (1979)
3. Bastide, P., et al.: Reconstructing graphs from connected triples (2023)
4. Bollobás, B.: Random graphs. In: Modern Graph Theory. GTM, vol. 184, pp. 215–252. Springer, New York (1998). https://doi.org/10.1007/978-1-4612-0619-4_7
5. Bondy, J.A., Hemminger, R.L.: Graph reconstruction - a survey. J. Graph Theory **1**(3), 227–268 (1977)
6. Bowler, A., Brown, P., Fenner, T.: Families of pairs of graphs with a large number of common cards. J. Graph Theory **63**(2), 146–163 (2010)

7. Brandes, U., Cornelsen, S.: Phylogenetic graph models beyond trees. Discrete Appl. Math. **157**(10), 2361–2369 (2009)
8. Cameron, P.J., Martins, C.: A theorem on reconstruction of random graphs. Comb. Probab. Comput. **2**(1), 1–9 (1993)
9. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, pp. 197–200. MIT press (2022)
10. Dey, T.K., Wang, J., Wang, Y.: Graph reconstruction by discrete Morse theory. In: Proceedings of the 34th International Symposium on Computational Geometry. Leibniz International Proceedings in Informatics (LIPIcs), vol. 99, pp. 31:1–31:15 (2018)
11. Even, S., Itai, A., Shamir, A.: On the complexity of timetable and multicommodity flow problems. SIAM J. Comput. **5**(4), 691–703 (1976)
12. Feder, T.: Network flow and 2-satisfiability. Algorithmica **11**(3), 291–319 (1994)
13. Fredman, M.L., Komlós, J., Szemerédi, E.: Storing a sparse table with 0(1) worst case access time. J. ACM **31**(3), 538–544 (1984)
14. Frieze, A., Karonski, M.: Introduction to Random Graphs. Cambridge University Press, New York (2015)
15. Giles, W.B.: The reconstruction of outerplanar graphs. J. Comb. Theory Ser. B **16**(3), 215–226 (1974)
16. Groenland, C., Guggiari, H., Scott, A.: Size reconstructibility of graphs. J. Graph Theory **96**(2), 326–337 (2021)
17. Harary, F.: A survey of the reconstruction conjecture. In: Bari, R.A., Harary, F. (eds.) Graphs and Combinatorics. LNCS, vol. 406, pp. 18–28. Springer, Berlin (1974). https://doi.org/10.1007/BFb0066431
18. Janson, S., Rucinski, A., Luczak, T.: Random Graphs. John Wiley & Sons, Hoboken (2011)
19. Kannan, S., Mathieu, C., Zhou, H.: Graph reconstruction and verification. ACM Trans. Algorithms **14**(4), 1–30 (2018)
20. Kassiano, V., Gounaris, A., Papadopoulos, A.N., Tsichlas, K.: Mining uncertain graphs: an overview. In: Sellis, T., Oikonomou, K. (eds.) ALGOCLOUD 2016. LNCS, vol. 10230, pp. 87–116. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57045-7_6
21. Kelly, P.J.: On Isometric Transformations. Ph.D. thesis, University of Wisconsin (1942)
22. Lauri, J.: The reconstruction of maximal planar graphs. J. Combi. Theory Ser. B **30**(2), 196–214 (1981)
23. Lauri, J., Scapellato, R.: Topics in Graph Automorphisms and Reconstruction. Cambridge University Press, Cambridge (2016)
24. Mordeson, J.N., Peng, C.S.: Operations on fuzzy graphs. Inf. Sci. **79**(3), 159–170 (1994)
25. Mossel, E., Ross, N.: Shotgun assembly of labeled graphs. IEEE Trans. Netw. Sci. Eng. **6**(2), 145–157 (2017)
26. Myrvold, W.: The degree sequence is reconstructible from $n-1$ cards. Discrete Math. **102**(2), 187–196 (1992)
27. Rosenfeld, A.: Fuzzy graphs. In: Zadeh, L.A., Fu, K.S., Tanaka, K., Shimura, M. (eds.) Fuzzy Sets and their Applications to Cognitive and Decision Processes, pp. 77–95. Elsevier (1975)
28. Tutte, W.: All the king's horses. A guide to reconstruction. Graph Theory Relat. Top., 15–33 (1979)
29. Ulam, S.M.: A Collection of Mathematical Problems, Interscience Tracts in Pure and Applied Mathematics, vol. 8. Interscience Publishers (1960)

Parameterized Complexity of Vertex Splitting to Pathwidth at Most 1

Jakob Baumann, Matthias Pfretzschner(✉), and Ignaz Rutter

Universität Passau, 94032 Passau, Germany
{baumannjak,pfretzschner,rutter}@fim.uni-passau.de

Abstract. Motivated by the planarization of 2-layered straight-line drawings, we consider the problem of modifying a graph such that the resulting graph has pathwidth at most 1. The problem Pathwidth-One Vertex Explosion (POVE) asks whether such a graph can be obtained using at most k vertex explosions, where a *vertex explosion* replaces a vertex v by $\deg(v)$ degree-1 vertices, each incident to exactly one edge that was originally incident to v. For POVE, we give an FPT algorithm with running time $O(4^k \cdot m)$ and an $O(k^2)$ kernel, thereby improving over the $O(k^6)$-kernel by Ahmed et al. [2] in a more general setting. Similarly, a *vertex split* replaces a vertex v by two distinct vertices v_1 and v_2 and distributes the edges originally incident to v arbitrarily to v_1 and v_2. Analogously to POVE, we define the problem variant Pathwidth-One Vertex Splitting (POVS) that uses the split operation instead of vertex explosions. Here we obtain a linear kernel and an algorithm with running time $O((6k+12)^k \cdot m)$. This answers an open question by Ahmed et al. [2].

Keywords: Vertex Splitting · Vertex Explosion · Pathwidth 1

1 Introduction

Crossings are one of the main aspects that negatively affect the readability of drawings [20]. It is therefore natural to try and modify a given graph in such a way that it can be drawn without crossings while preserving as much of the information as possible. We consider three different operations.

A *deletion operation* simply removes a vertex from the graph. A *vertex explosion* replaces a vertex v by $\deg(v)$ degree-1 vertices, each incident to exactly one edge that was originally incident to v. Finally, a *vertex split* replaces a vertex v by two distinct vertices v_1 and v_2 and distributes the edges originally incident to v arbitrarily to v_1 and v_2.

Nöllenburg et al. [18] have recently studied the vertex splitting problem, which is known to be NP-complete [11]. In particular, they gave a non-uniform FPT-algorithm for deciding whether a given graph can be planarized with at most k splits. We observe that, since degree-1 vertices can always be inserted into a planar drawing, the vertex explosion model and the vertex deletion model are

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 30–43, 2023.
https://doi.org/10.1007/978-3-031-43380-1_3

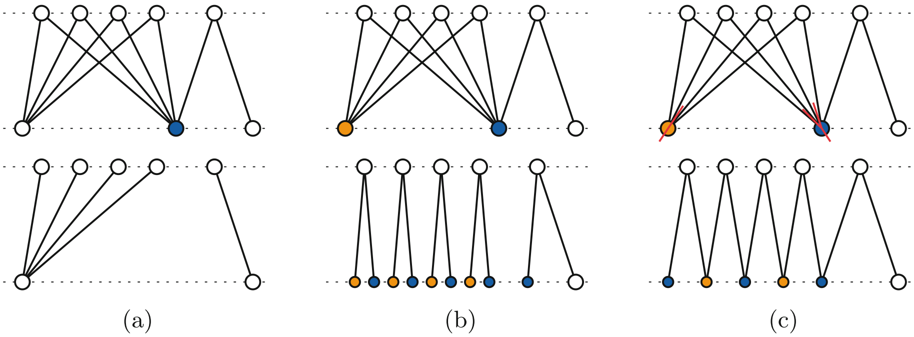


Fig. 1. Given the shown bipartite graph, a crossing-free 2-layered drawing can be obtained using one vertex deletion (a), two vertex explosions (b), or three vertex splits (c).

equivalent for obtaining planar graphs. Note that this is not necessarily the case for other target graph classes (see, for example, Fig. 1). The problem of deleting vertices to obtain a planar graph is also known as Vertex Planarization and has been studied extensively in the literature [13,15–17]. In particular, Jansen et al. [13] gave an FPT-algorithm with running time $O(2^{O(k \log k)} \cdot n)$.

Ahmed et al. [2] investigated the problem of splitting the vertices of a bipartite graph so that it admits a 2-layered drawing without crossings. They assume that the input graph is bipartite and only the vertices of one of the two sets in the bipartition may be split. Under this condition, they give an $O(k^6)$-kernel for the vertex explosion model, which results in an $O(2^{O(k^6)}m)$-time algorithm. They ask whether similar results can be obtained in the vertex splitting model. Figure 1 illustrates the three operations in the context of 2-layered drawings[1].

We note that a graph admits a 2-layer drawing without crossings if and only if it has pathwidth at most 1, i.e., it is a disjoint union of caterpillars [3,9]. Motivated by this, we more generally consider the problem of turning a graph $G = (V, E)$ into a graph of pathwidth at most 1 by the above operations. In order to model the restriction of Ahmed et al. [2] that only one side of their bipartite input graph may be split, we further assume that we are given a subset $S \subseteq V$, to which we may apply modification operations as part of the input. We define that the new vertices resulting from an operation are also included in S.

More formally, we consider the following problems, all of which have been shown to be NP-hard [1,19].

[1] In this context, minimizing the number of vertex explosions is equivalent to minimizing the number of vertices that are split, since it is always best to split a vertex as often as possible.

Pathwidth-One Vertex Explosion (POVE)
Input: An undirected graph $G = (V, E)$, a set $S \subseteq V$, and a positive integer k.
Question: Is there a set $W \subseteq S$ with $|W| \leq k$ such that the graph resulting from exploding all vertices in W has pathwidth at most 1?

Pathwidth-One Vertex Splitting (POVS)
Input: An undirected graph $G = (V, E)$, a set $S \subseteq V$, and a positive integer k.
Question: Is there a sequence of at most k splits on vertices in S such that the resulting graph has pathwidth at most 1?

We note that the analogous problem with the deletion operation has been studied extensively [8,19,23]. Here, a branching algorithm with running time $O(3.888^k \cdot n^{O(1)})$ [23] and a quadratic kernel [8] are known. Our results are as follows.

First, in Sect. 3, we show that POVE admits a kernel of size $O(k^2)$ and an algorithm with running time $O(4^k m)$, thereby improving over the results of Ahmed et al. [2] in a more general setting.

Second, in Sect. 4, we show that POVS has a kernel of size $16k$ and it admits an algorithm with running time $O((6k+12)^k \cdot m)$. This answers the open question of Ahmed et al. [2].

Finally, in Sect. 5, we consider the problem Π Vertex Splitting(Π-VS), the generalized version of the splitting problem where the goal is to obtain a graph of a specific graph class Π using at most k split operations. Eppstein et al. [10] recently studied the similar problem of deciding whether a given graph G is *k-splittable*, i.e., whether it can be turned into a graph of Π by splitting every vertex of G at most k times. For graph classes Π that can be expressed in monadic second-order graph logic (MSO$_2$, see [7]), they gave an FPT algorithm parameterized by the solution size k and the treewidth of the input graph. We adapt their algorithm for the problem Π-VS, resulting in an FPT algorithm parameterized by the solution size k for MSO$_2$-definable graph classes Π of bounded treewidth. Using a similar algorithm, we obtain the same result for the problem variant using vertex explosions.

2 Preliminaries

A parameterized problem L with parameter k is *non-uniformly fixed-parameter tractable* if, for every value of k, there exists an algorithm that decides L in time $f(k) \cdot n^{O(1)}$ for some computable function f. If there is a single algorithm that satisfies this property for all values of k, then L is *(uniformly) fixed-parameter tractable*.

Given a graph G, we let n and m denote the number of vertices and edges of G, respectively. Since we can determine the subgraph of G that contains no isolated vertices in $O(m)$ time, we assume, without loss of generality, that $n \in O(m)$. For a vertex $v \in V(G)$, we let $N(v) := \{u \in V(G) \mid \text{adj}(v, u)\}$ and $N[v] := N(v) \cup \{v\}$ denote the open and closed neighborhood of v in G, respectively.

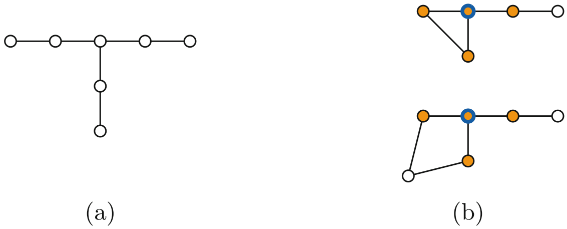


Fig. 2. (a) The graph T_2. (b) Two graphs that do not contain T_2 as a subgraph, but both contain N_2 (marked in orange) as a substructure. (Color figure online)

We refer to vertices of degree 1 as *pendant* vertices. For a vertex v of G, we let $\deg^*(v) := |\{u \in N(v) \mid \deg(u) > 1\}|$ denote the degree of v ignoring its pendant neighbors. If $\deg^*(v) = d$, we refer to v as a vertex of *degree* d*. A graph is a *caterpillar* (respectively a *pseudo-caterpillar*), if it consists of a simple path (a simple cycle) with an arbitrary number of adjacent pendant vertices. The path (the cycle) is called the *spine* of the (pseudo-)caterpillar.

Philip et al. [19] mainly characterized the graphs of pathwidth at most 1 as the graphs containing no cycles and no T_2 (three simple paths of length 2 that all share an endpoint; see Fig. 2a) as a subgraph. We additionally use slightly different sets of forbidden substructures. An N_2 *substructure* consists of a *root* vertex r adjacent to three distinct vertices of degree at least 2. Note that every T_2 contains an N_2 substructure, however, the existence of an N_2 substructure does not generally imply the existence of a T_2 subgraph; see Fig. 2b. In the following proposition, we state the different characterizations for graphs of pathwidth at most 1 that we use in this work.

Proposition 1 ($\star$[2]). *For a graph G, the following statements are equivalent.*

a) G has pathwidth at most 1
b) every connected component of G is a caterpillar
c) G is acyclic and contains no T_2 subgraph
d) G is acyclic and contains no N_2 substructure
e) G contains no N_2 substructure and no connected component that is a pseudo-caterpillar.

We define the *potential* of $v \in V(G)$ as $\mu(v) := \max(\deg^*(v) - 2, 0)$. The *global potential* $\mu(G) := \sum_{v \in V(G)} \mu(v)$ is defined as the sum of the potentials of all vertices in G. Observe that $\mu(G) = 0$ if and only if G contains no N_2 substructure. The global potential thus indicates how far away we are from eliminating all N_2 substructures from the instance.

Recall that, for the problems POVE and POVS, the set $S \subseteq V(G)$ marks the vertices of G that may be chosen for the respective operations. We say that a set $W \subseteq S$ is a *pathwidth-one explosion set* (POES) of G, if the graph resulting from exploding all vertices in W has pathwidth at most 1.

[2] The proofs of results marked with a star can be found in the full version [4].

3 FPT Algorithms for Pathwidth-One Vertex Explosion

In this section, we first show that POVE can be solved in time $O(4^k \cdot m)$ using bounded search trees. Subsequently, we develop a kernelization algorithm for POVE that yields a quadratic kernel in linear time.

3.1 Branching Algorithm

We start by giving a simple branching algorithm for POVE, similar to the algorithm by Philip et al. [19] for the deletion variant of the problem. For an N_2 substructure X, observe that exploding vertices not contained in X cannot eliminate X, because the degrees of the vertices in X remain the same due to the new degree-1 vertices resulting from the explosion. To obtain a graph of pathwidth at most 1, it is therefore always necessary to explode one of the four vertices of every N_2 substructure by Proposition 1. Our branching rule thus first picks an arbitrary N_2 substructure from the instance and then branches on which of the four vertices of the N_2 substructure belongs to the POES. Recall that S denotes the set of vertices of the input graph that can be exploded.

Branching Rule 1. *Let r be the root of an N_2 substructure contained in G and let x, y, and z denote the three neighbors of r in N_2. For every vertex $v \in \{r, x, y, z\} \cap S$, create a branch for the instance $(G', S \setminus \{v\}, k-1)$, where G' is obtained from G by exploding v.*
If $\{r, x, y, z\} \cap S = \emptyset$, reduce to a trivial no-instance instead.

Note that an N_2 substructure can be found in $O(m)$ time by checking, for every vertex v in G, whether v has at least three neighbors of degree at least 2. Also note that vertex explosions do not increase the number of edges of the graph. Since Branching Rule 1 creates at most four new branches, each of which reduces the parameter k by 1, exhaustively applying the rule takes $O(4^k \cdot m)$ time. By Proposition 1, it subsequently only remains to eliminate connected components that are a pseudo-caterpillar. Since a pseudo-caterpillar can (only) be turned into a caterpillar by exploding a vertex of its spine, the remaining instance can be solved in linear time.

Theorem 1. *The problem* Pathwidth-One Vertex Explosion *can be solved in time $O(4^k \cdot m)$.*

3.2 Quadratic Kernel

We now turn to our kernelization algorithm for POVE. In this section, we develop a kernel of quadratic size, which can be computed in linear time.

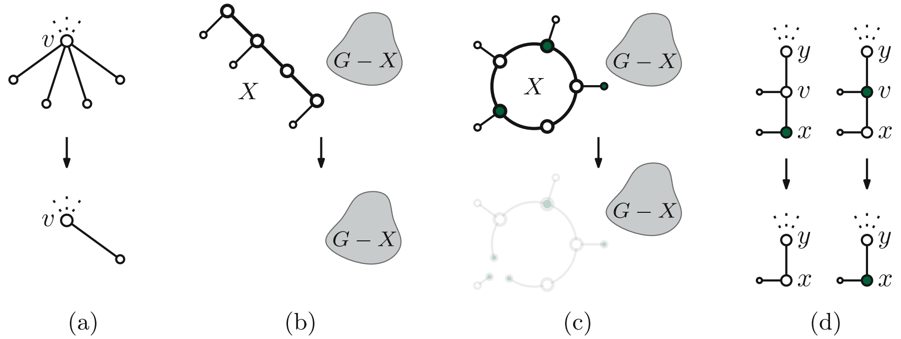


Fig. 3. Examples for Reduction Rules 1 (a), 2 (b), 3 (c), and 4 (d). The vertices of S are marked in green (Color figure online).

We adopt our first two reduction rules from the kernelization of the deletion variant by Philip et al. [19] and show that these rules are also safe for the explosion variant. The first rule reduces the number of pendant neighbors of each vertex to at most one; see Fig. 3a.

Reduction Rule 1. (⋆). *If G contains a vertex v with at least two pendant neighbors, remove all pendant neighbors of v except one to obtain the graph G' and reduce the instance to $(G',\ S \cap V(G'),\ k)$.*

Since a caterpillar has pathwidth at most 1 by Proposition 1, we can safely remove any connected component of G that forms a caterpillar; see Fig. 3b for an example.

Reduction Rule 2. *If G contains a connected component X that is a caterpillar, remove X from G and reduce the instance to $(G - X,\ S \setminus V(X),\ k)$.*

If G contains a connected component that is a pseudo-caterpillar, then exploding an arbitrary vertex of its spine yields a caterpillar. If the spine contains no vertex of S, the spine is a cycle that cannot be broken by a vertex explosion. However, by Proposition 1, acyclicity is a necessary condition for a graph of pathwidth at most 1. Hence we get the following reduction rule; see Fig. 3c for an illustration.

Reduction Rule 3. *Let X denote a connected component of G that is a pseudo-caterpillar. If the spine of X contains a vertex of S, remove X from G and reduce the instance to $(G - X,\ S \setminus V(X),\ k - 1)$. Otherwise reduce to a trivial no-instance.*

Recall that the degree* of a vertex is the number of its non-pendant neighbors. Our next goal is to shorten paths of degree*-2 vertices to at most two vertices. If we have a path x, y, z of degree*-2 vertices, we refer to y as a *2-enclosed* vertex. Note that exploding a 2-enclosed vertex y cannot eliminate any

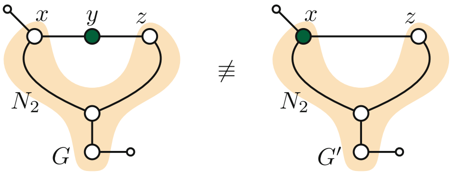


Fig. 4. A graph G that has no POES, because the highlighted N_2 substructure contains no vertex of S. For the graph G' resulting from contracting y into x, the set $\{x\}$ is a POES. The two instances are therefore not equivalent.

N_2 substructures from the instance. By Proposition 1, vertex y can thus only be part of an optimal solution if exploding y breaks cycles. If we want to shorten the chain x, y, z by contracting y into one of its neighbors, we therefore need to ensure that the shortened chain contains a vertex of S if and only if the original chain contained a vertex of S. If $y \in S$, we cannot simply add one of its neighbors, say x, to S in the reduced instance, because exploding x may additionally remove an N_2 substructure; see Fig. 4 for an example. While shortening paths of degree*-2 vertices to at most three vertices is simple, shortening them to length at most 2 (i.e., eliminating all 2-enclosed vertices) is therefore more involved. In the following, we briefly sketch how this can be achieved in linear time. For the specific reduction rules and the corresponding correctness proofs, we refer to the full version of the paper [4].

Lemma 1 ($\star$). *Given an instance of* POVE*, an equivalent instance without 2-enclosed vertices can be computed in $O(m)$ time.*

Sketch of Proof. Given a 2-enclosed vertex y, we show that we can decide greedily whether y is contained in an optimal solution or not. This means that we can either immediately explode y, or we can safely contract it into one of its degree*-2 neighbors. Since y is 2-enclosed, y is not contained in any N_2 substructures and we thus only have to consider cycles containing y. If there exists a cycle C in G with $C \cap S = \{y\}$ (i.e., y is the only splittable vertex of C), then we can immediately explode y. Otherwise, every cycle containing y contains at least one additional vertex of S. In this case, we can show that there exists a minimum POES of G that does not contain y, thus we can remove y from S and contract it into one of its neighbors, thereby preserving all cycles of the instance. To achieve linear running time, we can show that the set of 2-enclosed vertices that should be exploded can be computed globally using a specialized spanning tree. □

To simplify the instance even further, the following reduction rule removes all degree*-2 vertices v that are adjacent to a vertex x of degree* 1; see Fig. 3d for an illustration. Roughly speaking, since v cannot be contained in a cycle and x substitutes v in all N_2 substructures v is contained in, all forbidden substructures are preserved.

Reduction Rule 4 ($\star$). *Let v be a degree*-2 vertex of G with non-pendant neighbors x and y, such that x has degree* 1. Remove v from G and add a new edge xy. If $v \in S$, reduce to $(G - v + xy, (S \setminus \{v\}) \cup \{x\}, k)$. Otherwise reduce to $(G - v + xy, S \setminus \{x\}, k)$.*

Recall that the global potential $\mu(G)$ indicates how far away we are from our goal of eliminating all N_2 substructures from G. With the following lemma, we show that our reduction rules ensure that the number of vertices in the graph G is bounded linearly in the global potential of G.

Lemma 2. *After exhaustively applying Reduction Rules 1–4 and Lemma 1, it holds that $|V(G)| \leq 8 \cdot \mu(G)$.*

Proof. Reduction Rule 2 ensures that G contains no vertices of degree* 0. For $i \in \{1, 2\}$, let V_i denote the set of non-pendant degree*-i vertices of G and let V_3 denote the set of vertices with degree* at least 3. Recall that we defined the global potential as

$$\mu(G) = \sum_{v \in V(G)} \mu(v) = \sum_{v \in V(G)} \max(0, \deg^*(v) - 2).$$

Since all vertices of V_1 and V_2 have degree* at most 2, their potential is 0 and we get

$$\mu(G) = \sum_{v \in V_3} (\deg^*(v) - 2) = \sum_{v \in V_3} \deg^*(v) - 2 \cdot |V_3|.$$

Note that $|V_3| \leq \mu(G)$, because each vertex of degree* at least 3 contributes at least 1 to the global potential. We therefore get

$$\sum_{v \in V_3} \deg^*(v) \leq 3 \cdot \mu(G). \tag{1}$$

By Lemma 1, every vertex in $v \in V_2$ is adjacent to a vertex of $V_1 \cup V_3$, since otherwise, v would be 2-enclosed. However, Reduction Rule 4 additionally ensures that vertices of V_2 cannot be adjacent to vertices of V_1, thus every vertex of V_2 must be adjacent to a vertex of V_3. Note that two adjacent vertices of V_1 would form a caterpillar, which is prohibited by Reduction Rule 2. Therefore, every vertex of V_1 is also adjacent to a vertex of V_3.

Overall, every vertex of V_1 and V_2 is thus adjacent to a vertex of V_3. Note that every vertex $v \in V_1$ must additionally have a pendant neighbor, because otherwise, v itself would be a pendant vertex. Hence every vertex of V_1 and V_2 has degree at least 2 and thus contributes to the degree* of its neighbor in V_3. We therefore have $|V_1| + |V_2| \leq \sum_{v \in V_3} \deg^*(v)$, hence $|V_1| + |V_2| \leq 3 \cdot \mu(G)$ by Eq. 1. Recall that $|V_3| \leq \mu(G)$, thus $|V_1| + |V_2| + |V_3| \leq 4 \cdot \mu(G)$. By Reduction Rule 1, each of these vertices can have at most one pendant neighbor and thus $|V(G)| \leq 8 \cdot \mu(G)$.

With Lemma 2, it now only remains to find an upper bound for the global potential $\mu(G)$. We do this using the following two reduction rules.

Reduction Rule 5. *Let v be a vertex of G with potential $\mu(v) > k$. If $v \in S$, explode v to obtain the graph G' and reduce the instance to $(G',\ S \setminus \{v\},\ k-1)$. Otherwise reduce to a trivial no-instance.*

Proof of Safeness. Since exploding a vertex $u \in V(G) \setminus \{v\}$ decreases $\mu(v)$ by at most one, after exploding at most k vertices in $V(G) \setminus \{v\}$ we still have $\mu(v) > 0$. Because $\mu(v) > 0$ implies that G contains an N_2 substructure, it is therefore always necessary to explode vertex v by Proposition 1. □

Reduction Rule 6. *If $\mu(G) > 2k^2 + 2k$, reduce to a trivial no-instance.*

Proof of Safeness. By Reduction Rule 5 we have $\mu(v) \leq k$ and consequently $\deg^*(v) \leq k + 2$ for all $v \in V(G)$. Hence exploding a vertex v decreases the potential of v by at most k and the potential of each of its non-pendant neighbors by at most 1. Overall, k vertex explosions can therefore only decrease the global potential $\mu(G)$ by at most $k \cdot (2k + 2)$. □

Because Reduction Rule 6 gives us an upper bound for the global potential $\mu(G)$, we can now use Lemma 2 to obtain the kernel.

Theorem 2 (⋆). *The problem* Pathwidth-One Vertex Explosion *admits a kernel of size* $16k^2 + 16k$. *It can be computed in time* $O(m)$.

4 FPT Algorithms for Pathwidth-One Vertex Splitting

In this section, we briefly outline how the results from Sect. 3 can be adapted for the split operation. For detailed proofs, we refer to the full version [4].

4.1 Linear Kernel

One can prove that Reduction Rules 1–4 and Lemma 1 we used for POVE are also safe for the problem POVS. Since only these are needed to establish the upper bound of $|V(G)| \leq 8 \cdot \mu(G)$ in Lemma 2, the lemma also applies for POVS.

The main difference to the kernelization of POVE lies in the way the global potential changes due to splits. While a vertex explosion can decrease the global potential linearly in k, we can show that a single vertex split decreases $\mu(G)$ by at most 2. If $\mu(G) > 2k$, we can thus again reduce to a trivial no-instance. Using Lemma 2 with $\mu(G) \leq 2k$, we obtain the following result.

Theorem 3 (⋆). *The problem* Pathwidth-One Vertex Splitting *admits a kernel of size* $16k$. *It can be computed in time* $O(m)$.

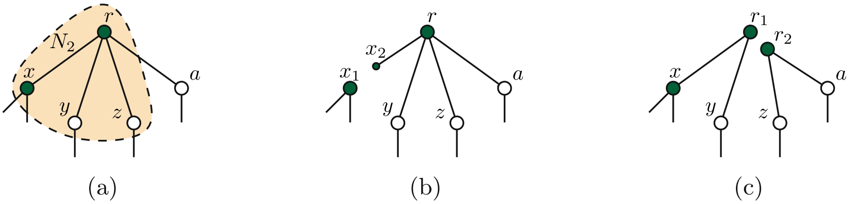


Fig. 5. (a) An N_2 substructure $\{r, x, y, z\}$. (b)-(c) Two possible branches eliminating the N_2 substructure. The former splits off edge rx at x, the latter splits off the edges rz and ra at r.

4.2 Branching Algorithm

As in Sect. 3.1, our branching algorithm for POVS eliminates every N_2 substructure of G by branching on which of its four vertices should be split. In this case, however, we need to additionally consider the possible ways to split a single vertex. The following lemma helps us limit the number of suitable splits.

Lemma 3 (⋆). *For every instance of* POVS*, there exists a minimum sequence of splits such that every split operation splits off at most two edges.*

Theorem 4 (⋆). *The problem* POVS *can be solved in time* $O((6k + 12)^k \cdot m)$.

Sketch of Proof. From the kernelization, we use Reduction Rule 1 reducing pendant vertices, and the above rule that yields the bound $\mu(G) \leq 2k$. Together, these two rules ensure that each vertex has degree at most $2k+3$. We now branch on the way of splitting an N_2 substructure with root r and neighbors $\{x, y, z\}$ as above (see Fig. 5). If we split r, then, by Lemma 3, we may assume that we split off one of the neighbors $\{x, y, z\}$, together with at most one other neighbor of r; these are $3 \cdot (2k+3)$ choices. If we split a vertex $v \in \{x, y, z\}$, then it is necessary that we only split off the edge rv at v, thus there is only one possibility for each of them. Overall, we thus find a branching vector of size $6k + 12$. □

5 FPT Algorithms for Splitting and Exploding to MSO$_2$-Definable Graph Classes of Bounded Treewidth

While Sect. 4 focused on the problem of obtaining graphs of pathwidth at most 1 using at most k vertex splits on the input graph, we now consider the problem of splitting vertices to obtain other graph classes. With the following problem, we generalize the problem POVS.

Π Vertex Splitting(*Π*-VS)
Input: An undirected graph $G = (V, E)$, a set $S \subseteq V$, and a positive integer k.
Question: Is there a sequence of at most k splits on vertices in S such that the resulting graph is contained in Π?

Nöllenburg et al. [18] showed that, for any minor-closed graph class Π, the graph class Π_k containing all graphs that can be modified to a graph in Π using at most k vertex splits is also minor-closed. Robertson and Seymour [21] showed that every minor-closed graph class has a constant-size set of forbidden minors and that it can be tested in cubic time whether a graph contains a given fixed graph as a minor. Since Π_k is minor-closed, this implies the existence of a non-uniform FPT-algorithm for the problem Π-VS. Because the graphs of pathwidth at most 1 form a minor-closed graph class, this includes the problem POVS.

Proposition 2 ([18]). *For every minor-closed graph class Π, the problem Π-*VS *is non-uniformly FPT parameterized by the solution size k.*

We say that a graph class Π is *MSO$_2$-definable*, if there exists an MSO$_2$ (monadic second-order graph logic, see [7]) formula φ such that $G \models \varphi$ if and only if $G \in \Pi$. In the following, we show that the problem Π-VS is uniformly FPT parameterized by k if Π is MSO$_2$-definable and has bounded treewidth. Since every minor-closed graph class is MSO$_2$-definable, this improves the result from Proposition 2 for graph classes of bounded treewidth.

Eppstein et al. [10] showed that the problem of deciding whether a given graph G can be turned into a graph of class Π by splitting each vertex of G at most k times can be expressed as an MSO$_2$ formula on G, if Π itself is MSO$_2$-definable. Using Courcelle's Theorem [6], this yields an FPT-algorithm parameterized by k and the treewidth of the input graph. Their algorithm exploits the fact that the split operations create at most k copies of each vertex in the graph. Since the same also applies for the problem Π-VS, where we may apply at most k splits overall, their algorithm can be straightforwardly adapted for Π-VS, thereby implying the following result.

Corollary 1. *For every MSO$_2$-definable graph class Π, the problem Π-*VS *is FPT parameterized by the solution size k and the treewidth of the input graph.*

For a graph class Π of bounded treewidth, we let $\mathrm{tw}(\Pi)$ denote the maximum treewidth among all graphs in Π. With the following lemma, we show that, if the target graph class Π has bounded treewidth, then every yes-instance of Π-VS must also have bounded treewidth.

Proposition 3. *For a graph class Π of bounded treewidth, let $\mathcal{I} = (G, S, k)$ be an instance of Π-*VS*. If $\mathrm{tw}(G) > k + \mathrm{tw}(\Pi)$, then $\mathcal{I}$ is a no-instance.*

Proof. We first show that a single split operation can reduce the treewidth of G by at most 1. Assume, for the sake of contradiction, that we can obtain a graph G' of treewidth less than $\mathrm{tw}(G)-1$ by splitting a single vertex v of G into vertices v_1 and v_2 of G'. Let $\mathcal{T}$ denote a minimum tree decomposition of G'. Remove all occurences of v_1 and v_2 in $\mathcal{T}$ and add v to every bag of $\mathcal{T}$. Observe that the result is a tree decomposition of size less than $\mathrm{tw}(G)$ for G, a contradiction. A single split operation thus decreases the treewidth of the graph by at most 1. Since every graph $G' \in \Pi$ has $\mathrm{tw}(G') \leq \mathrm{tw}(\Pi)$, it is thus impossible to obtain a graph of Π with at most k vertex splits if $\mathrm{tw}(G) > k + \mathrm{tw}(\Pi)$. □

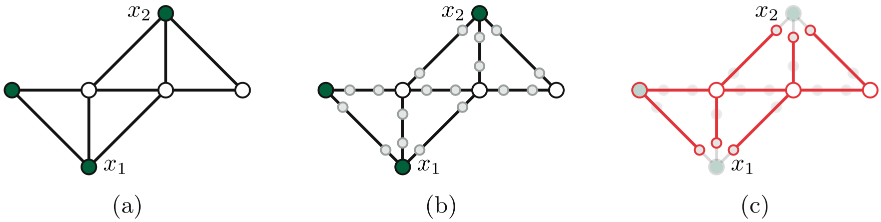


Fig. 6. (a) An instance $(G, S, 2)$ of Π-VE. (b) The corresponding auxiliary graph $G^\times$ obtained by subdividing each edge in G twice. (c) The graph obtained by exploding $\{x_1, x_2\}$ in G is the highlighted minor of $G^\times$. Since Π is MSO$_2$-definable, one can express Π-VE using an MSO$_2$ formula on $G^\times$.

Given a graph class Π of bounded treewidth, we first determine in time $f(k + \text{tw}(\Pi)) \cdot n$ whether the treewidth of G is greater than $k + \text{tw}(\Pi)$ [5]. If this is the case, then we can immediately report a no-instance by Proposition 3. Otherwise, we know that $\text{tw}(G) \leq k + \text{tw}(\Pi)$. Since $\text{tw}(\Pi)$ is a constant, we have $\text{tw}(G) \in O(k)$, and thus Corollary 1 yields the following result.

Theorem 5. *For every MSO$_2$-definable graph class Π of bounded treewidth, the problem Π-*VS *is FPT parameterized by the solution size k.*

Vertex Explosion. We now briefly sketch how these results extend to the problem variant Π Vertex Explosion(Π-VE) using vertex explosions instead of vertex splits. In this case, for minor-closed graph classes Π, the set of yes-instances of Π-VE is not minor-closed in general, thus the non-uniform FPT algorithm used to obtain Proposition 2 does not work for Π-VE. Additionally, the FPT-algorithm by Eppstein et al. [10] for MSO$_2$-definable graph classes cannot be straightforwardly adapted for Π-VE, since the number of new vertices resulting from explosions is not bounded by a function in k. However, using the approach illustrated in Fig. 6, we obtain the following results.

Lemma 4 ($\star$). *For every MSO$_2$-definable graph class Π, the problem Π-*VE *is FPT parameterized by the treewidth of the input graph.*

Theorem 6 ($\star$). *For every MSO$_2$-definable graph class Π of bounded treewidth, the problem Π-*VE *is FPT parameterized by the solution size k.*

We remark that, for arbitrary graph classes Π, the question whether a graph of Π can be obtained by applying arbitrarily many vertex splits to at most k vertices in the input graph is not equivalent to Π-VE.

6 Conclusion

In this work, we studied the problems Pathwidth-One Vertex Explosion and Pathwidth-One Vertex Splitting, obtaining an efficient branching algorithm and a small kernel for each variant. Subsequently, we more generally considered the problem of obtaining a graph of a specific graph class Π using at most k vertex splits (respectively explosions). For MSO_2-definable graph classes Π of bounded treewidth, we obtained an FPT algorithm parameterized by the solution size k. These graph classes include, for example, the outerplanar graphs, the pseudoforests, and the graphs of treewidth (respectively pathwidth) at most c for some constant c.

Instead of splitting vertices to obtain a graph of pathwidth at most 1, one can also consider obtaining graphs of treewidth at most 1, i.e., forests. Since, in this context, the degree-1 vertices resulting from an explosion can simply be reduced, the explosion model is equivalent to the problem Feedback Vertex Set, a well-studied NP-complete [14] problem that admits a quadratic kernel [22]. In the full version of this paper [4], we show that the problem of splitting vertices of a graph to obtain a forest is equivalent to the problem Feedback Edge Set, which asks whether a given graph can be made acyclic using at most k edge deletions; a problem that can be solved by computing an arbitrary spanning forest of the graph. Firbas [12] independently obtained the same result.

References

1. Ahmed, R., et al.: Splitting vertices in 2-layer graph drawings. IEEE Comput. Graph. Appl. **43**(3), 24–35 (2023). https://doi.org/10.1109/MCG.2023.3264244
2. Ahmed, R., Kobourov, S.G., Kryven, M.: An FPT algorithm for bipartite vertex splitting. In: Angelini, P., von Hanxleden, R. (eds.) Graph Drawing and Network Visualization - 30th International Symposium, GD 2022. LNCS, vol. 13764, pp. 261–268. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-22203-0_19
3. Arnborg, S., Proskurowski, A., Seese, D.: Monadic second order logic, tree automata and forbidden minors. In: Börger, E., Kleine Büning, H., Richter, M.M., Schönfeld, W. (eds.) CSL 1990. LNCS, vol. 533, pp. 1–16. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-54487-9_49
4. Baumann, J., Pfretzschner, M., Rutter, I.: Parameterized complexity of vertex splitting to pathwidth at most 1. CoRR abs/2302.14725 (2023). https://doi.org/10.48550/arXiv.2302.14725
5. Bodlaender, H.L.: A linear time algorithm for finding tree-decompositions of small treewidth. In: Kosaraju, S.R., Johnson, D.S., Aggarwal, A. (eds.) Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, pp. 226–234. ACM (1993). https://doi.org/10.1145/167088.167161
6. Courcelle, B.: The monadic second-order logic of graphs. i. recognizable sets of finite graphs. Inf. Comput. **85**(1), 12–75 (1990). https://doi.org/10.1016/0890-5401(90)90043-H
7. Cygan, M.: Parameterized Algorithms. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3
8. Cygan, M., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J.O.: An improved FPT algorithm and a quadratic kernel for pathwidth one vertex deletion. Algorithmica **64**(1), 170–188 (2012). https://doi.org/10.1007/s00453-011-9578-2
9. Eades, P., McKay, B.D., Wormald, N.C.: On an edge crossing problem. In: Proceedings of the 9th Australian Computer Science Conference, vol. 327, p. 334 (1986)

10. Eppstein, D., et al.: On the planar split thickness of graphs. Algorithmica **80**(3), 977–994 (2017). https://doi.org/10.1007/s00453-017-0328-y
11. Faria, L., de Figueiredo, C.M.H., Mendonça, C.F.X.: Splitting number is NP-complete. In: Hromkovič, J., Sýkora, O. (eds.) WG 1998. LNCS, vol. 1517, pp. 285–297. Springer, Heidelberg (1998). https://doi.org/10.1007/10692760_23
12. Firbas, A.: Establishing Hereditary Graph Properties via Vertex Splitting. Diploma thesis, Technische Universität Wien (2023). https://doi.org/10.34726/hss.2023.103864
13. Jansen, B.M.P., Lokshtanov, D., Saurabh, S.: A near-optimal planarization algorithm. In: Chekuri, C. (ed.) Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1802–1811. SIAM (2014). https://doi.org/10.1137/1.9781611973402.130
14. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Proceedings of a Symposium on the Complexity of Computer Computations, pp. 85–103. The IBM Research Symposia Series, Plenum Press, New York (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
15. Kawarabayashi, K.: Planarity allowing few error vertices in linear time. In: 50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, pp. 639–648. IEEE Computer Society (2009). https://doi.org/10.1109/FOCS.2009.45
16. Lewis, J.M., Yannakakis, M.: The node-deletion problem for hereditary properties is NP-complete. J. Comput. Syst. Sci. **20**(2), 219–230 (1980). https://doi.org/10.1016/0022-0000(80)90060-4
17. Marx, D., Schlotter, I.: Obtaining a planar graph by vertex deletion. In: Brandstädt, A., Kratsch, D., Müller, H. (eds.) WG 2007. LNCS, vol. 4769, pp. 292–303. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74839-7_28
18. Nöllenburg, M., Sorge, M., Terziadis, S., Villedieu, A., Wu, H., Wulms, J.: Planarizing graphs and their drawings by vertex splitting. In: Angelini, P., von Hanxleden, R. (eds.) GD 2022. LNCS, vol. 13764, pp. 232–246. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-22203-0_17
19. Philip, G., Raman, V., Villanger, Y.: A quartic kernel for pathwidth-one vertex deletion. In: Thilikos, D.M. (ed.) WG 2010. LNCS, vol. 6410, pp. 196–207. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16926-7_19
20. Purchase, H.C., Cohen, R.F., James, M.: Validating graph drawing aesthetics. In: Brandenburg, F.J. (ed.) GD 1995. LNCS, vol. 1027, pp. 435–446. Springer, Heidelberg (1996). https://doi.org/10.1007/BFb0021827
21. Robertson, N., Seymour, P.D.: Graph minors. XIII. The disjoint paths problem. J. Comb. Theory, Ser. B **63**(1), 65–110 (1995). https://doi.org/10.1006/jctb.1995.1006
22. Thomassé, S.: A $4k^2$ kernel for feedback vertex set. ACM Trans. Algorithms **6**(2), 32:1–32:8 (2010). https://doi.org/10.1145/1721837.1721848
23. Tsur, D.: Faster algorithm for pathwidth one vertex deletion. Theor. Comput. Sci. **921**, 63–74 (2022). https://doi.org/10.1016/j.tcs.2022.04.001

Odd Chromatic Number of Graph Classes

Rémy Belmonte[1], Ararat Harutyunyan[2], Noleen Köhler[2](✉), and Nikolaos Melissinos[3]

[1] Université Gustave Eiffel, CNRS, LIGM, 77454 Marne-la-Vallée, France
remy.belmonte@u-pem.fr

[2] Université Paris-Dauphine, PSL University, CNRS UMR7243, LAMSADE, Paris, France
ararat.harutyunyan@lamsade.dauphine.fr, noleen.kohler@dauphine.psl.eu

[3] Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, Prague, Czech Republic

Abstract. A graph is called *odd* (respectively, *even*) if every vertex has odd (respectively, even) degree. Gallai proved that every graph can be partitioned into two even induced subgraphs, or into an odd and an even induced subgraph. We refer to a partition into odd subgraphs as an *odd colouring* of G. Scott [Graphs and Combinatorics, 2001] proved that a graph admits an odd colouring if and only if it has an even number of vertices. We say that a graph G is k-odd colourable if it can be partitioned into at most k odd induced subgraphs. We initiate the systematic study of odd colouring and odd chromatic number of graph classes. In particular, we consider for a number of classes whether they have bounded odd chromatic number. Our main results are that interval graphs, graphs of bounded modular-width and graphs of bounded maximum degree all have bounded odd chromatic number.

Keywords: Graph classes · Vertex partition problem · Odd colouring · Colouring variant · Upper bounds

1 Introduction

A graph is called *odd* (respectively even) if all its degrees are odd (respectively even). Gallai proved the following theorem (see [8], Problem 5.17 for a proof).

Theorem 1. *For every graph G, there exist:*

- *a partition (V_1, V_2) of $V(G)$ such that $G[V_1]$ and $G[V_2]$ are both even;*
- *a partition (V_1', V_2') of $V(G)$ such that $G[V_1']$ is odd and $G[V_2']$ is even.*

Ararat Harutyunyan is supported by the grant from French National Research Agency under JCJC program (DAGDigDec: ANR-21-CE48-0012).
Noleen Köhler is supported by the grant from French National Research Agency under JCJC program (ASSK: ANR-18-CE40-0025-01).
Nikolaos Melissinos is partially supported by the CTU Global postdoc fellowship program.

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 44–58, 2023.
https://doi.org/10.1007/978-3-031-43380-1_4

This theorem has two main consequences. The first one is that every graph contains an induced even subgraph with at least $|V(G)|/2$ vertices. The second is that every graph can be *even coloured* with at most two colours, i.e., partitioned into two (possibly empty) sets of vertices, each of which induces an even subgraph of G. In both cases, it is natural to wonder whether similar results hold true when considering odd subgraphs.

The first question, known as the *odd subgraph conjecture* and mentioned already by Caro [3] as part of the graph theory folklore, asks whether there exists a constant $c > 0$ such that every graph G contains an odd subgraph with at least $|V(G)|/c$ vertices. In a recent breakthrough paper, Ferber and Krivelevich proved that the conjecture is true.

Theorem 2 **([5]).** *Every graph G with no isolated vertices has an odd induced subgraph of size at least $|V(G)|/10000$.*

The second question is whether every graph can be partitioned into a bounded number of odd induced subgraphs. We refer to such a partition as an *odd colouring*, and the minimum number of parts required to odd colour a given graph G, denoted by $\chi_{\text{odd}}(G)$, as its *odd chromatic number*. This can be seen as a variant of proper (vertex) colouring, where one seeks to partition the vertices of a graph into odd subgraphs instead of independent sets. An immediate observation is that in order to be odd colourable, a graph must have all its connected components be of even order, as an immediate consequence of the handshake lemma. Scott [11] proved that this necessary condition is also sufficient. Therefore, graphs can generally be assumed to have all their connected components of even order, unless otherwise specified.

Motivated by this result, it is natural to ask how many colours are necessary to partition a graph into odd induced subgraphs. As Scott showed [11], there exist graphs with arbitrarily large odd chormatic number. On the computational side, Belmonte and Sau [2] proved that the problem of deciding whether a graph is k-odd colourable is solvable in polynomial time when $k \leq 2$, and NP-complete otherwise, similarly to the case of proper colouring. They also show that the k-odd colouring problem can be solved in time $2^{O(k \cdot rw)} \cdot n^{O(1)}$, where k is the number of colours and rw is the rank-width of the input graphs. They then ask whether the problem can be solved in FPT time parameterized by rank-width alone, i.e., whether the dependency on k is necessary. A positive answer would provide a stark contrast with proper colouring, for which the best algorithms run in time $n^{2^{O(rw)^2}}$ (see, e.g., [7]), while Fomin et al. [6] proved that there is no algorithm that runs in time $n^{2^{o(rw)}}$, unless the ETH fails.[1]

On the combinatorial side, Scott showed that there exist graphs that require $\Theta(\sqrt{n})$ colours. In particular, the *subdivided clique*, i.e., the graph obtained from a complete graph on n vertices by subdividing[2] every edge once requires

[1] While Fomin et al. proved the lower bound for clique-width, it also holds for rank-width, since rank-width is always at most clique-width.

[2] Subdividing an edge uv consists in removing uv, adding a new vertex w, and making it adjacent to exactly u and v.

exactly n colours, as the vertices obtained by subdividing the edges force their two neighbours to be given distinct colours. More generally, and by the same argument, given any graph G, the graph H obtained from G by subdividing every edge once has $\chi_{\text{odd}}(H) = \chi(G)$, and H is odd colourable if and only if $|V(H)| = |V(G)|+|E(G)|$ is even. Note that a subdivided clique is odd colourable if and only if the subdivided complete graph K_n satisfies $n \in \{k : k \equiv 0 \vee k \equiv 3 \pmod 4\}$. Surprisingly, Scott also showed that only a sublinear number of colours is necessary to odd colour a graph, i.e., every graph of even order G has $\chi_{\text{odd}}(G) \leq cn(\log\log n)^{-1/2}$. As Scott observed, this bound is quite weak, and he instead conjectures that the lower bound obtained from the subdivided clique is essentially tight:

Conjecture 1 (Scott, 2001) . Every graph G of even order has $\chi_{\text{odd}}(G) \leq (1 + o(1))c\sqrt{n}$.

One way of seeing Conjecture 1 is to consider that subdivided cliques appear to be essentially the graphs that require most colours to be odd coloured. More specifically, consider the family $\mathcal{B}$ of graphs G' obtained from a graph G by adding, for every pair of vertices $u, v \in V(G)$, a vertex w_{uv} and edges uw_{uv} and vw_{uv}, and G' has even order. Note that subdivided cliques of even order are exactly those graphs in $\mathcal{B}$ where graph G is edgeless, and that the graphs in $\mathcal{B}$ have $\chi_{\text{odd}}(G') = |V(G)| \in \Theta(\sqrt{|V(G')|})$. A question closely related to Conjecture 1 is whether if a class of graphs $\mathcal{G}$ does not contain arbitrarily large graphs of $\mathcal{B}$ as induced subgraphs, then $\mathcal{G}$ has odd chromatic number $\mathcal{O}(\sqrt{n})$, i.e., they satisfy Conjecture 1. This question was already answered positively for some graph classes. In fact, the bounds provided were constant. It was shown in [2] that every cograph can be odd coloured using at most three colours, and that graphs of treewidth at most k can be odd coloured using at most $k + 1$ colours. In fact, those results can easily be extended to all graphs admitting a join, and H-minor free graphs, respectively. Using a similar argument, Aashtab et al. [1] showed that planar graphs are 4-odd colourable, and this is tight due to subdivided K_4 being planar and 4-odd colourable, as explained above. They also proved that subcubic graphs are 4-odd colourable, which is again tight due to subdivided K_4, and conjecture that this result can be generalized to all graphs, i.e., $\chi_{\text{odd}}(G) \leq \Delta + 1$, where Δ denotes the maximum degree of G. Observe that none of those graph classes contain arbitrarily large graphs from $\mathcal{B}$ as induced subgraphs. On the negative side, bipartite graphs and split graphs contain arbitrarily large graphs from $\mathcal{B}$, and therefore the bound of Conjecture 1 is best possible. In fact, Scott specifically asked whether the conjecture holds for the specific case of bipartite graphs.

Our Contribution. Motivated by these first isolated results and Conjecture 1, we initiate the systematic study of the odd chromatic number in graph classes, and determine which have bounded odd chromatic number. We focus on graph classes that do not contain large graphs from $\mathcal{B}$ as induced subgraphs. Our main results are that graphs of bounded maximum degree, interval graphs and graphs of bounded modular width all have bounded odd chromatic number.

In Sect. 3, we prove that every graph G of even order and maximum degree Δ has $\chi_{\text{odd}}(G) \leq 2\Delta - 1$, extending the result of Aashtab et al. on subcubic graphs to graphs of bounded degree. We actually prove a more general result, which provides additional corollaries for graphs of large girth. In particular, we obtain that planar graphs of girth 11 are 3-odd colourable. We also obtain that graphs of girth at least 7 are $\mathcal{O}(\sqrt{n})$-odd colourable. While this bound is not constant, it is of particular interest as subdivided cliques have girth exactly 6.

In Sect. 4 we prove that every graph with all connected components of even order satisfies $\chi_{\text{odd}}(G) \leq 3 \cdot mw(G)$, where $mw(G)$ denotes the modular-width of G. This significantly generalizes the cographs result from [2] and provides an important step towards proving that graphs of bounded rank-width have bounded odd chromatic number, which in turn would imply that the Odd Chromatic Number is FPT when parameterized by rank-width alone.

Finally, we prove in Sect. 5 that every interval graph with all components of even order is 6-odd colourable. Additionally, every proper interval graph with all components of even order is 3-odd colourable, and this bound is tight.

We would also like to point out that all our proofs are constructive and furthermore a (not necessarily) optimal odd-colouring with the number of colours matching the upper bound can be computed in polynomial time. In particular, the proof provided in [8] of Theorem 1, upon which we rely heavily is constructive, and both partitions can easily be computed in polynomial time. An overview of known results and open cases is provided in Fig. 1 below.

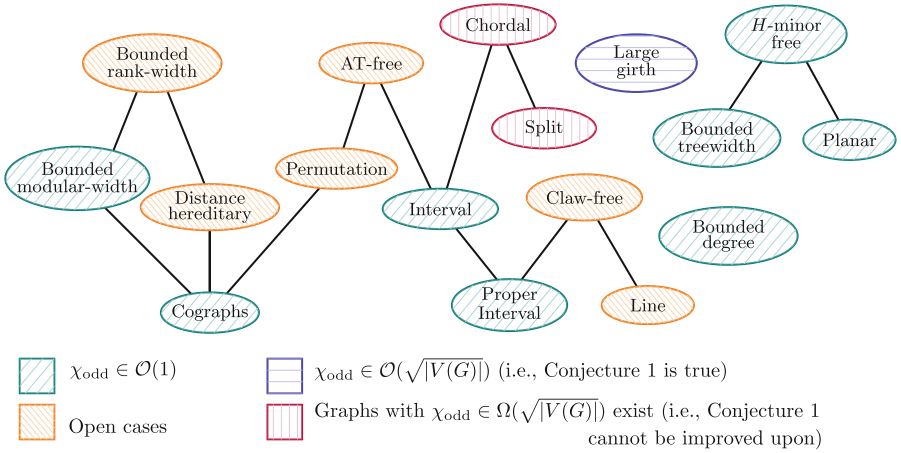

Chordal
Bounded rank-width
AT-free
Large girth
H-minor free
Split
Bounded treewidth
Planar
Bounded modular-width
Permutation
Distance hereditary
Interval
Claw-free
Bounded degree
Cographs
Proper Interval
Line
$\chi_{\text{odd}} \in \mathcal{O}(1)$
Open cases
$\chi_{\text{odd}} \in \mathcal{O}(\sqrt{|V(G)|})$ (i.e., Conjecture 1 is true)
Graphs with $\chi_{\text{odd}} \in \Omega(\sqrt{|V(G)|})$ exist (i.e., Conjecture 1 cannot be improved upon)

Fig. 1. Overview of known and open cases.

2 Preliminaries

For a positive integer i, we denote by $[i]$ the set of integers j such that $1 \leq j \leq i$. A partition of a set X is a tuple $\mathcal{P} = (P_1, \ldots, P_k)$ of subsets of X such that $X =$

$\bigcup_{i\in[k]} P_i$ and $P_i \cap P_j = \emptyset$, i.e., we allow parts to be empty. Let $\mathcal{P} = (P_1, \ldots, P_k)$ be a partition of X and $Y \subseteq X$. We let $\mathcal{P}|_Y$ be the partition of Y obtained from $(P_1 \cap Y, \ldots, P_k \cap Y)$ by removing all empty parts. A partition $(Q_1, \ldots, Q_\ell)$ of X is a *coarsening* of a partition $(P_1, \ldots, P_k)$ of X if for every P_i and every Q_j either $P_i \cap Q_j = \emptyset$ or $P_i \cap Q_j = P_i$, i.e., every Q_j is the union of P_i's.

Every graph in this paper is simple, undirected and finite. We use standard graph-theoretic notation, and refer the reader to [4] for any undefined notation. For a graph G we denote the set of vertices of G by $V(G)$ and the edge set by $E(G)$. Let G be a graph and $S \subseteq V(G)$. We denote an edge between u and v by uv. The *order* of G is $|V(G)|$. The *degree* (respectively, *open neighborhood*) of a vertex $v \in V(G)$ is denoted by $d_G(v)$ (respectively, $N_G(v)$). We denote the subgraph induced by S by $G[S]$. $G \setminus S = G[V(G) \setminus S]$. The *maximum* degree of any vertex of G is denoted by Δ. We denote paths and cycles by tuples of vertices. The *girth* of G is the length of a shortest cycle of G. Given two vertices u and v lying in the same connected component of G, we say an edge e *separates* u and v if they lie in different connected components of $G \setminus \{e\}$.

A graph is called odd (even, respectively) if every vertex has odd (respectively, even) degree. A partition $(V_1, \ldots, V_k)$ of $V(G)$ is a k*-odd colouring*[3] of G if $G[V_i]$ induces an odd subgraphs of G for every $i \in [k]$. We say a graph is k-odd colourable if it admits a k-odd colouring. The *odd chromatic number* of G, denoted by $\chi_{\text{odd}}(G)$, is the smallest integer k such that G is k-odd colourable. The empty graph (i.e., $V(G) = \emptyset$) is considered to be both even and odd. Since every connected component can be odd coloured separately, we only need to consider connected graphs.

Modular-width. A set S of vertices is called a *module* if, for all $u, v \in S, N(u) \cap S = N(v) \cap S$. A partition $\mathcal{M} = (M_1, \ldots, M_k)$ of $V(G)$ is a module partition of G if every M_i is a module in G. Without loss of generality, we further ask that any module partition $\mathcal{M}$ of G, unless $G = K_1$, is non-trivial, i.e., $\mathcal{M}$ has at least two non-empty parts. Given two sets of vertices X and Y, we say that X and Y are *complete to each other* (*completely non-adjacent*, respectively) if $uv \in E(G)$ ($uv \notin E(G)$, respectively) for every $u \in X, v \in Y$. Note that for any two modules M and N in G, either M and N are non-adjacent or complete to each other. We let $G_{\mathcal{M}}$ be the module graph of $\mathcal{M}$, i.e., the graph on vertex set $\mathcal{M}$ with an edge between M_i and M_j if and only if M_i and M_j are complete to each other (non-adjacency between modules M_i, M_j in $G_{\mathcal{M}}$ corresponds to M_i and M_j being non-adjacent in G). We define the modular width of a graph G, denoted by $\text{mw}(G)$, recursively as follows. $\text{mw}(K_1) = 1$, the width of a module partition $(M_1, \ldots, M_k)$ of G is the maximum over k and $\text{mw}(G[M_i])$ for all $i \in [k]$ and $\text{mw}(G)$ is the minimum width of any module partitions of G.

[3] This definition of odd colouring is not to be confused with the one introduced by Petrusevski and Skrekovski [10], which is a specific type of proper colouring.

3 Graphs of Bounded Degree and Graphs of Large Girth

In this section, we study Scott's conjecture (Conjecture 1) as well as the conjecture made by Aashtab et al. [1] which states that $\chi_{\mathrm{odd}}(G) \leq \Delta + 1$ for any graph G. We settle Conjecture 1 for graphs of girth at least 7, and prove that $\chi_{\mathrm{odd}}(G) \leq 2\Delta - 1$ for any graph G, thus obtaining a weaker version of the conjecture of Aashtab et al. To this end, we prove the following more general theorem, which implies both of the aforementioned results.

Theorem 3. *Let $\mathcal{H}$ be a class of graphs such that:*

- $K_2 \in \mathcal{H}$
- *$\mathcal{H}$ is closed under vertex deletion and*
- *there is a $k \geq 2$ such that any connected graph $G \in \mathcal{H}$ satisfies at least one of the following properties:*
  (I) G has two pendant vertices u, v such that $N_G(u) = N_G(v)$ or
  (II) G has two adjacent vertices u, v such that $d_G(u) + d_G(v) \leq k$.

Then every graph $G \in \mathcal{H}$ with all components of even order has $\chi_{\mathrm{odd}}(G) \leq k-1$.

Proof. First notice that $\mathcal{H}$ is well defined as K_2 has the desired properties. The proof is by induction on the number of vertices. Let $|V(G)| = 2n$.

For $n = 1$, since G is connected, we have that $G = K_2$ which is odd. Therefore, $\chi_{\mathrm{odd}}(G) = 1 \leq k - 1$ (recall that $k \geq 2$). Let G be a graph of order $2n$. Notice that we only need to consider the case where G is connected as, otherwise, we can apply the inductive hypothesis to each of the components of G. Assume first that G has two pendant vertices u, v such that $N_G(u) = N_G(v) = \{w\}$. Then, since $G \setminus \{u, v\}$ is connected and belongs to $\mathcal{H}$, by induction, there is an odd colouring of $G \setminus \{u, v\}$ that uses at most $k - 1$ colours. Let $(V_1, \dots, V_{k-1})$ be a partition of $V(G) \setminus \{u, v\}$ such that $G[V_i]$ is odd for all $i \in [k - 1]$. We may assume that $w \in V_1$. We give a partition $V_1', \dots, V_{k-1}'$ of $V(G)$ by setting $V_1' = V_1 \cup \{u, v\}$ and $V_i' = V_i$ for all $i \in [k] \setminus \{1\}$. Notice that for all $i \in [k - 1]$, $G[V_i']$ is odd. Therefore, $\chi_{\mathrm{odd}}(G) \leq k - 1$.

Thus, we assume that G has an edge $uv \in E(G)$ such that $d_G(u)+d_G(v) \leq k$. We may assume that $k \geq 3$ for otherwise the theorem follows. We consider two cases; $G \setminus \{u, v\}$ is connected and $G \setminus \{u, v\}$ is disconnected.

Assume that $G \setminus \{u, v\}$ is connected. Since $G \setminus \{u, v\}$ has $|V(G) \setminus \{u, v\}| = 2n - 2$ and belongs to $\mathcal{H}$, by induction, there is an odd colouring of it that uses at most $k - 1$ colours. Let $(V_1, \dots, V_{k-1})$ be a partition of $V(G) \setminus \{u, v\}$, such that $G[V_i]$ is odd of all $i \in [k - 1]$. We give a partition of G into $k - 1$ odd graphs as follows. Since $|N_G(\{u, v\})| \leq k - 2$, there exists $\ell \in [k - 1]$ such that $V_\ell \cap N_G(\{u, v\}) = \emptyset$. We define a partition $(U_1, \dots, U_{k-1})$ of $V(G)$ as follows. For all $i \in [k - 1]$, if $i \neq \ell$, we define $U_i = V_i$, otherwise we set $U_i = V_i \cup \{u, v\}$. Notice that for all $i \neq \ell$, $G[U_i]$ is odd since $U_i = V_i$. Also, since $N_{G[U_\ell]}[v] = N_{G[U_\ell]}[u] = \{u, v\}$ and $G[V_\ell]$ is odd, we conclude that $G[U_\ell]$ is odd. Thus, $\chi_{\mathrm{odd}}(G) \leq k - 1$.

Now, we consider the case where $G \setminus \{u, v\}$ is disconnected. First, we assume that there is at least one component in $G \setminus \{u, v\}$ of even order. Let U be the set of vertices of this component. By induction, $\chi_{\text{odd}}(G[U]) \leq k-1$ and $\chi_{\text{odd}}(G \setminus U) \leq k-1$. Furthermore, $|N_G(\{u,v\}) \cap U| \leq k-3$ because $G \setminus \{u,v\}$ has at least two components. Let $(U_1,\dots,U_{k-1})$ be a partition of U such that $G[U_i]$ is odd for all $i \in [k-1]$. Also, let $(V_1,\dots,V_{k-1})$ be a partition of $V(G) \setminus U$ such that $G[V_i]$ is odd for all $i \in [k-1]$. We may assume that $V_i \cap \{u,v\} = \emptyset$ for all $i \in [k-3]$. Since $|N_G(\{u,v\}) \cap U| \leq k-3$, there are at least two indices $l, l' \in [k-1]$ such that $U_l \cap N_G(\{u,v\}) = U_{l'} \cap N_G(\{u,v\}) = \emptyset$. We may assume that $l = k-2$ and $l' = k-1$. We define a partition $(V'_1,\dots,V'_{k-1})$ of $V(G)$ as follows. For all $i \in [k-1]$ we define $V'_i = U_i \cup V_i$. We claim that $G[V'_i]$ is odd for all $i \in [k-1]$. To show the claim, we consider two cases; either $V'_i \cap \{u,v\} = \emptyset$ or not. If $V'_i \cap \{u,v\} = \emptyset$, since the only vertices in $V(G) \setminus U$ that can have neighbours in U are v and u we have that $G[V'_i]$ is odd. Indeed, this holds because $U_i \cap N_G(V_i) = \emptyset$ and both $G[U_i]$ and $G[V_i]$ are odd. If $V'_i \cap \{u,v\} \neq \emptyset$, then $i = k-2$ or $i = k-1$. In both cases, we know that $U_i \cap N_G(V_i) = \emptyset$ because the only vertices in $V(G) \setminus U$ that may have neighbours in U are v and u and we have assumed that u, v do not have neighbours in $U_{k-2} \cup U_{k-1}$. So, $G[V'_i]$ is odd because $U_i \cap N_G(V_i) = \emptyset$ and both $G[U_i]$ and $G[V_i]$ are odd.

Thus, we can assume that all components of $G \setminus \{u,v\}$ are of odd order. Let $\ell > 0$ be the number of components, denoted by $V_1,\dots,V_\ell$, of $G \setminus \{u,v\}$ and note that ℓ must be even. We consider two cases, either for all $i \in [\ell]$, one of $G[V_i \cup \{u\}]$ or $G[V_i \cup \{v\}]$ is disconnected, or there is at least one $i \in [\ell]$ such that both $G[V_i \cup \{u\}]$ and $G[V_i \cup \{v\}]$ are connected.

In the first case, for each V_i, $i \in [\ell]$ we call w_i the vertex in $\{u,v\}$ such that $G[V_i \cup \{w_i\}]$ is connected. Note that w_i is uniquely determined, i.e., only one of u and v can be w_i for each $i \in [\ell]$. Now, by induction, for all $i \in [\ell]$, $G[V_i \cup \{w_i\}]$ has $\chi_{\text{odd}}(G[V_i \cup \{w_i\}]) \leq k-1$. Let, for each $i \in [\ell]$, $(V^i_1,\dots,V^i_{k-1})$ denote a partition of $V_i \cup \{w_i\}$ such that $G[V^i_j]$ be odd, for all $j \in [k-1]$. Furthermore, we may assume that for each $i \in [\ell]$, if $v \in V_i \cup \{w_i\}$, then $v \in V^i_{k-2}$. Also, we can assume that for each $i \in [\ell]$, if $u \in V_i \cup \{w_i\}$, then $u \in V^i_{k-1}$. Finally, let $I = \{i \in [\ell] \mid w_i = u\}$ and $J = \{i \in [\ell] \mid w_i = v\}$.

We consider two cases. If $|I|$ is odd, then $|J|$ is odd since $\ell = |I| + |J|$ is even. Then, we claim that for the partition $(U_1,\dots,U_{k-1})$ of $V(G)$ where $U_i = \bigcup_{j \in [\ell]} V^j_i$ it holds that $G[U_i]$ is odd for all $i \in [k-1]$. First notice that $(U_1,\dots,U_{k-1})$ is indeed a partition of $V(G)$. Indeed, the only vertices that may belong in more than one set are u and v. However, v belongs only to some sets V^i_{k-2}, and hence it is no set U_i except U_{k-2}. Similarly, u belongs to no set U_i except U_{k-1}. Therefore, it remains to show that $G[U_i]$ is odd for all $i \in [k-1]$. We will show that for any $i \in [k-1]$ and for any $x \in U_i$, $|N_G(x) \cap U_i|$ is odd. Let $x \in U_i \setminus \{u,v\}$, for some $i \in [k-1]$. Then we know that $N_G(x) \cap U_i = N_G(x) \cap V^j_i$ for some $j \in [\ell]$. Since $G[V^j_i]$ is odd for all $i \in [k-1]$ and $j \in [\ell]$ we have that $|N_G(x) \cap U_i| = |N_G(x) \cap V^j_i|$ is odd. Therefore, we only need to consider u and v. Notice that $v \in U_{k-2} = \bigcup_{j \in [\ell]} V^j_{k-2}$ (respectively, $u \in U_{k-1} = \bigcup_{j \in [\ell]} V^j_{k-1}$). Also, v (respectively, u) is included in V^j_{k-2} (respectively, V^j_{k-1}) only if $j \in I$ (respectively, $j \in J$). Since $G[V^j_{k-2}]$ (respectively, $G[V^j_{k-1}]$) is odd for any $j \in [\ell]$ we have that $|N(v) \cap V^j_{k-2}|$ (respectively, $|N(u) \cap V^j_{k-1}|$) is odd for any

$j \in I$ (respectively, $j \in J$). Finally, since $|I|$ and $|J|$ are odd, we have that $|N_G(v) \cap U_{k-2}| = \sum_{j \in I} |N(v) \cap V_{k-2}^j|$ and $|N_G(u) \cap U_{k-1}| = \sum_{j \in I} |N(u) \cap V_{k-1}^j|$ are both odd. Therefore, for any $i \in [k-1]$, $G[U_i]$ is odd and $\chi_{\text{odd}}(G) \leq k-1$.

Now, suppose that both $|I|$ and $|J|$ are even. We consider the partition $(U_1, \ldots, U_{k-1})$ of $V(G)$ where, for all $i \in [k-3]$ $U_i = \bigcup_{j \in [\ell]} V_i^j$, $U_{k-2} = \bigcup_{j \in J} V_{k-2}^j \cup \bigcup_{j \in I} V_{k-1}^j$ and $U_{k-1} = \bigcup_{j \in I} V_{k-2}^j \cup \bigcup_{j \in J} V_{k-1}^j$. We claim that for this partition it holds that $G[U_i]$ is odd for all $i \in [k-1]$. First notice that $(U_1, \ldots, U_{k-1})$ is indeed a partition of $V(G)$. Indeed, this is clear for all vertices except for v and u. However, v only belongs to sets of type V_{k-2}^i for $i \in I$, and u only belongs to sets of type V_{k-1}^i for $i \in J$. Therefore, u or v belong to no set U_i except U_{k-1}. We will show that for any $i \in [k-1]$ and $x \in U_i$, $|N_G(x) \cap U_i|$ is odd. Let $x \in U_i \setminus \{u, v\}$, for some $i \in [k-1]$. Then we know that $N_G(x) \cap U_i = N_G(x) \cap V_i^j$ for some $j \in [\ell]$. Since $G[V_i^j]$ is odd for all $i \in [k-1]$ and $j \in [\ell]$ we have that $|N_G(x) \cap U_i| = |N_G(x) \cap V_i^j|$ is odd. Therefore, we only need to consider v and u. Note that $u, v \in U_{k-1}$. Since both $|I|$ and $|J|$ are even and $U_{k-1} = \bigcup_{j \in I} V_{k-2}^j \cup \bigcup_{j \in J} V_{k-1}^j$, we have that $|N_G(v) \cap U_{k-1} \setminus \{u\}|$ and $|N_G(u) \cap U_{k-1} \setminus \{v\}|$ are both even. Finally, since $uv \in E(G)$ we have that $|N_G(v) \cap U_{k-1}|$ and $|N_G(u) \cap U_{k-1}|$ are both odd. Hence, $\chi_{\text{odd}}(G) \leq k-1$.

Now we consider the case where there is at least one $i \in [\ell]$ where both $G[V_i \cup \{v\}]$ and $G[V_i \cup \{u\}]$ are connected. We define the following sets I and J. For each $i \in [\ell]$, (i) $i \in J$, if $G[V_i \cup \{v\}]$ is disconnected, and (ii) $i \in I$, if $G[V_i \cup \{u\}]$ is disconnected. Finally, for the rest of the indices, $i \in [\ell]$, which are not in $I \cup J$, it holds that both $G[V_i \cup \{v\}]$ and $G[V_i \cup \{u\}]$ are connected. Call this set of indices X and note that by assumption $|X| \geq 1$. Since $|I| + |J| + |X|$ is even, it is easy to see that there is a partition of X into two sets X_1 and X_2 such that both $I' := I \cup X_1$ and $J' := J \cup X_2$ have odd size. Let $V_I = \bigcup_{i \in I'} V_i$ and $V_J = \bigcup_{i \in J'} V_i$. Now, by induction, we have that $\chi_{\text{odd}}(G[V_I \cup \{v\}]) \leq k-1$ and $\chi_{\text{odd}}(G[V_J \cup \{u\}]) \leq k-1$. Assume that $(V_1^I, \ldots, V_{k-1}^I)$ is a partition of V_I and $(V_1^J, \ldots, V_{k-1}^J)$ is a partition of V_J such that for any $i \in [k-1]$, $G[V_i^I]$ and $G[V_i^J]$ are odd. Without loss of generality, we may assume that $v \in V_1^I$ and $u \in V_{k-1}^J$. Since $|X| \geq 1$, note that both $d_G(u)$ and $d_G(v)$ are at least two, which implies that $d_G(u) \leq k-2$ and $d_G(v) \leq k-2$. Therefore, there exists $i_0 \in [k-2]$ such that $N_G(v) \cap V_{i_0}^J = \emptyset$ and $j_0 \in [k-1] \setminus \{1\}$ such that $N_G(v) \cap V_{j_0}^I = \emptyset$. We reorder the sets V_i^J, $i \in [k-2]$, so that $i_0 = 1$ and we reorder the sets V_i^I, $i \in [k-1] \setminus \{1\}$ so that $j_0 = k-1$. Note that this reordering does not change the fact that $v \in V_1^I$ and $u \in V_{k-1}^J$. Consider the partition $(U_1, \ldots, U_{k-1})$ of $V(G)$, where $U_i = V_i^I \cup V_i^J$. We claim that for all $i \in [k-1]$, $G[U_i]$ is odd. Note that for any $x \in U_i$, we have $N_G(x) \cap U_i = N_G(x) \cap V_i^I$ or $N_G(x) \cap U_i = N_G(x) \cap V_i^J$. Since for any $i \in [k-1]$, $G[V_i^I]$ and $G[V_i^J]$ are odd we conclude that $G[U_i]$ is odd for any $i \in [k-1]$. □

Notice that the class of graphs G of maximum degree Δ satisfies the requirements of Theorem 3. Indeed, this class is closed under vertex deletions and any connected graph in the class has least two adjacent vertices u, v such that $d_G(u) + d_G(v) \leq 2\Delta$. Therefore, the following corollary holds.

Corollary 1. *For every graph G with all components of even order, $\chi_{\text{odd}}(G) \leq 2\Delta - 1$.*

Next, we prove Conjecture 1 for graphs of girth at least seven.

Corollary 2. *For every graph G with all components of even order of girth at least 7, $\chi_{\text{odd}}(G) \leq \frac{3\sqrt{|V(G)|}}{2} + 1$.* $(*)$[4].

One may wonder if graphs of sufficiently large girth have bounded odd chromatic number. In fact, this is far from being true, which we show in the next.

Proposition 1. *For every integer g and k, there is a graph G such that every component of G has even order, G is of girth at least g and $\chi_{\text{odd}}(G) \geq k$.* $(*)$

Next, we obtain the following result for sparse planar graphs.

Corollary 3. *For every planar graph G with all components of even order of girth at least 11, $\chi_{\text{odd}}(G) \leq 3$.* $(*)$

The upper bound in Corollary 3 is tight as C_{14}, the cycle of length 14, has $\chi_{\text{odd}}(C_{14}) = 3$.

4 Graphs of Bounded Modular-Width

In this section we consider graphs of bounded modular-width and show that we can upper bound the odd chromatic number by the modular-width of a graph.

Theorem 4. *For every graph G with all components of even order, $\chi_{\text{odd}}(G) \leq 3\,\text{mw}(G)$.*

In order to prove Theorem 4 we show that every graph G is 3-colourable for which we have a module partition $\mathcal{M}$ such that the module graph $G_{\mathcal{M}}$ exhibits a particular structure, i.e., is either a star Lemma 1 or a special type of tree Lemma 2. The following is an easy consequence of Theorem 1 which will be useful to colour modules and gain control over the parity of parts in case of modules of even size.

Remark 1. For every non-empty graph G of even order, there exists a partition (V_1, V_2, V_3) of $V(G)$ with $|V_2|$, $|V_3|$ being odd such that $V[G_1]$ is odd and $G[V_2]$, $G[V_3]$ are even. This can be derived from Theorem 1 by taking an arbitrary vertex $v \in V(G)$, setting $V_3 := \{v\}$ and then using the existence of a partition (V_1, V_2) of $V(G) \setminus \{v\}$ such that $G[V_1]$ is odd and $G[V_2]$ is even.

Lemma 1. *For every connected graph G of even order with a module partition $\mathcal{M} = \{M_1, \ldots, M_k\}$ such that $G_{\mathcal{M}}$ is a star, $\chi_{\text{odd}}(G) \leq 3$.*

[4] For every result which is marked by $(*)$ the proof can be found in the full version of the paper.

Proof of A. ssume that in $G_{\mathcal{M}}$ the vertices $M_2, \dots, M_k$ have degree 1. We refer to M_1 as the centre and to $M_2, \dots, M_k$ as leaves of $G_{\mathcal{M}}$. We further assume that $|M_2|, \dots, |M_\ell|$ are odd and $|M_{\ell+1}|, \dots, |M_k|$ are even for some $\ell \in [k]$. We use the following two claims.

Claim 1. If $W \subseteq V(G)$ with $G[W \cap M_i]$ is odd for every $i \in [k]$, then $G[W]$ is odd.

Proof. First observe that the degree of any vertex $v \in W \cap M_1$ in $G[W]$ is $d_{G[W \cap M_1]}(v) + \sum_{i=2}^{k} |W \cap M_i|$. Since $d_{G[W \cap M_1]}(v)$ is odd and $|W \cap M_i|$ is even for every $i \in \{2, \dots, k\}$ (which follows from $G[W \cap M_i]$ being odd by the handshake lemma) we get that $d_{G[W]}(v)$ is odd. For every $i \in \{2, \dots, k\}$ the degree of any vertex $v \in W \cap M_i$ in $G[W]$ is $d_{G[W \cap M_i]}(v) + |W \cap M_1|$ which is odd (again, because $|W \cap M_1|$ must be even). Hence $G[W]$ is odd. $\diamondsuit$

Claim 2. If $W \subseteq V(G)$ such that $G[W \cap M_i]$ is even for every $i \in [k]$, $|W \cap M_1|$ is odd and $|\{i \in \{2, \dots, k\} : |W \cap M_i| \text{ is odd}\}|$ is odd, then $G[W]$ is odd.

Proof. Since $G_{\mathcal{M}}$ is a star and M_1 its centre we get that the degree of any vertex $v \in W \cap M_i$ for any $i \in \{2, \dots, k\}$ is $d_{G[W \cap M_i]}(v) + |W \cap M_1|$. Since $|W \cap M_1|$ is odd and $d_{G[W \cap M_i]}(v)$ is even we get that every $v \in W \cap M_i$ for every $i \in \{2, \dots, k\}$ has odd degree in $G[W]$. Moreover, the degree of $v \in W \cap M_1$ is $d_{G[W \cap M_1]}(v) + \sum_{i=2}^{k} |W \cap M_i|$. Since $d_{G[W \cap M_1]}(v)$ is even and $|\{i \in \{2, \dots, k\} : |W \cap M_i| \text{ is odd}\}|$ is odd $d_{G[W]}(v)$ is odd. We conclude that $G[W]$ is odd. $\diamondsuit$

First consider the case that $|M_1|$ is odd. Since G is of even order this implies that there must be an odd number of leaves of $G_{\mathcal{M}}$ of odd size and hence ℓ is even. Using Theorem 1 we let (W_1^i, W_2^i) be a partition of M_i such that $G[W_1^i]$ is odd and $G[W_2^i]$ is even for every $i \in [k]$. Note that since $G[W_1^i]$ is odd $|W_1^i|$ has to be even and hence $|W_2^i|$ is odd if and only if $i \in [\ell]$. We define $V_1 := \bigcup_{i \in [k]} W_1^i$ and $V_2 := \bigcup_{i \in [k]} W_2^i$. Note that (V_1, V_2) is a partition of G. Furthermore, $G[V_1]$ is odd by Claim 1 and $G[V_2]$ is odd by Claim 2. For an illustration see Fig. 2.

Now consider the case that $|M_1|$ is even. We first consider the special case that $\ell = 1$, i.e., there is no $i \in [k]$ such that $|M_i|$ is odd. In this case we let (W_1^i, W_2^i, W_3^i) be a partition of M_i for $i \in \{1, 2\}$ such that $G[W_1^i]$ is odd, $G[W_2^i]$, $G[W_3^i]$ are even and $|W_2^i|$, $|W_3^i|$ are odd which exists due to Remark 1. For $i \in \{3, \dots, k\}$ we let (W_1^i, W_2^i) be a partition of M_i such that $G[W_1^i]$ is odd and $G[W_2^i]$ is even which exists by Theorem 1. We define $V_1 := \bigcup_{i \in [k]} W_1^i$, $V_2 := \bigcup_{i \in [k]} W_2^i$ and $V_3 := W_3^1 \cup W_3^2$. As before we observe that (V_1, V_2, V_3) is a partition of $V(G)$, $G[V_1]$ is odd by Claim 1 and $G[V_2]$, $G[V_3]$ are even by Claim 2. For an illustration see Fig. 2.

Lastly, consider the case that $|M_1|$ is even and $\ell > 1$. By Remark 1 there is a partition (W_1^1, W_2^1, W_3^1) of M_1 such that $G[W_1^1]$ is odd, $G[W_2^1]$, $G[W_3^1]$ are even and $|W_2^1|$, $|W_3^1|$ are odd. For $i \in \{2, \dots, k\}$ we let (W_1^i, W_2^i) be a partition of M_i such that $G[W_1^i]$ is odd and $G[W_2^i]$ is even which exists by Theorem 1.

We define $V_1 := \bigcup_{i\in[k]} W_1^i$, $V_2 := W_2^1 \cup \bigcup_{i=3}^k W_2^i$ and $V_3 := W_3^1 \cup W_2^2$. Note that (V_1, V_2, V_3) is a partition of $V(G)$. Furthermore, $G[V_1]$ is odd by Claim 1 and $G[V_3]$ is odd by Claim 2. Additionally, since $|M_1|$ is even there is an even number of $i \in \{2, \dots, k\}$ such that $|M_i|$ is odd. Since for each $i \in \{2, \dots, k\}$ for which $|M_i|$ is odd, $|W_1^i|$ must be odd, we get that $\left|\{i \in \{2, \dots, k\} : |V_1 \cap M_i| \text{ is odd}\}\right|$ is odd (note that $V_1 \cap M_2 = \emptyset$ because $W_2^2 \subseteq V_3$). Hence we can use Claim 2 to conclude that $G[V_2]$ is odd. For an illustration see Fig. 2. □

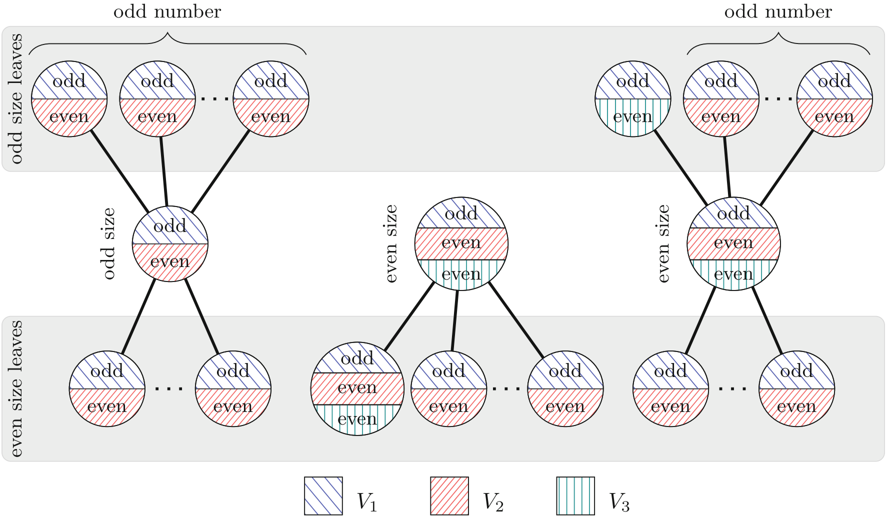


Fig. 2. Schematic illustration of the three cases in the proof of Lemma 1. Depicted is the module graph $G_{\mathcal{M}}$ along with a partition of the modules into sets V_1, V_2 and V_3 such that $G[V_i]$ is odd for $i \in [3]$.

Let G be a connected graph of even order with module partition $\mathcal{M} = (M_1, \dots, M_k)$ such that $G_{\mathcal{M}}$ is a tree. For an edge e of $G_{\mathcal{M}}$ we let X_e and Y_e be the two components of the graph obtained from $G_{\mathcal{M}}$ by removing e. We say that the tree $G_{\mathcal{M}}$ is colour propagating if the following properties hold.

(i) $|\mathcal{M}| \geq 3$.
(ii) Every non-leaf module has size one.
(iii) $|\bigcup_{M\in V(X_e)} M|$ is odd for every $e \in E(G_{\mathcal{M}})$ not incident to any leaf of $G_{\mathcal{M}}$.

Lemma 2. *For every connected graph G of even order with a module partition $\mathcal{M} = (M_1, \dots, M_k)$ such that $G_{\mathcal{M}}$ is a colour propagating tree, $\chi_{\text{odd}}(G) \leq 2$.*

Proof. To find an odd colouring (V_1, V_2) of G, we first let (W_1^i, W_2^i) be a partition of M_i such that $G[W_1^i]$ is odd and $G[W_2^i]$ is even for every $i \in [k]$. The partitions

(W_1^i, W_2^i) exist due to Theorem 1. Note that (ii) implies that for every module M_i which is not a leaf $|W_2^i| = 1$ and $W_1^i = \emptyset$. We define $V_1 := \bigcup_{i\in[k]} W_1^i$ and $V_2 := \bigcup_{i\in[k]} W_2^i$.

To argue that (V_1, V_2) is an odd colouring of G first consider any $v \in V(G)$ such that $v \in M_i$ for some leaf M_i of $G_\mathcal{M}$. Condition (i) implies that $G_\mathcal{M}$ must have at least three vertices and hence the neighbour M_j of M_i cannot be a leaf due to $G_\mathcal{M}$ being a tree. Hence $|M_j| = 1$ by (ii). Hence, if $v \in W_1^i$, then $d_{G[V_1]}(v) = d_{G[W_1^i]}(v)$ since $W_1^j = \emptyset$ and therefore $d_{G[V_1]}(v)$ is odd. Further, if $v \in W_2^i$, then $d_{G[V_2]}(v) = d_{G[W_2^i]}(v) + 1$ since $|W_2^j| = 1$ and hence $d_{G[V_2]}(v)$ is odd. Hence the degree of any vertex $v \in M_i$ is odd in $G[V_1]$, $G[V_2]$ respectively.

Now consider any vertex $v \in V(G)$ such that $M_i = \{v\}$ for some non-leaf M_i of $G_\mathcal{M}$. Let $M_{i_1}, \ldots, M_{i_\ell}$ be the neighbours of M_i in $G_\mathcal{M}$. Let e_j be the edge $M_i M_{i_j} \in E(G)$ for every $j \in [\ell]$. Without loss of generality, assume that $M_i \notin V(X_{e_j})$ for every $j \in [\ell]$. By (iii) we have that $|\bigcup_{M\in V(X_{e_j})} M|$ is odd whenever M_{i_j} is not a leaf in $G_\mathcal{M}$. Hence, by (ii), $|X_{e_j}| \equiv |M_{i_j}| \pmod 2$ for every $j \in [\ell]$ for which M_{i_j} is not a leaf in $G_\mathcal{M}$. On the other hand, as a consequence of the handshake lemma we get that $|W_2^{i_j}|$ is odd if and only if $|M_{i_j}|$ is odd. Hence the following holds for the parity of the degree of v in $G[V_2]$.

$$d_{G[V_2]}(v) = |\{j \in [m] : d_{G_\mathcal{M}}(M_{i_j}) \geq 2\}| + \bigcup_{\substack{j\in[m]\\ d_{G_\mathcal{M}}(M_{i_j})=1}} |W_2^{i_j}| \equiv |V(G) \setminus M_i| \pmod 2.$$

Since G has even order, $d_{G[V_2]}(v)$ is odd and (V_1, V_2) is an odd colouring of G. □

We now show that, given a graph G with module partition $\mathcal{M}$, we can decompose the graph in such a way that the module graph of any part of the decomposition is either a star or a colour propagating tree. Here we consider the module graph with respect to the module partition $\mathcal{M}$ restricted to the part of the decomposition we are considering. To obtain the decomposition we use a spanning tree $G_\mathcal{M}$ and inductively find a non-separating star, i.e., a star whose removal does not disconnect the graph, or a colour propagating tree. In order to handle parity during this process we might separate a module into two parts.

Lemma 3. *For every connected graph G of even order and module partition $\mathcal{M} = (M_1, \ldots, M_k)$ there is a partition $\widehat{\mathcal{M}}$ of $V(G)$ with at most $2k$ many parts such that there is a coarsening $\mathcal{P}$ of $\widehat{\mathcal{M}}$ with the following properties. $|P|$ is even for every part P of $\mathcal{P}$. Furthermore, for every part P of $\mathcal{P}$ we have that $\widehat{\mathcal{M}}|_P$ is a module partition of $G[P]$ and $G[P]_{\widehat{\mathcal{M}}|_P}$ is either a star (with at least two vertices) or a colour propagating tree.* $(*)$

Proof 1. Without loss of generality assume that G is connected. Furthermore, let $k := \mathrm{mw}(G)$ and $\mathcal{M} = (M_1, \ldots, M_k)$ be a module partition of G. Let $\widehat{\mathcal{M}}$ be a partition of $V(G)$ with at most $2k$ parts and $\mathcal{P}$ be a coarsening of $\widehat{\mathcal{M}}$ as in Lemma 3. First observe that $\widehat{\mathcal{M}}|_P$ must contain at least two parts for every part P of $\mathcal{P}$ as $\widehat{\mathcal{M}}|_P$ is a module partition of $G[P]$. Since $\widehat{\mathcal{M}}$ has at most $2k$ parts and

$\mathcal{P}$ is a coarsening of $\widehat{\mathcal{P}}$ this implies that $\mathcal{P}$ has at most k parts. Since $G[P]_{\widehat{\mathcal{M}}|_P}$ is either a star or a colour propagating tree we get that $\chi_{\text{odd}}(G[P]) \leq 3$ for every part P of $\mathcal{P}$ by Lemma 1 and Lemma 2. Using a partition (W_1^P, W_2^P, W_3^P) of $G[P]$ such that $G[W_i^P]$ is odd for every $i \in [3]$ for every part P we obtain a global partition of G into at most $3k$ parts such that each part induces an odd subgraph. □

Since deciding whether a graph is k-odd colourable can be solved in time $2^{\mathcal{O}(k\,\text{rw}(G))}$ [2, Theorem 6] and $\text{rw}(G) \leq \text{cw}(G) \leq \text{mw}(G)$, where $\text{cw}(G)$ denotes the clique-width of G and $\text{rw}(G)$ rank-width, we obtain the following as a corollary.

Corollary 4. *Given a graph G and a module partition of G of width m the problem of deciding whether G can be odd coloured with at most k colours can be solved in time $2^{\mathcal{O}(m^2)}$.*

5 Interval Graphs

In this section we study the odd chromatic number of interval graphs and provide an upper bound in the general case as well as a tight upper bound in the case of proper interval graphs. We use the following lemma in both proofs.

Lemma 4. *Let G be a connected interval graph and $P = (p_1, \ldots, p_k)$ a maximal induced path in G with the following property.*

(Π) $\ell_{p_1} = \min\{\ell_v : v \in V(G)\}$ and for every $i \in [k-1]$ we have that $r_{p_{i+1}} \geq r_v$ for every $v \in N_G(p_i)$.

Then every $v \in V(G)$ is adjacent to at least one vertex on P. $(*)$

To prove that the odd chromatic number of proper interval graphs is bounded by three we essentially partition the graph into maximal even sized cliques greedily in a left to right fashion.

Theorem 5. *For every proper interval graph G with all components of even order, $\chi_{\text{odd}}(G) \leq 3$ and this bound is tight.*

Proof. We assume that G is connected. Fix an interval representation of G and denote the interval representing vertex $v \in V(G)$ by $I_v = [\ell_v, r_v]$ where $\ell_v, r_v \in \mathbb{R}$. Let $P = (p_1, \ldots, p_k)$ be a maximal induced path in G as in Lemma 4. For every vertex $v \in V(G) \setminus \{p_1, \ldots, p_k\}$ let $i_v \in [k]$ be the index such that p_{i_v} is the first neighbour of v on P. Note that this is well defined by Lemma 4. For $i \in [k]$ we let Y_i be the set with the following properties.

(Π)1_i $\{v \in V(G) : i_v = i\} \subseteq Y_i \subseteq \{v \in V(G) : i_v = i\} \cup \{p_i, p_{i+1}\}$.

(Π)2_i $p_i \in Y_i$ if and only if $\left|\{p_1, \ldots, p_{i-1}\} \cup \bigcup_{j\in[i-1]}\{v \in V(G) : i_v = j\}\right|$ is even.

(Π)3_i $p_{i+1} \in Y_i$ if and only if $\left|\{p_1, \ldots, p_i\} \cup \bigcup_{j\in[i]}\{v \in V(G) : i_v = j\}\right|$ is odd.

First observe that $(Y_1, \ldots, Y_k)$ is a partition of $V(G)$ as $(\Pi 2)_i$ and $(\Pi 3)_i$ imply that every p_i is in exactly one set Y_i. Furthermore, $|Y_i|$ is even for every $i \in [k]$ since $(\Pi 1)_i$ and $(\Pi 3)_i$) imply that $\left|Y_i \cup \{p_1, \ldots, p_i\} \cup \bigcup_{j \in [i-1]} \{v \in V(G) : i_v = j\}\right|$ is even and $(\Pi 2)_i$ implies that $\left|(\{p_1, \ldots, p_i\} \cup \bigcup_{j \in [i-1]} \{v \in V(G) : i_v = j\}) \setminus Y_i\right|$ is even. Since $v \in V(G) \setminus \{p_1, \ldots, p_k\}$ is not adjacent to $p_{i_v - 1}$ we get that $\ell_v \in I_{p_{i_v}}$. Since G is a proper interval graph this implies that $r_{p_{i_v}} \leq r_v$ and hence v is adjacent to p_{i_v+1}. Hence $(\Pi 1)_i$ implies that $G[Y_i]$ must be a clique since $Y_i \cap \{p_1, \ldots, p_k\} \subseteq \{p_i, p_{i+1}\}$ for every $i \in [k]$. Furthermore, $N_G(Y_i)$ and Y_{i+3} are disjoint since $r_v \leq r_{p_{i+1}}$ for every $v \in Y_i$ by property (Π) and $r_{p_{i+1}} < \ell_{p_{i+3}} \leq r_w$ for every $w \in Y_{i+3}$ since P is induced. Hence we can define an odd-colouring (V_1, V_2, V_3) of G in the following way. We let $V_j := \bigcup_{i \equiv j \pmod 3} Y_i$ for $j \in [3]$. Note that since $N_G(Y_i) \cap Y_{i+3}$ we get that $d_{G[Y_i]}(v) = d_{G[V_j]}(v)$ for $i \equiv j \pmod 3$ which is odd (as Y_i is a clique of even size). Hence $G[V_j]$ is odd for every $j \in [3]$.

To see that the bound is tight consider the graph G consisting of K_4 with two pendant vertices u, w adjacent to different vertices of K_4. Clearly, G is a proper interval graph and further $\chi_{\text{odd}}(G) = 3$. □

We use a similar setup (i.e., a path P covering all vertices of the graph G) as in the proof of Theorem 5 to show our general upper bound for interval graphs. The major difference is that we are not guaranteed that sets of the form $\{p_i\} \cup \{v \in V(G) : i_v = i\}$ are cliques. To nevertheless find an odd colouring with few colours of such sets we use an odd/even colouring as in Theorem 1 of $\{v \in V(G) : i_v = i\}$ and the universality of p_i. Hence this introduces a factor of two on the number of colours. Furthermore, this approach prohibits us from moving the p_i around as in the proof of Theorem 5. As a consequence we get that the intervals of vertices contained in a set Y_i span a larger area of the real line than in the proof of Theorem 5. This makes the analysis more technical.

Theorem 6. *For every interval graph G with all components of even order, $\chi_{\text{odd}}(G) \leq 6$.* (∗)

Note that we currently are unaware whether the bound from Theorem 6 is tight or even whether there is an interval graph G with $\chi_{\text{odd}}(G) > 3$.

6 Conclusion

We initiated the systematic study of odd colouring on graph classes. Motivated by Conjecture 1, we considered graph classes that do not contain large graphs from a given family as induced subgraphs. Put together, these results provide evidence that Conjecture 1 is indeed correct. Answering it remains a major open problem, even for the specific case of bipartite graphs.

Several other interesting classes remain to consider, most notably line graphs and claw-free graphs. Note that odd colouring a line graph $L(G)$ corresponds to colouring the edges of G in such a way that each colour class induces a bipartite graph where every vertex in one part of the bipartition has odd degree, and every vertex in the other colour part has even degree. This is not to be confused with the notion of odd k-edge colouring, which is a (not necessarily proper) edge colouring with at most k colours such that each nonempty colour class induces a graph in which every vertex is of odd degree. It is known that all simple graphs can be odd 4-edge coloured, and every loopless multigraph can be odd 6-edge coloured (see e.g., [9]). While (vertex) odd colouring line graphs is not directly related to odd edge colouring, this result leads us to believe that line graphs have bounded odd chromatic number.

Finally, determining whether Theorem 4 can be extended to graphs of bounded rank-width remains open. We also believe that the bounds in Theorem 6 and Corollary 1 are not tight and can be further improved. In particular, we believe that the following conjecture, first stated in [1], is true:

Conjecture 2 (Aashtab et al., 2023). Every graph G of even order has $\chi_{\text{odd}}(G) \leq \Delta + 1$.

References

1. Aashtab, A., Akbari, S., Ghanbari, M., Shidani, A.: Vertex partitioning of graphs into odd induced subgraphs. Discuss. Math. Graph Theory **43**(2), 385–399 (2023)
2. Belmonte, R., Sau, I.: On the complexity of finding large odd induced subgraphs and odd colorings. Algorithmica **83**(8), 2351–2373 (2021)
3. Caro, Y.: On induced subgraphs with odd degrees. Discret. Math. **132**(1–3), 23–28 (1994)
4. Diestel, R.: Graph Theory, 4th Edn., vol. 173 of Graduate texts in mathematics. Springer, Cham (2012)
5. Ferber, A., Krivelevich, M.: Every graph contains a linearly sized induced subgraph with all degrees odd. Adv. Math. **406**, 108534 (2022)
6. Fomin, F.V., Golovach, P.A., Lokshtanov, D., Saurabh, S., Zehavi, M.: Clique-width III: Hamiltonian cycle and the odd case of graph coloring. ACM Trans. Algorithms **15**(1), 9:1-9:27 (2019)
7. Ganian, R., Hlinený, P., Obdrzálek, J.: A unified approach to polynomial algorithms on graphs of bounded (bi-)rank-width. Eur. J. Comb. **34**(3), 680–701 (2013)
8. Lovász, L.: Combinatorial Problems and Exercises. North-Holland (1993)
9. Petrusevski, M.: Odd 4-edge-colorability of graphs. J. Graph Theory **87**(4), 460–474 (2018)
10. Petrusevski, M., Skrekovski, R.: Colorings with neighborhood parity condition (2021)
11. Scott, A.D.: On induced subgraphs with all degrees odd. Graphs Comb. **17**(3), 539–553 (2001)

Deciding the Erdős-Pósa Property in 3-Connected Digraphs

Julien Bensmail[1], Victor Campos[2], Ana Karolinna Maia[2], Nicolas Nisse[1](✉), and Ana Silva[2]

[1] Université Côte d'Azur, CNRS, Inria, I3S, Sophia Antipolis, France
{julien.bensmail,nicolas.nisse}@inria.fr
[2] ParGO, Universidade Federal do Ceará, Fortaleza, Brazil

Abstract. A (di)graph H has the Erdős-Pósa (EP) property for (butterfly) minors if there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that, for any $k \in \mathbb{N}$ and any (di)graph G, either G contains at least k pairwise vertex-disjoint copies of H as (butterfly) minor, or there exists a subset T of at most $f(k)$ vertices such that H is not a (butterfly) minor of $G - T$. It is a well known result of Robertson and Seymour that an undirected graph has the EP property if and only if it is planar. This result was transposed to digraphs by Amiri, Kawarabayashi, Kreutzer and Wollan, who proved that a strong digraph has the EP property for butterfly minors if, and only if, it is a butterfly minor of a cylindrical grid. Contrary to the undirected case where a graph is planar if, and only if, it is the minor of some grid, not all planar digraphs are butterfly minors of a cylindrical grid. In this work, we characterize the planar digraphs that have a butterfly model in a cylindrical grid. In particular, this leads to a linear-time algorithm that decides whether a weakly 3-connected strong digraph has the EP property.

Keywords: Erdős-Pósa property · Planar digraphs · Butterfly minor

1 Introduction

A classical result by Erdős and Pósa [5] states that there is a function $f : \mathbb{N} \to \mathbb{N}$ such that, for every k, every graph G contains either k pairwise vertex-disjoint cycles or a set T of at most $f(k)$ vertices such that $G - T$ is acyclic. The generalization of Erdős and Pósa's result for digraphs and directed cycles was conjectured by Younger [13] and proved by Reed et al. [7].

(Partially) supported by: FUNCAP MLC-0191-00056.01.00/22 and PNE-0112-00061.01.00/16, and CNPq 303803/2020-7, the CAPES-Cofecub project Ma 1004/23, by the project UCA JEDI (ANR-15-IDEX-01) and EUR DS4H Investments in the Future (ANR-17-EURE-004), the ANR Digraphs, the ANR Multimod and the Inria Associated Team CANOE. The full versions of omitted or sketched proofs can be found in [3].

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 59–71, 2023.
https://doi.org/10.1007/978-3-031-43380-1_5

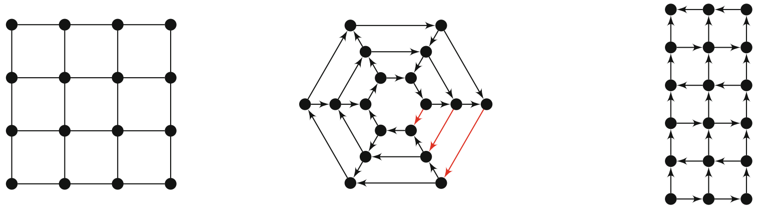


Fig. 1. The (4×4)-grid (left), the (3×6)-cylindrical grid $C_{3,6}$ (middle), and the directed wall (right) obtained from $C_{3,6}$ by removing the three red arcs. (Color figure online)

We say that H is a *minor* of G if H is obtained from a subgraph of G by a sequence of edge contractions. If H is a digraph and we restrict the contractions in the previous definition to *butterfly contractions* [6], we get the definition of a butterfly minor. We say that a graph H has the *Erdős-Pósa (EP) property for minors* if there is a function $f : \mathbb{N} \to \mathbb{N}$ such that, for every k, every graph G contains either k pairwise vertex-disjoint copies of H as a minor or a set T of at most $f(k)$ vertices such that H is not a minor of $G - T$. By changing graph into digraph and minor into butterfly minor, the previous definition can be adapted into the *EP property for butterfly minors in digraphs.* In this view, if H is the undirected graph with a unique vertex and a unique loop on it and D is the digraph obtained from H by orienting its loop edge, then Erdős and Pósa proved that H has the EP property for minors while Reed et al. proved that D has the EP property for butterfly minors.

The results of Erdős and Pósa and Reed et al. were generalized by Robertson and Seymour [8] for undirected graphs and by Amiri et al. [1] for digraphs. Robertson and Seymour [8] proved that an undirected graph G has the EP property for minors if, and only if, G is planar. Amiri et al. [1] proved that a strong digraph D has the EP property for butterfly minors if, and only if, D is a butterfly minor of a cylindrical grid (see Fig. 1). The results of Robertson and Seymour [8] and Amiri et al. [1] are similar since an undirected graph is planar if, and only if, it is a minor of some grid [9]. Contrary to the undirected case, not all planar digraphs are butterfly minors of a cylindrical grid. In this paper, we provide a structural characterization of planar digraphs that are butterfly minors of a cylindrical grid. In particular, such characterization leads to a linear-time algorithm that decides whether a weakly 3-connected strong digraph has the EP property for butterfly minors.

Although planarity is a necessary condition for a digraph to be a butterfly minor of a cylindrical grid, it is not sufficient. For example, the two planar digraphs of Fig. 2 are not butterfly minors of any cylindrical grid. To see this, first note that they are planar, weakly 3-connected, and have essentially a unique (up to the outerface) embedding in the plane, according to Whitney's Theorem [12]. Note also that, in a cylindrical grid, any embedding is such that there is a point in the plane around which all directed cycles go, and in the same direction. We refer to this as being concentric and with same orientation. Now, in the

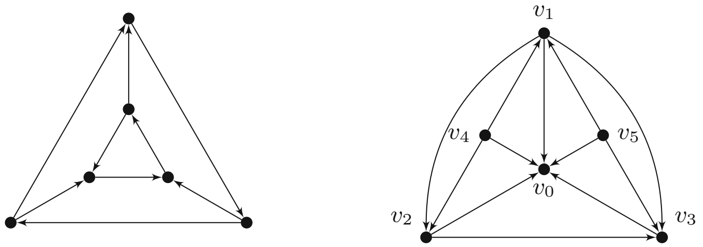


Fig. 2. Two planar digraphs L (left) and R (right) which are not butterfly minors of any cylindrical grid.

digraph L of Fig. 2, the matching between the two directed triangles forces that, in any planar embedding, either the two triangles are not concentric or they have opposite orientations. On the other hand, the digraph R of Fig. 2 is acyclic but it is not a minor of any cylindrical grid. To see why, note that, if R was a butterfly minor of a cylindrical grid, then, because R is acyclic, it would also be a butterfly minor of a *directed wall*, which is the digraph obtained by cutting a cylindrical grid along "parallel" arcs (see Fig. 1). Note that, in an embedding of a directed wall similar to the one given in Fig. 1, no arc goes down. This means that some relative positions of the vertices of R in a directed wall are forced. Namely, the two sources v_4 and v_5 of R must be below each of their out-neighbors, vertex v_1 must be below its three out-neighbors, and the universal sink v_0 must be above every other vertex. It can then be checked that these positions must lead to some crossing arcs. This second example shows that sources and sinks may play an important role in the fact that a planar digraph may or may not be a butterfly minor of a cylindrical grid. In a way, our main result tells that the above two examples fully characterize the reasons why a planar digraph cannot be a butterfly minor of a cylindrical grid.

To formally state our main result, we need a few definitions. Given a digraph $D = (V, A)$ and $\emptyset \neq X \subset V(D)$, the set of arcs between X and $V \backslash X$ is denoted by $(X, V \backslash X)$. We say that $(X, V \backslash X)$ is a *dicut* if there are no arcs from $V \backslash X$ to X. A *dijoin path* P of D is a directed path in D whose arc-set intersects the arc-set of every dicut of D. A *plane digraph* is a planar digraph together with a planar embedding. Recall also that, given a plane digraph H, H^* denotes its *dual*. That is, the dual digraph H^* of H (with a fixed planar embedding) is the digraph that has a vertex for each face of the embedding of H and H^* has an arc $e^* = \{u, v\}$ for each two faces u and v in the embedding of H that are separated from each other by an arc $e \in E$. Moreover, each dual arc e^* is oriented by a 90° clockwise turn from the corresponding primal arc e. For instance, if a face of a plane digraph H is "surrounded" by a directed cycle oriented clockwise (resp., counter-clockwise), then the corresponding vertex of H^* is a source (resp., a sink).

We can now state our main result.

Theorem 1. *A digraph D is a butterfly minor of a cylindrical grid if, and only if, D has a plane spanning supergraph H with neither sources nor sinks such that H^* admits a dijoin path.*

To get further intuition about Theorem 1, consider the definition of a *feedback arc set* $F \subseteq A$ of a digraph $D = (V, A)$, which is any subset of arcs such that $D - F$ is acyclic. Given a plane digraph D, it is known that every directed cycle of D is associated to a dicut in D^*. This implies that a set of arcs is a feedback arc set of D if, and only if, the corresponding set of its dual edges intersects the arc-set of every dicut of D^* [2]. Therefore, the fact that D^* admits a dijoin path means that such path intersects the arcs of a feedback arc set of D "in the same direction", i.e., intersects the drawing of each directed cycle of D, with each intersection occurring in the same orientation. This is equivalent to being concentric and with the same orientation. This condition (D^* admits a dijoin path) allows avoiding the kind of planar digraphs as exemplified by the digraph L in Fig. 2. In turn, the difficulties exemplified in digraph R in Fig. 2 are dealt with by the existence of a supergraph H with neither sources nor sinks.

Structure of the Paper and Algorithmic Applications. We first prove that if D is a plane digraph with neither sources nor sinks such that D^* has a dijoin path, then D is a butterfly minor of a cylindrical grid (Theorem 4). Observe that this gives us the sufficiency part of Theorem 1. We then show that if D is a butterfly minor of a cylindrical grid, then D has a planar embedding such that D^* admits a dijoin path (Theorem 5). Observe that D might still have sources and sinks, so the remainder of the proof consists in adding arcs to D in order kill all sources and sinks (Lemma 2).

Theorems 4 and 5 have the following important corollary:

Corollary 1. *Any digraph D without sources or sinks is a butterfly minor of a cylindrical grid if and only if D admits a planar embedding s.t. D^* has a dijoin path.*

Note that the planar digraph R in Fig. 2 is acyclic. So, whatever be its planar embedding, the dual is strongly connected, i.e., R^* has no dicuts. Therefore, every planar embedding of R is such that R^* has a trivial dijoin path (the empty path). Therefore, unfortunately, there is no hope that the condition on sources and sinks can be removed from Corollary 1.

Note that any *strongly connected digraph* (or *strong*) D satisfies the conditions of Corollary 1. Together with the result of Amiri et al. [1], this implies that:

Corollary 2. *Any strong digraph D has the EP property for butterfly minors if, and only if, D admits a planar embedding such that D^* has a dijoin path.*

By Whitney's Theorem [12], any weakly 3-connected planar digraph D has a unique (up to the outerface) planar embedding (computable in linear time). Since deciding whether the dual of a plane digraph admits a dijoin path can be done in linear time, then our result has the following algorithmic application:

Corollary 3. *Deciding whether a weakly* 3*-connected strong digraph has the EP property for butterfly minors can be done in linear time.*

Section 2 is devoted to defining the main notions and to present previously known results used in this paper. Section 3 is devoted to digraphs with neither sources nor sinks. Section 4 is devoted to obtaining the supergraph with neither sinks nor sources.

2 Preliminaries

Planar Digraphs and Duality. In this section, we present a number of simple known facts concerning planar graphs and their duals. The interested reader can find formal definitions and proofs for such facts in most books on graph theory (e.g., [11]).

Given a digraph $D = (V, A)$ and $e \in A$, let $D \backslash e = (V, A \backslash \{e\})$ and let D/e be the digraph obtained from D after contracting the arc e.

Observation 1. *Let $D = (V, A)$ be a plane digraph, and $e \in A$ be any arc of D. Then, $(D \backslash e)^* = D^*/e^*$ and $(D/e)^* = D^* \backslash e^*$.*

A *dicut* of a digraph $D = (V, A)$ is a partition $(X, V \backslash X)$ of the vertex-set such that X is a non empty proper subset of V and there are no arcs from $V \backslash X$ to X. The arc-set of $(X, V \backslash X)$ is the set of arcs from X to $(X, V \backslash X)$. A *dijoin* $X \subseteq A(D)$ of D is a set of arcs intersecting all dicuts' arc-sets of D. A *dijoin path* (resp., *dijoin walk*) of D is a dijoin inducing a directed path (resp., directed walk) in D. That is, a dijoin path/walk P of D is a directed path/walk whose arc-set intersects the arc-set of every dicut of D.

Observation 2. *A digraph D admits a dijoin path if, and only if, the decomposition of D into strongly connected components has a single source component and a single sink component.*

Observation 3. *Let $D = (V, A)$ be a digraph with a dijoin path P, and $e \in A \backslash A(P)$. Then, P is a dijoin path of $D \backslash e$.*

Observation 4. *Let $D = (V, A)$ be a digraph with a dijoin path P, and $e \in A$. Let P' be obtained from P by contracting e if $e \in A(P)$, and $P' = P$ otherwise. Then, P' is a dijoin walk of D/e.*

Observation 5. *Let $D = (V, A)$ be a digraph with a dijoin path P, and $v \in V$ be an isolated vertex. Then, P is a dijoin path of $D \backslash v$.*

Observation 6. *Every digraph with a dijoin walk admits a dijoin path.*

Butterfly Models and Cylindrical Grids. We now present the formal definition of butterfly models. Let G and H be two digraphs. A (*butterfly*) *model* of G in H is a function $\eta : V(G) \cup A(G) \to \mathcal{S}(H)$, where $\mathcal{S}(H)$ denotes the set of all subdigraphs of H, such that:

- for every $v \in V(G)$, $\eta(v)$ is a subdigraph of H being the orientation of some tree such that $V(\eta(v))$ can be partitioned into $(\{r_v\}, I_v, O_v)$ where
  - $\eta(v)[O_v \cup \{r_v\}]$ is an out-arborescence rooted in r_v (thus in which all non-root vertices have in-degree 1), called the *out-tree* of v,
  - $\eta(v)[I_v \cup \{r_v\}]$ is an in-arborescence rooted in r_v (thus in which all non-root vertices have out-degree 1), called the *in-tree* of v;
- for every two distinct $u, v \in V(G)$, $\eta(u)$ and $\eta(v)$ are vertex-disjoint;
- for every $(x, y) \in A(G)$, $\eta(xy)$ is a directed path of H from the out-tree of x to the in-tree of y, with internal vertices disjoint from every vertex of $\eta(u)$ for every $u \in V(G)$, and from every internal vertex of $\eta(uv)$ for every $(u, v) \in A(G) \backslash \{(x, y)\}$.

Throughout this work, given a model of G in H, we will refer to the arcs $e \in A(H) \cap \bigcup_{f \in A(G)} A(\eta(f))$ as the **blue arcs** of the model, and to the arcs $e \in A(H) \cap \bigcup_{v \in V(G)} A(\eta(v))$ as the **black arcs**. A vertex of H incident to at least one black arc will be referred to as a **black vertex**.

A model of G in H is *minimal* if, for every $v \in V(G)$ and for every leaf w of $\eta(v)$, w is incident to some blue arc. Note that, up to removing the leaves that do not satisfy this property from $\eta(v)$, we can always assume to be working on a minimal model.

Butterfly contracting an arc $(u, v) \in A(D)$ of some digraph D consists in contracting the arc (u, v) if $d^-(v) = 1$ or $d^+(u) = 1$. A digraph G is a *butterfly minor* of some digraph H if G can be obtained from H by deleting arcs, deleting vertices, and butterfly contracting arcs. Note that if G is a butterfly minor of H, then G can be obtained by first removing some arcs, then removing isolated vertices, and finally performing butterfly contractions.

Observation 7 [1]. *A digraph G is a butterfly minor of some digraph H if, and only if, G has a butterfly model in H.*

We now deal with cylindrical grids. Let $n, m \in \mathbb{N}^*$. The *cyclindrical grid* $C_{n,2m}$ can be seen as a set of n concentric directed cycles having the same direction and linked through $2m$ directed paths that alternate directions (see Fig. 3). Formally, $C_{n,2m}$ is the digraph with vertex-set $\{(i, j) \mid 0 \leq i < n, 0 \leq j < 2m\}$, and with the following arc-set. For every $0 \leq i < n$ and $0 \leq j < 2m$, we have $((i, j), (i, j+1 \mod m)) \in A(C_{n,2m})$, and the directed cycle induced by $\{(i, j) \mid 0 \leq j < m\}$ is called the i^{th} *column* of $C_{n,2m}$. For every $0 \leq i < n-1$ and $0 \leq j < m$, we have $((i, 2j), (i+1, 2j)) \in A(C_{n,2m})$ and $((i, 2j+1), (i-1, 2j+1)) \in A(C_{n,2m})$. Moreover, for every $0 \leq j < 2m$, the directed path induced by $\{(i, j) \mid 0 \leq i < n\}$ is called the j^{th} *row* of $C_{n,2m}$.

Throughout this work, we consider that any $C_{n,2m}$ is embedded in the plane so that its first column coincides with the outerface (see Fig. 3). Hence, we may naturally refer to left/right and top/bottom such that the first (last) column is the leftmost (rightmost) and the first (last) row is the bottommost (topmost). The arcs of a column are referred to as *vertical arcs*. Note that all vertical arcs are going up. The arcs of a row are the *horizontal arcs*. Moreover, the arcs of even (resp., odd) rows are horizontal to the right (resp., to the left).

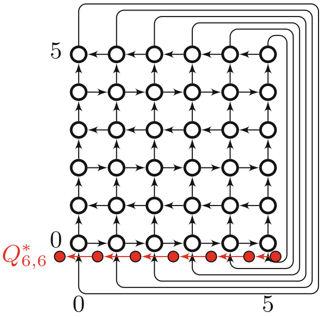
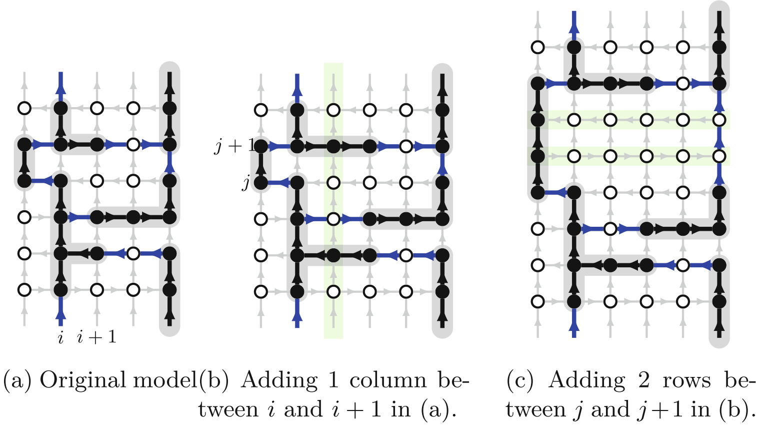


Fig. 3. A planar embedding of the cylindrical grid $C_{6,6}$. The red directed path $Q^*_{6,6}$ is the dijoin path defined and used in Sect. 4. (Color figure online)



(a) Original model

(b) Adding 1 column between i and $i+1$ in (a).

(c) Adding 2 rows between j and $j+1$ in (b).

Fig. 4. Green rows and columns are added. Blue arcs belong to the images of some arcs of G by η. Grey subtrees (with black vertices and arcs) are the images of some vertices of G by η. (Color figure online)

Since $C_{n,2m}$ is strong, we get that $C^*_{n,2m}$ is a DAG. Moreover, $C^*_{n,2m}$ has a unique sink t^*, corresponding to the outerface of the given embedding of $C_{n,2m}$, and a unique source s^*, corresponding to the face of $C_{n,2m}$ bounded by the last column of $C_{n,2m}$. Note that if P^* is any directed path from s^* to t^* in $C^*_{n,2m}$, then P^* is a dijoin path, i.e., it intersects all dicuts of $C^*_{n,2m}$ (or, equivalently, P^* "crosses" all directed cycles of $C_{n,2m}$).

Let η be a butterfly model of a digraph G in $C_{n,2m}$. We will deal with η through a few operations. Due to lack of space, we only present them informally.

- *Adding one column between columns i and $i+1$ in η* consists in considering the new model η' of G in the cylindrical grid $C_{n+1,2m}$ obtained as follows. Roughly, the left part of the model (between columns 0 to i) does not change, one new column is added (with abscissa $i+1$), and the right part of the model (between former columns $i+1$ to n) is translated by one column to the right.

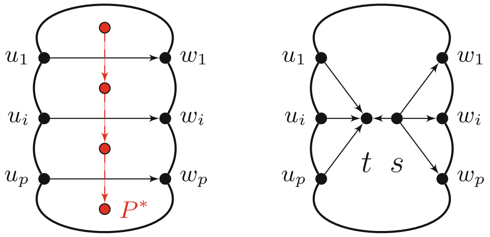


Fig. 5. Construction of $D_{P^*}(s,t)$. On the left, a dijoin path P^* is represented by dashed red arcs. On the right, the obtained digraph $D_{P^*}(s,t)$ is depicted. (Color figure online)

The horizontal arcs of the model that were going from former columns i to $i+1$ are subdivided once, i.e., they are now directed paths that go from column i to column $i+2$. Note that no vertical arcs of the added column belong to the new model η' of D. See Fig. 4 for an illustration.

- *Adding two rows between rows j and $j+1$ in η* consists in considering the new model η' of G in the cylindrical grid $C_{n,2(m+1)}$ defined as follows. All the elements of the model below row j, or in row j, remain the same, all elements of the model above row j are translated up from two rows, and all vertical arcs from former row j to former row $j+1$ are subdivided twice, i.e., they are now vertical directed paths with three arcs from row j to row $j+3$. Note that no horizontal arcs of the two added rows belong to the new model η' of D. See Fig. 4 for an illustration.

3 Digraphs with Neither Sources nor Sinks

Let D be a plane digraph such that D^* has a dijoin path P^* with arcs $(e_1^*, \cdots, e_p^*)$. Let $D_{P^*}(s,t)$ be obtained from D as follows (see Fig. 5 to follow the construction). For every $i \in \{1, \cdots, p\}$, let $e_i = (u_i, w_i)$ be the arc of D corresponding to e_i^*. Subdivide e_i into three arcs (u_i, t_i), (t_i, s_i), and (s_i, w_i). Then, remove (t_i, s_i), and, for every $i \in \{1, \cdots, p\}$, identify the vertices $t_1, \cdots, t_p$ into one vertex t, and the vertices $s_1, \cdots, s_p$ into one vertex s. Finally, add an arc from s to t. Note that $V(D_{P^*}(s,t)) = V(D) \cup \{s,t\}$ and, for every $v \in V(D)$, the in-degree (resp., out-degree) of v in D is the same as in $D_{P^*}(s,t)$. Since P^* is a dijoin path of D^*, the set $\{e_i\}_{i \leq m}$ is a feedback arc set of D [2]. Therefore:

Observation 8. *Let D be a plane digraph such that D^* has a dijoin path P^*. If D has neither sources nor sinks, then $D_{P^*}(s,t)$ is a planar DAG having s as unique source and t as unique sink.*

A *visibility representation* of a graph G is a mapping of $V(G)$ into non-intersecting horizontal segments[1] $\{h_u\}_{u \in V(G)}$, together with a mapping $\{t_e\}_{e \in E(G)}$ of the edges into vertical segments such that for every $uv \in E(G)$,

[1] Here, segment means line segment in the plane.

we get that t_{uv} has endpoints in h_u and h_v, and t_{uv} does not cross h_w for every $w \neq u, v$.

Theorem 2 ([4]). *Every planar graph admits a visibility representation.*

Here, we apply the approach presented in [10] to our context in order to obtain a butterfly model of a planar digraph D into a cylindrical grid, if one exists. For this, we slightly adapt their definitions to our purposes.

We consider a visibility representation $(\{h_u\}_{u \in V}, \{t_e\}_{e \in A})$ of $D = (V, A)$ to be drawn on the plane, and, given two horizontal (vertical) segments s_1, s_2, we write $s_1 \leq s_2$ if the y-coordinate (x-coordinate) of s_1 is smaller than the one of s_2. Now, given a DAG $D = (V, A)$, we say that a visibility representation $(\{h_u\}_{u \in V}, \{t_e\}_{e \in A})$ of D is *increasing* if $h_u \leq h_v$ for every arc $(u, v) \in A$ (in other words, the arcs are all directed upwards).

In [10], in order to construct a visibility representation, the authors show that they can obtain an orientation D of a graph G that is acyclic, has exactly one source s and exactly one sink t, and $(s, t) \in A(D)$ (they call such a digraph a *PERT-digraph*). After they obtain this orientation, they use a total order $(v_1, \ldots, v_n)$ of $V(G)$ that meets the orientation D, and then construct a visibility representation such that $s_1 < s_2 < \ldots < s_n$, where s_i denotes the y-coordinate of h_{v_i}, for every $i \in \{1, \ldots, n\}$. Observe that, because the order meets the orientation, we get that this is an increasing visibility representation. Their representation also has the property that the x-coordinate of arc (s, t) is smaller than the x-coordinate of every other edge of G. In short, even though they use a different terminology, the results presented in [10] actually show that the theorem below holds. The interested reader can check this is true by observing, in their algorithm W-VISIBILITY, that after they obtain the desired orientation D (line 2), they only work on D itself; also, the increasing order over the y-coordinates is ensured in line 5.1 of their algorithm.

Theorem 3 ([10]). *Let D be a planar DAG with unique source s and unique sink t, and such that $(s, t) \in A(D)$. Then, D admits an increasing visibility representation such that each horizontal segment has a distinct y-coordinate, and the x-coordinate of the segment of (s, t) is smaller than the x-coordinate of the segment of every other arc of D.*

Theorem 4. *Let $D = (V, A)$ be a digraph without sources or sinks. If D has a planar embedding such that D^* admits a dijoin path P^*, then D has a butterfly model in $C_{n,2m}$ for some $n, m \in \mathbb{N}^*$.*

Sketch of the Proof. By Observation 8, $D_{P^*}(s, t)$ is a DAG with a unique source s, a unique sink t, and $(s, t) \in A(D_{P^*}(s, t))$. By Theorem 3, there exists an increasing visibility representation of $D_{P^*}(s, t)$. Let $V(D_{P^*}(s, t)) = \{s = v_1, \ldots, t = v_n\}$ be ordered increasingly according to their y-coordinates on the representation and suppose, without loss of generality, that the y-coordinate of $h_{v_1} = h_s$ is 0 and the difference between the y-coordinates of h_{v_i} and $h_{v_{i-1}}$ is 2 (their value on the constructed increasing visibility representation are all different, so we just need to adjust it). Observe that, in this case, the y-coordinate of h_{v_i} is $2i-2$. We will build a model of D in some cylindrical grid as follows.

For each $v_i \in V(D)$ (hence $i \notin \{1, n\}$), let h'_{v_i} be the segment equivalent to h_{v_i}, but in the upper row. In other words, h'_{v_i} has y-coordinate $2i-1$, and leftmost and rightmost x-coordinates equal to the ones of h_{v_i}. The idea of the proof is to relate v_i with the path formed by the union of paths associated to h'_{v_i} and h_{v_i} in the cylindrical grid. Since all arcs of D point upwards, we get that the subpath associated to h'_{v_i} (i.e., in row $2i-1$) corresponds to the out-tree of v_i, while the subpath associated to h_{v_i} (i.e., in row $2i-2$) corresponds to the in-tree of v_i. □

Note that Theorem 4 allows us to prove the "if" part of Theorem 1.

Theorem 5. *If a digraph $D = (V, A)$ has a butterfly model in $C_{n,2m}$ for some $n, m \in \mathbb{N}^*$, then D has a planar embedding such that D^* admits a dijoin path.*

Proof. Consider the planar embedding of $C_{n,2m}$ such that the outerface contains its first column (see Fig. 3) and let P^* be any directed path from the single source of $C^*_{n,2m}$ to its single sink. Note that P^* is a dijoin path of $C^*_{n,2m}$.

By Observation 7, D is a butterfly minor of $C_{n,2m}$. Let $s_1, \cdots, s_q$ be the sequence of operations allowing to get D from $C_{n,2m}$ where these operations are ordered in such a way that first arcs are removed, then isolated vertices are removed and, finally, butterfly contractions are performed. For every $0 \leq i \leq q$, let G_i be the digraph obtained after the i^{th} operation (so $G_0 = C_{n,2m}$ and $G_q = D$). We show, by induction on $0 \leq i \leq q$, how to obtain a directed path P_i which is a dijoin path of G^*_i. In particular, it holds for $i = 0$ by taking $P_0 = P^*$.

Let $i \geq 1$. If s_i consists in removing an arc e_i of G_{i-1} then, if $e^*_i \in A(P_{i-1})$, let $P'_i = P_{i-1}/e^*_i$, and let $P'_i = P_{i-1}$ otherwise. By Observations 1 and 4, P'_i is a dijoin walk of G^*_i and, by Observation 6, G^*_i admits a dijoin path P_i. If s_i consists in removing an isolated vertex, then, by Observation 5, $P_i = P_{i-1}$ is a dijoin path of G^*_i. And if s_i is a butterfly contraction of the arc $e_i \in A(G_{i-1})$, where $e^*_i \notin A(P_{i-1})$, then Observations 1 and 3 ensure us that $P_i = P_{i-1}$ is a dijoin path of G^*_i.

Finally, let us consider the case when s_i consists in butterfly contracting an arc $e_i = (u, v) \in A(G_{i-1})$ such that $e^*_i \in A(P_{i-1})$. Let us assume that $d^-(v) = 1$ (the case when $d^+(u) = 1$ is symmetric). Observe that $d^+(v) > 0$ as otherwise e^*_i would be a loop, contradicting that P_{i-1} is a directed path. Then, let $\{f_1, \cdots, f_q\}$ be the set of out-arcs of v ordered clockwise in the embedding of D in the plane. Then, let P'_i be the directed walk obtained by replacing e^*_i in P_{i-1} by the directed walk consisting of the arcs $f^*_1, f^*_2, \cdots, f^*_q$. Note that P'_i is a dijoin walk in G^*_i. Indeed, consider the set of arcs K of a dicut of G^*_i. If K is also a dicut in G^*_{i-1}, then $e^*_i \notin K$ and P'_i intersects K since P_{i-1} is a dijoin path and so intersects K. Otherwise, $e^*_i \in K$ which implies that $\{f^*_1, f^*_2, \cdots, f^*_q\} \cap K \neq \emptyset$, and so P'_i intersects K. Finally, by Observation 6, G^*_i admits a dijoin path P_i. □

Theorems 4 and 5 prove the following corollary which corresponds to Theorem 1 in the case of digraphs with neither sources nor sinks (and in particular, Corollary 3 is a special case of the following corollary).

Corollary 4. *A digraph D without sources or sinks is a butterfly minor of a cylindrical grid if, and only if, D admits a planar embedding such that D^* has a dijoin path. Moreover, if D is weakly 3-connected, then this can be decided in linear time.*

Proof. Due to the weakly 3-connectivity of D, and by Whitney's Theorem, D has a unique (up to the outerface) planar embedding (and a unique dual). Given such an embedding, checking the existence of a dijoin path can be done in linear time by Observation 2. □

Recall that a strong digraph D has the EP property for butterfly minors if and only if D is a butterfly minor of a cylindrical grid [1]. Together with our result, we get:

Corollary 3. *Deciding whether a weakly 3-connected strong digraph D has the EP property for butterfly minors can be done in linear time.*

4 Digraphs with Sources and Sinks

We have seen that if D is a butterfly minor of a cylindrical grid, then D has a planar embedding such that D^* admits a dijoin path (Theorem 5). As we want to show that this holds for a *spanning supergraph with neither sources nor sinks*, it remains to "kill" sources and sinks in D. This is done in this section.

Given a cylindrical grid $C_{n,2m}$ with the canonical planar embedding described previously, let $Q^*_{n,2m}$ be the directed path of the dual $C^*_{n,2m}$ whose arcs correspond exactly to all arcs of $C_{n,2m}$ that go from the last (topmost) row to the first (bottommost) row. Note that $Q^*_{n,2m}$ is a dijoin path of $C^*_{n,2m}$ (see Fig. 3). The next lemma states that if a digraph D has a model in a cylindrical grid, then it is possible to get a model such that $Q^*_{n,2m}$ only crosses blue arcs of this model.

Lemma 1. *If a digraph D has a butterfly model η in $C_{n,2m}$, then D has a butterfly model in $C_{n',2m'}$ for some $n' \geq n$ and $m' \geq m$ such that no black arcs of this model are dual of an arc of $Q^*_{n',2m'}$.*

Lemma 2. *If a digraph $D = (V, A)$ has a butterfly model in $C_{n,2m}$, then D has a spanning supergraph with neither sources nor sinks that has a butterfly model in $C_{n',2m'}$ for some $n' \geq n$ and $m' \geq m$.*

Sketch of the Proof. If D has no sources nor sinks, then we are done, so suppose otherwise. In what follows, given a source s in D (resp., a sink t), we describe a process that builds a model for an in-arc that we add to s (resp., an out-arc that we add to t), so that the obtained supergraph has also a model in a cylindrical grid and has one less source (resp., sink). By iteratively applying such process, we get the desired conclusions.

Let us consider a butterfly model η of D with a dijoin path P^* as in Lemma 1, and suppose that D has a source s (the case of a sink is symmetric). We will add a new arc (z, s) for some $z \in V(D)$, and a directed path Q (finishing in the in-tree of the model of s) for modelling this arc in the existing model η. The difficulty is to find the vertex of an out-tree in which we can start Q from. First let us add two columns between any two consecutive columns and let us add two rows between any two consecutive rows. We also add one column to the left and one column to the right of the cylindrical grid.

Let r_s be the root of $\eta(s)$. Note that, by assuming η to be minimal, we get that the in-tree of the model of s in η is reduced to its root. Let a_1 be the vertex below r_s and b_1 the vertex below a_1. Since we have just added two rows between any two rows, and because the in-tree of s is reduced to r_s, we get that a_1, b_1 are not part of the model of any vertex nor arc. Let Q initially contain just the arc (a_1, r_s) (this will actually be the last arc of Q). Let us assume that Q has been built up to some vertex a_h, i.e., $Q = (a_h, a_{h-1}, \cdots, a_1, r_s)$, and additionally assume that the vertex b_h below a_h is not part of the model of any vertex nor arc (this is the case for $h = 1$). Let w be the vertex below b_h.

- If w is not part of the model of any vertex nor arc, then let $a_{h+1} = b_h$ and let $b_{h+1} = w$, and we continue to build Q.
- If w is in the out-tree of some vertex, then add $(w, b_h), (b_h, a_h)$ to the end of Q to be done.
- If w is part of the in-tree of some vertex a (and not of its out-tree, i.e., w is not the root of the model of a), then assume that the row of a_h goes to the right (the other case is symmetric). Let x be the left neighbor of a_h, y be the vertex below x (and to the left of b_h), and z the vertex below y (and to the left of w). Note that, since w is part of the model and b_h is not, then the rows of b_h and a_h are rows that have been added just before starting the process. In particular, this implies that either both x and y belong to the model of some $e \in V(D) \cup A(D)$, or neither x nor y is part of any model. We can then prove that the former case is not possible because it would contradict the fact that w is part of the in-tree (and not of the out-tree) of the model of a. In the latter case, we set $a_{h+1} = x$ and $b_{h+1} = y$ and go on.
- If w is part of the model of some arc $e = (u, v) \in A(D)$. We apply similar arguments and omit the proof because of space constraints.

The above process is not ensured to finish because if might happen that vertices a_h and b_h are already on the outerface of the model η. The next two cases allow to ensure that our process will actually terminate. For this purpose, we use the dijoin path P^*.

- If (b_h, a_h) crosses the dijoin path P^* and P^* does not cross any blue arc, then let us consider the closest row under P^* that contains a vertex of the model. W.l.o.g., let us assume that this row goes to the right and let x be the rightmost vertex of this row that is part of the model of some vertex v^* of D. By minimality of the model and because x has no out-neigbour in the model of any vertex or arc, then, x must be the root of $\eta(v^*)$ (which is actually an in-tree). Now, we add to $\eta(v^*)$: the up-going arc (x, y) (y being the up-neighbor of x), and the horizontal directed path Y starting from y to the leftmost vertex of this row (then y becomes the new root of $\eta(v^*)$ and the path Y will be considered as its out-tree). To conclude this case, add at the beginning of Q, the directed path from the path Y of $\eta(v^*)$ (added in previous paragraph) to b_h.

- If (b_h, a_h) crosses P^* which crosses some blue arc, then we apply similar arguments and omit the proof because of space constraints. □

Further Work. An interesting question is whether there exists a structural condition on the sources and sinks of a digraph D that corresponds to being a butterfly minor of a cylindrical grid (avoiding to invoke a supergraph without sources or sinks). This may help to answer the question of the computational complexity of deciding if a strong digraph D has the EP property when D is not weakly 3-connected. Since the class of digraphs that are butterfly minors of a cylindrical grid is closed under taking butterfly minors, it would also be interesting to characterize the minimal forbidden butterfly minors for this class.

References

1. Amiri, A., Kawarabayashi, K., Kreutzer, S., Wollan, P.: The Erdos-Pósa property for directed graphs. arXiv preprint arXiv:1603.02504 (2016)
2. Bang-Jensen, J., Gutin, G.: Digraphs: Theory, Algorithms and Applications, 2nd edn. Springer, London (2008). https://doi.org/10.1007/978-1-84800-998-1
3. Bensmail, J., Campos, V., Maia, A.K., Nisse, N., Silva, A.: Deciding the Erdös-Pósa property in 3-connected digraphs (2023). www.inria.hal.science/hal-04084227
4. Duchet, P., Hamidoune, Y., Las Vergnas, M., Meyniel, H.: Representing a planar graph by vertical lines joining different levels. Discrete Math. **46**, 319–321 (1983)
5. Erdős, P., Pósa, L.: On independent circuits contained in a graph. Can. J. Math. **17**, 347–352 (1965)
6. Johnson, T., Robertson, N., Seymour, P., Thomas, R.: Directed tree-width. J. Comb. Theory Ser. B **82**(1), 138–154 (2001)
7. Reed, B., Robertson, N., Seymour, P., Thomas, R.: Packing directed circuits. Combinatorica **16**(4), 535–554 (1996). https://doi.org/10.1007/BF01271272
8. Robertson, N., Seymour, P.: Graph minors. V. Excluding a planar graph. J. Comb. Theory Ser. B **41**, 92–114 (1986)
9. Robertson, N., Seymour, P.: Graph minors. VII. Disjoint paths on a surface. J. Comb. Theory Ser. B **45**(2), 212–254 (1988)
10. Tamassia, R., Tollis, I.G.: A unified approach to visibility representations of planar graphs. Discrete Comput. Geom. **1**, 321–341 (1986). https://doi.org/10.1007/BF02187705
11. West, D.B.: Introduction to Graph Theory, 2 edn. Prentice Hall (2000)
12. Whitney, H.: 2-isomorphic graphs. Am. J. Math. **55**(1), 245–254 (1933)
13. Younger, D.: Graphs with interlinked directed circuits. In: Proceedings of the Mid-West Symposium on Circuit Theory, vol. 2 (1973)

New Width Parameters for Independent Set: One-Sided-Mim-Width and Neighbor-Depth

Benjamin Bergougnoux[1], Tuukka Korhonen[2](✉), and Igor Razgon[3]

[1] University of Warsaw, Warsaw, Poland
benjamin.bergougnoux@mimuw.edu.pl
[2] University of Bergen, Bergen, Norway
tuukka.korhonen@uib.no
[3] Birkbeck University of London, London, UK
i.razgon@bbk.ac.uk

Abstract. We study the tractability of the maximum independent set problem from the viewpoint of graph width parameters, with the goal of defining a width parameter that is as general as possible and allows to solve independent set in polynomial-time on graphs where the parameter is bounded. We introduce two new graph width parameters: one-sided maximum induced matching-width (o-mim-width) and neighbor-depth. O-mim-width is a graph parameter that is more general than the known parameters mim-width and tree-independence number, and we show that independent set and feedback vertex set can be solved in polynomial-time given a decomposition with bounded o-mim-width. O-mim-width is the first width parameter that gives a common generalization of chordal graphs and graphs of bounded clique-width in terms of tractability of these problems.

The parameter o-mim-width, as well as the related parameters mim-width and sim-width, have the limitation that no algorithms are known to compute bounded-width decompositions in polynomial-time. To partially resolve this limitation, we introduce the parameter neighbor-depth. We show that given a graph of neighbor-depth k, independent set can be solved in time $n^{O(k)}$ even without knowing a corresponding decomposition. We also show that neighbor-depth is bounded by a polylogarithmic function on the number of vertices on large classes of graphs, including graphs of bounded o-mim-width, and more generally graphs of bounded sim-width, giving a quasipolynomial-time algorithm for independent set on these graph classes. This resolves an open problem asked by Kang, Kwon, Strømme, and Telle [TCS 2017].

Keywords: Graph width parameters · Mim-width · Sim-width · Independent set

Due to space limits, most of technicals details are omitted or just sketched. The full version of the paper is available on arXiv [2]. Tuukka Korhonen was supported by the Research Council of Norway via the project BWCA (grant no. 314528).

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 72–85, 2023.
https://doi.org/10.1007/978-3-031-43380-1_6

1 Introduction

Graph width parameters have been successful tools for dealing with the intractability of NP-hard problems over the last decades. While tree-width [25] is the most prominent width parameter due to its numerous algorithmic and structural properties, only sparse graphs can have bounded tree-width. To capture the tractability of many NP-hard problems on well-structured dense graphs, several graph width parameters, including clique-width [7], mim-width [26], Boolean-width [6], tree-independence number [9,27], minor-matching hypertree width [27], and sim-width [20] have been defined. A graph parameter can be considered to be more general than another parameter if it is bounded whenever the other parameter is bounded. For a particular graph problem, it is natural to look for the most general width parameter so that the problem is tractable on graphs where this parameter is bounded. In this paper, we focus on the maximum independent set problem (Independent Set).

Let us recall the standard definitions on branch decompositions. Let V be a finite set and $\mathtt{f} : 2^V \to \mathbb{Z}_{\geq 0}$ a symmetric set function, i.e., for all $X \subseteq V$ it holds that $\mathtt{f}(X) = \mathtt{f}(V \setminus X)$. A branch decomposition of $\mathtt{f}$ is a pair (T, δ), where T is a cubic tree and δ is a bijection mapping the elements of V to the leaves of T. Each edge e of T naturally induces a partition (X_e, Y_e) of the leaves of T into two non-empty sets, which gives a partition $(\delta^{-1}(X_e), \delta^{-1}(Y_e))$ of V. We say that the width of the edge e is $\mathtt{f}(e) = \mathtt{f}(\delta^{-1}(X_e)) = \mathtt{f}(\delta^{-1}(Y_e))$, the width of the branch decomposition (T, δ) is the maximum width of its edges, and the branchwidth of the function $\mathtt{f}$ is the minimum width of a branch decomposition of $\mathtt{f}$. When G is a graph and $\mathtt{f} : 2^{V(G)} \to \mathbb{Z}_{\geq 0}$ is a symmetric set function on $V(G)$, we say that the $\mathtt{f}$-*width* of G is the branchwidth of $\mathtt{f}$.

Vatshelle [26] defined the maximum induced matching-width (mim-width) of a graph to be the $\mathtt{mim}$-width where $\mathtt{mim}(A)$ for a set of vertices A is defined to be the size of a maximum induced matching in the bipartite graph $G[A, \overline{A}]$ given by edges between A and $\overline{A}$, where $\overline{A} = V(G) \setminus A$. He showed that given a graph together with a branch decomposition of mim-width k, any locally checkable vertex subset and vertex partitioning problem (LC-VSVP), including Independent Set, Dominating Set, and Graph Coloring with a constant number of colors, can be solved in time $n^{\mathcal{O}(k)}$. Mim-width has gained a lot of attention recently [1,4,5,17–19,22]. While mim-width is more general than clique-width and bounded mim-width captures many graph classes with unbounded clique-width (e.g. interval graphs), there are many interesting graph classes with unbounded mim-width where Independent Set is known to be solvable in polynomial-time. Most notably, chordal graphs, and even their subclass split graphs, have unbounded mim-width, but it is a classical result of Gavril [15] that Independent Set can be solved in polynomial-time on them. More generally, all width parameters in a general class of parameters that contains mim-width and was studied by Eiben, Ganian, Hamm, Jaffke, and Kwon [11] are unbounded on split graphs.

With the goal of providing a generalization of mim-width that is bounded on chordal graphs, Kang, Kwon, Strømme, and Telle [20] defined the parameter *special induced matching-width* (sim-width). Sim-width of a graph G is the $\mathtt{sim}$-width where $\mathtt{sim}(A)$ for a set of vertices A is defined to be the maximum size of an induced matching in G whose every edge has one endpoint in A and another in $\overline{A}$. The key difference of $\mathtt{mim}$ and $\mathtt{sim}$ is that $\mathtt{mim}$ ignores the edges in $G[A]$ and $G[\overline{A}]$ when determining if the matching is induced, while $\mathtt{sim}$ takes them into account, and therefore the sim-width of a graph is always at most its mim-width. Chordal graphs have sim-width at most one [20]. However, it is not known if Independent Set can be solved in polynomial-time on graphs of bounded sim-width, and indeed Kang, Kwon, Strømme, and Telle asked as an open question if Independent Set is NP-complete on graphs of bounded sim-width [20].

In this paper, we introduce a width parameter that for the Independent Set problem, captures the best of both worlds of mim-width and sim-width. Our parameter is inspired by a parameter introduced by Razgon [24] for classifying the OBDD size of monotone 2-CNFs. For a set of vertices A, let $E(A)$ denote the edges of the induced subgraph $G[A]$. For a set $A \subseteq V(G)$, we define the upper-induced matching number $\mathtt{umim}(A)$ of A to be the maximum size of an induced matching in $G - E(\overline{A})$ whose every edge has one endpoint in A and another in $\overline{A}$. Then, we define the *one-sided maximum induced matching-width* (o-mim-width) of a graph to be the $\mathtt{omim}$-width where $\mathtt{omim}(A) = \min(\mathtt{umim}(A), \mathtt{umim}(\overline{A}))$. In particular, o-mim-width is like sim-width, but we ignore the edges on one side of the cut when determining if a matching is induced. Clearly, the o-mim-width of a graph is between its mim-width and sim-width. Our first result is that the polynomial-time solvability of Independent Set on graphs of bounded mim-width generalizes to bounded o-mim-width. Moreover, we show that the interest of o-mim-width is not limited to Independent Set by proving that the Feedback Vertex Set problem is also solvable in polynomial time on graphs of bounded o-mim-width.

Theorem 1. *Given an n-vertex graph together with a branch decomposition of o-mim-width k,* Independent Set *and* Feedback Vertex Set *can be solved in time $n^{\mathcal{O}(k)}$.*

We also show that o-mim-width is bounded on chordal graphs. In fact, we show a stronger result that o-mim-width of any graph is at most its *tree-independence number* ($\mathsf{tree}\text{-}\alpha$), which is a graph width parameter defined by Dallard, Milanič, and Štorgel [9] and independently by Yolov [27], and is known to be at most one on chordal graphs.

Theorem 2. *Any graph with tree-independence number k has o-mim-width at most k.*

We do not know if there is a polynomial-time algorithm to compute a branch decomposition of bounded o-mim-width if one exists, and the corresponding question is notoriously open also for both mim-width and sim-width. Because of this, it is also open whether Independent Set can be solved in polynomial-time on graphs of bounded mim-width, and more generally on graphs of bounded o-mim-width.

In our second contribution we partially resolve the issue of not having algorithms for computing branch decompositions with bounded mim-width, o-mim-width, or sim-width. We introduce a graph parameter *neighbor-depth.*

Definition 3. *The neighbor-depth (*$\mathtt{nd}$*) of a graph* G *is defined recursively as follows:*

1. $\mathtt{nd}(G) = 0$ *if and only if* $V(G) = \emptyset$*,*
2. *if* G *is not connected, then* $\mathtt{nd}(G)$ *is the maximum value of* $\mathtt{nd}(G[C])$ *where* $C \subseteq V(G)$ *is a connected component of* G*,*
3. *if* $V(G)$ *is non-empty and* G *is connected, then* $\mathtt{nd}(G) \leq k$ *if and only if there exists a vertex* $v \in V(G)$ *such that* $\mathtt{nd}(G \setminus N[v]) \leq k - 1$ *and* $\mathtt{nd}(G \setminus \{v\}) \leq k$*.*

In the case (3) of Definition 3, we call the vertex v *the* pivot-vertex *witnessing* $\mathtt{nd}(G) \leq k$*.*

By induction, the neighbor-depth of all graphs is well-defined. We show that neighbor-depth can be computed in $n^{\mathcal{O}(k)}$ time and also Independent Set can be solved in time $n^{\mathcal{O}(k)}$ on graphs of neighbor-depth k.

Theorem 4. *There is an algorithm that given a graph* G *of neighbor-depth* k*, determines its neighbor-depth and solves* Independent Set *in time* $n^{\mathcal{O}(k)}$*.*

We show that graphs of bounded sim-width have neighbor-depth bounded by a polylogarithmic function on the number of vertices.

Theorem 5. *Any* n*-vertex graph of sim-width* k *has neighbor-depth* $\mathcal{O}(k \log^2 n)$*.*

Theorems 4 and 5 combined show that Independent Set can be solved in time $n^{\mathcal{O}(k \log^2 n)}$ on graphs of sim-width k, which in particular is quasipolynomial time for fixed k. This resolves, under the mild assumption that $\mathsf{NP} \not\subseteq \mathsf{QP}$, the question of Kang, Kwon, Strømme, and Telle, who asked if Independent Set is NP-complete on graphs of bounded sim-width [20, Question 2].

Neighbor-depth characterizes branching algorithms for Independent Set in the following sense. We say that an independent set branching tree of a graph G is a binary tree whose every node is labeled with an induced subgraph of G, so that (1) the root is labeled with G, (2) every leaf is labeled with the empty graph, and (3) if a non-leaf node is labeled with a graph $G[X]$, then either (a) its children are labeled with the graphs $G[L]$ and $G[R]$ where (L, R) is a partition of X with no edges between L and R, or (b) its children are labeled with the graphs $G[X \setminus N[v]]$ and $G[X \setminus \{v\}]$ for some vertex $v \in X$. Note that such a tree corresponds naturally to a branching approach for Independent Set, where we branch on a single vertex and solve connected components independently of each other. Let $\beta(G)$ denote the smallest number of nodes in an independent set branching tree of a graph G. Neighbor-depth gives both lower- and upper-bounds for $\beta(G)$.

Theorem 6. *For all graphs* G*, it holds that* $2^{\mathtt{nd}(G)} \leq \beta(G) \leq n^{\mathcal{O}(\mathtt{nd}(G))}$*.*

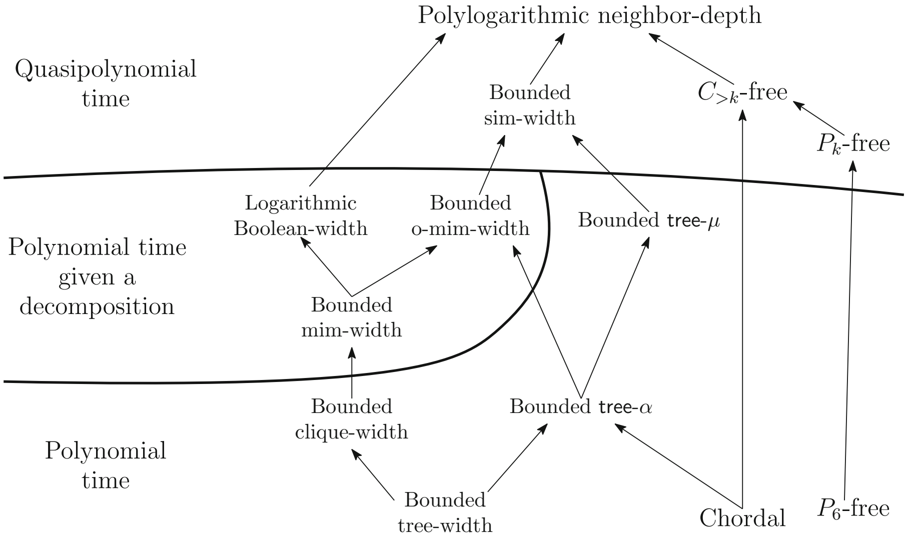


Fig. 1. Hierarchy of some graph classes with polylogarithmically bounded neighbor-depth, divided vertically on whether the best known algorithm for Independent Set on the class is polynomial time, polynomial time given a decomposition (and quasipolynomial without a decomposition), or quasipolynomial time.

By observing that some known algorithms for Independent Set in fact construct independent set branching trees implicitly, we obtain upper bounds for neighbor-depth on some graph classes purely by combining the running times of such algorithms with Theorem 6. In particular, for an integer k, we say that a graph is $C_{>k}$-free if it does not contain induced cycles of length more than k. Gartland, Lokshtanov, Pilipczuk, Pilipczuk and Rzazewski [14] showed that Independent Set can be solved in time $n^{\mathcal{O}(\log^3 n)}$ on $C_{>k}$-free graphs for any fixed k, generalizing a result of Gartland and Lokshtanov on P_k-free graphs [13]. By observing that their algorithm is a branching algorithm that (implicitly) constructs an independent set branching tree, it follows from Theorem 6 that the neighbor-depth of $C_{>k}$-free graphs is bounded by a polylogarithmic function on the number of vertices.

Proposition 7. *For every fixed integer k, $C_{>k}$-free graphs with n vertices have neighbor-depth at most $\mathcal{O}(\log^4 n)$.*

Along the same lines as Proposition 7, a polylogarithmic upper bound for neighbor-depth can be also given for graphs with bounded induced cycle packing number, using the quasipolynomial algorithm of Bonamy, Bonnet, Déprés, Esperet, Geniet, Hilaire, Thomassé, and Wesolek [3].

In Fig. 1 we show the hierarchy of inclusions between some of the graph classes discussed in this paper, and the known algorithmic results for Independent Set on those classes. All the inclusions shown are proper, and all the inclusions between these classes appear in the figure. Some of the inclusions are proven in Sects. 3 and 4, and some of the non-inclusions in the full version of the paper [2]. Note that bounded Boolean-width is equivalent to bounded clique-width [26]. The polynomial-time algorithm for Independent Set on P_6-free graphs is from [16], the definition of tree-μ and polynomial-time algorithm for Independent Set on graphs of bounded tree-μ is from [27], and the definition of Boolean-width and a polynomial-time algorithm for Independent Set on graphs of logarithmic Boolean-width is from [6]. The inclusion of logarithmic Boolean-width in polylogarithmic neighbor-depth follows from Theorem 5 and the fact the sim-width of a graph is at most its Boolean-width. Polynomial-time algorithm for Independent Set on graphs of bounded clique-width follows from [8,23].

Organization of this Paper. We prove the part of Theorem 1 on Independent Set and Theorem 2 in Sect. 3. Theorem 5 is proved in Sect. 4. Proofs omitted in this version of the paper due to space constraints are provided in the full version in [2].

2 Preliminaries

The size of a set V is denoted by $|V|$ and its power set is denoted by 2^V. We let $\max(\emptyset) := -\infty$. Our graph terminology is standard and we refer to [10].

The subgraph of G induced by a subset X of its vertex set is denoted by $G[X]$. We also use the notation $G \setminus X = G[V(G) \setminus X]$. For two disjoint subsets of vertices X and Y of $V(G)$, we denote by $G[X, Y]$ the bipartite graph with vertex set $X \cup Y$ and edge set $\{xy \in E(G) : x \in X \text{ and } y \in Y\}$. Given two disjoint set of vertices X, Y, we denote by $E(X)$ the set of edges of $G[X]$ and by $E(X, Y)$ the set of edges of $G[X, Y]$. For a set of edges E' of G, we denote by $G - E'$ the graph with vertex set $V(G)$ and edge set $E(G) \setminus E'$.

An *independent set* is a set of vertices that induces an edgeless graph. Given a graph G with a weight function $w : V(G) \to \mathbb{Z}_{\geq 0}$, the problem Independent Set asks for an independent set of maximum weight, where the weight of a set $X \subseteq V(G)$ is $\sum_{x \in X} w(x)$. A *feedback vertex set* is the complement of a set of vertices inducing a forest (i.e. acyclic graph). The problem Feedback Vertex Set asks for a feedback vertex set of minimum weight.

A *matching* in a graph G is a set $M \subseteq E(G)$ of edges having no common endpoint. We denote by $V(M)$ the set of vertices incident to M. An *induced matching* is a matching M such that $G[V(M)]$ does not contain any other edges than M. Given two disjoint subsets A, B of $V(G)$, we say that a matching M is a (A, B)-matching if every edge of M has one endpoint in A and the other in B.

Width Parameters. We refer to the introduction for the definitions of branch-decomposition and f-width, we recall below the definitions of mim-width, sim-width and o-mim-width.

- The maximum induced matching-width (mim-width) [26] of a graph G is the $\mathtt{mim}$-width where $\mathtt{mim}(A)$ is the size of a maximum induced matching of the graph $G[A, \overline{A}]$.
- The special induced matching-width (sim-width) [20] of a graph G is the $\mathtt{sim}$-width where $\mathtt{sim}(A)$ is the size of maximum induced $(A, \overline{A})$-matching in the graph G.
- Given a graph G and $A \subseteq V(G)$, the *upper-mim-width* $\mathtt{umim}(A)$ of A is the size of maximum induced $(A, \overline{A})$-matching in the graph $G - E(\overline{A})$. The one-sided maximum induced matching-width (o-mim-width) of G is the $\mathtt{omim}$-width where $\mathtt{omim}(A) := \min(\mathtt{umim}(A), \mathtt{umim}(\overline{A}))$.

The following is a standard lemma that $\mathtt{f}$-width at most k implies balanced cuts with $\mathtt{f}$-width at most k.

Lemma 8. *Let G be a graph, $X \subseteq V(G)$ a set of vertices with $|X| \geq 2$, and $\mathtt{f} : 2^{V(G)} \to \mathbb{Z}_{\geq 0}$ a symmetric set function. If the $\mathtt{f}$-width of G is at most k, then there exists a bipartition $(A, \overline{A})$ of $V(G)$ with $\mathtt{f}(A) \leq k$, $|X \cap A| \leq \frac{2}{3}|X|$, and $|X \cap \overline{A}| \leq \frac{2}{3}|X|$.*

A tree decomposition of a graph G is a pair $(T, \mathtt{bag})$, where T is a tree and $\mathtt{bag} : V(T) \to 2^{V(G)}$ is a function from the nodes of T to subsets of vertices of G called *bags*, satisfying that (1) for every edge $uv \in E(G)$ there exists a node $t \in V(T)$ so that $\{u, v\} \subseteq \mathtt{bag}(t)$, and (2) for every vertex $v \in V(G)$, the set of nodes $\{t \in V(T) \colon v \in \mathtt{bag}(t)\}$ induces a non-empty and connected subtree of T. The width of a tree decomposition is the maximum size of $\mathtt{bag}(t)$ minus one, and the treewidth of a graph is the minimum width of a tree decomposition of the graph.

For a set of vertices $X \subseteq V(G)$, we denote by $\alpha(X)$ the maximum size of an independent set in $G[X]$. The independence number of a tree decomposition $(T, \mathtt{bag})$ is the maximum of $\alpha(\mathtt{bag}(t))$ over $t \in V(T)$ and it is denoted by $\alpha(T, \mathtt{bag})$. The tree-independence number of a graph ($\mathtt{tree}$-α) is the minimum independence number of a tree decomposition of the graph [9,27].

For a set of vertices $X \subseteq V(G)$, we denote by $\mu(X)$ the maximum size of an induced matching in G so that for each edge of the matching, at least one of the endpoints of the edge is in X. For a tree decomposition $(T, \mathtt{bag})$, we denote by $\mu(T, \mathtt{bag})$ the maximum of $\mu(\mathtt{bag}(t))$ over $t \in V(T)$. Yolov [27] defined the minor-matching hypertree width ($\mathtt{tree}$-μ) of a graph to be the minimum $\mu(T, \mathtt{bag})$ of a tree decomposition $(T, \mathtt{bag})$ of G.

3 O-Mim-Width

In this section, we prove the part of Theorem 1 on Independent Set and Theorem 2. We start with some intermediary results. The following reveals an important property of cuts of bounded upper-mim-width. Razgon proved a similar statement in [24]. To simplify the statements of this section, we fix an n-vertex graph G with a weight function $w : V(G) \to \mathbb{Z}_{\geq 0}$.

Lemma 9. *Let $A \subseteq V(G)$. For every $X \subseteq A$ that is the union of t independent sets, there exists $X' \subseteq X$ of size at most $t \cdot \mathtt{umim}(A)$ such that $N(X) \setminus A = N(X') \setminus A$. In particular, we have $|\{N(X) \setminus A : X \in \mathsf{IS}(A)\}| \leq n^{\mathtt{umim}(A)}$ where $\mathsf{IS}(A)$ is the set of independent sets of $G[A]$.*

Proof. It is sufficient to prove the lemma for $t = 1$, since if X is the union of t independent sets $X_1, \ldots, X_t$, then the case $t = 1$ implies that, for each $i \in [1, t]$, there exits $X_i' \subseteq X_i$ such that $N(X_i) \setminus A = N(X_i') \setminus A$ and $|X_i'| \leq \mathtt{umim}(A)$. It follows that $X' = X_1' \cup \cdots \cup X_t' \subseteq X$, $N(X) \setminus A = N(X') \setminus A$ and $|X'| \leq t \cdot \mathtt{umim}(A)$.

Let X be an independent set of $G[A]$. If for every vertex $x \in X$, there exists a vertex $y_x \in \overline{A}$ such that $N(y_x) \cap X = \{x\}$, then $\{xy_x : x \in X\}$ is an induced $(A, \overline{A})$-matching in $G - E(\overline{A})$. We deduce that either $|X| \leq \mathtt{umim}(A)$ or there exists a vertex $x \in X$ such that $N(X) \setminus A = N(X \setminus \{x\}) \setminus A$. Thus, we can recursively remove vertices from X to find a set $X' \subseteq X$ of size at most $\mathtt{umim}(A)$ and such that $N(X) \setminus A = N(X') \setminus A$. In particular, the latter implies that $\{N(X) \setminus A : X \in \mathsf{IS}(A)\} = \{N(X) \setminus A : X \in \mathsf{IS}(A) \wedge |X| \leq \mathtt{umim}(A)\}$. We conclude that $|\{N(X) \setminus A : X \in \mathsf{IS}(A)\}| \leq n^{\mathtt{umim}(A)}$. □

To solve Independent Set and Feedback Vertex Set, we use the general toolkit developed in [1] with a simplified notation adapted to our two problems. This general toolkit is based on the following notion of representativity between sets of partial solutions. In the following, the collection $\mathcal{S}$ represents the set of solutions, in our setting $\mathcal{S}$ consists of either all the independent sets or all the set of vertices inducing a forest.

Definition 10. *Given $\mathcal{S} \subseteq 2^{V(G)}$, for every $\mathcal{A} \subseteq 2^{V(G)}$ and $Y \subseteq V(G)$, we define* $\mathrm{best}_{\mathcal{S}}(\mathcal{A}, Y) := \max\{w(X) : X \in \mathcal{A} \wedge X \cup Y \in \mathcal{S}\}$. *Given $A \subseteq V(G)$ and $\mathcal{A}, \mathcal{B} \subseteq 2^A$, we say that $\mathcal{B}$ $(\mathcal{S}, A)$-represents $\mathcal{A}$ if for every $Y \subseteq \overline{A}$, we have* $\mathrm{best}_{\mathcal{S}}(\mathcal{A}, Y) = \mathrm{best}_{\mathcal{S}}(\mathcal{B}, Y)$.

Observe that if there is no $X \in \mathcal{B}$ such that $X \cup Y \in \mathcal{S}$, then $\mathrm{best}_{\mathcal{S}}(\mathcal{B}, Y) = \max(\emptyset) = -\infty$. It is easy to see that the relation "$(\mathcal{S}, A)$-represents" is an equivalence relation.

The following is an application of Theorem 4.1 from [1]. It proves that a routine for computing small representative sets can be used to design a dynamic programming algorithm.

Theorem 11 **([1]).** *Let $\mathcal{S} \subseteq 2^{V(G)}$. Assume that there exists a constant c and an algorithm that, given $A \subseteq V(G)$ and $\mathcal{A} \subseteq 2^A$, computes in time $|\mathcal{A}| n^{\mathcal{O}(\mathtt{omim}(A))}$ a subset $\mathcal{B}$ of $\mathcal{A}$ such that $|\mathcal{B}| \leq n^{c \cdot \mathtt{omim}(A)}$ and $\mathcal{B}$ $(\mathcal{S}, A)$-represents $\mathcal{A}$. Then, there exists an algorithm, that given a branch decomposition $\mathcal{L}$ of G, computes in time $n^{\mathcal{O}(\mathtt{omim}(\mathcal{L}))}$ a set of size at most $n^{c \cdot \mathtt{omim}(A)}$ that contains an element in $\mathcal{S}$ of maximum weight.*

The following lemma provides a routine to compute small representative sets for Independent Set. We denote by $\mathcal{I}$ the set of all independent sets of G.

Lemma 12. *Let $k = \mathtt{omim}(A)$. Given a collection $\mathcal{A} \subseteq 2^A$, we can compute in time $|\mathcal{A}| n^{\mathcal{O}(k)}$ a subset $\mathcal{B}$ of $\mathcal{A}$ such that $\mathcal{B}$ $(\mathcal{I}, A)$-represents $\mathcal{A}$ and $|\mathcal{B}| \leq n^k$.*

Proof. Let $\mathcal{A} \subseteq 2^A$. We compute $\mathcal{B}$ from the empty set as follows:

- If $\mathtt{umim}(A) = k$, then, for every $Y \in \{N(X) \setminus A : X$ is an independent in $\mathcal{A}\}$, we add to $\mathcal{B}$ an independent set $X \in \mathcal{A}$ of maximum weight such that $Y = N(X) \setminus A$.
- If $\mathtt{umim}(A) > k$, then, for each subset $Y \subseteq \overline{A}$ with $|Y| \leq k$, we add to $\mathcal{B}$ a set $X \in \mathcal{A}$ of maximum weight such that $X \cup Y$ is an independent set (if such X exists).

It remains to prove the runtime. First, we prove that $|\mathcal{B}| \leq n^k$. This is straightforward when $\mathtt{umim}(A) > k$. When $\mathtt{umim}(A) = k$, Lemma 9 implies that $|\{N(X) \setminus A : X$ is an independent in $\mathcal{A}\}| \leq n^k$ and thus, we have $|\mathcal{B}| \leq n^k$.

Next, we prove that $\mathcal{B}$ $(\mathcal{I}, A)$-represents $\mathcal{A}$, i.e. for every $Y \subseteq \overline{A}$, we have that $\text{best}_{\mathcal{I}}(\mathcal{A}, Y) = \text{best}_{\mathcal{I}}(\mathcal{B}, Y)$. Let $Y \subseteq \overline{A}$. As $\mathcal{B}$ is subset of $\mathcal{A}$, we have $\text{best}_{\mathcal{I}}(\mathcal{B}, Y) \leq \text{best}_{\mathcal{I}}(\mathcal{A}, Y)$. In particular, if there is no $X \in \mathcal{A}$ such that $X \cup Y$ is an independent set, then we have $\text{best}_{\mathcal{I}}(\mathcal{A}, Y) = \text{best}_{\mathcal{I}}(\mathcal{B}, Y) = -\infty$.

Suppose from now that $\text{best}_{\mathcal{I}}(\mathcal{A}, Y) \neq -\infty$ and let $X \in \mathcal{A}$ such that $X \cup Y$ is an independent set and $w(X) = \text{best}_{\mathcal{I}}(\mathcal{A}, Y)$. We distinguish the following cases:

- If $\mathtt{umim}(A) = k$, then, by construction, there exists an independent set $W \in \mathcal{B}$ such that $N(X) \setminus A = N(W) \setminus A$ and $w(X) \leq w(W)$. As $X \cup Y$ is an independent set, we deduce that $N(X) \cap Y = N(W) \cap Y = \emptyset$ and thus $W \cup Y$ is an independent set.
- If $\mathtt{umim}(A) > k$, then $\mathtt{umim}(\overline{A}) = k$ as $\mathtt{omim}(A) = \min(\mathtt{umim}(A), \mathtt{umim}(\overline{A})) = k$. By Lemma 9, there exists an independent set $Y' \subseteq Y$ of size at most k such that $N(Y) \setminus \overline{A} = N(Y') \setminus \overline{A}$. As $Y' \subseteq Y$, we know that $X \cup Y'$ is an independent set. Thus, by construction there exists a set $W \in \mathcal{B}$ such that $W \cup Y'$ is an independent set and $w(X) \leq w(W)$. Since $N(Y) \setminus A = N(Y') \setminus A$, we deduce that $W \cup Y$ is an independent set.

In both cases, there exists $W \in \mathcal{B}$ such that $W \cup Y$ is an independent set and $w(X) \leq w(W) \leq \text{best}_{\mathcal{I}}(\mathcal{B}, Y)$. Since $\text{best}_{\mathcal{I}}(\mathcal{B}, Y) \leq \text{best}_{\mathcal{I}}(\mathcal{A}, Y) = w(X)$, it follows that $w(X) = \text{best}_{\mathcal{I}}(\mathcal{A}, Y) = \text{best}_{\mathcal{I}}(\mathcal{B}, Y)$. As this holds for every $Y \subseteq \overline{A}$, we conclude that $\mathcal{B}$ $(\mathcal{I}, A)$-represents $\mathcal{A}$.

It remains to prove the running time. Computing $\mathtt{omim}(A) = k$ and checking whether $\mathtt{umim}(A) = k$ can be done by looking at every set of $k+1$ edges and check whether one of these sets is an induced $(A, \overline{A})$-matching in $G - E(\overline{A})$ and in $G - E(A)$. This can be done in time $\mathcal{O}(\binom{n^2}{k+1} n^2) = n^{\mathcal{O}(k)}$ time. When $\mathtt{umim}(A) > k$, it is clear that computing $\mathcal{B}$ can be done in time $|\mathcal{A}| n^{\mathcal{O}(k)}$. This is also possible when $\mathtt{umim}(A) = k$ as Lemma 9 implies that $|\{N(X) \setminus A : X$ is an independent set in $\mathcal{A}\}| \leq n^k$. □

We obtain the following by using Theorem 11 with the routine of Lemma 12.

Theorem 13. *Given an n-vertex graph with a branch decomposition of o-mim-width k, we can solve* Independent Set *in time $n^{\mathcal{O}(k)}$.*

We show that the o-mim-width of a graph is upper bounded by its tree-independence number.

We say that a branch decomposition is *on a set* $V(G)$ if it is a branch decomposition of some function $\mathtt{f} : 2^{V(G)} \to \mathbb{Z}_{\geq 0}$. Next we give a general lemma for turning tree decompositions of G into branch decompositions on $V(G)$.

Lemma 14. *Let* $(T, \mathtt{bag})$ *be a tree decomposition of a graph* G*. There exists a branch decomposition* (T', δ) *on the set* $V(G)$ *so that for every bipartition* $(A, \overline{A})$ *of* $V(G)$ *given by an edge of* (T', δ)*, there exists a bag of* $(T, \mathtt{bag})$ *that contains either* $N(A)$ *or* $N(\overline{A})$*.*

Then we restate Theorem 2 and prove it using Lemma 14.

Theorem 2. *Any graph with tree-independence number* k *has o-mim-width at most* k*.*

Proof. Let G be a graph with tree-independence number k and $(T, \mathtt{bag})$ a tree decomposition of G with independence number $\alpha(T, \mathtt{bag}) = k$. By applying Lemma 14 we turn $(T, \mathtt{bag})$ into a branch decomposition on $V(G)$ so that for every partition $(A, \overline{A})$ of $V(G)$ given by the decomposition, either $N(A)$ or $N(\overline{A})$ has independence number at most k. Now, if $N(A)$ has independence number at most k, then $\mathtt{umim}(\overline{A}) \leq k$, and if $N(\overline{A})$ has independence number at most k, then $\mathtt{umim}(A) \leq k$, so we have that $\mathtt{omim}(A) \leq k$, and therefore the o-mim-width of the branch decomposition is at most k. □

With similar arguments, we also prove the following.

Theorem 15. *Any graph with minor-matching hypertreewidth* k *has sim-width at most* k*.*

4 Neighbor-Depth of Graphs of Bounded Sim-Width

In this section we show that graphs of bounded sim-width have poly-logarithmic neighbor-depth, i.e., Theorem 5. The idea of the proof will be that given a cut of bounded sim-width, we can delete a constant fraction of the edges going over the cut by deleting the closed neighborhood of a single vertex. This allows to first fix a balanced cut according to an optimal decomposition for sim-width, and then delete the edges going over the cut in logarithmic depth.

We say that a vertex $v \in V(G)$ *neighbor-controls* an edge $e \in E(G)$ if e is incident to a vertex in $N[v]$. In other words, v neighbor-controls e if $e \notin E(G \setminus N[v])$.

Lemma 16. *Let* G *be a graph and* $A \subseteq V(G)$ *so that* $\mathtt{sim}(A) \leq k$*. There exists a vertex* $v \in V(G)$ *that neighbor-controls at least* $|E(A, \overline{A})|/2k$ *edges in* $E(A, \overline{A})$*.*

Proof. Suppose the contradiction, i.e., that all vertices of G neighbor-control less than $|E(A, \overline{A})|/2k$ edges in $E(A, \overline{A})$. Let $M \subseteq E(A, \overline{A})$ be a maximum induced $(A, \overline{A})$-matching, having size at most $|M| \leq \mathtt{sim}(A) \leq k$, and let $V(M)$ denote the set of vertices incident to M. Now, an edge in $E(A, \overline{A})$ cannot be added to M if and only if one of its endpoints is in $N[V(M)]$. In particular, an edge in $E(A,\overline{A})$ cannot be added to M if and only if there is a vertex in $V(M)$ that neighbor-controls it. However, by our assumption, the vertices in $V(M)$ neighbor-control strictly less than

$$|V(M)| \cdot |E(A,\overline{A})|/2k = |E(A,\overline{A})|$$

edges of $E(A,\overline{A})$, so there exists an edge in $E(A,\overline{A})$ that is not neighbor-controlled by $V(M)$, and therefore we contradict the maximality of M. □

Now, the idea will be to argue that because sim-width is at most k, there exists a balanced cut $(A,\overline{A})$ with $\mathtt{sim}(A) \leq k$, and then select the vertex v given by Lemma 16 as the pivot-vertex. Here, we need to be careful to persistently target the same cut until the graph is disconnected along it.

Theorem 5. *Any n-vertex graph of sim-width k has neighbor-depth $\mathcal{O}(k \log^2 n)$*

Proof. For integers $n \geq 2$ and $k, t \geq 0$, we denote by $\mathtt{nd}(n,k,t)$ the maximum neighbor-depth of a graph that

1. has at most n vertices,
2. has sim-width at most k, and
3. has a cut $(A,\overline{A})$ with $\mathtt{sim}(A) \leq k$, $|E(A,\overline{A})| \leq t$, $|A| \leq 2n/3$, and $|\overline{A}| \leq 2n/3$.

We observe that if a graph G satisfies all of the conditions 1–3, then any induced subgraph of G also satisfies the conditions. In particular, note that n can be larger than $|V(G)|$, and in the condition 3, the cut should be balanced with respect to n but not necessarily with respect to $|V(G)|$.

We will prove by induction that

$$\mathtt{nd}(n,k,t) \leq 1 + 4k(\log_{3/2}(n) \cdot \log(n^2+1) + \log(t+1)). \tag{1}$$

This will then prove the statement, because by Lemma 8 any graph with n vertices and sim-width k satisfies the conditions with $t = n^2$.

First, when $n \leq 2$ this holds because any graph with at most two vertices has neighbor-depth at most one. We then assume that $n \geq 3$ and that Eq. (1) holds for smaller values of n and first consider the case $t = 0$.

Let G be a graph that satisfies the conditions 1–3 with $t = 0$. Because $t = 0$, each connected component of G has at most $2n/3$ vertices, and therefore satisfies the conditions with $n' = 2n/3$, $k' = k$, and $t' = (2n/3)^2$. Therefore, by induction each component of G has neighbor-depth at most $\mathtt{nd}(2n/3, k, (2n/3)^2)$. Because the neighbor-depth of G is the maximum neighbor-depth over its components, we get that

$$\begin{aligned}
\mathtt{nd}(G) \leq & \,\mathtt{nd}(2n/3, k, (2n/3)^2) \\
\leq & \,1 + 4k(\log_{3/2}(2n/3) \cdot \log((2n/3)^2+1) + \log((2n/3)^2+1)) \\
\leq & \,1 + 4k((\log_{3/2}(n) - 1) \cdot \log((2n/3)^2+1) + \log((2n/3)^2+1)) \\
\leq & \,1 + 4k(\log_{3/2}(n) \cdot \log((2n/3)^2+1)) \leq 1 + 4k(\log_{3/2}(n) \cdot \log(n^2+1)),
\end{aligned}$$

which proves that Eq. (1) holds when $t = 0$.

We then consider the case when $t \geq 1$. Assume that Eq. (1) does not hold and let G be a counterexample that is minimal under induced subgraphs. Note that this implies that G is connected, and every proper induced subgraph G' of G has neighbor-depth at most $1 + 4k(\log_{3/2}(n) \cdot \log(n^2+1) + \log(t+1))$. We can also assume that $t = |E(A, \overline{A})|$.

Now, by Lemma 16 there exists a vertex $v \in V(G)$ that neighbor-controls at least $t/2k$ edges in $E(A, \overline{A})$. We will select v as the pivot-vertex. By the minimality of G, we have that $\mathtt{nd}(G \setminus \{v\}) \leq 1 + 4k(\log_{3/2}(n) \cdot \log(n^2+1) + \log(t+1))$, so it suffices to prove that $\mathtt{nd}(G \setminus N[v]) \leq 1 + 4k(\log_{3/2}(n) \cdot \log(n^2 + 1) + \log(t+1)) - 1$. Because v neighbor-controls at least $t/2k$ edges in $E(A, \overline{A})$, the graph $G \setminus N[v]$ satisfies the conditions with $n' = n$, $k' = k$, and $t' = t - t/2k$. We denote

$$\alpha = \frac{t'+1}{t+1} = 1 - \frac{t/2k}{t+1} \leq 1 - \frac{t/2k}{2t} \leq 1 - \frac{1}{4k}.$$

Now we have that

$$\begin{aligned}
\mathtt{nd}(G) &\leq \mathtt{nd}(n, k, t - t/2k) + 1 \leq\ 2 + 4k(\log_{3/2}(n) \cdot \log(n^2+1) + \log(\alpha \cdot (t+1))) \\
&\leq 2 + 4k(\log_{3/2}(n) \cdot \log(n^2+1) + \log(\alpha) + \log(t+1)) \\
&\leq 2 + 4k\log(\alpha) + 4k(\log_{3/2}(n) \cdot \log(n^2+1) + \log(t+1)) \\
&\leq 2 - 4k \cdot \frac{1}{4k} + 4k(\log_{3/2}(n) \cdot \log(n^2+1) + \log(t+1)) \\
&\leq 1 + 4k(\log_{3/2}(n) \cdot \log(n^2+1) + \log(t+1)),
\end{aligned}$$

which proves that Eq. (1) holds when $t \geq 1$, and therefore completes the proof.

□

5 Conclusion

We conclude with some open problems. First, as already discussed, it is still open if independent set can be solved in polynomial-time on graphs of bounded mim-width, because it is not known how to construct a decomposition of bounded mim-width if one exists. It would be very interesting to resolve this problem by either giving an algorithm for computing decompositions of bounded mim-width, or by defining an alternative width parameter that is more general than mim-width and allows to solve Independent Set in polynomial-time when the parameter is bounded.

The class of graphs of polylogarithmic neighbor-depth generalizes several classes where Independent Set can be solved in (quasi)polynomial time. Another interesting class where Independent Set can be solved in polynomial-time and which, to our knowledge, could have polylogarithmic neighbor-depth is the class of graphs with polynomial number of minimal separators [12]. It would be interesting to show that this class has polylogarithmic neighbor-depth. More generally, Korhonen [21] studied a specific model of dynamic programming algorithms for Independent Set, in particular, tropical circuits for independent set, and it appears plausible that all graphs with polynomial size tropical circuits for independent set could have polylogarithmic neighbor-depth.

References

1. Bergougnoux, B., Dreier, J., Jaffke, L.: A logic-based algorithmic meta-theorem for mim-width. In: Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 3282–3304. SIAM (2023). https://doi.org/10.1137/1.9781611977554.ch125
2. Bergougnoux, B., Korhonen, T., Razgon, I.: New width parameters for independent set: one-sided-mim-width and neighbor-depth. CoRR abs/2302.10643 (2023). https://doi.org/10.48550/arXiv.2302.10643
3. Bonamy, M., et al.: Sparse graphs with bounded induced cycle packing number have logarithmic treewidth. In: Bansal, N., Nagarajan, V. (eds.) Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, 22–25 January 2023, pp. 3006–3028. SIAM (2023). https://doi.org/10.1137/1.9781611977554.ch116
4. Brettell, N., Horsfield, J., Munaro, A., Paesani, G., Paulusma, D.: Bounding the mim-width of hereditary graph classes. J. Graph Theory **99**(1), 117–151 (2022). https://doi.org/10.1002/jgt.22730
5. Brettell, N., Horsfield, J., Munaro, A., Paulusma, D.: List k-colouring P_t-free graphs: a mim-width perspective. Inf. Process. Lett. **173**, 106168 (2022). https://doi.org/10.1016/j.ipl.2021.106168
6. Bui-Xuan, B., Telle, J.A., Vatshelle, M.: Boolean-width of graphs. Theor. Comput. Sci. **412**(39), 5187–5204 (2011). https://doi.org/10.1016/j.tcs.2011.05.022
7. Courcelle, B., Engelfriet, J., Rozenberg, G.: Handle-rewriting hypergraph grammars. J. Comput. Syst. Sci. **46**(2), 218–270 (1993). https://doi.org/10.1016/0022-0000(93)90004-G
8. Courcelle, B., Makowsky, J.A., Rotics, U.: Linear time solvable optimization problems on graphs of bounded clique-width. Theory Comput. Syst. **33**(2), 125–150 (2000). https://doi.org/10.1007/s002249910009
9. Dallard, C., Milanič, M., Štorgel, K.: Treewidth versus clique number. II. Tree-independence number. CoRR abs/2111.04543 (2022). https://doi.org/10.48550/arXiv.2111.04543
10. Diestel, R.: Graph Theory. Graduate Texts in Mathematics, vol. 173, 4th edn. Springer, Cham (2012)
11. Eiben, E., Ganian, R., Hamm, T., Jaffke, L., Kwon, O.: A unifying framework for characterizing and computing width measures. In: Braverman, M. (ed.) 13th Innovations in Theoretical Computer Science Conference, ITCS 2022, Berkeley, CA, USA, 31 January–3 February 2022. LIPIcs, vol. 215, pp. 63:1–63:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.ITCS.2022.63
12. Fomin, F.V., Todinca, I., Villanger, Y.: Large induced subgraphs via triangulations and CMSO. SIAM J. Comput. **44**(1), 54–87 (2015). https://doi.org/10.1137/140964801
13. Gartland, P., Lokshtanov, D.: Independent set on P_k-free graphs in quasi-polynomial time. In: Irani, S. (ed.) 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, 16–19 November 2020, pp. 613–624. IEEE (2020). https://doi.org/10.1109/FOCS46700.2020.00063

14. Gartland, P., Lokshtanov, D., Pilipczuk, M., Pilipczuk, M., Rzazewski, P.: Finding large induced sparse subgraphs in $C_{>t}$-free graphs in quasipolynomial time. In: Khuller, S., Williams, V.V. (eds.) STOC 2021: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, 21–25 June 2021, pp. 330–341. ACM (2021). https://doi.org/10.1145/3406325.3451034
15. Gavril, F.: Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. SIAM J. Comput. **1**(2), 180–187 (1972). https://doi.org/10.1137/0201013
16. Grzesik, A., Klimosová, T., Pilipczuk, M., Pilipczuk, M.: Polynomial-time algorithm for maximum weight independent set on P_6-free graphs. ACM Trans. Algorithms **18**(1), 4:1–4:57 (2022). https://doi.org/10.1145/3414473
17. Jaffke, L., Kwon, O., Strømme, T.J.F., Telle, J.A.: Mim-width III. Graph powers and generalized distance domination problems. Theor. Comput. Sci. **796**, 216–236 (2019). https://doi.org/10.1016/j.tcs.2019.09.012
18. Jaffke, L., Kwon, O., Telle, J.A.: Mim-width I. Induced path problems. Discret. Appl. Math. **278**, 153–168 (2020). https://doi.org/10.1016/j.dam.2019.06.026
19. Jaffke, L., Kwon, O., Telle, J.A.: Mim-width II. The feedback vertex set problem. Algorithmica **82**(1), 118–145 (2020). https://doi.org/10.1007/s00453-019-00607-3
20. Kang, D.Y., Kwon, O., Strømme, T.J.F., Telle, J.A.: A width parameter useful for chordal and co-comparability graphs. Theor. Comput. Sci. **704**, 1–17 (2017). https://doi.org/10.1016/j.tcs.2017.09.006
21. Korhonen, T.: Lower bounds on dynamic programming for maximum weight independent set. In: 48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, Glasgow, Scotland, 12–16 July 2021 (Virtual Conference), pp. 87:1–87:14 (2021). https://doi.org/10.4230/LIPIcs.ICALP.2021.87
22. Munaro, A., Yang, S.: On algorithmic applications of sim-width and mim-width of (H_1, H_2)-free graphs. CoRR abs/2205.15160 (2022). https://doi.org/10.48550/arXiv.2205.15160
23. Oum, S., Seymour, P.D.: Approximating clique-width and branch-width. J. Comb. Theory Ser. B **96**(4), 514–528 (2006). https://doi.org/10.1016/j.jctb.2005.10.006
24. Razgon, I.: Classification of OBDD size for monotone 2-CNFs. In: Golovach, P.A., Zehavi, M. (eds.) 16th International Symposium on Parameterized and Exact Computation, IPEC 2021, Lisbon, Portugal, 8–10 September 2021. LIPIcs, vol. 214, pp. 25:1–25:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.IPEC.2021.25
25. Robertson, N., Seymour, P.D.: Graph minors. III. Planar tree-width. J. Comb. Theory Ser. B **36**(1), 49–64 (1984). https://doi.org/10.1016/0095-8956(84)90013-3
26. Vatshelle, M.: New width parameters of graphs. Ph.D. thesis, University of Bergen, Norway (2012). www.hdl.handle.net/1956/6166
27. Yolov, N.: Minor-matching hypertree width. In: Czumaj, A. (ed.) Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, 7–10 January 2018, pp. 219–233. SIAM (2018). https://doi.org/10.1137/1.9781611975031.16

Nonplanar Graph Drawings with k Vertices per Face

Carla Binucci[1(✉)], Giuseppe Di Battista[2], Walter Didimo[1],
Seok-Hee Hong[3], Michael Kaufmann[4], Giuseppe Liotta[1], Pat Morin[5],
and Alessandra Tappini[1]

[1] Università degli Studi di Perugia, Perugia, Italy
{carla.binucci,walter.didimo,giuseppe.liotta,alessandra.tappini}@unipg.it
[2] Università degli Studi Roma Tre, Rome, Italy
giuseppe.dibattista@uniroma3.it
[3] University of Sydney, Camperdown, Australia
seokhee.hong@sydney.edu.au
[4] University of Tübingen, Tübingen, Germany
mk@informatik.uni-tuebingen.de
[5] Carleton University, Ottawa, Canada
morin@scs.carleton.ca

Abstract. The study of nonplanar graph drawings with forbidden or desired crossing configurations has a long tradition in geometric graph theory, and received an increasing attention in the last two decades, under the name of *beyond-planar graph drawing*. In this context, we introduce a new hierarchy of graph families, called k^+*-real face graphs*. For any integer $k \geq 1$, a graph G is a k^+-real face graph if it admits a drawing Γ in the plane such that the boundary of each face (formed by vertices, crossings, and edges) contains at least k vertices of G. We give tight upper bounds on the maximum number of edges of k^+-real face graphs. In particular, we show that 1^+-real face and 2^+-real face graphs with n vertices have at most $5n - 10$ and $4n - 8$ edges, respectively. Also, if all vertices are constrained to be on the boundary of the external face, then 1^+-real face and 2^+-real face graphs have at most $3n - 6$ and $2.5n - 4$ edges, respectively. We also study relationships between k^+-real face graphs and beyond-planar graph families with hereditary property.

Keywords: beyond-planar graph drawing · k^+-real face graphs · edge density

1 Introduction

The study of nonplanar graph drawings with forbidden substructures has a long tradition in geometric graph theory (see, e.g., [31]). In the last two decades, this

Research started at the Summer Workshop on Graph Drawing (SWGD) 2022, and partially supported by: (*i*) MIUR, grant 20174LF3T8 "AHeAD: efficient Algorithms for HArnessing networked Data"; (*ii*) Dipartimento di Ingegneria - Università degli Studi di Perugia, Ricerca di Base, grants RICBA21LG and RICBA22CB.

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 86–100, 2023.
https://doi.org/10.1007/978-3-031-43380-1_7

topic, often recognized as *beyond-planar graph drawing*, has become increasingly popular. This growth in interest is due in part to human cognitive experiments aimed at estimating the impact of crossing configurations on graph visualization readability. See [22,25,27] for recent surveys or books on the subject.

With a widely-accepted terminology, a *beyond-planar graph family* is a type of nonplanar graphs that can be drawn in the plane by avoiding some given edge crossing configurations or by guaranteeing some properties about edge crossings. For example, for a given positive integer k, the family of *k-planar graphs* consists of those graphs that can be drawn in the plane with at most k crossings per edge [28,34], while *k-quasi planar graphs* are graphs that can be drawn in the plane without k mutually crossing edges [1,4,6,33,35]. Again, *right-angle-crossing graphs* (also known as *RAC graphs*) are those graphs that admit a straight-line drawing in which any two crossing edges form angles of 90° at their crossing point [20,21]; generalizations and variants of RAC drawings have also been proposed (see, e.g., [3,18,24]). Refer to [22] for other notable beyond-planar graph families. Given a family $\mathcal{F}$ of beyond-planar graphs, one of the most relevant problems is establishing the maximum *edge density* for the elements of $\mathcal{F}$, i.e., the maximum number of edges that an n-vertex graph in $\mathcal{F}$ can have with respect to n. This is a classical *Turán-type* problem with a long tradition in extremal graph theory [9,15,29] and represents one of the core research topics in the literature on graph drawing beyond planarity. For example, it is known that n-vertex 1-planar graphs and 2-planar graphs have at most $4n - 8$ edges and $5n - 10$ edges, respectively, and both these bounds are tight, in the sense that there are graphs in these families that can actually achieve them [34]. A graph of $\mathcal{F}$ that is maximally dense (i.e., whose number of edges is the maximum possible over its number of vertices) is usually called an *optimal* graph of $\mathcal{F}$.

A similar research direction investigates how different beyond-planar graph families relate to each other in terms of inclusion, partly exploiting edge density results [22]. For instance, it has been shown that the family of simple k-planar graphs is a subset of $(k + 1)$-quasi planar graphs for any $k \geq 2$ [7].

Contribution. In this paper, we propose a new hierarchy of graph families, which we call *k^+-real face graphs*, for any positive integer k. Namely, consider any drawing Γ of a graph G in the plane, where edge crossing points are regarded as dummy vertices. The drawing Γ divides the plane into connected regions, called *faces*: if no two edges of G cross in Γ, the boundary of each face consists of vertices (and edges) of G; otherwise, the boundaries of some faces contain dummy vertices. We say that G is a *k^+-real face graph* if it admits a drawing such that each face boundary contains at least k real vertices, i.e., k vertices of G (k^+ stands for "k or more"). By definition, for any $k \geq 1$, the family of $(k + 1)^+$-real face graphs is included in the family of k^+-real face graphs. From a theoretical perspective, studying k^+-real face graphs generalizes to nonplanar graphs the study of planar graphs that admit a crossing-free drawing whose face sizes are above a desired threshold [5,23,30]. Also, finding k^+-real face graphs can be regarded as a generalization to nonplanar graphs of the classical guarding planar graph problem [16], where the vertices that cover the face set

Table 1. Summary of density results in this paper; n denotes the number of vertices.

Graph Family	Crossings ($\chi \leq$)	Edges ($m \leq$)	Ref.
k^+-real face graphs ($k \geq 3$)	$\frac{2-k}{k} \cdot m + n - 2$	$\frac{k}{k-2}(n-2)$	Lemma 1, Theorem 1
2^+-real face graphs	$n - 2$	$4n - 8$	Lemma 1, Theorem 2
1^+-real face graphs	$m + n - 2$	$5n - 10$	Lemma 1, Theorem 3
outer k^+-real face graphs ($k \geq 3$)	$\frac{2-k}{k} \cdot m + \frac{k-1}{k} \cdot n - 1$	$\frac{k-1}{k-2} \cdot n - \frac{k}{k-2}$	Lemma 2, Theorem 4
outer 2^+-real face graphs	$\frac{1}{2}n - 1$	$2.5n - 4$	Lemma 2, Theorem 5
outer 1^+-real face graphs	$m - 1$	$3n - 6$	Lemma 2, Theorem 6

are the (real) vertices of G. From a more practical perspective, the interest in k^+-real face graphs is motivated by the intuition that faces that mostly consist of crossing points could make the graph layout less readable; indeed, the number of real vertices per face can be regarded as a measure of how much the drawing is far from being planar; in particular, one would avoid, when possible, faces formed only by crossing points. Our results can be summarized as follows:

- We provide tight upper bounds on the edge density of k^+-real face graphs, for all values of k, both in the general case (Sect. 3) and in the constrained scenario in which all vertices of the graph are forced to stay on the external face (Sect. 4); see Table 1 for a summary of our results. The constrained scenario can be regarded as a generalization of the study of outerplanar graphs to our graphs, which we call *outer k^+-real face graphs*. We note that similar constraints have been previously studied in other families of beyond-planar graphs (see, e.g., [10,12,13,19,26]).
- We establish inclusion relationships between k^+-real face graphs and families of beyond-planar graphs with hereditary property, such as h-planar and h-quasi planar graphs (Sect. 5). In particular, we show that, for any positive integer k, the family of k^+-real face graphs is not included in any beyond-planar graph family with hereditary property. However, this is not always the case if we restrict our attention to optimal graphs.

For space reasons, some proofs have been omitted or sketched.

2 Basic Definitions

Let G be a graph. We assume that G is connected and simple, meaning that it contains neither multiple edges nor self-loops (if G is not connected we can treat each connected component of G independently). We denote by $V(G)$ and $E(G)$ the set of vertices and the set of edges of G, respectively. A *drawing* Γ of G maps each vertex $v \in V(G)$ to a distinct point in the plane and each edge $uv \in E(G)$ to a simple Jordan arc between the points corresponding to u and v. We assume that Γ is a *simple* drawing, that is: (*i*) adjacent edges do not intersect, except at their common endpoint; (*ii*) two independent (i.e., non-adjacent) edges intersect at most in one of their interior points, called a *crossing point*; and (*iii*) no three edges intersect at a common crossing point.

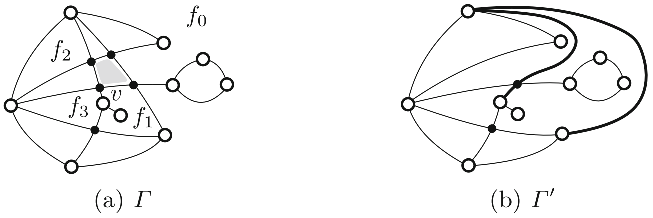


Fig. 1. (a) A nonplanar drawing Γ of a graph G with 5 crossings. White circles are the real-vertices of Γ (i.e., the vertices of G) and black circles are the crossing-vertices of Γ. Graph G has $n = 10$ vertices and $m = 14$ edges. Drawing Γ has $\nu = 15$ vertices, $\mu = 24$ edges, and $\varphi = 11$ faces. Face f_0 is the external face. The shaded face is a 0-real face. Face f_2 is a 2-real triangle and f_3 is a 2-real quadrilateral. The boundary of f_1 is not a simple cycle as vertex v is traversed twice while walking along its boundary. It follows that $\deg^r_\Gamma(f_1) = 4$, $\deg^c_\Gamma(f_1) = 3$, and $\deg_\Gamma(f_1) = 7$. (b) A 2^+-real face drawing Γ' of G, obtained from the previous one by rerouting two edges of G (the thicker ones).

Refer to Fig. 1 for an illustration of the next definitions. Let Γ be a drawing of G. A *vertex* of Γ is either a point corresponding to a vertex of G, called a *real-vertex*, or a point corresponding to a crossing point, called a *crossing-vertex*. Observe that a crossing-vertex has degree 4. We remark that in the literature a plane graph obtained by replacing crossing points with dummy vertices is often referred to as a *planarization* [17].

We denote by $V(\Gamma)$ the set of vertices of Γ. An *edge* of Γ is a curve connecting two vertices of Γ; an edge of Γ whose endpoints are both real-vertices coincides with an edge of G. We denote by $E(\Gamma)$ the set of edges of Γ. Drawing Γ subdivides the plane into topologically connected regions, called *faces*. The boundary of each face consists of a circular sequence of vertices and edges of Γ. We denote by $F(\Gamma)$ the set of faces of Γ. Exactly one face in $F(\Gamma)$ corresponds to an infinite region of the plane, called the *external face* (or *outer face*) of Γ; the other faces are the *internal faces* of Γ. When the boundary of a face f of Γ contains a vertex v (or an edge e), we also say that f contains v (or e).

From now on, we denote by $n = |V(G)|$ and $m = |E(G)|$ the number of vertices and the number of edges of G, respectively. For a drawing Γ of G, we denote by $\nu = |V(\Gamma)|$, $\mu = |E(\Gamma)|$, and $\varphi = |F(\Gamma)|$ the number of vertices, edges, and faces of Γ, respectively. Also, we denote by $\chi = |V(\Gamma) \setminus V(G)| = \nu - n$ the number of crossing-vertices of Γ.

- **Degree of vertices and faces.** For a vertex $v \in V(G)$, denote by $\deg_G(v)$ the *degree of v in G*, i.e., the number of edges incident to v. Analogously, for a vertex $v \in V(\Gamma)$, denote by $\deg_\Gamma(v)$ the *degree of v in Γ*. For a face $f \in F(\Gamma)$, denote by $\deg_\Gamma(f)$ the *degree of f*, i.e., the number of times we traverse vertices (either real- or crossing-vertices) while walking on the boundary of f clockwise. Each vertex contributes to $\deg_\Gamma(f)$ the number of times we traverse it (possibly more than once if the boundary of f is not a simple cycle). Also, denote by $\deg^r_\Gamma(f)$ the *real-vertex degree of f*, i.e., the number of times we traverse a real-vertex of Γ while walking on the boundary of f clockwise. Again, each real-vertex contributes to $\deg^r_\Gamma(f)$ the number of times we traverse it. Finally, $\deg^c_\Gamma(f)$ denotes the number of times we traverse a crossing-vertex of Γ while walking on the boundary of f clockwise. Clearly, $\deg_\Gamma(f) = \deg^r_\Gamma(f) + \deg^c_\Gamma(f)$.

- **k^+-real face drawings and graphs.** Given a graph G and a positive integer k, a *k^+-real face drawing* of G is a drawing Γ of G such that the boundary of each face of Γ has at least k real-vertices. If G admits a k^+-real face drawing, we say that G is a *k^+-real face graph.* An *outer k^+-real face drawing* of G is a k^+-real face drawing Γ of G such that all its real-vertices are on the boundary of the outer face. If G admits an outer k^+-real face drawing we say that G is an *outer k^+-real face graph.* We say that a face $f \in F(\Gamma)$ is an *h-real face*, where h is a non-negative integer, if $\deg^r_\Gamma(f) = h$. An h-real face of degree d is called an *h-real d-gon.* An h-real 3-gon is also called an *h-real triangle*, and an h-real 4-gon is also called an *h-real quadrilateral.* We say that an edge $e = uv \in E(\Gamma)$ is an *h-real edge* ($h \in \{0, 1, 2\}$) if $|\{u, v\} \cap V(G)| = h$, i.e., e contains h real-vertices.

3 Density of k^+-Real Face Graphs

In this section, we prove tight upper bounds on the number of edges that a k^+-real face graph can have. We start by proving the following upper bound on the number χ of crossing-vertices in a k^+-real face drawing.

Lemma 1. *Let Γ be a k^+-real face drawing of a graph G. We have:*

$$\chi \leq \frac{2-k}{k} \cdot m + n - 2 \tag{1}$$

Proof. By hypothesis, each face $f \in F(\Gamma)$ contains at least k real-vertices (i.e., at least k vertices of G). Since each real-vertex $v \in V(G)$ can belong to at most $\deg_G(v)$ faces of Γ and since $\sum_{v \in V(G)} \deg_G(v) = 2m$, we have that the number φ of faces of Γ is such that $\varphi \leq \frac{2m}{k}$. Also, the number of edges μ of Γ is such that $\mu = m + 2\chi$. Hence, by Euler's formula applied to Γ, we have $\varphi = \mu + 2 - \nu = m + 2\chi + 2 - n - \chi$, and hence $\varphi = m + \chi + 2 - n$. It follows that $\chi = \varphi - m + n - 2 \leq \frac{2m}{k} - m + n - 2 = \frac{2-k}{k} \cdot m + n - 2$. □

3.1 k^+-Real Face Graphs, with $k \geq 2$

We first consider the case $k \geq 3$ and then the case $k = 2$.

Theorem 1. *Let k be a positive integer such that $k \geq 3$. If G is a k^+-real face graph with n vertices and m edges, then $m \leq \frac{k}{k-2}(n-2)$, and this bound is tight. Also, the optimal n-vertex k^+-real face drawings are exactly the n-vertex planar drawings in which each face is a simple k-gon.*

Proof (Sketch). Let Γ be any k^+-real face drawing of G. When $k \geq 3$, the term $\frac{2-k}{k}$ is negative and, equivalently, $\frac{k}{k-2}$ is positive. Since the number χ of crossing-vertices of Γ cannot be negative, i.e., $\chi \geq 0$, by Eq. (1) of Lemma 1 we have that the number of edges m must satisfy the inequality $m \leq \frac{k}{k-2}(n-2)$.

For the tightness of the bound, just consider the family of planar embedded graphs such that each face has k vertices. Any n-vertex graph in this family has $m = \frac{k}{k-2}(n-2)$ edges. Also, one can prove that every k^+-real face drawing with $\frac{k}{k-2}(n-2)$ edges is planar and all its faces have degree k. □

Theorem 2. *If G is a 2^+-real face graph with n vertices and m edges, then $m \leq 4n - 8$, and this bound is tight. Also, the optimal n-vertex 2^+-real face graphs are exactly the optimal 1-planar graphs.*

Proof (Sketch). Let Γ be any 2^+-real face drawing of G. By Eq. (1) of Lemma 1, with $k = 2$, we get $\chi \leq n - 2$. Since $\mu \leq 3\nu - 6$, and since $\mu = m + 2\chi$ and $\nu = n + \chi$, we have $m \leq \chi + 3n - 6$, and therefore $m \leq n - 2 + 3n - 6 = 4n - 8$. This proves that $4n - 8$ is an upper bound on the number of edges of G.

About the tightness of the bound, consider the family of 1-planar graphs, i.e., graphs that admit a drawing Γ with at most one crossing per edge. Each face of Γ has at least two real-vertices (see also [36]), thus Γ is a 2^+-real face drawing. In particular, for $n = 8$ and for every $n \geq 12$, there exists an optimal 1-planar graph with n vertices and $4n - 8$ edges [14,34]. Also, it can be proven that every optimal 2^+-real face drawing Γ of G is also a 1-planar drawing of G. □

3.2 1^+-Real Face Graphs

To prove an upper bound on the number of edges in 1^+-real face graphs, we use *discharging* techniques. See for example [2,4,24] for other papers that use similar approaches. Following [4], we consider a *charging function* $\mathrm{ch} : F(\Gamma) \to \mathbb{R}$ such that, for each $f \in F(\Gamma)$, we set:

$$\mathrm{ch}(f) = \deg_\Gamma(f) + \deg^r_\Gamma(f) - 4 = 2\deg^r_\Gamma(f) + \deg^c_\Gamma(f) - 4 \qquad (2)$$

The value $\mathrm{ch}(f)$ is called the *initial charge* of f. By using Euler's formula, it is not difficult to prove that the following relationship holds (for details, see [4]):

$$\sum_{f \in F(\Gamma)} \mathrm{ch}(f) = 4n - 8 \qquad (3)$$

The idea of a discharging technique is to derive from the initial charging function ch a new function ch′ that satisfies the next two properties (see also [4]):
C1. $\mathrm{ch}'(f) \geq \alpha \deg^r_\Gamma(f)$, for some real number $\alpha > 0$;
C2. $\sum_{f \in F(\Gamma)} \mathrm{ch}'(f) \leq \sum_{f \in F(\Gamma)} \mathrm{ch}(f)$
If $\alpha > 0$ is a real number for which a charging function ch′ satisfies C1 and C2, by Eq. (3) we have: $4n - 8 = \sum_{f \in F(\Gamma)} \mathrm{ch}(f) \geq \sum_{f \in F(\Gamma)} \mathrm{ch}'(f) \geq$

$\alpha \sum_{f \in F(\Gamma)} \deg^r_\Gamma(f)$. Also, since $\sum_{f \in F(\Gamma)} \deg^r_\Gamma(f) = \sum_{v \in V(G)} \deg_G(v) = 2m$, we get the following:

$$m \leq \frac{2}{\alpha}(n-2) \tag{4}$$

Thus, Eq. (4) can be exploited to prove upper bounds on the edge-density of a graph for specific values of α, whenever we find a charging function ch$'$ that satisfies C1 and C2. We are now ready to present the main result of this section.

Theorem 3. *Let G be a 1^+-real face graph with n vertices and m edges. We have that $m \leq 5n - 10$, and this bound is tight.*

Proof (Sketch). Let Γ be a 1^+-real face drawing of G. We first augment Γ and G as follows. If some face f of Γ contains a pair u and v of real-vertices but does not contain an edge uv on its boundary, then we augment Γ (and G) with an edge uv drawn in the interior of f, in such a way that it does not create any crossing. We then repeat this process until every pair of real-vertices in each face f is connected by an edge on the boundary of f. Note that, this augmentation is not unique and may introduce multiple edges in G. However, it does not create any 0-real faces and any faces of degree two in the drawing; also, the drawing remains a 1^+-real face drawing. If Γ' denotes the drawing resulting from the augmentation on Γ, for each face $f \in F(\Gamma')$ we have that: (a) $\deg_{\Gamma'}(f) \geq 3$; and (b) $3 \geq \deg^r_{\Gamma'}(f) \geq 1$. Also, denoted by G' the graph resulting from the augmentation on G, we have $V(G') = V(G)$ and $E(G) \subseteq E(G')$; hence, an upper bound on the number of edges m' of G' is also an upper bound on the number of edges m of G.

Suppose given on Γ' the initial charging function ch : $F(\Gamma') \to \mathbb{R}$ of Eq. (2). If we are able to define a charging function ch$'$: $F(\Gamma') \to \mathbb{R}$ that satisfies C1 and C2 for $\alpha = \frac{2}{5}$, then by Eq. (4) we get $m \leq m' \leq 5n - 10$, and we are done.

We show how to define ch$'$. For every face $f \in \Gamma'$, we initially set $\text{ch}'(f) = \text{ch}(f) = 2\deg^r_{\Gamma'}(f) + \deg^c_{\Gamma'}(f) - 4$. With this choice and with $\alpha = \frac{2}{5}$, function ch$'$ satisfies C2. Also, C1 becomes $2\deg^r_{\Gamma'}(f) + \deg^c_{\Gamma'} - 4 \geq \frac{2}{5}\deg^r_{\Gamma'}(f)$, that is, $8\deg^r_{\Gamma'}(f) + 5\deg^c_{\Gamma'}(f) \geq 20$. Hence, since $\deg_{\Gamma'}(f) \geq 3$, C1 is always satisfied for each face f such that either $\deg^r_{\Gamma'}(f) \geq 2$, or $\deg^r_{\Gamma'}(f) = 1$ and $\deg^c_{\Gamma'}(f) \geq 3$. It follows that, the only faces that do not satisfy C1 are the 1-real triangles, i.e., each face t for which $\deg^r_{\Gamma'}(t) = 1$ and $\deg^c_{\Gamma'}(t) = 2$. Indeed, for a 1-real triangle t the initial charge equals 0, thus we need to suitably increase the value of ch$'(t)$.

For each 1-real triangle t, let f be the face incident to the unique 0-real edge of t; see Fig. 2a. Observe that $\deg_{\Gamma'}(f) \geq 4$. Indeed, if it were $\deg_{\Gamma'}(f) = 3$ then G would contain two parallel edges (which is impossible because G is simple) or there would be two adjacent edges of G that cross in Γ (which is impossible because Γ is simple). Also, since Γ' is a 1^+-real face drawing, we have $\deg^r_{\Gamma'}(f) \geq 1$. We apply a discharging operation, by moving a fraction $\frac{2}{5}$ of charge from f to t across their shared 0-real edge. In this way, we set ch$'(t) = \frac{2}{5}$ and reduce $ch'(f)$ by $\frac{2}{5}$. The total charge of Γ' determined by ch$'$ does not change (hence C2 is still satisfied) but now ch$'(t)$ satisfies C1.

Since for a face f the reduction of ch$'(f)$ by $\frac{2}{5}$ occurs across a 0-real edge of f, the number of times this happens is at most $\deg^c_{\Gamma'}(f) - 1$. Therefore, after we

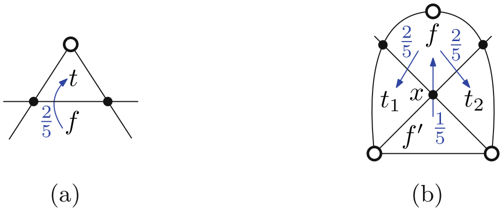


Fig. 2. Illustration for the proof of Theorem 3: (a) A 1-real triangle t and an adjacent face f that moves a charge of $\frac{2}{5}$ towards t across a 0-real edge. (b) A 1-real quadrilateral f that moves two charges of $\frac{2}{5}$ towards two adjacent 1-real triangles t_1 and t_2; face f recovers a charge of $\frac{1}{5}$ from a 2-real triangle f' that shares a vertex x with f, t_1, and t_2

have applied a discharging operation for each 1-real triangle, the charge $\mathrm{ch}'(f)$ of each face f of degree at least four is such that:

$$\mathrm{ch}'(f) \geq 2\deg^r_{\Gamma'}(f) + \deg^c_{\Gamma'}(f) - 4 - \frac{2}{5}\deg^c_{\Gamma'}(f) + \frac{2}{5} = 2\deg^r_{\Gamma'}(f) + \frac{3}{5}\deg^c_{\Gamma'}(f) - \frac{18}{5}$$

Hence, f satisfies C1 (i.e., $\mathrm{ch}'(f) \geq \frac{2}{5}\deg^r_{\Gamma'}(f)$) if this relation holds:

$$8\deg^r_{\Gamma'}(f) + 3\deg^c_{\Gamma'}(f) \geq 18 \tag{5}$$

It can be easily verified that the above relation is always satisfied for a face f of degree at least four, except when f is a 1-real quadrilateral (which consists of one real-vertex and 3 crossing-vertices). Indeed, if f is a 1-real quadrilateral it could have moved a fraction $\frac{2}{5}$ of charge towards a 1-real triangle t_1 and a fraction $\frac{2}{5}$ of charge towards another 1-real triangle t_2; see Fig. 2b. Both t_1 and t_2 share a crossing-vertex x with f and with another face f'. In this case $\mathrm{ch}'(f) = \mathrm{ch}(f) - \frac{4}{5} = 1 - \frac{4}{5} = \frac{1}{5} = \frac{2}{5}\deg^r_{\Gamma'}(f) - \frac{1}{5}$, thus f has a deficit of $\frac{1}{5}$. Observe that the boundary of f' contains two real-vertices adjacent to x, which are connected by an edge due to the edge augmentation initially performed on Γ. Hence, f' is a 2-real triangle and at this point we have $\mathrm{ch}'(f') = \mathrm{ch}(f') = 1 = \frac{2}{5}\deg^r_{\Gamma'}(f') + \frac{1}{5}$. It follows that $\mathrm{ch}'(f')$ has a surplus of $\frac{1}{5}$, and we can move this surplus from f' to f, i.e., we increase $\mathrm{ch}'(f)$ by $\frac{1}{5}$ and decrease $\mathrm{ch}'(f')$ by $\frac{1}{5}$. Since this reduction of $\mathrm{ch}(f')$ can happen at most once for f', both f and f' satisfy C1 at the end of this operation. This completes the proof that $m \leq 5n - 10$.

As for the tightness of the bound, consider any n-vertex optimal 2-planar drawing Γ. Such a drawing has $5n - 10$ edges and it is composed of a planar pentangulation (i.e., every face is a simple cycle of degree five) plus five crossing edges inside each pentagon [11]. A 1^+-real face drawing with $n' > n$ vertices and $5n' - 10$ edges is obtained from Γ by adding a vertex inside each pentagon and connecting it to all vertices of the pentagon (see Fig. 3 for an illustration). □

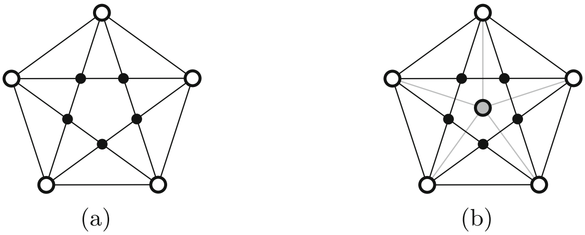


Fig. 3. (a) Pentagonal face of an optimal 2-planar drawing. (b) Augmenting each pentagonal face with a vertex and five edges (in gray) to make the drawing 1^+-real face.

4 Density of Outer k^+-Real Face Graphs

In this section, we provide tight upper bounds on the maximum number of edges that an outer k^+-real face graph can have, depending on k. For an outer k^+-real face drawing Γ of a graph G, we denote by $F_{\text{int}}(\Gamma) \subset F(\Gamma)$ the subset of internal faces of Γ. Additionally, we denote by φ_{int} the number of internal faces of Γ, that is $\varphi_{\text{int}} = |F_{\text{int}}(\Gamma)|$. Notice that $\varphi = \varphi_{\text{int}} + 1$. As for k^+-real face graphs, we first give an upper bound on the number χ of crossing-vertices in an outer k^+-real face drawing (the proof relies on similar arguments).

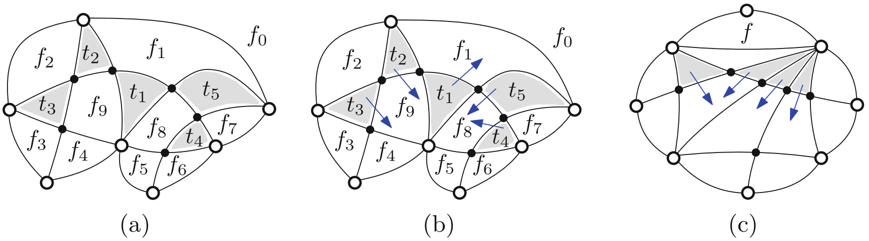


Fig. 4. (a) An edge-maximal outer 1^+-real face drawing of a graph with 7 vertices and 14 edges. Face f_1 is a 2-real 4-gon; $f_2, \ldots, f_7$ are 2-real triangles; f_8 and f_9 are 1-real quadrilaterals; $t_1, \ldots, t_5$ are 1-real triangles. (b) Arrows show a mapping of the 1-real triangles that satisfies Property (c) of Lemma 3. (c) Another example of an edge-maximal outer 1^+-real face drawing with 8 vertices and 16 edges. Face f is a 3-real triangle. The shaded faces are the 1-real triangles; a mapping of these triangles that satisfies Property (c) of Lemma 3 is shown.

Lemma 2. *Let G be a graph and let k be a positive integer. If Γ is an outer k^+-real face drawing of G then the following holds:*

$$\chi \leq \frac{2-k}{k} \cdot m + \frac{k-1}{k} \cdot n - 1 \tag{6}$$

4.1 Outer k^+-Real Face Graphs, with $k \geq 2$

We first consider the case $k \geq 3$ and then the case $k = 2$. The proof of the next theorem is similar to the proof of Theorem 1 and it has been omitted.

Theorem 4. *Let k be a positive integer such that $k \geq 3$. If G is an outer k^+-real face graph with n vertices and m edges, then $m \leq \frac{k-1}{k-2} \cdot n - \frac{k}{k-2}$, and this bound is tight. Also, the optimal n-vertex outer k^+-real face drawings are exactly the n-vertex outerplanar drawings in which each internal face is a simple k-gon.*

Theorem 5. *Let G be an outer 2^+-real face graph with n vertices and m edges. We have that $m \leq 2.5n - 4$, and this bound is tight. Also, the n-vertex optimal outer 2^+-real face graphs are exactly the optimal outer-1-planar graphs.*

Proof (Sketch). Let Γ be any outer 2^+-real face drawing of G. By Lemma 2, with $k = 2$, we get $\chi \leq \frac{n}{2} - 1$. If we remove from Γ exactly one edge of G per crossing-vertex, we get an outerplanar graph with $m' = m - \chi$ edges and n vertices. Since a maximal outerplanar graph with n vertices has at most $2n - 3$ edges, we have $m - \chi \leq 2n - 3$, and therefore $m \leq 2n - 3 + \chi \leq 2n - 3 + \frac{n}{2} - 1$, that is, $m \leq \frac{5}{2}n - 4 = 2.5n - 4$. This proves that $2.5n - 4$ is an upper bound on the number of edges of G. Also, it can be proven that the bound is tight and that every optimal outer 2^+-real face drawing Γ is also 1-planar; since optimal outer-1-planar graphs have at most $2.5n - 4$ edges [8,19], this implies that optimal outer 2^+-real face graphs are exactly the optimal outer-1-planar graphs. □

4.2 Outer 1^+-Real Face Graphs

As for 1^+-real face graphs, we use discharging techniques to prove an upper bound on the number of edges of outer 1^+-real face graphs. An outer 1^+-real face Γ is *edge-maximal* if the drawing obtained by adding to Γ any new edge between two of its real-vertices is no longer outer 1^+-real face. An example of edge-maximal outer 1^+-real face drawing is illustrated in Fig. 4a; as Theorem 6 will show, this graph is however not optimal, as for any $n \geq 3$ there exist outer 1^+-real face graphs that contain $3n - 6$ edges. Another edge-maximal outer 1^+-real face drawing that is not optimal is shown in Fig. 4c.

We now present a key result about the structure of edge-maximal outer 1^+-real face drawings.

Lemma 3. *Let G be an n-vertex outer 1^+-real face graph, with $n \geq 4$, and let Γ be an edge-maximal outer 1^+-real face drawing of G. The following properties hold:*

- **a)** *The boundary of the external face is a simple cycle that consists of exactly n real-vertices and no crossing-vertices.*
- **b)** *Each internal face of Γ is either a 3-real triangle, or a 2-real d-gon ($d \geq 3$), or a 1-real triangle, or a 1-real quadrilateral.*

c) *We can map each 1-real triangle to exactly one face of Γ that is either a 2-real d-gon, for $d \geq 4$, or a 1-real quadrilateral, in such a way that: (i) at most $(d-3)$ 1-real triangles are mapped to the same 2-real d-gon; and (ii) at most two 1-real triangles are mapped to the same 1-real quadrilateral.*

d) *The number of 3-real triangles plus the number of 2-real d-gons is exactly n, and the number of 1-real quadrilaterals is at most $n-4$.*

Theorem 6. *If G is an outer 1^+-real face graph with n vertices and m edges, then $m \leq 3n-6$, and this bound is tight.*

Proof (Sketch). To prove the upper bound, it is enough to concentrate on edge-maximal outer 1^+-real face drawings of G. Let Γ be such a drawing. If G has three vertices, then Γ is a 3-cycle and the statement trivially holds. Assume that $n \geq 4$. We exploit a discharging technique as for Theorem 3. In this case, we want to show the existence of a charging function ch$'$ that satisfies C1 and C2 for $\alpha = \frac{2}{3}$. If such a function exists then, by Eq. (4), we get $m \leq 3n-6$. For each face $f \in F(\Gamma)$, initially set $\mathrm{ch}'(f) = \mathrm{ch}(f)$, where $\mathrm{ch}(f)$ is the charging function of Eq. (2). Denote by f_0 the external face of Γ. Based on Properties (a) and (b) of Lemma 3, $\deg_\Gamma(f_0) = \deg_\Gamma^r(f_0) = n$ and each internal face of Γ is either a 3-real triangle, or a 2-real d-gon, or a 1-real triangle, or a 1-real quadrilateral. At this point, we have:

- $\mathrm{ch}'(f_0) = 2\deg_\Gamma^r(f_0) + \deg_\Gamma^c(f_0) - 4 = 2n-4$; the charge excess of f_0 with respect to $\frac{2}{3}\deg_\Gamma^r(f_0)$ is $\frac{4}{3}n - 4$;
- If f is a 3-real triangle, then $\mathrm{ch}'(f) = 2$; it has no charge excess/deficit;
- If f is a 2-real d-gon, $\mathrm{ch}'(f) = d-2$; hence, if $d = 3$ (i.e., f is a 2-real triangle) f has a charge deficit of $\frac{1}{3}$, while if $d \geq 4$ it has an excess of $d - \frac{10}{3}$;
- If f is a 1-real triangle then $\mathrm{ch}(f) = 0$ and f has a charge deficit of $\frac{2}{3}$;
- If f is a 1-real quadrilateral then $\mathrm{ch}(f) = 1$ and f has a charge excess of $\frac{1}{3}$.

We modify ch$'$ by moving charges from faces with an excess to faces with a deficit, in such a way that C1 is satisfied. Based on the above analysis, the only faces with a deficit are the 2-real triangles and the 1-real triangles. We map each 1-real triangle to either a 2-real d-gon (with $d \geq 4$) or to a 1-real quadrilateral, so that the mapping satisfies Property (c) of Lemma 3. This mapping tells for each face with a deficit from which face it will receive charges; see Fig. 4. Property (d) of Lemma 3 is used to prove that the new charging function satisfies C1. □

5 Inclusion Relationships

In this section, we study inclusion relationships between the families of k^+-real face graphs and other beyond-planar graph families. As already observed, k^+-real face graphs form a hierarchy of families, that is, for each integer $k \geq 1$, the family of $(k+1)^+$-real face graphs is properly included in the family of k^+-real face graphs.

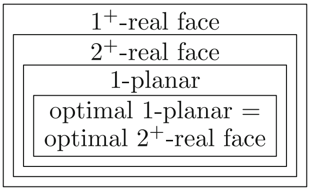


Fig. 5. Inclusion relationships between k^+-real face graphs and k-planar graphs, for $k = 1, 2$.

Note that the hierarchy of k-planar graphs has the opposite behavior, i.e., each k-planar graph is also a $(k+1)$-planar graph (for any $k \geq 1$). The results of Sect. 3 provide insights about inclusion relationships between the hierarchies of k-planar graphs and of k^+-real face graphs, for $k \in \{1, 2\}$. Namely, Theorem 2 shows that 1-planar graphs are 2^+-real face graphs and that the families of optimal 1-planar graphs and optimal 2^+-real face graphs coincide. These relationships are summarized in Fig. 5.

We now show a more general result about the relationship between k^+-real face graphs and any other beyond-planar graph family with hereditary property. This result (Theorem 7) excludes that, for any fixed positive integer k, there exists some beyond-planar graph families with hereditary property that contain all k^+-real face graphs. In the following, we formalize this concept.

Let $\mathcal{F}$ be a family of beyond-planar graphs. We say that $\mathcal{F}$ has the *hereditary property* if any subgraph of a graph in $\mathcal{F}$ also belongs to $\mathcal{F}$. Most of the beyond-planar graph families studied in the literature (see, e.g., [22]) have the hereditary property. On the contrary, the family of k^+-real face graphs (for any $k \geq 1$) does not necessarily satisfy this property, as removing vertices from a k^+-real face graph makes it impossible in some cases to guarantee at least k real-vertices per face; for example, if we remove the central vertex in the drawing of Fig. 3b, the drawing is no longer a 1^+-real face drawing. Nonetheless, it is immediate to see that if we remove from a k^+-real face graph any subset of edges but no vertices, the resulting subgraph is still a k^+-real face graph.

Lemma 4. *For any integer $k > 0$ and for any family $\mathcal{F}$ of beyond-planar graphs with hereditary property, there exists a k^+-real face graph not belonging to $\mathcal{F}$.*

Proof. Let G be any (connected) graph such that $G \notin \mathcal{F}$. Consider any drawing Γ of G. If Γ is already a k^+-real face drawing, we are done. Otherwise, we augment Γ into a new drawing by suitably adding new vertices and edges; refer to Fig. 6 for an example. We first consider the set of 0-real faces of Γ (i.e., faces whose boundary contains only crossings). If this set is not empty, there must be a 0-real face f that is adjacent to a face f' containing a real-vertex v. Add to Γ a new real-vertex u in the interior of f and connect u to v with an edge that crosses exactly one edge shared by f and f'. In this way, the set of 0-real faces is decreased by one element. Iterate this procedure until there is no more 0-real faces in the drawing. Now, consider every face f of Γ that contains $1 \leq h < k$ real-vertices (if any). Arbitrarily select a real-vertex v of f, and attach to v a chain of $k - h$ vertices in the interior of f. This creates a new face f' in place of f, which contains k real-vertices. Once all those faces have been processed, the underlying graph G' of the resulting drawing is a (connected) k^+-real face graph. Also, since $G \subseteq G'$ and $\mathcal{F}$ has the hereditary property, we have that $G' \notin \mathcal{F}$. □

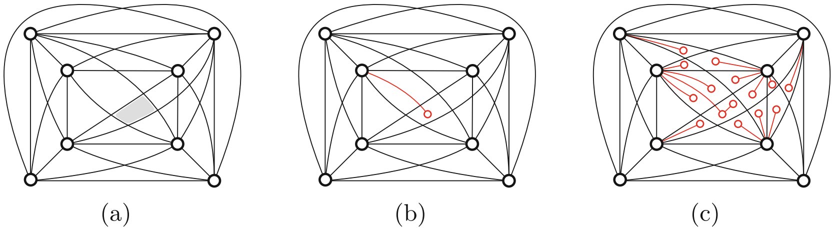


Fig. 6. (a) An initial drawing of a graph that is not 1-planar; it has one 0-real face (shaded). (b) An augmentation that removes the 0-real face; the new elements are red. (c) A further augmentation that makes the drawing 2^+-real face. (Color figure online)

Lemma 4 immediately implies the following.

Theorem 7. *For any positive integer k, the family of k^+-real face graphs is not included in any beyond-planar graph family with hereditary property.*

A consequence of Theorem 7 is that, for any integer $k > 0$, the family $\mathcal{K}$ of k^+-real face graphs is incomparable with any beyond-planar graph family $\mathcal{F}$ with hereditary property whose edge density is higher than the edge density of $\mathcal{K}$. For instance, each family of k^+-real face graphs is incomparable with the families of h-planar graphs for $h \geq 3$. Indeed, Theorem 7 proves the existence of a 1^+-real face graph that is not h-planar; on the other hand, since the maximum number of edges of an h-planar graph, for $h \geq 3$, can be higher than $5n-10$ [32,34], there exist h-planar graphs that are not 1^+-real face graphs. Similarly, each family of k^+-real face graphs is incomparable with the family of h-quasi planar graphs, for every $h \geq 3$, as 3-quasi planar graphs can have up to $6.5n - 20$ edges [1].

6 Open Problems

We conclude with two open research questions.

OP(1) The maximum edge density of 2-planar graphs is the same as the one of 1^+-real face graphs, and Theorem 6 implies that there are 1^+-real face graphs that are not 2-planar. An open question is whether there exist 2-planar graphs that are not 1^+-real face graphs. Note that, every 2-planar drawing of an optimal 2-planar graph G is not 1^+-real face, as it contains 0-real faces. However, one cannot exclude in principle that G admits a 1^+-real face drawing that is not 2-planar.

OP(2) Another interesting research direction is to establish the complexity of testing whether a graph is k^+-real face or outer k^+-real face for a given k. In particular, are these problems NP-hard?

Acknowledgments. We thank Vida Dujmović for valuable discussion.

References

1. Ackerman, E.: On the maximum number of edges in topological graphs with no four pairwise crossing edges. Discret. Comput. Geom. **41**(3), 365–375 (2009). https://doi.org/10.1007/s00454-009-9143-9
2. Ackerman, E.: On topological graphs with at most four crossings per edge. Comput. Geom. **85** (2019). https://doi.org/10.1016/j.comgeo.2019.101574
3. Ackerman, E., Fulek, R., Tóth, C.D.: On the size of graphs that admit polyline drawings with few bends and crossing angles. In: Brandes, U., Cornelsen, S. (eds.) GD 2010. LNCS, vol. 6502, pp. 1–12. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-18469-7_1
4. Ackerman, E., Tardos, G.: On the maximum number of edges in quasi-planar graphs. J. Comb. Theory Ser. A **114**(3), 563–571 (2007). https://doi.org/10.1016/j.jcta.2006.08.002
5. Ali, P., Dankelmann, P., Mukwembi, S.: The radius of k-connected planar graphs with bounded faces. Discret. Math. **312**(24), 3636–3642 (2012). https://doi.org/10.1016/j.disc.2012.08.019
6. Alon, N., Erdős, P.: Disjoint edges in geometric graphs. Discret. Comput. Geom. **4**, 287–290 (1989). https://doi.org/10.1007/BF02187731
7. Angelini, P., et al.: Simple k-planar graphs are simple (k+1)-quasiplanar. J. Comb. Theory Ser. B **142**, 1–35 (2020). https://doi.org/10.1016/j.jctb.2019.08.006
8. Auer, C., et al.: Outer 1-planar graphs. Algorithmica **74**(4), 1293–1320 (2016). https://doi.org/10.1007/s00453-015-0002-1
9. Avital, S., Hanani, H.: Graphs. Gilyonot Lematematika **3**, 2–8 (1966)
10. Bekos, M.A., Cornelsen, S., Grilli, L., Hong, S., Kaufmann, M.: On the recognition of fan-planar and maximal outer-fan-planar graphs. Algorithmica **79**(2), 401–427 (2017). https://doi.org/10.1007/s00453-016-0200-5
11. Bekos, M.A., Kaufmann, M., Raftopoulou, C.N.: On optimal 2- and 3-planar graphs. In: SoCG. LIPIcs, vol. 77, pp. 16:1–16:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017). https://doi.org/10.4230/LIPIcs.SoCG.2017.16
12. Binucci, C., et al.: Algorithms and characterizations for 2-layer fan-planarity: from caterpillar to stegosaurus. J. Graph Algorithms Appl. **21**(1), 81–102 (2017). https://doi.org/10.7155/jgaa.00398
13. Binucci, C., et al.: Fan-planarity: properties and complexity. Theor. Comput. Sci. **589**, 76–86 (2015). https://doi.org/10.1016/j.tcs.2015.04.020
14. Bodendiek, R., Schumacher, H., Wagner, K.: Über 1-optimale graphen. Math. Nachr. **117**, 323–339 (1984)
15. Bollobás, B.: Extremal Graph Theory. Academic Press, New York (1978)
16. Bose, P., Kirkpatrick, D.G., Li, Z.: Worst-case-optimal algorithms for guarding planar graphs and polyhedral surfaces. Comput. Geom. **26**(3), 209–219 (2003). https://doi.org/10.1016/S0925-7721(03)00027-0
17. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice-Hall, Hoboken (1999)
18. Di Giacomo, E., Didimo, W., Liotta, G., Meijer, H.: Area, curve complexity, and crossing resolution of non-planar graph drawings. Theory Comput. Syst. **49**(3), 565–575 (2011). https://doi.org/10.1007/s00224-010-9275-6
19. Didimo, W.: Density of straight-line 1-planar graph drawings. Inf. Process. Lett. **113**(7), 236–240 (2013). https://doi.org/10.1016/j.ipl.2013.01.013
20. Didimo, W.: Right angle crossing drawings of graphs. In: Hong, S.-H., Tokuyama, T. (eds.) Beyond Planar Graphs, pp. 149–169. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-6533-5_9

21. Didimo, W., Eades, P., Liotta, G.: Drawing graphs with right angle crossings. Theor. Comput. Sci. **412**(39), 5156–5166 (2011). https://doi.org/10.1016/j.tcs.2011.05.025
22. Didimo, W., Liotta, G., Montecchiani, F.: A survey on graph drawing beyond planarity. ACM Comput. Surv. **52**(1), 4:1–4:37 (2019). https://doi.org/10.1145/3301281
23. Du Preez, B.: Plane graphs with large faces and small diameter. Australas. J. Comb. **80**(3), 401–418 (2021)
24. Dujmovic, V., Gudmundsson, J., Morin, P., Wolle, T.: Notes on large angle crossing graphs. Chicago J. Theor. Comput. Sci. **2011** (2011)
25. Hong, S.-H.: Beyond planar graphs: introduction. In: Hong, S.-H., Tokuyama, T. (eds.) Beyond Planar Graphs, pp. 1–9. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-6533-5_1
26. Hong, S., Eades, P., Katoh, N., Liotta, G., Schweitzer, P., Suzuki, Y.: A linear-time algorithm for testing outer-1-planarity. Algorithmica **72**(4), 1033–1054 (2015). https://doi.org/10.1007/s00453-014-9890-8
27. Hong, S., Kaufmann, M., Kobourov, S.G., Pach, J.: Beyond-planar graphs: algorithmics and combinatorics (Dagstuhl Seminar 16452). Dagstuhl Rep. **6**(11), 35–62 (2016). https://doi.org/10.4230/DagRep.6.11.35
28. Kobourov, S.G., Liotta, G., Montecchiani, F.: An annotated bibliography on 1-planarity. Comput. Sci. Rev. **25**, 49–67 (2017). https://doi.org/10.1016/j.cosrev.2017.06.002
29. Kupitz, Y.S.: Extremal problems in combinatorial geometry. Lecture notes series, Matematisk institut, Aarhus universitet (1979)
30. Lan, Y., Shi, Y., Song, Z.: Extremal h-free planar graphs. Electron. J. Comb. **26**(2), 2 (2019). https://doi.org/10.37236/8255
31. Pach, J.: Geometric graph theory. In: Handbook of Discrete and Computational Geometry, 2nd edn., pp. 219–238. Chapman and Hall/CRC (2004). https://doi.org/10.1201/9781420035315.ch10
32. Pach, J., Radoicic, R., Tardos, G., Tóth, G.: Improving the crossing lemma by finding more crossings in sparse graphs. Discret. Computat. Geom. **36**(4), 527–552 (2006). https://doi.org/10.1007/s00454-006-1264-9
33. Pach, J., Töröcsik, J.: Some geometric applications of Dilworth's theorem. Discret. Comput. Geom. **12**, 1–7 (1994). https://doi.org/10.1007/BF02574361
34. Pach, J., Tóth, G.: Graphs drawn with few crossings per edge. Combinatorica **17**(3), 427–439 (1997). https://doi.org/10.1007/BF01215922
35. Suk, A., Walczak, B.: New bounds on the maximum number of edges in k-quasi-planar graphs. Comput. Geom. **50**, 24–33 (2015). https://doi.org/10.1016/j.comgeo.2015.06.001
36. Suzuki, Y.: 1-planar graphs. In: Hong, S.-H., Tokuyama, T. (eds.) Beyond Planar Graphs, pp. 47–68. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-6533-5_4

Computational Complexity of Covering Colored Mixed Multigraphs with Degree Partition Equivalence Classes of Size at Most Two (Extended Abstract)

Jan Bok[1,3](✉), Jiří Fiala[2], Nikola Jedličková[2], Jan Kratochvíl[2](✉), and Michaela Seifrtová[2]

[1] Computer Science Institute, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

[2] Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic
{fiala,jedlickova,honza,mikina}@kam.mff.cuni.cz

[3] Université Clermont Auvergne, CNRS, Clermont Auvergne INP, Mines Saint-Étienne, LIMOS, 63000 Clermont-Ferrand, France
jan.bok@uca.fr

Abstract. The notion of graph covers (also referred to as locally bijective homomorphisms) plays an important role in topological graph theory and has found its computer science applications in models of local computation. For a fixed target graph H, the H-Cover problem asks if an input graph G allows a graph covering projection onto H. Despite the fact that the quest for characterizing the computational complexity of H-Cover had been started more than 30 years ago, only a handful of general results have been known so far.

In this paper, we present a complete characterization of the computational complexity of covering colored graphs for the case that every equivalence class in the degree partition of the target graph has at most two vertices. We prove this result in a very general form. Following the lines of current development of topological graph theory, we study graphs in the most relaxed sense of the definition - the graphs are mixed (they may have both directed and undirected edges), may have multiple edges, loops, and semi-edges. We show that a strong P/NP-co dichotomy holds true in the sense that for each such fixed target graph H, the H-Cover problem is either polynomial time solvable for arbitrary inputs, or NP-complete even for simple input graphs.

1 Introduction

The notion of *graph covers* stems from topology and is viewed as a discretization of the notion of covers of topological spaces. Apart from being used in combinatorics as a tool for constructing large highly symmetric graphs [3–6], this notion has found computer science applications in the theory of local computation [2,13–15,17,30]. In this paper we aim to contribute to the kaleidoscope of

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 101–115, 2023.
https://doi.org/10.1007/978-3-031-43380-1_8

results about computational complexity of graph covers. We first briefly comment on the known results and show where our main result is placed among them. The formal definitions of graphs under consideration (Definition 1) and of graph covering projections (Definitions 2 and 3) are presented in Sect. 2, as well as the detailed definition of the so called degree reducing reduction (Definition 4), the concept of the degree partition of a graph (Proposition 1) and identification of several special graphs which play the key role in our characterization in Theorem 2 (Definition 5).

Despite the efforts and attention that graph covers received in the computer science community, their computational complexity is still far from being fully understood. Bodlaender [9] proved that deciding if one graph covers another one is an NP-complete problem, if both graphs are part of the input. Abello et al. [1] considered the variant when the target graph, say H, is fixed, i.e., a parameter of the problem, and the question is if an input graph covers H (this decision problem will be referred to as H-Cover). They showed examples of graphs H for which the problem is polynomial time solvable as well as examples for which it is NP-complete, but most importantly, they were the first to formulate the goal of a complete characterization of the computational complexity of the H-Cover problem, depending on the target graph H. Some of the explicit questions of Abello et al. [1] were answered by Kratochvíl et al. in [25,27], some of the NP-hardness results have been strengthened to planar input graphs by Bílka et al. [8]. A connection to a generalization of the Frequency Assignment Problem has been identified through partial covers [7,18,21]. The computationally even more sophisticated problem of *regular covers* has been treated in [19]. In a recent paper [11], the authors initiated the study of the complexity of H-Cover for graphs that allow multiple edges and loops, and also semi-edges. This is motivated by the recent development of topological graph theory where it has now become standard to consider this more general model of graphs [29,31–34]. The graphs with semi-edges were also introduced and used in mathematical physics, e.g. by Getzler and Karpanov [22]. It should be pointed out right away that considering loops, multiple edges and directed edges was shown necessary already in [26], where it is proved that in order to fully understand the computational complexity of H-Cover for *simple* undirected graphs H (i.e., undirected graphs without multiple edges, loops, and semi-edges), it is necessary and sufficient to understand the complexity of the problem for colored mixed multigraphs of minimum degree greater than 2. All papers from that era restrict their attention to covers of connected graphs. Disconnected target graphs are carefully treated in detail only in [10], where it is argued that the right way to define covers of disconnected graphs is to request that the preimages of all vertices have the same size. Such covers are called *equitable covers* in [10], and in the current paper we adopt this view and require graph covers to be equitable in case of covering disconnected graphs.

Apart from several isolated results (which also include a complete characterization of the complexity of H-Cover for connected simple undirected graphs

H with at most 6 vertices [25]), the following general results have been known about the complexity of H-Cover for infinite classes of graphs:

1. Polynomial time solvability of H-Cover for connected simple undirected graphs H which have at most two vertices in every equivalence class of their degree partitions [25].
2. NP-completeness of H-Cover for regular simple undirected graphs H of valency at least three [20,27].
3. Complete characterization of the complexity of H-Cover for undirected (multi)graphs H (without semi-edges) on at most three vertices [28].
4. Complete characterization of the complexity of H-Cover for colored mixed (multi)graphs H on at most two vertices [26] (for graphs without semi-edges) and [11] (with semi-edges allowed).

It turns out that so far all the known NP-hard instances of H-Cover remain NP-hard for simple graphs on input. This has led Bok et al. to formulating the following conjecture.

Strong Dichotomy Conjecture for Graph Covers [12]**.** *For every graph H, the H-Cover problem is either polynomial time solvable for arbitrary input graphs, or it is NP-complete for simple graphs as input.*

The main result of our paper is a complete characterization of the computational complexity of H-Cover for graphs H, each of whose equivalence classes of the degree partition has at most 2 vertices. This provides a common generalization of results 1 and 4.

Theorem 1. *The H-Cover problem satisfies Strong Dichotomy for graphs H such that each equivalence class of the degree partition has at most 2 vertices.*

The actual characterization is somewhat technical and it follows from Theorem 2 in Sect. 2, presented after the formal definitions of all the notions and special graphs that are needed for it. The characterization goes much farther beyond the motivating results from [25,26]. The main novel points are the following:

- For simple graphs H, the H-Cover problem is always polynomial time solvable (if H has all equivalence classes of size at most 2), while for general graphs, already graphs with 2 vertices may define NP-complete cases (and even graphs with 1 vertex when semi-edges are allowed).
- For simple graphs H, the polynomial time algorithm is based on 2-Satisfiability, while in case of general graphs, our polynomial time algorithm is a blend of 2-Satisfiability and Perfect Matching algorithms; this is somewhat surprising, since these two approaches are known to be incompatible in some other situations.
- The NP-complete cases are proved for simple input graphs, which is in line with the Strong Dichotomy Conjecture as stated in [12] (in contrast to many previous results which allowed multiple edges and loops in the input graphs).

2 Preliminaries

2.1 Definitions

Throughout the paper we will be working with the most general notion of a *graph* which allows multiple edges, loops, directed edges and also semi-edges and whose elements – both edges and vertices – are colored. A semi-edge is a pendant edge, incident to just one vertex (and adding just 1 to the degree of this vertex, unlike the loop, which adds 2 to the degree). In figures, semi-edges are depicted as lines with one loose end, the other one being the vertex incident to the semi-edge. To avoid any possible confusion, we present a formal definition.

Definition 1. *A* graph *is a quadruple* $G = (V, \Lambda, \iota, c)$*, where* V *is a (finite) set of* vertices*,* $\Lambda = \overline{E} \cup \overrightarrow{E} \cup \overline{L} \cup \overrightarrow{L} \cup S$ *is the set of edges of* G*,* $\iota : \Lambda \longrightarrow \binom{V}{2} \cup (V \times V) \cup V$ *is the incidence mapping of edges, and* $c : V \cup E \longrightarrow C$ *is a coloring of the vertices and edges. The edges of* $\overline{E}$ *are called* normal undirected edges *and they satisfy* $\iota(e) \in \binom{V}{2}$*, the edges of* $\overrightarrow{E}$ *are* normal directed edges *(and* $\iota(e) \in (V \times V) \setminus \{(u, u) : u \in V\}$*), the edges of* $\overline{L}$ *(*$\overrightarrow{L}$*) are* undirected (directed, *respectively)* loops *(and we have* $\iota(e) \in V$ *in both cases), and finally the edges of* S *are called* semi-edges *(and again* $\iota(e) \in V$*).*

The vertex set and edge set of a graph G will be denoted by $V(G)$ and $\Lambda(G)$, respectively, and a similar notation will be used for $\overline{E}(G), \overrightarrow{E}(G), \overline{L}(G), \overrightarrow{L}(G)$ and $S(G)$. Since we can distinguish vertices from edges, and directed edges from the undirected ones, we assume without loss of generality that colors of vertices, of directed edges and of undirected ones are different. However, we allow directed loops and directed normal edges to have the same color, as well as undirected normal edges, undirected loops and semi-edges. Edges with the same incidence function are called *parallel*. A graph is called *simple* if it has no parallel edges, no pair of opposite directed normal edges, no loops and no semi-edges. When talking about a disjoint union of graphs, we assume that the graphs are vertex (and therefore also edge) disjoint. The following definition presents the main notion of the paper.

Definition 2. *Let* G *and* H *be connected graphs colored by the same sets of colors. A* covering projection *from* G *to* H *is a pair of color-preserving mappings* $f_V : V(G) \longrightarrow V(H)$*,* $f_E : \Lambda(G) \longrightarrow \Lambda(H)$ *such that*

- *the preimage of an undirected normal edge of* H *incident with vertices* $u, v \in V(H)$ *is a perfect matching in* G *spanning* $f^{-1}(u) \cup f^{-1}(v)$*, each edge of the matching being incident with one vertex of* $f^{-1}(u)$ *and with one vertex of* $f^{-1}(v)$*;*
- *the preimage of a directed normal edge of* H *leading from a vertex* $u \in V(H)$ *to a vertex* $v \in V(H)$ *is a perfect matching in* G *spanning* $f^{-1}(u) \cup f^{-1}(v)$*, each edge of the matching being oriented from a vertex of* $f^{-1}(u)$ *to a vertex of* $f^{-1}(v)$*;*

- *the preimage of an undirected loop of H incident with a vertex $u \in V(H)$ is a disjoint union of cycles in G spanning $f^{-1}(u)$;*
- *the preimage of a directed loop of H incident with a vertex $u \in V(H)$ is a disjoint union of directed cycles in G spanning $f^{-1}(u)$; and*
- *the preimage of a semi-edge of H incident with a vertex $u \in V(H)$ is a disjoint union of semi-edges and normal edges spanning $f^{-1}(u)$ (each vertex of $f^{-1}(u)$ being incident to exactly one semi-edge and no normal edges, or exactly one normal edge and no semi-edges, from the preimage).*

We say that G *covers* H, and write $G \longrightarrow H$, if there exists a covering projection from G to H. Informally speaking, if G covers H via a covering projection (f_V, f_E) and if an agent moves along the edges of G and in every moment sees only the label $f_V(u)$ (or $f_E(e)$) of the vertex (edge) he/she is currently visiting, plus the labels of the incident edges (vertices, respectively), then the agent cannot distinguish whether he/she is moving through the covering graph G or the target graph H. Mind the significant difference between undirected loops and semi-edges. The presence of an undirected loop incident with a vertex, say u, means that there are two ways how to move from u to u along this loop, while for a semi-edge, there is just one way. The same holds true for their preimages in covering projections (undirected cycles, or isolated edges). An example of a graph and a possible cover is depicted in Fig. 1 right.

In [11], a significant role of semi-edges was noted. A color-preserving vertex-mapping $f_V : V(G) \longrightarrow V(H)$ is called *degree-obedient* if for any edge color α, any vertex $u \in V(G)$ and any vertex $x \in V(H)$, the number of edges of color α that lead from u to a vertex from $f_V^{-1}(x)$ in G is the same as the number of edges of color α leading from $f_V(u)$ to x in H, counting those edges that may map onto each other in a covering projection (e.g., if $x = f_V(u)$ and H has ℓ undirected loops and s semi-edges incident with x, and u is incident with k loops, n normal undirected edges with both end-vertices in $f_V^{-1}(x)$ and t semi-edges, then $t \leq s$ and $2k+n+t = 2\ell+s$; analogously for other types of edges). It is proved in [11] that every degree-obedient vertex-mapping extends to a covering projection if H has no semi-edges, and also when G has no semi-edges and is bipartite.

It follows straightforwardly from the definition of graph covering that the preimages of any two vertices have the same size. For disconnected graphs, we add this requirement to the definition.

Definition 3. *Let G and H be graphs and let $f = (f_V, f_E)\colon G \longrightarrow H$ be a pair of incidence-compatible color-preserving mappings. Then f is a* covering projection *of G to H if for each component G_i of G, the restricted mapping $f|_{G_i} : G_i \longrightarrow H$ is a covering projection of G_i onto some component of H, and for every two vertices $u, v \in V(H)$, $|f^{-1}(u)| = |f^{-1}(v)|$.*

Another notion we need to recall is that of the *degree partition* of a graph. This is a standard notion for simple undirected graphs, cf. [16], and it can be naturally generalized to graphs in general. A partition of the vertex set of a graph G is *equitable* if every two vertices of the same class of the partition a) have the same color, and b) have the same number of neighbors along edges of the same color in every class (including its own). The *degree partition* of a graph is then the coarsest equitable partition. It can be found in polynomial time, and moreover, a canonical linear ordering of the classes of the degree partition comes out from the algorithm. Let $V(G) = \bigcup_{i=1}^{k} V_i$ be the degree partition of G, in the canonical ordering. The *degree refinement matrix* of G is a $k \times k$ matrix M_G whose entries are vectors indexed by edge colors expressing that every vertex $u \in V_i$ has $M_{i,j,c}$ neighbors in V_j along edges of color c (if $i = j$ and c is a color of directed edges, then every vertex $u \in V_i$ has $M_{i,j,c}$ in-neighbors and $M_{i,j,c}$ out-neighbors in V_i along edges of color c). The following is proved in [26] for graphs without semi-edges, the extension to graphs with semi-edges is straightforward.

Proposition 1. *Let G and H be graphs and let $V(G) = \bigcup_{i=1}^{k} V_i$ and $V(H) = \bigcup_{i=1}^{\ell} W_i$ be the degree partitions of their vertex sets, in the canonical orderings. If G covers H, then $k = \ell$, the degree refinement matrices of G and H are equal, and for any covering projection $f : G \longrightarrow H$, $f(V_i) = W_i$ holds true for every $i = 1, 2, \ldots, k$.*

The classes of the degree partition will be further referred to as *blocks*. Once we have determined the degree partition of a graph, we will re-color the vertices so that vertices in different blocks are distinguished by vertex-colors (representing the membership to blocks), and recolor and de-orient the edges so that edges connecting vertices from different blocks are undirected and so that for any edge color, either all edges of this color belong to the same block, or they are connecting vertices from the same pair of blocks. The degree partition will remain unchanged after such a re-coloring.

A *block graph* of G is a subgraph G' of G whose vertex set is the union of some blocks of G, and such that for every edge color α, G' either contains all edges of color α that G contains, or none. A block graph G' of G is *induced* if G' contains all edges of G on the vertices of $V(G')$. A block graph is *monochromatic* if it contains edges of at most one color. A *uniblock graph* is a block graph whose vertex set is a single block of G. An *interblock graph* of G is a block graph whose vertices belong to two blocks of G, and each of its edges is incident with vertices from both blocks (i.e., with one vertex from each block).

As a local bijection, any graph covering maintains vertex degrees. In particular, vertices of degree one are mapped onto vertices of degree one and once we choose the image of such a vertex, the image of its neighbor is uniquely determined. Applied inductively, this proves the well known fact that the only connected cover of a (rooted) tree is an isomorphic copy of the tree itself. (Note here, that by definition a tree is a connected graph that does not contain cycles, parallel edges, oppositely oriented directed edges, loops, and semi-edges.) As a special case, the only connected cover of a path is the path itself. These observations are the basis of the following degree reducing reduction which has been introduced in [26] for graphs without semi-edges, and generalized to graphs with semi-edges in [10].

Definition 4. (Degree reducing reduction) *Let G be a non-tree graph.*

1. *Determine all vertices that belong to cycles in G or that are incident with semi-edges or that lie on paths connecting aforementioned vertices. Determine all maximal subtrees pending on these vertices. Determine the isomorphism types of these subtrees, introduce a new vertex color for each isomorphism type, delete each subtree and color its root by the color corresponding to the ismomorphism type of the deleted tree. In this way we obtain a graph with minimum degree at least 2 (or a single-vertex graph).*
2. *Determine all maximal paths with at least one end-vertex of degree greater than 2 and all inner vertices being of degree exactly 2. (For this step, a cycle is viewed as a path whose end-vertices are equal.) Determine all color patterns of the sequences of vertex colors, edge colors and edge directions along such paths, and introduce a new color for each such pattern. Replace each such path by a new edge of this color as follows:*
   - *2.1. If both end-vertices of the path are of degree greater than 2 and the color pattern, say π, is symmetric, the path gets replaced by an undirected edge (or loop) of color π.*
   - *2.2. If both end-vertices of the path are of degree greater than 2 and the color pattern π is asymmetric, the path gets replaced by a directed edge (or loop) of color π.*
   - *2.3. If the path ends with a semi-edge (the other end of the path must be a vertex of degree greater than 2), replace it by a semi-edge incident with its end-vertex of degree greater than 2, and color it with color $\pi\alpha$, where π is the color pattern along the path without the ending semi-edge, and α is the color of the semi-edge. In this case, consider the colors corresponding to $\pi\alpha$ (on one sided open paths) and $\pi\alpha\pi^{-1}$ on symmetric paths ending with vertices of degree greater than 2 on both sides, as the same color (this enables a path of color pattern $\pi\alpha\pi^{-1}$ be mapped on the one sided open path in a covering projection).*

Denote the resulting graph by G. Note that G is a path or a cycle (if Step 2 was void) or has minimum degree greater than 2.

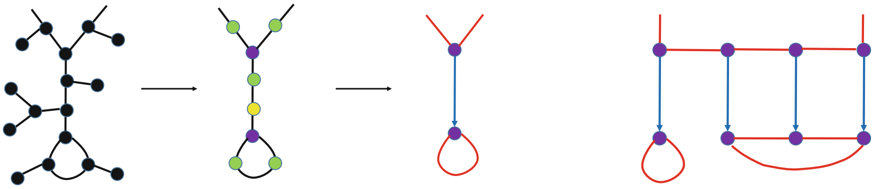


Fig. 1. An example of the application of the degree reducing reduction. An example of a graph cover of the reduced graph is depicted in the right.

The reduced graph can be constructed in polynomial time. The usefulness of this reduction is observed in [26] and [10]:

Observation 1. *Given graphs G and H, perform the degree reducing reduction on both of them simultaneously. Then $G \longrightarrow H$ if and only if $G \longrightarrow H$.*

Finally, for a subset $W \subseteq V(G)$, we denote by $G[W]$ the subgraph induced by W. If α is an edge color, then G^α denotes the spanning subgraph of G containing exactly the edges of color α.

2.2 Our Results

In order to describe the results, we introduce the formal notation of certain small graphs. We denote by

- $F(b, c)$ the one-vertex graph with b semi-edges and c loops;
- $FD(c)$ the one-vertex graph with c directed loops;
- $W(k, m, \ell, p, q)$ the two-vertex graph with ℓ parallel undirected edges joining its two vertices and with k (q) semi-edges and m (p) undirected loops incident with one (the other one, respectively) of its vertices;
- $WD(m, \ell, m)$ the directed two-vertex graph with m directed loops incident with each of its vertices, the two vertices being connected by ℓ directed edges in each direction;
- $FF(c)$ the two-vertex graph connected by c parallel undirected edges, with the two vertices being distinguishable to belong to different blocks;
- $FW(b)$ the three-vertex graph with bundles of b parallel edges connecting one vertex to each of the remaining two; and
- $WW(b, c)$ the graph on four vertices obtained from a 4-cycle by replacing the edges of a perfect matching by bundles of b parallel edges, and replacing the edges of the complementary matching by bundles of c parallel edges, the two independent sets of size 2 belonging to different blocks.

Edges of all of these graphs are uncolored (or, equivalently, monochromatic). We shall only consider W graphs having $k + 2m = 2p + q$. See the illustration in Fig. 2 for the graphs defined here.

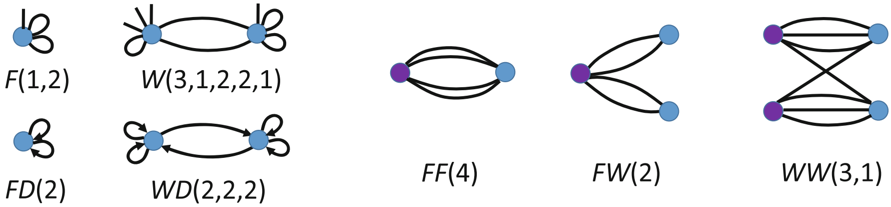


Fig. 2. Examples of the small graphs we are considering.

Definition 5. *For the convenience of the reader, the maximal harmless monochromatic uniblock and interblock graphs are depicted in Fig. 3. A regular monochromatic uniblock graph with at most two vertices is called*

- harmless *if it is isomorphic to* $F(b,0)$, $b \leq 2$, $F(1,c)$, $F(0,c)$, $FD(c)$, $W(2,0,0,0,2)$, $W(2,0,0,1,0)$, $W(0,c,0,c,0)$, $W(1,c,0,c,1)$, $W(0,0,c,0,0)$, $W(1,0,1,0,1)$, $WD(c,0,c)$, $WD(0,c,0)$, $WD(1,1,1)$ *(c being an arbitrary nonnegative integer),*
- harmful *if it is isomorphic to* $F(b,c)$ *such that* $b \geq 2$ *and* $b+c \geq 3$, *or to* $W(k,m,\ell,p,q)$ *such that* $\ell \geq 1$ *and* $k+2m+\ell = q+2p+\ell \geq 3$, *or to the disjoint union of* $F(b,c)$ *and* $F(b',c')$ *such that at least one of them is harmful, or to* $WD(c,b,c)$ *such that* $b \geq 1, c \geq 1$ *and* $b+c \geq 3$.

A monochromatic interblock graph is called

- harmless *if it is isomorphic to* $FF(c)$ *or* $WW(0,c)$ *(with c being an arbitrary nonnegative integer), or to* $FW(0)$, $FW(1)$, *or* $WW(1,1)$,
- dangerous *if it is isomorphic to* $FW(2)$, *and*
- harmful *if it is isomorphic to* $FW(c)$ *for* $c \geq 3$, *or to* $WW(b,c)$ *such that* $b \geq 1$, $c \geq 1$ *and* $b+c \geq 3$.

Note that under the assumption that each degree partition equivalence class has size at most two, every monochromatic uniblock graph as well as every monochromatic interblock graph fall in exactly one of the above described categories. The choice of the terminology is explained by the following theorem.

Theorem 2. *Suppose all blocks of a graph H have sizes at most 2. Then the following hold true:*

1. *If all monochromatic uniblock and interblock graphs of H are harmless, then the H-*Cover *problem is solvable in polynomial time (for arbitrary input graphs).*
2. *If at least one of the monochromatic uniblock or interblock graphs of H is harmful, then the H-*Cover *problem is NP-complete even for simple input graphs.*
3. *If the minimum degree of H is greater than 2 and H contains a dangerous monochromatic interblock graph, then the H-*Cover *problem is NP-complete even for simple input graphs.*

Observe that Theorem 2 implies that H-Cover is polynomial time solvable if and only if every monochromatic uniblock graph defines a polynomial time solvable instance and the monochromatic interblock graphs are such that either each vertex has at most one neighbor, or each vertex has degree at most two. For the interblock graphs, this is also very close to saying that each monochromatic interblock graph itself defines a polynomial time solvable instance, but not quite. The one and only exception is the graph $FW(2)$. Indeed, $FW(2)$-Cover is polynomial time solvable (since it reduces to $F(2)$-Cover), but with the additional condition that all vertices have degrees greater than 2, the presence of $FW(2)$ in H leads to NP-completeness of H-Cover (this will be shown in detail in Sect. 4).

3 Proof of Theorem 2 - Polynomial Cases

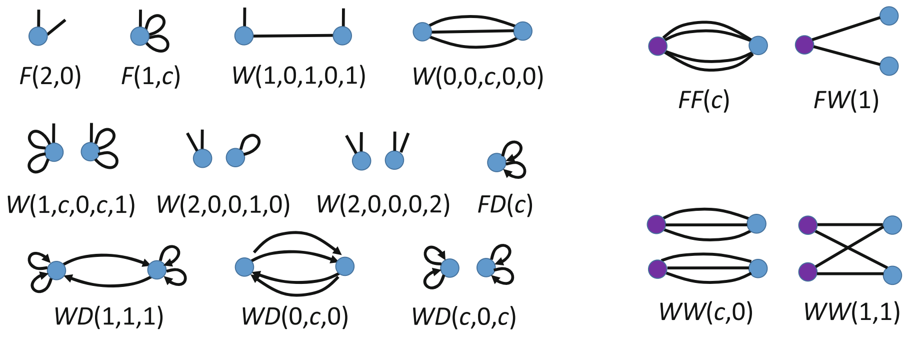


Fig. 3. The maximal harmless monochromatic uniblock (left) and interblock (right) graphs (c is an arbitrary non-negative integer).

Here we sketch an algorithm that proves Part 1 of Theorem 2. It clearly runs in polynomial time, the details can be found in the journal version of the paper.

Algorithm

1. Compute the degree partitions of G (the input graph) and H (the target graph). Reorder the equivalence classes W_i of the degree partition of H so that $W_1, \dots, W_s$ are singletons and $W_{s+1}, \dots, W_k$ contain two vertices each, and reorder the degree partition equivalence classes V_i of the input graph G accordingly. Denote further, for every $i = 1, \dots, s$, by a_i the vertex in W_i, and, for every $i = s+1, \dots, k$, by b_i, c_i the vertices of W_i.
2. Check that the degree refinement matrices of G and H are indeed the same.
3. Decide if the edges within $G[V_i]$ can be mapped onto the edges of $H[W_i]$ to form a covering projection, for each $i = 1, 2, \dots, s$. (This step amounts to checking degrees and the numbers of semi-edges incident with the vertices, as well as checking that monochromatic subgraphs contain perfect matchings in case of semi-edges in the target graph).
4. Preprocess the two-vertex equivalence classes W_i, $i = s+1, \dots, k$ when $H[W_i]$ contains semi-edges (this may impose conditions on some vertices of V_i, whether they can map on b_i or c_i).
5. Using 2-Satisfiability, find a degree-obedient vertex mapping from V_i onto W_i for each $i = s+1, \dots, k$, which fulfills the conditions observed in Step 4. (For every vertex $u \in V_i$, introduce a variable x_u with the interpretation that x_u is true if u is mapped onto b_i and it is false when x_u is mapped onto c_i. The harmless block graphs are such that either all neighbors of a vertex u must be mapped onto the same vertex, and thus the value of the corresponding variables are all the same (e.g., for $WW(0, c)$), or u has exactly two neighbors which should map onto different vertices (e.g., for $WW(1, 1)$)

meaning that the corresponding variables must get opposite values. All the situations that arise from harmless block graphs can be described by clauses of size 2).

6. Complete the covering projection by defining the mapping on edges in case a degree-obedient vertex mapping was found in Step 5, or conclude that G does not cover H otherwise. (The existence and polynomial time constructability of covering projections from degree-obedient vertex mappings for such instances have been proven in [11]).

4 Proof of Theorem 2 - NP-Hard Cases

The proof is technical and involves several NP-hardness reductions. We will provide an overview of its main steps, the details will appear in the full version of the paper.

Step 1. We first argue that H-Cover restricted to simple input graphs is NP-complete for every harmful uniblock or interblock graph H. These cases have been proved previously in [1,10,11,26,28], however, some of them only for input graphs allowing parallel edges. An extra care was thus needed to strengthen the NP-hardness results for simple input graphs.
Step 2. Covering the dangerous graph $FW(2)$ itself is polynomial time decidable, since red$FW(2) = F(2)$ is harmless. However, if $FW(2)$ is a monochromatic interblock graph of H, all vertices of H have degrees greater than 2, and H does not contain any harmful monochromatic uniblock or interblock graph, then H contains a block graph which is reducible to one of the graphs from Fig. 4 (if subscript k is used in the name of a graph from this figure, it refers to the number of parallel red edges or loops). This can be proven by a straightforward case analysis.
Step 3. For every graph H from Fig. 4, H-Cover is NP-complete for simple input graphs (for those graphs indexed by k, we claim the statement for every $k \geq 3$ in case of H_k and H'_k, for every $k \geq 2$ in case of B'_k, for every $k \geq 1$ in case of $B_k, C'_k, D_k, D'_k, L_k, L'_k, M_k$ and M'_k, and for every $k \geq 0$ in case of C_k). We prove this by reductions from Monotone 2-in-4-Satisfiability which is known to be NP-complete [23]. The 25 graphs can be grouped into a few groups which are handled en bloc by unified reductions tailored on the groups.
Step 4. The last step is to show that H-Cover for simple input graphs polynomially reduces to H'-Cover for simple input graphs, when H is a block graph of H', and H is a harmful graph or one of the graphs from Fig. 4. This is a step which is usually called the garbage collection. We describe a way how to construct a simple graph G' from a simple graph G (an input of H-Cover), so that $G' \longrightarrow H'$ if and only if $G \longrightarrow H$. Note that this is somewhat simpler in case when H is balanced in the sense that in every two-vertex block and for every edge color, both vertices are incident with the same number of semi-edges of this color.

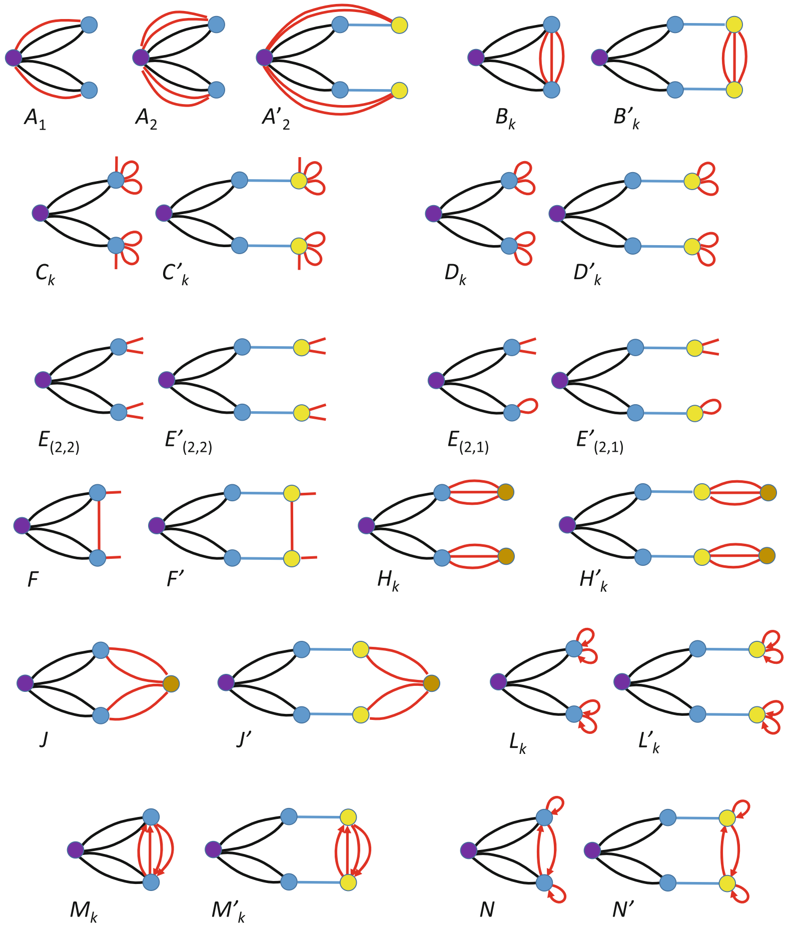


Fig. 4. Block graphs forced by $FW(2)$.

5 Proof of Theorem 1

Suppose H is a connected graph each of whose equivalence classes of the degree partition has at most 2 vertices. The H-Cover problem can be solved in constant or linear time if H is a tree or a cycle or a path (possibly ending with semi-edges). Otherwise, consider the reduced graph H, reduced via the degree reducing reduction of Definition 4. It is important that H also has at most two vertices in each equivalence class of its degree partition. If H is a path or a cycle, then H-Cover is solvable in polynomial time, and so is H-Cover, due to Observation 1.

If H contains a vertex of degree greater than 2, then all vertices of H have degrees greater than 2. If all monochromatic uniblock and interblock graphs of H are harmless, then H-Cover is polynomially solvable for general input graphs, and so is H-Cover, due to Observation 1. If H contains a harmful or a dangerous uniblock or interblock graph, then H-Cover is NP-complete for simple input graphs by Parts 2 and 3 of Theorem 2. If G is a simple graph as an input to the H-Cover problem, the reverse operation to the degree reducing reduction gives a simple graph G such that $G \longrightarrow H$ if and only if $G \longrightarrow H$. Hence H-Cover is also NP-complete for simple input graphs.

6 Concluding Remarks

The polynomial algorithm of Sect. 3 combines two approaches - finding perfect matchings and solving 2-Satisfiability. It is well known that these two problems are polynomial time solvable. It may be somewhat surprising that so is their combination, e.g., in comparison with the so called compatible 2-factor problem [24], whose instances solvable in polynomial time are of two types, one solved by a reduction to perfect matching, the other one solved by 2-Satisfiability, but if restrictions of both types are present in the same instance, the problem becomes NP-complete.

Note further that the polynomiality of the polynomial time solvable case does not depend on the target graph being fixed. If H is a graph with at most 2 vertices in each block of the degree partition, and all monochromatic block and interblock graphs are harmless, then the algorithm described in Sect. 3 remains polynomial time even if H is part of the input.

We believe that the method developed above has a much wider potential and we conjecture the following:

Conjecture. *Let H be a block graph of a graph H'. Then H-*Cover *for simple input graphs polynomially reduces to H'-*Cover *for simple input graphs.*

And of course, the ultimate goal is to prove (or disprove) the Strong Dichotomy Conjecture for graph covers parameterized by the target graph, ideally with a complete catalog of the polynomially solvable cases.

Acknowledgments. All co-authors were supported by research grant GAČR 20-15576S of the Czech Science Foundation, Nikola Jedličková was further supported by SVV–2020–260578 and GAUK 370122. Jan Bok was partially financed by the ANR project GRALMECO (ANR-21-CE48-0004) and the French government IDEX-ISITE initiative 16-IDEX-0001 (CAP 20-25).

References

1. Abello, J., Fellows, M.R., Stillwell, J.C.: On the complexity and combinatorics of covering finite complexes. Aust. J. Comb. **4**, 103–112 (1991)
2. Angluin, D.: Local and global properties in networks of processors. In: Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing, STOC 1980, pp. 82–93. Association for Computing Machinery, New York (1980)
3. Biggs, N.: Algebraic Graph Theory. Cambridge University Press, Cambridge (1974)

4. Biggs, N.: Covering biplanes. In: The Theory and Applications of Graphs, Fourth International Conference, Kalamazoo, pp. 73–79. Wiley (1981)
5. Biggs, N.: Constructing 5-arc transitive cubic graphs. J. Lond. Math. Soc. **II**(26), 193–200 (1982)
6. Biggs, N.: Homological coverings of graphs. J. Lond. Math. Soc. **II**(30), 1–14 (1984)
7. Bílka, O., Lidický, B., Tesař, M.: Locally injective homomorphism to the simple weight graphs. In: Ogihara, M., Tarui, J. (eds.) TAMC 2011. LNCS, vol. 6648, pp. 471–482. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20877-5_46
8. Bílka, O., Jirásek, J., Klavík, P., Tancer, M., Volec, J.: On the complexity of planar covering of small graphs. In: Kolman, P., Kratochvíl, J. (eds.) WG 2011. LNCS, vol. 6986, pp. 83–94. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25870-1_9
9. Bodlaender, H.L.: The classification of coverings of processor networks. J. Parallel Distrib. Comput. **6**, 166–182 (1989)
10. Bok, J., Fiala, J., Jedličková, N., Kratochvíl, J., Seifrtová, M.: Computational complexity of covering disconnected multigraphs. In: Bampis, E., Pagourtzis, A. (eds.) FCT 2021. LNCS, vol. 12867, pp. 85–99. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86593-1_6
11. Bok, J., Fiala, J., Hliněný, P., Jedličková, N., Kratochvíl, J.: Computational complexity of covering multigraphs with semi-edges: small cases. In: Bonchi, F., Puglisi, S.J. (eds.) 46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, Tallinn, Estonia, 23–27 August 2021. LIPIcs, vol. 202, pp. 21:1–21:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021)
12. Bok, J., Fiala, J., Jedličková, N., Kratochvíl, J., Rzażewski, P.: List covering of regular multigraphs. In: Bazgan, C., Fernau, H. (eds.) IWOCA 2022. LNCS, vol. 13270, pp. 228–242. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-06678-8_17
13. Chalopin, J.: Local computations on closed unlabelled edges: the election problem and the naming problem. In: Vojtáš, P., Bieliková, M., Charron-Bost, B., Sýkora, O. (eds.) SOFSEM 2005. LNCS, vol. 3381, pp. 82–91. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30577-4_11
14. Chalopin, J., Métivier, Y., Zielonka, W.: Local computations in graphs: the case of cellular edge local computations. Fund. Inform. **74**(1), 85–114 (2006)
15. Chalopin, J., Paulusma, D.: Graph labelings derived from models in distributed computing: a complete complexity classification. Networks **58**(3), 207–231 (2011)
16. Corneil, D.G., Gotlieb, C.C.: An efficient algorithm for graph isomorphism. J. Assoc. Comput. Mach. **17**, 51–64 (1970)
17. Courcelle, B., Métivier, Y.: Coverings and minors: applications to local computations in graphs. Eur. J. Comb. **15**, 127–138 (1994)
18. Fiala, J., Kratochvíl, J.: Complexity of partial covers of graphs. In: Eades, P., Takaoka, T. (eds.) ISAAC 2001. LNCS, vol. 2223, pp. 537–549. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45678-3_46
19. Fiala, J., Klavík, P., Kratochvíl, J., Nedela, R.: Algorithmic aspects of regular graph covers with applications to planar graphs. CoRR abs/1402.3774 (2014)
20. Fiala, J., Kratochvíl, J.: Locally constrained graph homomorphisms – structure, complexity, and applications. Comput. Sci. Rev. **2**(2), 97–111 (2008)
21. Fiala, J., Kratochvíl, J., Pór, A.: On the computational complexity of partial covers of theta graphs. Electron. Notes Discret. Math. **19**, 79–85 (2005)
22. Getzler, E., Kapranov, M.M.: Modular operads. Compos. Math. **110**(1), 65–125 (1998)

23. Kratochvíl, J.: Complexity of hypergraph coloring and Seidel's switching. In: Bodlaender, H.L. (ed.) WG 2003. LNCS, vol. 2880, pp. 297–308. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39890-5_26
24. Kratochvíl, J., Poljak, S.: Compatible 2-factors. Discret. Appl. Math. **36**(3), 253–266 (1992)
25. Kratochvíl, J., Proskurowski, A., Telle, J.A.: Complexity of graph covering problems. In: Mayr, E.W., Schmidt, G., Tinhofer, G. (eds.) WG 1994. LNCS, vol. 903, pp. 93–105. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-59071-4_40
26. Kratochvíl, J., Proskurowski, A., Telle, J.A.: Complexity of colored graph covers I. Colored directed multigraphs. In: Möhring, R.H. (ed.) WG 1997. LNCS, vol. 1335, pp. 242–257. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0024502
27. Kratochvíl, J., Proskurowski, A., Telle, J.A.: Covering regular graphs. J. Comb. Theory Ser. B **71**(1), 1–16 (1997)
28. Kratochvíl, J., Telle, J.A., Tesař, M.: Computational complexity of covering three-vertex multigraphs. Theor. Comput. Sci. **609**, 104–117 (2016)
29. Kwak, J.H., Nedela, R.: Graphs and their coverings. Lecture Notes Ser. **17**, 118 (2007)
30. Litovsky, I., Métivier, Y., Zielonka, W.: The power and the limitations of local computations on graphs. In: Mayr, E.W. (ed.) WG 1992. LNCS, vol. 657, pp. 333–345. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-56402-0_58
31. Malnič, A., Marušič, D., Potočnik, P.: Elementary abelian covers of graphs. J. Algebraic Combin. **20**(1), 71–97 (2004)
32. Malnič, A., Nedela, R., Škoviera, M.: Lifting graph automorphisms by voltage assignments. Eur. J. Comb. **21**(7), 927–947 (2000)
33. Mednykh, A.D., Nedela, R.: Harmonic Morphisms of Graphs: Part I: Graph Coverings. Vydavatelstvo Univerzity Mateja Bela v Banskej Bystrici, 1st edn. (2015)
34. Nedela, R., Škoviera, M.: Regular embeddings of canonical double coverings of graphs. J. Comb. Theory Ser. B **67**(2), 249–277 (1996)

Cutting Barnette Graphs Perfectly is Hard

Édouard Bonnet, Dibyayan Chakraborty(✉), and Julien Duron

Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1,
LIP UMR5668, Lyon, France
{edouard.bonnet,dibyayan.chakraborty,julien.duron}@ens-lyon.fr

Abstract. A *perfect matching cut* is a perfect matching that is also a cutset, or equivalently a perfect matching containing an even number of edges on every cycle. The corresponding algorithmic problem, Perfect Matching Cut, is known to be NP-complete in subcubic bipartite graphs [Le & Telle, TCS '22] but its complexity was open in planar graphs and in cubic graphs. We settle both questions at once by showing that Perfect Matching Cut is NP-complete in 3-connected cubic bipartite planar graphs or *Barnette graphs*. Prior to our work, among problems whose input is solely an undirected graph, only Distance-2 4-Coloring was known NP-complete in Barnette graphs. Notably, Hamiltonian Cycle would only join this private club if Barnette's conjecture were refuted.

1 Introduction

Deciding if an input graph admits a perfect matching, i.e., a subset of its edges touching each of its vertices exactly once, notoriously is a tractable task. There is indeed a vast literature, starting arguably in 1947 with Tutte's characterization via determinants [38], of polynomial-time algorithms deciding Perfect Matching (or returning actual solutions) and its optimization generalization Maximum Matching.

In this paper, we are interested in another containment of a spanning set of disjoint edges –perfect matching– than as a subgraph. As containing such a set of edges as an induced subgraph is a trivial property[1] (only shared by graphs that are themselves disjoint unions of edges), the meaningful other containment is as a *semi-induced subgraph*. By that we mean that we look for a bipartition of the vertex set or *cut* such that the edges of the perfect matching are "induced" in the corresponding cutset (i.e., the edges going from one side of the bipartition to the other), while we do not set any requirement on the presence or absence of edges within each side of the bipartition.

[1] Note however that the induced variant of Maximum Matching is an interesting problem that happens to be NP-complete [36].

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 116–129, 2023.
https://doi.org/10.1007/978-3-031-43380-1_9

This problem was in fact introduced as the Perfect Matching Cut (PMC for short) problem[2] by Heggernes and Telle who show that it is NP-complete [18]. As the name Perfect Matching Cut suggests, we indeed look for a perfect matching that is also a cutset. Le and Telle further show that PMC remains NP-complete in subcubic bipartite graphs of arbitrarily large girth, whereas it is polynomial-time solvable in a superclass of chordal graphs, and in graphs without a particular subdivided claw as an induced subgraph [26]. An in-depth study of the complexity of PMC when forbidding a single induced subgraph or a finite set of subgraphs has been carried out [14,28].

We look at Le and Telle's hardness constructions and wonder what other properties could make PMC tractable (aside from chordality, and forbidding a finite list of subgraphs or induced subgraphs). A simpler reduction for bipartite graphs is first presented. Let us briefly sketch their reduction (without thinking about its correctness) from Monotone Not-All-Equal 3-SAT, where given a negation-free 3-CNF formula, one seeks a truth assignment that sets in each clause a variable to true and a variable to false. Every variable is represented by an edge, and each 3-clause, by a (3-dimensional) cube with three anchor points at three pairwise non-adjacent vertices of the cube. One endpoint of the variable gadget is linked to the anchor points corresponding to this variable among the clause gadgets. Note that this construction creates three vertices of degree 4 in each clause gadget, and vertices of possibly large degree in the variable gadgets. Le and Telle then reduce the maximum degree to at most 3, by appropriately subdividing the cubes and tweaking the anchor points, and replacing the variable gadgets by cycles.

Notably the edge subdivision of the clause gadgets creates degree 2-vertices, which are not easy to "pad" with a third neighbor (even more so while keeping the construction bipartite). And indeed, prior to our work, the complexity of PMC in cubic graphs was open. Let us observe that on cubic graphs, the problem becomes equivalent to partitioning the vertex set into two sets each inducing a disjoint union of (independent) cycles. The close relative, Matching Cut, where one looks for a mere matching that is also a cutset, while NP-complete in general [5], is polynomial-time solvable in *subcubic* graphs [2,33]. The complexity of Matching Cut has further been examined in subclasses of planar graphs [2,35], when forbidding some (induced) subgraphs [13,14,28,29], on graphs of bounded diameter [25,29], and on graphs of large minimum degree [4]. Matching Cut has also been investigated with respect to parameterized complexity, exact exponential algorithms [21,24], and enumeration [16].

It was also open if PMC is tractable on planar graphs. Note that Bouquet and Picouleau [3] show that a related problem, Disconnected Perfect Matching, where one looks for a perfect matching that contains a cutset, is NP-

[2] The authors consider the framework of (k, σ, ρ)-partition problem, where k is a positive integer, and σ, ρ are sets of non-negative integers, and one looks for a vertex-partition into k parts such that each vertex of each part has a number of neighbors in its own part in σ, and a number of other neighbors in ρ; hence, PMC is then the $(2, \mathbb{N}, \{1\})$-partition problem.

complete on planar graphs of maximum degree 4, on planar graphs of girth 5, and on 5-regular bipartite graphs [3]. They incidentally call this related problem Perfect Matching Cut but subsequent references [14,26] use the name Disconnected Perfect Matching to avoid confusion. We will observe that PMC is equivalent to asking for a perfect matching containing an even number of edges from every cycle of the input graph (See Lemma 1 and 2). The sum of even numbers being even, it is in fact sufficient that the perfect matching contains an even number of edges from every element of a cycle basis. There is a canonical cycle basis for planar graphs: the bounded faces. This gives rise to the following neat reformulation of PMC in planar graphs: is there a perfect matching containing an even number of edges along each face?

While Matching Cut is known to be NP-complete on planar graphs [2, 35], it could have gone differently for PMC for the following "reasons." Not-All-Equal 3-SAT, which appears as the *right* starting point to reduce to PMC, is tractable on planar instances [32]. In planar graphs, perfect matchings are *simpler* than arbitrary matchings in that they alone [39] can be counted efficiently [20,37]. Let us finally observe that Maximum Cut can be solved in polynomial time in planar graphs [17].

In fact, we show that the reformulations for cubic and planar graphs cannot help algorithmically, by simultaneously settling the complexity of PMC in cubic and in planar graphs, with the following stronger statement.

Theorem 1. Perfect Matching Cut *is NP-hard in 3-connected cubic bipartite planar graphs.*

Not very many problems are known to be NP-complete in cubic bipartite planar graphs. Of the seven problems defined on mere undirected graphs from Karp's list of 21 NP-complete problems [19], only Hamiltonian Path is known to remain NP-complete in this class, while the other six problems admit a polynomial-time algorithm. Restricting ourselves to problems where the input is purely an undirected graph[3], besides Hamiltonian Path/Cycle [1,34], Minimum Independent Dominating Set was also shown NP-complete in cubic bipartite planar graphs [27], as well as P_3-Packing [23] (hence, an equivalent problem phrased in terms of disjoint dominating and 2-dominating sets [31]), and Distance-2 4-Coloring [11]. To our knowledge, Minimum Dominating Set is only known NP-complete in *subcubic* bipartite planar graphs [15,22].

It is interesting to note that the reductions for Hamiltonian Path, Hamiltonian Cycle, Minimum Independent Dominating Set, and P_3-Packing all produce cubic bipartite planar graphs that are *not* 3-connected. Notoriously, lifting the NP-hardness of Hamiltonian Cycle to the 3-connected case would require to disprove Barnette's conjecture[4] (and that would be indeed suffi-

[3] Among problems with edge orientations, vertex or edge weights, or prescribed subsets of vertices or edges, the list is significantly longer, and also includes Minimum Weighted Edge Coloring [7], List Edge Coloring and Precoloring Extension [30], k-In-A-Tree [8], etc.

[4] Which precisely states that every polyhedral (that is, 3-connected planar) cubic bipartite graphs admits a hamiltonian cycle.

cient [12]). Note that hamiltonicity in cubic graphs is equivalent to the existence of a perfect matching that is *not* an edge cut (i.e., whose removal is not disconnecting the graph). We wonder whether there is something inherently simpler about *3-connected* cubic bipartite planar graphs, which would go beyond hamiltonicity (assuming that Barnette's conjecture is true).

Let us call *Barnette* a 3-connected cubic bipartite planar graph. It appears that, prior to our work, Distance-2 4-Coloring was the only *vanilla* graph problem shown NP-complete in Barnette graphs [11]. Arguing that Distance-2 4-Coloring is a problem on *squares* of Barnette graphs more than it is on Barnette graphs, a case can be made for Perfect Matching Cut to be the first natural problem proven NP-complete in Barnette graphs.

Outline of the Proof. We reduce the NP-complete problem Monotone Not-All-Equal 3-SAT with exactly 4 occurrences of each variable [6] to PMC. Observe that flipping the value of every variable of a satisfying assignment results in another satisfying assignment. We thus see a solution to Monotone Not-All-Equal 3-SAT simply as a bipartition of the set of variables.

As we already mentioned, Not-All-Equal 3-SAT restricted to planar instances (i.e., where the variable-clause incidence graph is planar) is in P. We thus have to design *crossing* gadgets in addition to *variable* and *clause* gadgets. Naturally our gadgets are bipartite graphs with vertices of degree 3, except for some special *anchors*, vertices of degree 2 with one incident edge leaving the gadget.

The variable gadget is designed so that there is a unique way a perfect matching cut can intersect it. It might seem odd that no "binary choice" happens within it. The role of this gadget is only to serve as a baseline for which side of the bipartition the variable lands in, while the "truth assignments" take place in the clause gadgets. (Actually the same happens with Le and Telle's first reduction [26], where the variable gadget is a single edge, which has to be in any solution).

Our variable gadget consists of 36 vertices, including 8 anchor points; see Fig. 1. (We will later explain why we have 8 anchor points and not simply 4, that is, one for each occurrence of the variable.) Note that in all the figures, we adopt the following convention:

- black edges cannot (or can no longer) be part of a perfect matching cut,
- red edges are in every perfect matching cut,
- each blue edge e is such that at least one perfect matching cut within its gadget includes e, and at least one excludes e, and
- brown edges are blue edges that were indeed chosen in the solution.

Let us recall that PMC consists of finding a perfect matching containing an even number of edges from each cycle. Thus we look for a perfect matching M such that every path (or walk) between v and w contains a number of edges of M whose parity only depends on v and w. If this parity is even v and w are on the *same side*, and if it is odd, v and w are on *opposite sides*. The 8 anchor points of each variable gadget are forced on the same side. This is the *side of the variable*.

At the core of the clause gadget is a subdivided cube of blue edges; see Fig. 2. There are three vertices (u_1, u_8, u_{14} on the picture) of the subdivided cube that are forced on the same side as the corresponding three variables. Three perfect matching cuts are available in the clause gadget, each separating (i.e., putting on opposite sides) a different vertex of $\{u_1, u_8, u_{14}\}$ from the other two. Note that this is exactly the semantics of a not-all-equal 3-clause. We in fact need two copies of the subdivided cube, partly to increase the degree of some subdivided vertices, partly for the same reason we duplicated the anchor vertices in the variable gadgets. (The latter will be explained when we present the crossing gadgets.) Increasing the degree of all the subdivided vertices complicate further the gadget and create two odd faces. Fortunately these two odd faces have a common neighboring even face. We can thus "fix" the parity of the two odd faces by plugging the sub-gadget D_j in the even face. We eventually need a total of 112 vertices, including 6 anchor points.

Let us now describe the crossing gadgets. Basically we want to replace every intersection point of two edges by a 4-vertex cycle. This indeed propagates black edges (those that cannot be in any solution). The issue is that going through such a crossing gadget flips one's side. As we cannot guarantee that a variable "wire" has the same parity of intersection points towards each clause gadget it is linked to, we duplicate these wires. At a previous intersection point, we now have two parallel wires crossing two other parallel wires, making four crossings. The gadget simply consists of four 4-vertex cycles; see Fig. 3. This explains why we have 8 anchor points (not 4) in each variable gadget, and 6 anchor points (not 3) in each clause gadget.

2 Preliminaries

For a graph G, we denote by $V(G)$ its set of vertices and by $E(G)$ its set of edges. For $U \subseteq V(G)$, the *subgraph of G induced by U*, denoted as $G[U]$, is the graph obtained from G by removing the vertices not in U. We shall use $E_G(U)$ (or $E(U)$ when G is clear) as a shorthand for $E(G[U])$. For $M \subset E(G)$, $G - M$ is the spanning subgraph of G obtained by removing the edges in M (while preserving their endpoints). We may use *k-cycle* as a short-hand for the k-vertex cycle.

Given two disjoint sets $X, Y \subseteq V(G)$ we denote by $E(X, Y)$ the set of edges between X and Y. A set $M \subseteq E(G)$ is a *cutset*[5] of G if there is a proper bipartition $X \uplus Y = V(G)$, called *cut*, such that $M = E(X, Y)$. Note that a cut fully determines a cutset, and among connected graphs a cutset fully determines a cut. When dealing with connected graphs, we may speak of *the* cut of a cutset. For $X \subseteq V(G)$ the set of *outgoing edges of X* is $E(X, V(G) \setminus X)$. For a cutset M of a connected graph G, and $u, v \in V(G)$, we say that u and v are on the

[5] We avoid using the term "edge cut" since, for some authors, an edge cut is, more generally, a subset of edges whose deletion increases the number of connected components.

same side (resp. on *opposite sides*) of M if u and v are on the same part (resp. on different parts) of the cut of M.

A *matching* (resp. *perfect matching*) of G is a set $M \subset E(G)$ such that each vertex of G is incident to at most (resp. exactly) one edge of M. A *perfect matching cut* is a perfect matching that is also a cutset. For $M \subseteq E(G)$ and $U \subseteq V(G)$, we say that M is a *perfect matching cut of* $G[U]$ if $M \cap E(U)$ is so.

Due to space constraints, the proof of statements marked with ($\star$) are deferred to the long version [10].

3 Proof of Theorem 1

Before we give our reduction, we start with a handful of useful lemmas and observations, which we will later need.

3.1 Preparatory Lemmas

Lemma 1 ($\star$). *Let G be a graph, and $M \subseteq E(G)$. Then M is a cutset if and only if for every cycle C of G, $|E(C) \cap M|$ is even.*

Lemma 2. *Let G be a plane graph, and $M \subseteq E(G)$. Then M is a cutset if and only if for any facial cycle C of G, $|E(C) \cap M|$ is even.*

Proof. The forward implication is a direct consequence of Lemma 1. The converse comes from the known fact that the bounded faces form a cycle basis; see for instance [9]. If H is a subgraph of G, let $\tilde{H}$ be the vector of $\mathbb{F}_2^{E(G)}$ with 1 entries at the positions corresponding to edges of H. Thus, for any cycle C of G, we have $\tilde{C} = \Sigma_{1 \leqslant i \leqslant k} \tilde{F_i}$ where F_i are facial cycles of G. And $|M \cap E(C)|$ has the same parity as $\Sigma_{1 \leqslant i \leqslant k} |M \cap E(F_i)|$, a sum of even numbers. □

Lemma 3. *Let M be a perfect matching cut of a cubic graph G. Let C be an induced 4-vertex cycle of G. Then, exactly one of the following holds:*

(a) *$E(C) \cap M = \emptyset$ and the four outgoing edges of $V(C)$ belong to M.*
(b) *$|E(C) \cap M| = 2$, the two edges of $E(C) \cap M$ are disjoint, and none of the outgoing edges of $V(C)$ belongs to M.*

Proof. The number of edges of M within $E(C)$ is even by Lemma 2. Thus $|E(C) \cap M| \in \{0, 2\}$, as all four edges of $E(C)$ do not make a matching.

Suppose that $E(C) \cap M = \emptyset$. As M is a perfect matching, for every $v \in V(C)$ there is an edge in M incident to v and not in $E(C)$. As G is cubic, every outgoing edge of $V(C)$ is in M.

Suppose instead that $|E(C) \cap M| = 2$. As M is a matching, the two edges of $E(C) \cap M$ do not share an endpoint. It implies that all the four vertices of C are touched by these two edges. Thus no outgoing edge of $V(C)$ can be in M. □

Corollary 1 ($\star$). *Let M be a perfect matching of a cubic graph G. Let C_1, C_2 two vertex-disjoint induced 4-vertex cycles of G such that there is an edge between $V(C_1)$ and $V(C_2)$. Then $E(C_1) \cap M \neq \emptyset$ if and only if $E(C_2) \cap M \neq \emptyset$.*

Lemma 4. *Let M be a perfect matching cut of a cubic graph G. If a 6-cycle has three outgoing edges in M, then all six outgoing edges are in M.*

Proof. Let C be a 6-cycle. Since M is a perfect matching cut, $|E(C) \cap M|$ is even. Hence, $|E(C) \cap M|$ is either 0 or 2. If $|E(C) \cap M| = 2$, four vertices of C are touched by $E(C) \cap M$, which rules out that three outgoing edges of $V(C)$ are in M. Thus $E(C) \cap M = \emptyset$ and, G being cubic, all outgoing edges of $V(C)$ are in M. □

Lemma 5. *Let M a perfect matching cut of a cubic bipartite graph G. Suppose C is a 6-cycle $v_1v_2 \dots v_6$ of G, such that v_2v_3, v_3v_4, v_5v_6 and v_6v_1 are in some induced 4-cycles. Then $M \cap E(C) = \emptyset$.*

Proof. By applying Lemma 3 on the 4-cycle containing v_2v_3, and the one containing v_6v_1, it holds that $v_1v_2 \in M \Leftrightarrow v_3v_4 \in M \Leftrightarrow v_5v_6 \in M$. Thus none of these three edges can be in M, because C would have an odd number of edges in M. Symmetrically, no edge among v_2v_3, v_4v_5 and v_6v_1 can be in M. Thus no edge of C is in M. □

Observation 1. *Let G be a graph and M be a perfect matching cut of G. Let u, v be two vertices of G. Then for any path P between u and v, $|E(P) \cap M|$ is even if and only if u and v are on the same side of M. Note that implies that for any paths P, Q from u to v, $|E(P) \cap M|$ and $|E(Q) \cap M|$ have same parity.*

3.2 Reduction

We will prove Theorem 1 by reduction from the NP-complete Monotone Not-All-Equal 3SAT-E4 [6]. In Monotone Not-All-Equal 3SAT-E4, the input is a 3-CNF formula where each variable occurs exactly four times, each clause contains exactly three distinct literals, and no clause contains a negated literal. Here we say that a truth assignment on the variables *satisfies* a clause C if at least one literal of C is true and at least least one literal of C is false. The objective is to decide whether there is a truth assignment that satisfies all clauses. We can safely assume (and we will) that the variable-clause incidence graph $\text{inc}(I)$ of I has no cutvertex among its "variable" vertices; see long version.

Let I be an instance of Monotone Not-All-Equal 3SAT-E4 with variables $x_1, x_2, \dots, x_n$ and clauses $m = 4n/3$ clauses $C_1, C_2, \dots, C_m$. We shall construct, in polynomial time, an equivalent PMC-instance $G(I)$ that is Barnette.

Our reduction consists of three steps. First we construct a cubic graph $H(I)$ by introducing *variable gadgets* and *clause gadgets*. Then we *draw* $H(I)$ on the plane, i.e., we map the vertices of $H(I)$ to a set of points on the plane, and the edges of $H(I)$ to a set of simple curves on the plane. We shall refer to this drawing as $\mathcal{R}$. Note that, this drawing may not be planar, i.e., two simple curves (or analogously the corresponding edges) might intersect at a point which is not their endpoints. Finally, we eliminate the crossing points by introducing *crossing gadgets*. (Recall that if the clause-variable graph of an Monotone Not-All-Equal 3SAT-E4 instance is planar, then its satisfiability can be tested in polynomial time; hence, we do need crossing gadgets.) The resulting graph $G(I)$ is Barnette, and we shall prove that $G(I)$ has a perfect matching if and only if I is a positive instance of Monotone Not-All-Equal 3SAT-E4. Below we formally describe the above steps.

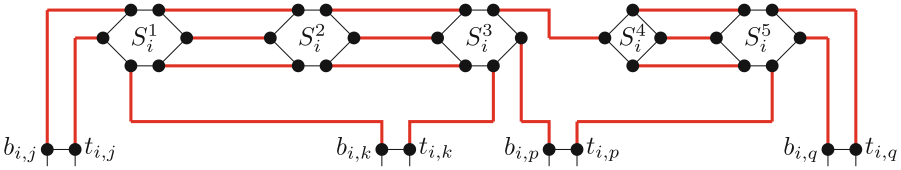


Fig. 1. Variable Gadget $\mathcal{X}_i$ corresponding to the variable x_i appearing in the clauses C_j, C_k, C_p, C_q with $j < k < p < q$.

1. For each variable x_i, let $\mathcal{X}_i$ denote a fresh copy of the graph shown in Fig. 1. Note that the variable x_i appears in exactly four clauses, say, C_j, C_k, C_p, C_q with $j < k < p < q$. The *variable gadget* $\mathcal{X}_i$ contains the special vertices $t_{i,j}$, $b_{i,j}$, $t_{i,k}$, $b_{i,k}$, $t_{i,p}$, $b_{i,p}$, $t_{i,q}$, $b_{i,q}$ as shown in the figure. We recall that red edges are those forced in any perfect matching cut, while black edges cannot be in any solution. An essential part of the proof will consist of justifying the edge colors in our figures.
   For each clause $C_j = (x_a, x_b, x_c)$ with $a < b < c$ let $\mathcal{C}_j$ denote a new copy of the graph shown in Fig. 2. The *clause gadget* $\mathcal{C}_j$ contains the special vertices $t'_{a,j}$, $b'_{a,j}$, $t'_{b,j}$, $b'_{b,j}$, $t'_{c,j}$, $b'_{c,j}$, as shown in the figure. Then for each variable x_i that appears in the clause C_j, introduce two new edges $E_{ij} = \{t_{i,j}t'_{i,j}, b_{i,j}b'_{i,j}\}$. Let $H(I)$ denote the graph defined as follows.

$$V(H(I)) = \bigcup_{i=1}^{n} V(\mathcal{X}_i) \cup \bigcup_{j=1}^{m} V(\mathcal{C}_j)$$

$$E(H(I)) = \bigcup_{i=1}^{n} E(\mathcal{X}_i) \cup \bigcup_{j=1}^{m} E(\mathcal{C}_j) \cup \bigcup_{x_i \in C_j} E_{ij}.$$

   We assign to each edge $e \in E_{i,j}$ its variable as $\text{var}(e) = i$. Note that, for a variable gadget $\mathcal{X}_i$, there are exactly eight edges that have one endpoint in $V(\mathcal{X}_i)$ and the other endpoint not in $V(\mathcal{X}_i)$.
2. In the next step, we generate a drawing $\mathcal{R}$ of $H(I)$ on the plane according to the following procedure.
   (a) For each variable x_i, we embed $\mathcal{X}_i$ as a translate of the variable gadget of Fig. 1 into $[0, 1] \times [2i, 2i + 1]$.
   (b) For each clause C_j, we embed $\mathcal{C}_j$ as a translate of the clause gadget of Fig. 2 into $[2, 3] \times [2j, 2j + 1]$.

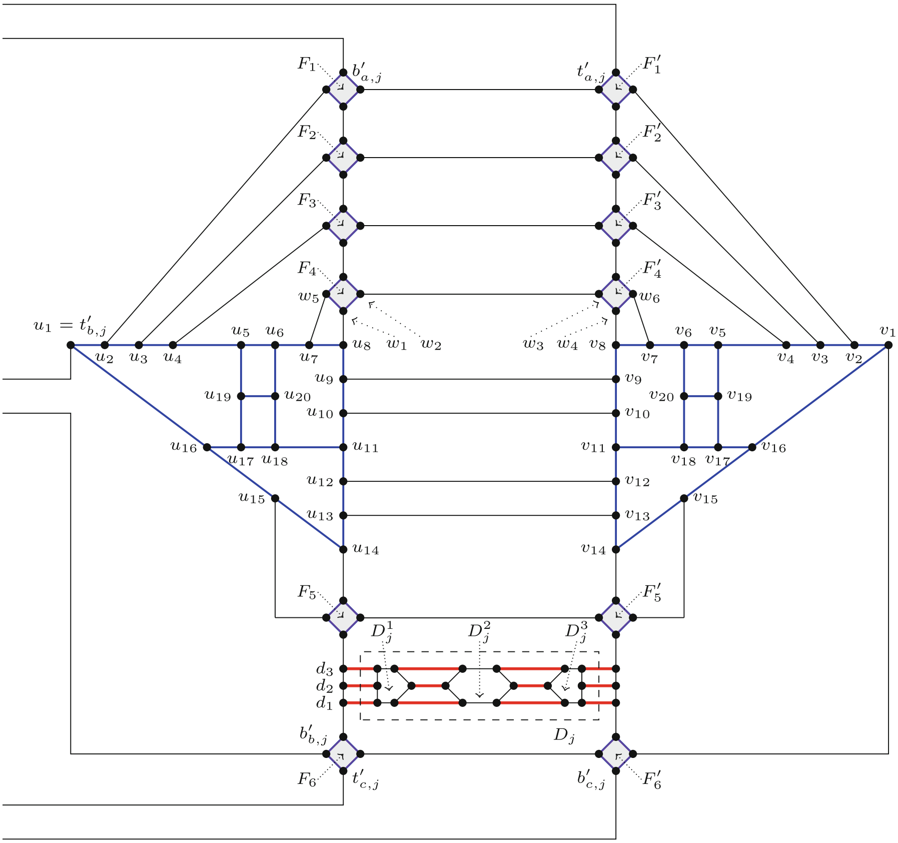


Fig. 2. Clause gadget $C_j = (x_a, x_b, x_c)$ with $a < b < c$. A red edge is selected in any perfect matching cut. A blue edge is selected in some perfect matching cut. A black edge is never selected in any perfect matching cut. (Color figure online)

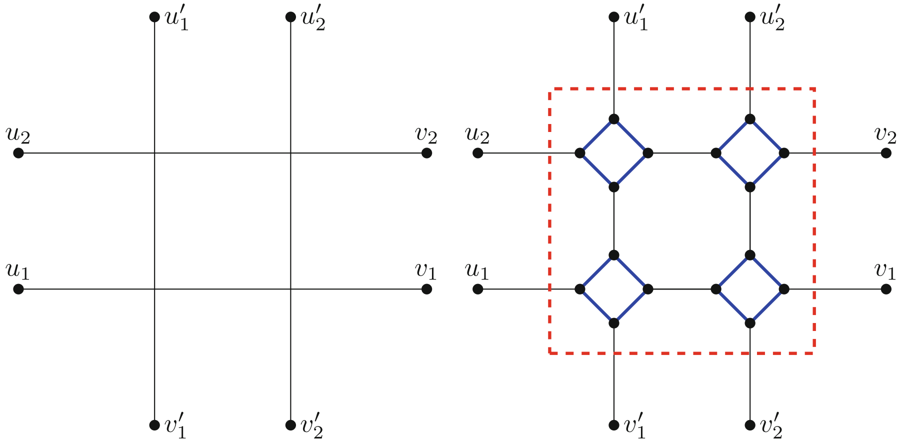
(c) Two edges incident to vertices in the same variable gadget or same clause gadget do not intersect in $\mathcal{R}$. For two variables $x_i, x_{i'}$ and clauses $C_j, C_{j'}$ with $x_i \in C_j, x_{i'} \in C_{j'}$, exactly one of the following holds:

   i For each pair of edges $(e, e') \in E_{ij} \times E_{i'j'}$, e and e' intersect exactly once in $\mathcal{R}$. When this condition is satisfied, we call $(E_{ij}, E_{i'j'})$ a *crossing quadruple*. Moreover, we ensure that the interior of the subsegment of $e \in E_{ij}$ between its two intersection points with edges of $E_{i'j'}$ is not crossed by any edge;

   ii There is no pair of edges $(e, e') \in E_{ij} \times E_{i'j'}$ such that e and e' intersect in $\mathcal{R}$;

3. For each crossing quadruples $(E_{ij}, E_{i'j'})$ replace the four crossing points shown in Fig. 3a by the crossing gadget shown in Fig. 3b.

Let $G(I)$ denote the resulting graph. We shall need the following definitions.



(a) Crossing quadruples.

(b) Replacement of step 3 (b).

Fig. 3. Replacement of a crossing by a crossing gadget.

Definition 1. *Any edge of $G(I)$ whose both endpoints are not contained withing the same gadget (variable, clause, or crossing) is a* connector edge. *Any endpoint of a connector edge is called a* connector vertex. *For a connector edge e incident to a crossing gadget, $var(e)$ is the index of the variable gadget it was originally going to. To each connector edge uv, we associate the variable $var(uv)$ to both u and v, denoted $var(u), var(v)$.*

Now we shall distinguish some 4-cycles of $G(I)$.

Definition 2. *An (induced) 4-cycle C of $G(I)$ is a* crossover *4-cycle if it belongs to some crossing gadget.*

Definition 3. *An induced 4-cycle C of $G(I)$ is* special *if C is identical to F_i or F'_i of some $\mathcal{C}_j$.*

The special 4-cycles of a particular clause gadget $\mathcal{C}_j$ are highlighted in Fig. 2. In the next section, we show that $G(I)$ is indeed a 3-connected cubic bipartite planar graph.

3.3 $G(I)$ Is Barnette

Lemma 6 (⋆). *The graph $G(I)$ is 3-connected.*

Lemma 7 (⋆). *The graph $G(I)$ is Barnette.*

3.4 Properties of Variable and Crossing Gadgets

Lemma 8. *Let M be a perfect matching cut of $G(I)$. Then for any variable gadget $\mathcal{X}_i$, $M \cap V(\mathcal{X}_i)$ is the matching formed by the red edges in Fig. 1. In particular, M does not contain any connector edge incident to a variable gadget.*

Proof. Consider the variable gadget $\mathcal{X}_i$. By applying Lemma 5 on the 6-cycle S_i^2 (which satisfies the requirement of having four particular edges in some 4-cycles), we get that all outgoing edges of $V(S_i^2)$ are in M. We can thus apply Lemma 4 on the 6-cycles S_i^1 and S_i^3, and obtain that all outgoing edges of these cycles are in M. Now there is an outgoing edge of the 4-cycle S_i^4 that is in M, hence by Lemma 3, all of them are. We can finally apply Lemma 4 on the 6-cycle S_i^5, and get that all the red edges of Fig. 1 should indeed be in M. In particular, as all the vertices of $\mathcal{X}_i$ are touched by red edges, the connector edges incident to a variable gadget cannot be in M. □

Now we prove a property of the crossover 4-cycles.

Lemma 9 (⋆). *Let M be a perfect matching cut of $G(I)$ and F be a crossover 4-cycle. Then $|E(F)| = 2$.*

Corollary 2 (⋆). *For any perfect matching M of $G(I)$, M contains no connector edges.*

3.5 Properties of Clause Gadgets

Observe that D_j is an induced subgraph of the variable gadget $\mathcal{C}_j$.

Lemma 10 (⋆). *Any perfect matching cut of $G(I)$ contains the edges of D_j drawn in red in Fig. 2.*

Lemma 11 (⋆). *Let M be a perfect matching cut of $G(I)$ and F be a special 4-cycle of $\mathcal{C}_j$. Then $|E(F) \cap M| = 2$, and no outgoing edge of $V(F)$ is in M.*

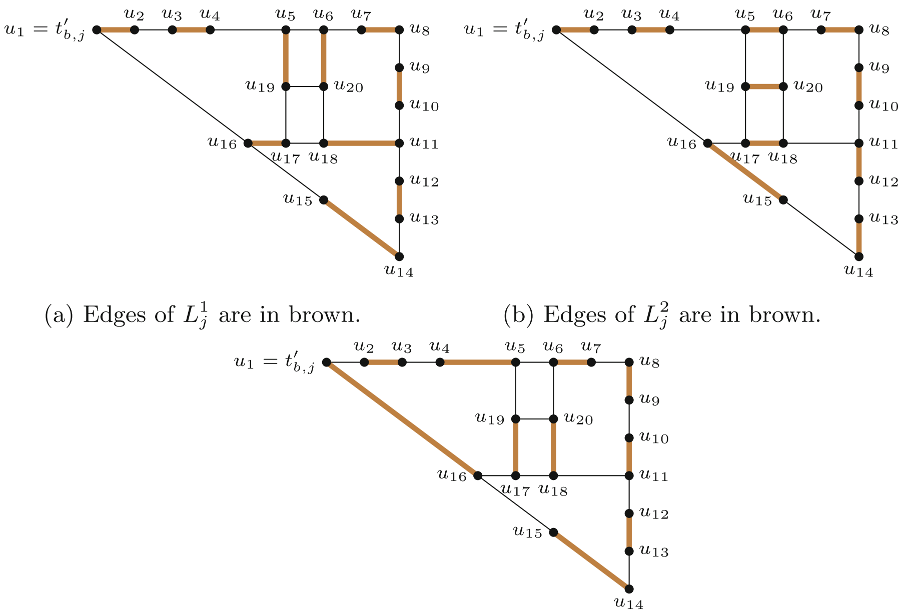
Lemma 12 (⋆). *Let M be a perfect matching cut of $G(I)$ and $\mathcal{C}_j$ be a clause gadget. Let $U_j = \{u_1, \dots, u_{20}\}$, and $V_j = \{v_1, \dots, v_{20}\}$. Then no outgoing edge of U_j or of V_j is in M.*

See the definition of L_j^i (and the symmetric R_j^i in V_j) in Fig. 4.

Definition 4. *We say that a perfect matching cut M of $G(I)$ is of* type i *in $\mathcal{C}_j$ with $i \in \{1, 2, 3\}$, if $M \cap E(U_j \cup V_j) = L_j^i \cup R_j^i$.*

Lemma 13 (⋆). *Let M be a perfect matching cut of $G(I)$ and $\mathcal{C}_j$ be a clause gadget. Then there exists exactly one integer $i \in \{1, 2, 3\}$ such that M is of type i in $\mathcal{C}_j$.*

As a direct consequence of Lemma 13, we get the following.



(a) Edges of L_j^1 are in brown.

(b) Edges of L_j^2 are in brown.

(c) Edges of L_j^3 are in brown.

Fig. 4. The three types of perfect matching cuts within a clause gadget.

Lemma 14. *Let M be a perfect matching cut of $G(I)$ and (A, B) be the cut of M. The vertices u_1, u_8, u_{14} of a clause gadget $\mathcal{C}_j$ cannot all be on the same side of M. More precisely:*

1. *L_j^1 sets u_1 to one side of M, and u_8, u_{14} to the other;*
2. *L_j^2 sets u_{14} to one side of M, and u_1, u_8 to the other;*
3. *L_j^3 sets u_8 to one side of M, and u_1, u_{14} to the other.*

3.6 Existence of Perfect Matching Cut Implies Satisfiability

Lemma 15 ($\star$). *If $G(I)$ has a perfect matching cut then I is a positive instance.*

3.7 Satisfiability Implies the Existence of a Perfect Matching Cut

Lemma 16 ($\star$). *If I has a satisfying assignment then $G(I)$ has a perfect matching cut.*

We finally get Theorem 1, due to Lemmas 15, 16, 7.

Acknowledgments. We are much indebted to Carl Feghali for introducing us to the topic of (perfect) matching cuts, and presenting us with open problems that led to the current paper. We also wish to thank him and Kristóf Huszár for helpful discussions on an early stage of the project.

References

1. Akiyama, T., Nishizeki, T., Saito, N.: NP-completeness of the Hamiltonian cycle problem for bipartite graphs. J. Inf. Process. **3**(2), 73–76 (1980)
2. Bonsma, P.S.: The complexity of the matching-cut problem for planar graphs and other graph classes. J. Graph Theory **62**(2), 109–126 (2009)
3. Bouquet, V., Picouleau, C.: The complexity of the perfect matching-cut problem. arXiv preprint arXiv:2011.03318 (2020)
4. Chen, C.-Y., Hsieh, S.-Y., Le, H.-O., Le, V.B., Peng, S.-L.: Matching cut in graphs with large minimum degree. Algorithmica **83**(5), 1238–1255 (2021)
5. Chvátal, V.: Recognizing decomposable graphs. J. Graph Theory **8**(1), 51–53 (1984)
6. Darmann, A., Döcker, J.: On a simple hard variant of not-all-equal 3-SAT. Theor. Comput. Sci. **815**, 147–152 (2020)
7. De Werra, D., Demange, M., Escoffier, B., Monnot, J., Paschos, V.T.: Weighted coloring on planar, bipartite and split graphs: complexity and approximation. Discret. Appl. Math. **157**(4), 819–832 (2009)
8. Derhy, N., Picouleau, C.: Finding induced trees. Discret. Appl. Math. **157**(17), 3552–3557 (2009)
9. Diestel, R.: Graph Theory. Graduate Texts in Mathematics, vol. 173, 4th edn. Springer, Heidelberg (2012)
10. Bonnet, É., Chakraborty, D., Duron, J.: Cutting Barnette graphs perfectly is hard. arXiv:2302.11667 (2023)
11. Feder, T., Hell, P., Subi, C.S.: Distance-two colourings of Barnette graphs. Eur. J. Comb. **91**, 103210 (2021)
12. Feder, T., Subi, C.S.: On Barnette's conjecture. Electron. Colloquium Comput. Complex. TR06-015 (2006)
13. Feghali, C.: A note on matching-cut in P_t-free graphs. Inf. Process. Lett. **179**, 106294 (2023)
14. Feghali, C., Lucke, F., Paulusma, D., Ries, B.: New hardness results for (perfect) matching cut and disconnected perfect matching. CoRR, abs/2212.12317 (2022)
15. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)
16. Golovach, P.A., Komusiewicz, C., Kratsch, D., Le, V.B.: Refined notions of parameterized enumeration kernels with applications to matching cut enumeration. J. Comput. Syst. Sci. **123**, 76–102 (2022)
17. Hadlock, F.: Finding a maximum cut of a planar graph in polynomial time. SIAM J. Comput. **4**(3), 221–225 (1975)
18. Heggernes, P., Telle, J.A.: Partitioning graphs into generalized dominating sets. Nord. J. Comput. **5**(2), 128–142 (1998)
19. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Proceedings of a Symposium on the Complexity of Computer Computations. The IBM Research Symposia Series, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA, 20–22 March 1972, pp. 85–103. Plenum Press, New York (1972)
20. Kasteleyn, P.: Graph theory and crystal physics. Graph Theory Theor. Phys. 43–110 (1967)
21. Komusiewicz, C., Kratsch, D., Le, V.B.: Matching cut: kernelization, single-exponential time FPT, and exact exponential algorithms. Discret. Appl. Math **283**, 44–58 (2020)

22. Korobitsin, D.V.: On the complexity of domination number determination in monogenic classes of graphs (1992)
23. Kosowski, A., Małafiejski, M., Żyliński, P.: Parallel processing subsystems with redundancy in a distributed environment. In: Wyrzykowski, R., Dongarra, J., Meyer, N., Waśniewski, J. (eds.) PPAM 2005. LNCS, vol. 3911, pp. 1002–1009. Springer, Heidelberg (2006). https://doi.org/10.1007/11752578_121
24. Kratsch, D., Le, V.B.: Algorithms solving the matching cut problem. Theor. Comput. Sci. **609**, 328–335 (2016)
25. Le, H.-O., Le, V.B.: A complexity dichotomy for matching cut in (bipartite) graphs of fixed diameter. Theor. Comput. Sci. **770**, 69–78 (2019)
26. Le, V.B., Telle, J.A.: The perfect matching cut problem revisited. Theor. Comput. Sci. **931**, 117–130 (2022)
27. Loverov, Ya.A., Orlovich, Y.L.: NP-completeness of the independent dominating set problem in the class of cubic planar bipartite graphs. J. Appl. Ind. Math. **14**, 353–368 (2020)
28. Lucke, F., Paulusma, D., Ries, B.: Finding matching cuts in H-free graphs. In: Bae, S.W., Park, H. (eds.) 33rd International Symposium on Algorithms and Computation, ISAAC 2022. LIPIcs, Seoul, Korea, 19–21 December 2022, vol. 248, pp. 22:1–22:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022)
29. Lucke, F., Paulusma, D., Ries, B.: On the complexity of matching cut for graphs of bounded radius and H-free graphs. Theor. Comput. Sci. **936**, 33–42 (2022)
30. Marx, D.: NP-completeness of list coloring and precoloring extension on the edges of planar graphs. J. Graph Theory **49**(4), 313–324 (2005)
31. Miotk, M., Topp, J., Żyliński, P.: Disjoint dominating and 2-dominating sets in graphs. Discret. Optim. **35**, 100553 (2020)
32. Moret, B.M.E.: Planar NAE3SAT is in P. SIGACT News **19**(2), 51–54 (1988)
33. Moshi, A.M.: Matching cutsets in graphs. J. Graph Theory **13**(5), 527–536 (1989)
34. Munaro, A.: On line graphs of subcubic triangle-free graphs. Discret. Math. **340**(6), 1210–1226 (2017)
35. Patrignani, M., Pizzonia, M.: The complexity of the matching-cut problem. In: Brandstädt, A., Le, V.B. (eds.) WG 2001. LNCS, vol. 2204, pp. 284–295. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45477-2_26
36. Stockmeyer, L.J., Vazirani, V.V.: NP-completeness of some generalizations of the maximum matching problem. Inf. Process. Lett. **15**(1), 14–19 (1982)
37. Temperley, H.N.V., Fisher, M.E.: Dimer problem in statistical mechanics-an exact result. Philos. Mag. **6**(68), 1061–1063 (1961)
38. Tutte, W.T.: The factorization of linear graphs. J. Lond. Math. Soc. **1**(2), 107–111 (1947)
39. Vadhan, S.P.: The complexity of counting in sparse, regular, and planar graphs. SIAM J. Comput. **31**(2), 398–427 (2001)

Metric Dimension Parameterized by Treewidth in Chordal Graphs

Nicolas Bousquet, Quentin Deschamps(✉), and Aline Parreau

Univ. Lyon, Université Lyon 1, CNRS, LIRIS UMR 5205, 69621 Lyon, France
{nicolas.bousquet,quentin.deschamps,aline.parreau}@univ-lyon1.fr

Abstract. The metric dimension has been introduced independently by Harary, Melter [11] and Slater [15] in 1975 to identify vertices of a graph G using its distances to a subset of vertices of G. A *resolving set* X of a graph G is a subset of vertices such that, for every pair (u, v) of vertices of G, there is a vertex x in X such that the distance between x and u and the distance between x and v are distinct. The metric dimension of the graph is the minimum size of a resolving set. Computing the metric dimension of a graph is NP-hard even on split graphs and interval graphs. Bonnet and Purohit [2] proved that the metric dimension problem is W[1]-hard parameterized by treewidth. Li and Pilipczuk strengthened this result by showing that it is NP-hard for graphs of treewidth 24 in [14]. In this article, we prove that metric dimension is FPT parameterized by treewidth in chordal graphs.

1 Introduction

Determining the position of an agent on a network is a central problem. One way to determine its position is to place sensors on nodes of the network and the agents try to determine their positions using their positions with respect to these sensors. More formally, assume that agents know the topology of the graph. Can they, by simply looking at their position with respect to the sensors determine for sure their position in the network? Conversely, where do sensors have to be placed to ensure that any agent at any possible position can easily determine for sure its position? These questions received a considerable attention in the last decades and have been studied in combinatorics under different names such as metric dimension, identifying codes, locating dominating sets...

Let $G = (V, E)$ be a graph and s, u, v be three vertices of G. We say that s *resolves* the pair (u, v) if the distance between s and u is different from the distance between s and v. A *resolving set* of a graph $G = (V, E)$ is a subset S of vertices of G such that any vertex of G is identified by its distances to the vertices of the resolving set. In other words, S is a resolving set if for every pair (u, v) of vertices of G, there is a vertex s of S such that s resolves (u, v). The *metric dimension* of G, denoted by $\dim(G)$, is the smallest size of a resolving set of G.

This notion has been introduced in 1975 by Slater [15] for trees and by Harary and Melter [11] for graphs to simulate the moves of a sonar. The associated

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 130–142, 2023.
https://doi.org/10.1007/978-3-031-43380-1_10

decision problem, called the Metric Dimension problem, is defined as follows: given a graph G and an integer k, is the metric dimension of G is at most k?

The Metric Dimension problem is NP-complete [9] even for restricted classes of graphs like planar graphs [4]. Epstein et al. [6] proved that this problem is NP-complete on split graphs, bipartite and co-bipartite graphs. The problem also is NP-complete on interval graphs [8] or sub-cubic graphs [12]. On the positive side, computing the metric dimension is linear on trees [11,15] and polynomial in outer-planar graphs [4].

Parameterized Algorithms. In this paper, we consider the Metric Dimension problem from a parameterized point of view. We say a problem Π is *fixed parameter tractable* (FPT) for a parameter k if any instance of size n and parameter k can be decided in time $f(k) \cdot n^{O(1)}$. Two types of parameters received a considerable attention in the literature: the size of the solution and the "width" of the graph (for various widths, the most classical being the treewidth).

Hartung and Nichterlein proved in [12] that the Metric Dimension problem is W[2]-hard parameterized by the size of the solution. Foucaud et al. proved that it is FPT parameterized by the size of the solution in interval graphs in [8]. This result was extended by Belmonte et al. who proved in [1] that Metric Dimension is FPT parameterized by the size of the solution plus the tree-length of the graph. In particular, it implies that computing the metric dimension for chordal graph is FPT parameterized by the size of the solution.

Metric Dimension is FPT parameterized by the modular width [1]. Using Courcelle's theorem, one can also remark that it is FPT parameterized by the treedepth of the graph as observed in [10]. Metric dimension has been proven W[1]-hard parameterized by the treewidth by Bonnet and Purohit in [2]. Li and Pilipczuk strengthened this result by showing that it is NP-complete for graphs of treewidth, and even pathwidth, 24 in [14]. While Metric dimension is polynomial on graphs of treewidth 1 (forests), its complexity is unknown for graphs of treewidth 2 is open (even if it is known to be polynomial for outerplanar graphs). Our main result is the following:

Theorem 1. Metric Dimension *is FPT parameterized by treewidth on chordal graphs. That is,* Metric Dimension *can be decided in time $O(n^3 + n^2 \cdot f(\omega))$ on chordal graphs of clique number ω.*

Recall that, on chordal graphs, the treewidth is equal to the size of a maximum clique minus one. Our proof is based on a dynamic programming algorithm. One of the main difficulty to compute the metric dimension is that a pair of vertices might be resolved by a vertex far from them in the graph. This non-locality implies that it is not simple to use classical algorithmic strategies like divide-and-conquer, induction or dynamic programming since a single edge or vertex modification somewhere in the graph might change the whole solution[1].

[1] The addition of a single edge in a graph might modify the metric dimension by $\Omega(n)$, see e.g. [7].

The first ingredient of our algorithm consists in proving that, given a chordal graph, if we are using a clique tree of a desirable form and make some simple assumptions on the shape of an optimal solution, we can ensure that resolving a pair of vertices close to a separator implies that we resolve all the pairs of vertices in the graph. Using this lemma, we build a dynamic programming algorithm that computes the minimum size of a resolving set containing a given vertex in FPT-time parameterized by treewdith.

The special type of clique tree used in the paper, inspired from [13], is presented in Sect. 2.1. We then give some properties of resolving sets in chordal graphs in Sect. 2.2. These properties will be needed to prove the correctness and the running time of the algorithm. Then, we present the definition of the extended problem in Sect. 3.1 and the rules of the dynamic programming in Sect. 3.2 where we also prove the correction of the algorithm. We end by an analysis of the complexity of the algorithm in Sect. 4.

Further Work. The function of the treewidth in our algorithm is probably not optimal and we did not try to optimize it to keep the algorithm as simple as possible. A first natural question is the existence of an algorithm running in time $2^{\omega} \cdot Poly(n)$ for chordal graphs.

We know that Theorem 1 cannot be extended to bounded treewidth graphs since Metric Dimension is NP-hard on graphs of treewidth at most 24 [14]. One can nevertheless wonder if our proof technique can be adapted to design polynomial time algorithms for graphs of treewidth at most 2 on which the complexity status of Metric Dimension is still open.

Our proof crucially relies on the fact that a separator X of a chordal graph is a clique and then the way a vertex in a component of $G \setminus X$ interacting with vertices in another component of $G \setminus X$ is simple. One can wonder if there is a tree decomposition in G where all the bags have diameter at most C, is it true that Metric Dimension is FPT parameterized by the size of the bags plus C. Note that, since Metric Dimension is NP-complete on chordal graphs, the problem is indeed hard parameterized by the diameter of the bags only.

2 Preliminaries

2.1 Nice Clique Trees

Unless otherwise stated, all graphs considered in this paper are undirected, simple, finite and connected. For standard terminology and notations on graphs, we refer the reader to [3]. Let us first define some notations we use throughout the article.

Let $G = (V, E)$ be a graph where V is the set of vertices of G and E the set of edges; we let $n = |V|$. For two vertices x and y in G, we denote by $d(x, y)$ the length of a shortest path between x and y and call it *distance between* x *and* y. For every $x \in V$ and $U \subseteq V$, the *distance between* x *and* U, denoted by $d(x, U)$, is the minimum distance between x and a vertex of U. Two vertices x and y are *adjacent* if $xy \in E$. A *clique* is a graph where all the pairs of vertices are adjacent. We denote by ω the size of a maximum clique. Let U be a set of vertices of G. We denote by $G \setminus U$ the subgraph of G induced by the set of vertices $V \setminus U$. We say that U is a *separator* of G if $G \setminus U$ is not connected. If two vertices x and y of $V \setminus U$ belong to two different connected components in $G \setminus U$, we say that U *separates* x and y. If a separator U induces a clique, we say that U is a *clique separator* of G.

Definition 1. *A* tree-decomposition *of a graph G is a pair (X, T) where T is a tree and $X = \{X_i | i \in V(T)\}$ is a collection of subsets (called bags) of $V(G)$ such that:*

- $\bigcup_{i \in V(T)} X_i = V(G)$.
- *For each edge $xy \in E(G), x, y \in X_i$ for some $i \in V(T)$.*
- *For each $x \in V(G)$, the set $\{i | x \in X_i\}$ induces a connected sub-tree of T.*

Let G be a graph and (X, T) a tree-decomposition of G. The *width* of the tree-decomposition (X, T) is the biggest size of a bag minus one. The *treewidth* of G is the smallest width of (X, T) amongst all the tree-decompositions (X, T) of G.

Chordal graphs are graphs with no induced cycle of length at least 4. A characterization given by Dirac in [5] ensures chordal graphs are graphs where minimal vertex separators are cliques. Chordal graphs admit tree-decompositions such that all the bags are cliques. We call such a tree-decomposition a *clique tree.*

Our dynamic programming algorithm is performed in a bottom-up way on a clique tree of the graph with more properties than the one given by Definition 1. These properties permit to simplify the analysis of the algorithm. We adapt the decomposition of [13, Lemma 13.1.2] to get this tree-decomposition.

Lemma 2. *Let $G = (V, E)$ be a chordal graph and r a vertex of G. There exists a clique tree (X, T) such that (i) T contains at most $7n$ nodes, (ii) T is rooted in a node that contains only the vertex r, (iii) T contains only four types of nodes, that are:*

- *Leaf nodes, $|X_i| = 1$ which have no child.*
- *Introduce nodes i which have exactly one child j, and that child satisfies $X_i = X_j \cup \{v\}$ for some vertex $v \in V(G) \setminus X_j$.*
- *Forget nodes i which have exactly one child j, and that child satisfies $X_i = X_j \setminus \{v\}$ for some vertex $v \in X_j$.*
- *Join node i which have exactly two children i_1 and i_2, and these children satisfy $X_i = X_{i_1} = X_{i_2}$.*

Moreover, such a clique tree can be found in linear time.

In the following, a clique tree with the properties of Lemma 2 will be called a *nice clique tree* and we will only consider nice clique trees (X, T) of chordal graphs G.

Given a rooted clique tree (T, X) of G, for any node i of T, we define the *subgraph of G rooted in X_i*, denoted by $T(X_i)$, as the subgraph induced by the subset of vertices of G contained in at least one of the bags of the sub-tree of T rooted in i (i.e. in the bag of i or one of its descendants).

2.2 Clique Separators and Resolving Sets

In this section, we give some technical lemmas that will permit to bound by $f(\omega)$ the amount of information we have to remember in the dynamic programming algorithm.

Lemma 3. *Let K be a clique separator of G and G_1 be a connected component of $G \setminus K$. Let G_{ext} be the subgraph of G induced by the vertices of $G_1 \cup K$ and $G_{int} = G \setminus G_{ext}$. Let $x_1, x_2 \in V(G_{int})$ be such that $|d(x_1, K) - d(x_2, K)| \geq 2$. Then, every vertex $s \in V(G_{ext})$ resolves the pair (x_1, x_2).*

Before proving Lemma 5, let us state a technical lemma.

Lemma 4. *Let G be a chordal and T be a nice clique tree of G. Let X, Y be two bags of T such that $X \cap Y = \emptyset$. Assume that there exist $x \in X, y \in Y$ such that $d(x, y) \geq 2$ and let z be a neighbour of x that appears in the bag the closest to Y in T amongst all the bags on the path between X and Y. Then z belongs to a shortest path between x and y.*

Lemma 5. *Let S be a subset of vertices of a chordal graph G. Let X, Y and Z be three bags of a nice tree-decomposition T of G such that Z is on the path P between X and Y in T. Denote by $P = X_1, \ldots Z \ldots X_p$ the bags of P with $X = X_1$ and $Y = X_p$. Let x be a vertex of X and y a vertex of Y with $d(x, Z) \geq 2$ and $d(y, Z) \geq 2$. Assume that any pair of vertices (u, v) with $u \in X_2 \cup \ldots \cup Z$, $v \in Z \cup \ldots \cup X_p$, $d(u, Z) < d(x, Z)$ and $d(v, Z) < d(y, Z)$ is resolved by S. Then the pair (x, y) is resolved by S.*

Proof. Let i_1 be such that $X_{i_1} \cap N[x] \neq \emptyset$ and for every $j > i_1$, $X_j \cap N[x] = \emptyset$ and i_2 be such that $X_{i_2} \cap N[y] \neq \emptyset$ and for $j < i_2$, $X_j \cap N[y] = \emptyset$. Let x' be the only neighbour of x in X_{i_1} and y' be the only neighbour of y in X_{i_2}. They are unique by definition of nice tree-decomposition. Note that $d(x, y) \geq 4$ since $d(x, Z) \geq 2$ and $d(y, Z) \geq 2$. So $N[x]$ is not adjacent to $N[y]$ and then $i_1 < i_2$. By Lemma 4, x' is on a shortest path between x and Z and y' is on a shortest path between y and Z. So $d(x', Z) < d(x, Z)$ and $d(y', Z) < d(y, Z)$. By hypothesis, there is a vertex $s \in S$ resolving the pair (x', y'). Let us prove that s resolves the pair (x, y).

If s belongs to $N[x]$ or to $N[y]$ then s resolves the pair (x, y) since $d(x, y) \geq 4$. So we can assume that $d(s, x) \geq 2$ and $d(s, y) \geq 2$. Let X_s be a bag of T containing s and X'_s be the closest bag to X_s on P between X and Y.

Case 1: $s \in X_{i_1}$ and $s \in X_{i_2}$. Then, $d(s, x') \leq 1$ and $d(s, y') \leq 1$. The vertex s resolves the pair (x', y') so $d(s, x') \neq d(s, y')$ so $s = x'$ or $s = y'$. Assume by symmetry that $s = x'$, then $d(s, x) = 1$ and $d(s, y) \geq 3$ because $d(x, y) \geq 4$. So s resolves the pair (x, y).

Case 2: s belongs to exactly one of X_{i_1} or X_{i_2}. By symmetry assume that $s \in X_{i_1}$. By Lemma 4, y' is on a shortest path between y and s. So $d(s, y) = d(s, y') + 1$. As s belongs to X_{i_1} then $d(x', s) \leq 1$ and $d(x, s) \leq 2$. As $d(y', s) \neq d(x', s)$ we have $d(y', s) \geq 2$, so $d(s, y) \geq 3$. Thus s resolves the pair (x, y).

Case 3: $s \notin X_{i_1}$ and $s \notin X_{i_2}$. First, we consider the case where X'_s is between X_{i_1} and X_{i_2}. Then, $d(s,x) = d(s,x') + 1$ and $d(s,y) = d(s,y') + 1$ by Lemma 4 as X_{i_1} separates y and s and X_{i_2} separates x and s. Thus, s resolves the pair (x,y).

By symmetry, we can now assume that X'_s is between X and X_{i_1}. Since $i_1 < i_2$, X_{i_2} separates s and y. So $d(s,y) = d(s,y') + 1$ by Lemma 4. To conclude we prove that $d(s,x') < d(s,y')$. Let Q be a shortest path between s and y'. The bag X_{i_1} separates s and y' so $Q \cap X_{i_1} \neq \emptyset$. Let $y_1 \in Q \cap X_{i_1}$. By definition of Q, $d(s,y') = d(s,y_1) + d(y_1,y')$. Since $y_1, x' \in X_{i_1}$ and X_{i_1} is a clique, we have that $y_1 \in N[x']$ and so, $y_1 \neq y'$. So $d(y_1,y') \neq 0$. We also have $d(s,x') \leq d(s,y_1) + 1$ because y_1 is a neighbour of x'. As $d(s,x') \neq d(s,y')$, this ensures $d(s,x') < d(s,y')$. So s resolves the pair (x,y) because $d(s,x) \leq d(s,x') + 1 < d(s,y') + 1 = d(s,y)$. □

The following lemma is essentially rephrasing Lemma 5 to get the result on a set of vertices.

Lemma 6. *Let G be a chordal graph and S be a subset of vertices of G. Let T be a nice clique tree of G. Let X be a bag of T and let $T_1 = (X_1, E_1)$ and $T_2 = (X_2, E_2)$ be two connected components of $T \setminus X$. Assume that any pair of vertices (u,v) of $(X_1 \cup X) \times (X_2 \cup X)$ with $d(u,X) \leq 2$ and $d(v,X) \leq 2$ is resolved by S. Then any pair of vertices (u,v) of (X_1, X_2) with $|d(u,X) - d(v,X)| \leq 1$ is resolved by S.*

3 Algorithm Description

In this section, we fix a vertex v of a chordal graph G and consider a nice clique tree (T, X) rooted in v which exists by Lemma 2. We present an algorithm computing the smallest size of a resolving set of G containing v.

3.1 Extension of the Problem

Our dynamic programming algorithm computes the solution of a generalization of metric dimension which is easier to manipulate when we combine solutions. In this new problem, we will represent some vertices by vectors of distances. We define notations to edit vectors.

Definition 7. *Given a vector $\mathbf{r}$, the notation $\mathbf{r}_i$ refers to the i-th coordinate of $\mathbf{r}$.*

- *Let $\mathbf{r} = (r_1, \ldots, r_k) \in \mathbb{N}^k$ be a vector of size k and $m \in \mathbb{N}$. The vector $\mathbf{r}' = \mathbf{r}|\mathbf{m}$ is the vector of size $k+1$ with $r'_i = r_i$ for $1 \leq i \leq k$ and $r'_{k+1} = m$.*
- *Let $\mathbf{r} = (r_1, \ldots, r_k) \in \mathbb{N}^k$ be a vector of size k. The vector $\mathbf{r}^-$ is the vector of size $k-1$ with $r^-_i = r_i$ for $1 \leq i \leq k-1$.*

Definition 8. *Let i be a node of T and let $X_i = \{v_1, \dots, v_k\}$ be the bag of i. For a vertex x of G, the* distance vector $\mathbf{d_{X_i}(x)}$ *of x to X_i is the vector of size k such that, for $1 \leq j \leq k$, $\mathbf{d_{X_i}(x)}_j = d(x, v_j)$. We define the set $d_{\leq 2}(X_i)$ as the set of distance vectors of the vertices of $T(X_i)$ at distance at most 2 of X_i in G (i.e. one of the coordinate is at most 2).*

Definition 9. *Let G be a graph and $K = \{v_1, \dots, v_k\}$ be a clique of G. Let x be a vertex of G. The* trace *of x on K, denoted by $\mathbf{Tr_K}(x)$, is the vector $\mathbf{r}$ of $\{0,1\}^k \setminus \{1, \dots, 1\}$ such that for every $1 \leq i \leq k$, $d(x, v_i) = a + \mathbf{r}_i$ where $a = d(x, K)$.*

Let S be a subset of vertices of G. The trace $Tr_K(S)$ of S in K is the set of vectors $\{\mathbf{Tr_K}(x), x \in S\}$.

The trace is well-defined because for a vertex x and a clique K, the distance between x and a vertex of K is either $d(x, K)$ or $d(x, K) + 1$.

Definition 10. *Let $\mathbf{r_1}, \mathbf{r_2}$ and $\mathbf{r_3}$ be three vectors of same size k. We say that $\mathbf{r_3}$* resolves *the pair $(\mathbf{r_1}, \mathbf{r_2})$ if*

$$\min_{1 \leq i \leq k} (\mathbf{r_1} + \mathbf{r_3})_i \neq \min_{1 \leq i \leq k} (\mathbf{r_2} + \mathbf{r_3})_i.$$

Lemma 11. *Let K be a clique separator of G and G_1 be a connected component of $G \setminus K$. Let (x, y) be a pair of vertices of $G \setminus G_1$ and let $\mathbf{r}$ be a vector of size $|K|$. If $\mathbf{r}$ resolves the pair $(\mathbf{d_K(x)}, \mathbf{d_K(y)})$, then any vertex $s \in V(G_1)$ with $\mathbf{Tr_K}(s) = \mathbf{r}$ resolves the pair (x, y).*

Proof. Let s be a vertex of G_1 such that $\mathbf{Tr_K}(s) = \mathbf{r}$. The clique K separates s and x (resp. y) so $d(x, s) = \min_{1 \leq i \leq |K|} (\mathbf{d_K(x)} + \mathbf{Tr_K}(s))_i + d(K, s)$ (resp. $d(y, s) = \min_{1 \leq i \leq |K|} (\mathbf{d_K(y)} + \mathbf{Tr_K}(s))_i + d(K, s)$). The vector $\mathbf{r}$ resolves the pair $(\mathbf{d_K(x)}, \mathbf{d_K(y)})$. So $d(x, s) \neq d(y, s)$ and s resolves the pair (x, y). □

Definition 12. *Let K be a clique separator of G and G_1, G_2 be two (non necessarily distinct) connected components of $G \setminus K$. Let M be a set of vectors and let $x \in V(G_1) \cup K$ and $y \in V(G_2) \cup K$. If a vector $\mathbf{r}$ resolves the pair $(\mathbf{d_K(x)}, \mathbf{d_K(y)})$, we say that $\mathbf{r}$* resolves *the pair (x, y). We say that the pair of vertices (x, y)* is resolved *by M if there exists a vector $\mathbf{r} \in M$ that resolves the pair (x, y).*

We can now define the generalised problem our dynamic programming algorithm actually solves. We call it the EXTENDED METRIC DIMENSION problem (EMD for short). We first define the instances of this problem.

Definition 13. *Let i be a node of T. An* instance for a node i *of the* EMD *problem is a 5-uplet $I = (X_i, S_I, D_{int}(I), D_{ext}(I), D_{pair}(I))$ composed of the bag X_i of i, a subset S_I of X_i and three sets of vectors satisfying*

- *$D_{int}(I) \subseteq \{0,1\}^{|X_i|}$ and $D_{ext}(I) \subseteq \{0,1\}^{|X_i|}$,*
- *$D_{pair}(I) \subseteq \{0,1,2,3\}^{|X_i|} \times \{0,1,2,3\}^{|X_i|}$,*

- *$D_{ext}(I) \neq \emptyset$ or $S_I \neq \emptyset$,*
- *For each pair of vectors $(\mathbf{r_1}, \mathbf{r_2}) \in D_{pair}(I)$, there exist two vertices $x \in T(X_i)$ with $\mathbf{d_{X_i}(x)} = \mathbf{r_1}$ and $d(x, X_i) \leq 2$ and $y \notin T(X_i)$ with $\mathbf{d_{X_i}(y)} = \mathbf{r_2}$ and $d(y, X_i) \leq 2$.*

Definition 14. *A set $S \subseteq T(X_i)$ is a solution for an instance I of the* EMD *problem if*

- ***(S1)** Every pair of vertices of $T(X_i)$ is either resolved by a vertex in S or resolved by a vector of $D_{ext}(I)$.*
- ***(S2)** For each vector $\mathbf{r} \in D_{int}(I)$ there exists a vertex $s \in S$ such that $\mathbf{Tr_{X_i}}(s) = \mathbf{r}$.*
- ***(S3)** For each pair of vector $(\mathbf{r_1}, \mathbf{r_2}) \in D_{pair}(I)$, for any vertex $x \in T(X_i)$ with $\mathbf{d_{X_i}(x)} = \mathbf{r_1}$ and any vertex $y \notin T(X_i)$ with $\mathbf{d_{X_i}(y)} = \mathbf{r_2}$, if $d(x, X_i) \leq 2$ and $d(y, X_i) \leq 2$ the pair (x, y) is resolved by S.*
- ***(S4)** $S \cap X_i = S_I$.*

In the rest of the paper, for shortness, we will refer to an instance of the EMD problem only by an instance.

Definition 15. *Let I be an instance. We denote by* $\dim(I)$ *the minimum size of a set $S \subseteq T(X_i)$ which is a solution of I. If such a set does not exist we define* $\dim(I) = +\infty$*. We call this value the* extended metric dimension *of I.*

We now explain the meaning of each element of I. Firstly, a solution S must resolve any pair in $T(X_i)$, possibly with a vector of $D_{ext}(I)$ which represents a vertex of $V \setminus T(X_i)$ in the resolving set. Secondly, for all $\mathbf{r}$ in $D_{int}(I)$, we are forced to select a vertex in $T(X_i)$ whose trace is $\mathbf{r}$. This will be useful to combine solutions since it will be a vector of D_{ext} in other instances. The elements in $D_{pair}(I)$ will also be useful for combinations. In some sense $D_{pair}(I)$ is the additional gain of S compared to the main goal to resolve $T(X_i)$. The set S_I constrains the intersection between S and X_i by forcing a precise subset of X_i to be in S.

The following lemma is a consequence of Definition 14. It connects the definition of the extended metric dimension with the metric dimension.

Lemma 16. *Let G be a graph, T be a nice tree-decomposition of G and r be the root of T. Let I_0 be the instance $(\{r\}, \{r\}, \emptyset, \emptyset, \emptyset)$, then* $\dim(I_0)$ *is the smallest size of a resolving set of G containing r.*

To ensure that our algorithm works well, we will need to use Lemma 3 in some subgraphs of G. This is possible only if we know that the solution is not included in the subgraph. This corresponds to the condition $D_{ext}(I) \neq \emptyset$ or $S_I \neq \emptyset$ and this is why the algorithm computes the size of a resolving set containing the root of T.

3.2 Dynamic Programming

We explain how we can compute the extended metric dimension of an instance I given the extended metric dimension of the instances on the children of X_i in T. The proof is divided according to the different type of nodes.

Leaf Node. Computing the extended metric dimension of an instance for a leaf node can be done easily with the following lemma:

Lemma 17. *Let I be an instance for a leaf node i and v be the unique vertex of X_i. Then,*

$$\dim(I) = \begin{cases} 0 & \textit{if } S_I = \emptyset,\ D_{int}(I) = \emptyset \textit{ and } D_{pair}(I) = \emptyset \\ 1 & \textit{if } S_I = \{v\} \textit{ and } D_{int}(I) \subseteq \{(\mathbf{0})\} \\ +\infty & \textit{otherwise} \end{cases}$$

Proof. Let I be an instance for i. If $S_I = \emptyset$, only the set $S = \emptyset$ can be a solution for I. This set is a solution only if $D_{int}(I) = \emptyset$ and $D_{pair}(I) = \emptyset$. If $S_I = \{v\}$, only the set $S = \{v\}$ can be a solution for I. This is a solution only if $D_{int}(I)$ is empty or only contains the vector $\mathbf{Tr_{x_i}}(v)$. □

In the rest of the section, we treat the three other types of nodes. For each type of nodes we will proceed as follows: define some conditions on the instances on children to be compatible with I, and prove an equality between the extended metric dimension on compatible children instances and the extended metric dimension of the instance of the node.

Join Node. Let I be an instance for a join node i and let i_1 and i_2 be the children of i.

Definition 18. *A pair of instances (I_1, I_2) for (i_1, i_2) is* compatible *with I if*

- ***(J1)*** $S_{I_1} = S_{I_2} = S_I$,
- ***(J2)*** $D_{ext}(I_1) \subseteq D_{ext}(I) \cup D_{int}(I_2)$ *and* $D_{ext}(I_2) \subseteq D_{ext}(I) \cup D_{int}(I_1)$,
- ***(J3)*** $D_{int}(I) \subseteq D_{int}(I_1) \cup D_{int}(I_2)$,
- ***(J4)*** *Let* $C_1 = \{(\mathbf{r}, \mathbf{t}) \in D_{pair}(I_1)$ *such that* $\mathbf{r} \notin d_{\leq 2}(X_{i_1})\}$ *and* $C_2 = \{(\mathbf{r}, \mathbf{t}) \in D_{pair}(I_2)$ *such that* $\mathbf{r} \notin d_{\leq 2}(X_{i_2})\}$. *Let* $D_1 = \{(\mathbf{r}, \mathbf{t}) \in d_{\leq 2}(X_{i_1}) \times d_{\leq 2}(G \backslash X_{i_1})$ *such that there exists* $\mathbf{u} \in D_{int}(I_2)$ *resolving the pair* $(\mathbf{r}, \mathbf{t})\}$ *and* $D_2 = \{(\mathbf{r}, \mathbf{t}) \in d_{\leq 2}(X_{i_2}) \times d_{\leq 2}(G \setminus X_{i_2}))$ *such that there exists* $\mathbf{u} \in D_{int}(I_1)$ *resolving the pair* $(\mathbf{r}, \mathbf{t})\}$ *Then* $D_{pair}(I) \subseteq (C_1 \cup D_1 \cup D_{pair}(I_1)) \cap (C_2 \cup D_2 \cup D_{pair}(I_2))$,
- ***(J5)*** *For all* $\mathbf{r_1} \in d_{\leq 2}(X_{i_1})$, *for all* $\mathbf{r_2} \in d_{\leq 2}(X_{i_2})$, $(\mathbf{r_1}, \mathbf{r_2}) \in D_{pair}(I_1)$ *or* $(\mathbf{r_2}, \mathbf{r_1}) \in D_{pair}(I_2)$ *or there exists* $\mathbf{t} \in D_{ext}(I)$ *such that* $\mathbf{t}$ *resolves the pair* $(\mathbf{r_1}, \mathbf{r_2})$.

Condition **(J4)** represents how the pairs of vertices of $V(T(X_{i_1})) \times V(T(X_{i_2}))$ can be resolved. A pair $(\mathbf{r}, \mathbf{t})$ is in $(C_1 \cup D_1 \cup D_{pair}(I_1))$ if all the pairs of vertices (x, y) with $x \in V(T(X_{i_1}))$ and $y \in V(T(X_{i_2}))$ are resolved. If $(\mathbf{r}, \mathbf{t})$ is in C_1, no pair (x, y) with $x \in V(T(X_{i_1}))$ and $y \in V(T(X_{i_2}))$ exists, if $(\mathbf{r}, \mathbf{t})$ is in D_1 the pairs of vertices are resolved by a vertex outside of $V(T(X_{i_1}))$ and if $(\mathbf{r}, \mathbf{t})$ is in $D_{pair}(I_1)$ the pairs of vertices are resolved by a vertex of $V(T(X_{i_1}))$. So a pair $(\mathbf{r}, \mathbf{t})$ is resolved if the pair is in $(C_1 \cup D_1 \cup D_{pair}(I_1))$ and in $(C_2 \cup D_2 \cup D_{pair}(I_2))$.

Let $\mathcal{F}_J(I)$ be the set of pairs of instances compatible with I. We want to prove the following lemma:

Lemma 19. *Let I be an instance for a join node i. Then,*

$$\dim(I) = \min_{(I_1,I_2)\in\mathcal{F}_J(I)} (\dim(I_1) + \dim(I_2) - |S_I|).$$

We prove the equality by proving the two inequalities in the next lemmas.

Lemma 20. *Let (I_1, I_2) be a pair of instances for (i_1, i_2) compatible with I with finite values for $\dim(I_1)$ and $\dim(I_2)$. Let $S_1 \subseteq V(T(X_{i_1}))$ be a solution for I_1 and $S_2 \subseteq V(T(X_{i_2}))$ be a solution for I_2. Then $S = S_1 \cup S_2$ is a solution for I. In particular,*

$$\dim(I) \leq \min_{(I_1,I_2)\in\mathcal{F}_J(I)} (\dim(I_1) + \dim(I_2) - |S_I|).$$

Proof. Let us prove that the conditions of Definition 14 are satisfied.
(S1) Let (x, y) be a pair of vertices of $T(X_i)$. Assume first that $x \in V(T(X_{i_1}))$ and $y \in V(T(X_{i_1}))$. Either (x, y) is resolved by a vertex of S_1 and then by a vertex of S or (x, y) is resolved by a vector $\mathbf{r} \in D_{ext}(I_1)$. By condition **(J2)**, $\mathbf{r} \in D_{ext}(I)$ or $\mathbf{r} \in D_{int}(I_2)$. If $\mathbf{r} \in D_{ext}(I)$ then (x, y) is resolved by a vector of $D_{ext}(I_1)$. Otherwise, there exists a vertex $t \in S_2$ such that $\mathbf{Tr}_{\mathbf{X_{i_2}}}(t) = \mathbf{r}$. So $t \in S$ and t resolves the pair (x, y). The case $x \in V(T(X_{i_2}))$ and $y \in V(T(X_{i_2}))$ is symmetric. So we can assume that $x \in V(T(X_{i_1}))$ and $y \in V(T(X_{i_2}))$. If $d(x, X_i) \leq 2$ and $d(y, X_i) \leq 2$, the condition **(J5)** ensures that the pair (x, y) is resolved by S or by a vector of $D_{ext}(I)$. Otherwise, either $|d(x, X_i)-d(y, X_i)| \leq 1$ and (x, y) is resolved by Lemma 6 or $|d(x, X_i)-d(y, X_i)| \geq 2$ and (x, y) is resolved by Lemma 3 because $D_{ext}(I) \neq \emptyset$ or $S_I \neq \emptyset$.
(S2) Let $\mathbf{r} \in D_{int}(I)$. By compatibility, the condition **(J3)** ensures that $\mathbf{r} \in D_{int}(I_1)$ or $\mathbf{r} \in D_{int}(I_2)$. As $S = S_1 \cup S_2$, S contains a vertex s such that $\mathbf{Tr}_{\mathbf{X_i}}(s) = \mathbf{r}$.
(S3) Let $(\mathbf{r}, \mathbf{t}) \in D_{pair}(I)$ and (x, y) with $x \in V(T(X_i))$ such that $\mathbf{d}_{\mathbf{X_i}}(\mathbf{x}) = \mathbf{r}$ and $y \notin T(X_i)$ such that $\mathbf{d}_{\mathbf{X_i}}(\mathbf{y}) = \mathbf{t}$. Without loss of generality assume that $x \in V(T(X_{i_1}))$.

By compatibility, $(\mathbf{r}, \mathbf{t}) \in (C_1 \cup D_1 \cup D_{pair}(I_1)) \cap (C_2 \cup D_2 \cup D_{pair}(I_2))$ so in $C_1 \cup D_1 \cup D_{pair}(I_1)$. If $(\mathbf{r}, \mathbf{t}) \in D_{pair}(I)_1$, then there exists $s \in S_1$ that resolves the pair (x, y) so the pair is resolved by S. If $(\mathbf{r}, \mathbf{t}) \in D_1$, there exists $\mathbf{u} \in D_{int}(I_2)$ such that $\mathbf{u}$ resolves the pair $(\mathbf{r}, \mathbf{t})$. By compatibility, there exists $s \in S_2$ such that $\mathbf{Tr}_{\mathbf{X_i}}(s) = \mathbf{u}$. So s resolves the pair (x, y). And $(\mathbf{r}, \mathbf{t}) \notin C_1$ since x belongs to $T(X_{i_1})$ with vector distance $\mathbf{r}$.
(S4) is clear since $X_{i_1} = X_{i_2} = X_i$.

Thus, $\dim(I) \leq \dim(I_1) + \dim(I_2) - |S_I|$ is true for any pair of compatible instances (I_1, I_2) so $\dim(I) \leq \min_{(I_1,I_2)\in\mathcal{F}_J(I)}(\dim(I_1) + \dim(I_2) - |S_I|)$. □

Lemma 21. *Let I be an instance for a join node i and let i_1 and i_2 be the children of i. Then,*

$$\dim(I) \geq \min_{(I_1,I_2)\in\mathcal{F}_J(I)} (\dim(I_1) + \dim(I_2) - |S_I|).$$

Proof. If $\dim(I) = +\infty$ then the result indeed holds. So we can assume that $\dim(I)$ is finite. Let S be a solution for I of minimal size. Let $S_1 = S \cap T(X_{i_1})$ and $S_2 = S \cap T(X_{i_2})$. We define now two instances I_1 and I_2 for i_1 and i_2. Let $S_{I_1} = S_{I_2} = S_I$, $D_{int}(I_1) = Tr_{X_i}(S_1)$, $D_{int}(I_2) = Tr_{X_i}(S_2)$, $D_{ext}(I_1) = D_{ext}(I) \cup D_{int}(I_2)$ and $D_{ext}(I_2) = D_{ext}(I) \cup D_{int}(I_1)$. To build the sets $D_{pair}(I_1)$ and $D_{pair}(I_2)$ we make the following process that we explain for $D_{pair}(I_1)$. For all pairs of vectors $(\mathbf{r}, \mathbf{t})$ of $(d_{\leq 2}(X_{i_1}), d_{\leq 2}(G \setminus X_{i_1}))$, consider all the pairs of vertices (x, y) with $x \in V(T(X_{i_1}))$, $y \in V(G \setminus T(X_{i_1}))$, $\mathbf{r} \in d_{\leq 2}(X_i)$, $\mathbf{t} \in d_{\leq 2}(G \setminus X_{i_1}))$, $\mathbf{d_{X_i}}(\mathbf{x}) = \mathbf{r}$ and $\mathbf{d_{X_i}}(\mathbf{y}) = \mathbf{t}$. If all the pairs are resolved by vertices of S_1 (that is for each pair, there exists a vertex of S_1 that resolves the pair), then add $(\mathbf{r}, \mathbf{t})$ to $D_{pair}(I_1)$.

Checking that (I_1, I_2) is compatible with I, that S_1 is a solution of I_1, and that S_2 is a solution of I_2 is straightforward. It consists of checking conditions of respectively Definition 18 and Definition 14.

Finally we prove the announced inequality. Since S is a minimal solution for I, we have $\dim(I) = |S|$. The sets S_1 and S_2 are solutions for S_1 and S_2 so $\dim(I_1) \leq |S_1|$ and $\dim(I_2) \leq |S_2|$. Since $|S| = |S_1| + |S_2| - |S_I|$, $\dim(I) \geq \dim(I_1) + \dim(I_2) - |S_I|$, giving the result. □

Lemma 19 is a direct consequence of Lemma 20 and Lemma 21.

Introduce Node. We now consider an instance I for an introduce node i. Let j be the child of i and $v \in V$ be such that $X_i = X_j \cup \{v\}$. Let $X_i = \{v_1, \ldots, v_k\}$ with $v = v_k$. The tree $T(X_i)$ contains one more vertex than its child. The definition of the compatibility is slightly different if we consider the same set as a solution (type 1) or if we add this vertex to the resolving set (type 2).

Definition 22. *An instance I_1 is compatible with I of type 1 (resp. 2) if*

- ***(I1)*** $S_I = S_{I_1}$ *(resp.* $= S_{I_1} \cup \{v\}$*).*
- ***(I2)*** *For all* $\mathbf{r} \in D_{ext}(I)$, $\mathbf{r}^- \in D_{ext}(I_1)$ *(resp. or* $\mathbf{r} = (0, \ldots, 0)$*).*
- ***(I3)*** *For all* $\mathbf{r} \in D_{int}(I), \mathbf{r}_k = 1$ *and* $\mathbf{r}^- \in D_{int}(I_1)$ *(resp. or* $\mathbf{r} = (1, \ldots, 1, 0)$*).*
- ***(I4)*** *For all* $(\mathbf{r}, \mathbf{t}) \in D_{pair}(I)$, $(\mathbf{r}^-, \mathbf{t}^-) \in D_{pair}(I_1)$.
- ***(I5)*** *If I_1 is of type 1, for all* $(\mathbf{r}, \mathbf{t})$ *with* $\mathbf{t} = (0, \ldots, 0)$, $(\mathbf{r}, \mathbf{t}) \in D_{pair}(I_1)$.

Lemma 23. *Let I be an instance for an introduce node i. Let $\mathcal{F}_1(I)$ be the set of instances I_1 for i_1 compatible with I of type 1 and $\mathcal{F}_2(I)$ be the set of instances I_2 for i_1 compatible with I of type 2. Then,*

$$\dim(I) = \min\{\min_{I_1 \in \mathcal{F}_1(I)}\{\dim(I_1)\}; \min_{I_2 \in \mathcal{F}_2(I)}\{\dim(I_2) + 1\}\}.$$

The proof of Lemma 23 consists in proving both inequalities similarly to Lemma 19. One inequality comes from the fact that we can get a solution of I from any compatible instance. The other consists in building a solution for a compatible instance from a minimal solution for I.

Forget Node. The construction for the forget nodes is similar to the one for introduce nodes. The main difference is that a vertex is removed from the bag so we have to keep the information about this vertex. The full construction leads to a similar equality between the extended metric dimension of an instance and the extended metric dimension of the compatible instances of its child.

3.3 Algorithm

Given as input a nice clique tree, the algorithm computes the extended metric dimension bottom up from the leaves. The algorithm computes the extended metric dimension for leaves using Lemma 17, for join nodes using Lemma 19, for introduce nodes using Lemma 23 and forget nodes using a similar lemma. The correction of the algorithm is straightforward by these lemmas.

We denote this algorithm by IMD in the following which takes as input a nice clique tree T and outputs the minimal size of a resolving set of G containing the root of T.

4 Proof of Theorem 1

Let us finally explain how we can compute the metric dimension of G. The following lemma is a consequence of Lemma 16.

Lemma 24. *The metric dimension of G is $\min_{v\in V(G)}\{IMD(T(v))\}$ where $T(v)$ is a nice clique tree of G rooted in v.*

So, n executions of the IMD algorithm with different inputs are enough to compute the metric dimension. Lemma 2 ensures that we can find for any vertex v of G a nice clique tree in linear time, the last part is to compute the complexity of the IMD algorithm.

Lemma 25. *The algorithm for IMD runs in time $O(n(T)^2+n(T)\cdot f(\omega))$ where $n(T)$ is the number of vertices of the input tree T and $f = O(\omega^2 \cdot 2^{O(4^{2^{\omega}})})$ is a function that only depends on the size of a maximum clique ω.*

We now have all the ingredients to prove Theorem 1:

Proof. For each vertex v of G, one can compute a nice clique tree of size at most $7n$ according to Lemma 2. Given this clique tree, the IMD algorithm outputs the size of a smallest resolving set containing v by Lemma 16 in time $O(n(T)^2 + n(T) \cdot f(\omega))$ for a computable function f according to Corollary 25. Repeat this for all vertices of G permits to compute the metric dimension of G by Lemma 24 in time $O(n^3 + n^2 \cdot f(\omega))$. □

References

1. Belmonte, R., Fomin, F.V., Golovach, P.A., Ramanujan, M.S.: Metric dimension of bounded tree-length graphs. CoRR abs/1602.02610 (2016)
2. Bonnet, É., Purohit, N.: Metric dimension parameterized by treewidth. Algorithmica **83**(8), 2606–2633 (2021)
3. Chartrand, G., Lesniak, L., Zhang, P.: Graphs and Digraphs, 6th edn. Chapman and Hall/CRC (2015)
4. Díaz, J., Pottonen, O., Serna, M., van Leeuwen, E.J.: On the complexity of metric dimension. In: Epstein, L., Ferragina, P. (eds.) Algorithms - ESA 2012 (2012)
5. Dirac, G.A.: On rigid circuit graphs. Abh. Math. Semin. Univ. Hambg. **25**, 71–76 (1961). https://doi.org/10.1007/BF02992776
6. Epstein, L., Levin, A., Woeginger, G.J.: The (weighted) metric dimension of graphs: hard and easy cases. Algorithmica **72**(4), 1130–1171 (2015)
7. Eroh, L., Feit, P., Kang, C.X., Yi, E.: The effect of vertex or edge deletion on the metric dimension of graphs. J. Comb **6**(4), 433–444 (2015)
8. Foucaud, F., Mertzios, G.B., Naserasr, R., Parreau, A., Valicov, P.: Identification, location-domination and metric dimension on interval and permutation graphs. II. Algorithms and complexity. Algorithmica **78**(3), 914–944 (2017)
9. Garey, J.: A guide to the theory of NP-completeness. J. Algorithms (1979)
10. Gima, T., Hanaka, T., Kiyomi, M., Kobayashi, Y., Otachi, Y.: Exploring the gap between treedepth and vertex cover through vertex integrity. Theor. Comput. Sci. **918**, 60–76 (2022)
11. Harary, F., Melter, R.A.: On the metric dimension of a graph. Ars Combinatoria **2**, 191–195 (1975)
12. Hartung, S., Nichterlein, A.: On the parameterized and approximation hardness of metric dimension. In: 2013 IEEE Conference on Computational Complexity, pp. 266–276. IEEE (2013)
13. Kloks, T.: Treewidth: Computations and Approximations. Springer, Heidelberg (1994)
14. Li, S., Pilipczuk, M.: Hardness of metric dimension in graphs of constant treewidth. Algorithmica **84**(11), 3110–3155 (2022)
15. Slater, P.J.: Leaves of trees. Congressus Numerantium **14** (1975)

Efficient Constructions for the Győri-Lovász Theorem on Almost Chordal Graphs

Katrin Casel(✉), Tobias Friedrich, Davis Issac,
Aikaterini Niklanovits(✉), and Ziena Zeif

Hasso Plattner Institute, University of Potsdam, 14482 Potsdam, Germany
{Katrin.Casel,Tobias.Friedrich,Davis.Issac,Aikaterini.Niklanovits,
Ziena.Zeif}@hpi.de

Abstract. In the 1970s, Győri and Lovász showed that for a k-connected n-vertex graph, a given set of terminal vertices $t_1, \dots, t_k$ and natural numbers $n_1, \dots, n_k$ satisfying $\sum_{i=1}^{k} n_i = n$, a connected vertex partition $S_1, \dots, S_k$ satisfying $t_i \in S_i$ and $|S_i| = n_i$ exists. However, polynomial time algorithms to actually compute such partitions are known so far only for $k \leq 4$. This motivates us to take a new approach and constrain this problem to particular graph classes instead of restricting the values of k. More precisely, we consider k-connected chordal graphs and a broader class of graphs related to them. For the first class, we give an algorithm with $\mathcal{O}(n^2)$ running time that solves the problem exactly, and for the second, an algorithm with $\mathcal{O}(n^4)$ running time that deviates on at most one vertex from the required vertex partition sizes.

Keywords: Győri-Lovász theorem · chordal graphs · HHD-free graphs

1 Introduction

Partitioning a graph into connected subgraphs is a fundamental task in graph algorithms. Such *connected* partitions occur as desirable structures in many application areas such as image processing [8], road network decomposition [9], and robotics [17].

From a theoretical point of view, the existence of a partition into connected components with certain properties also gives insights into the graph structure. In theory as well as in many applications, one is interested in a connected partition that has a given number of subgraphs of chosen respective sizes. With the simple example of a star-graph, it is observed that not every graph admits a connected partition for any such choice of subgraph sizes. More generally speaking, if there exists a small set of t vertices whose removal disconnects a graph (*separator*), then any connected partition into $k > t$ subgraphs has limited choice of subgraph sizes. Graphs that do not contain such a separator of size less than k are called *k-connected*.

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 143–156, 2023.
https://doi.org/10.1007/978-3-031-43380-1_11

On the other hand, Győri and Lovász independently showed that k-connectivity is not just necessary but also sufficient to enable a connected partitioning into k subgraphs of required sizes, formally stated by the following result.

Győri-Lovász Theorem ([4,7]). Let $k \geq 2$ be an integer, $G = (V, E)$ a k-connected graph, $t_1, \ldots, t_k \in V$ distinct vertices and $n_1, \ldots, n_k \in \mathbb{N}$ such that $\sum_{i=1}^{k} n_i = |V|$. Then G has disjoint connected subgraphs $G_1, \ldots G_k$ such that $|V(G_i)| = n_i$ and $t_i \in V(G_i)$ for all $i \in [k]$.

The caveat of this famous theorem is that the constructive proof of it yields an exponential time algorithm. Despite this result being known since 1976, to this day we only know polynomial constructions for restricted values of k. Specifically, in 1990 Suzuki et al. [15] provided such an algorithm for $k = 2$ and also for $k = 3$ [14]. Moreover in 1994 Wada et al. [16] also provided an extended result for $k = 3$. Nakano et al. [10] gave a linear time algorithm for the case where $k = 4$, G is planar and the given terminals are located on the same face of a plane embedding of G, while in 2016 Hoyer and Thomas [5] provided a polynomial time algorithm for the general case of $k = 4$. And so far, this is where the list ends, thus for $k \geq 5$ it remains open whether there even exists a polynomial time construction.

Towards a construction for general k, we consider restricting the class of k-connected graphs instead of the values of k. More precisely, we consider (generalizations of) *chordal* k-connected graphs. A graph is called *chordal*, if it does not contain an induced cycle of length more than three. The restriction to chordal graphs is known to often yield tractability for otherwise NP-hard problems, for example chromatic number, clique number, independence number, clique covering number and treewidth decomposition [13]. Apart from the interest chordal graphs have from a graph theoretic point of view, their structural properties have also been proven useful in biology when it comes to studying multidomain proteins and network motifs (see e.g. [11,12]).

Our Contribution. To the best of our knowledge, this paper is the first to pursue the route of restricting the Győri-Lovász Theorem to special graph classes in order to develop a polynomial construction for general values of k on a non-trivial subclass of k-connected graphs. We believe that in general considering the structure of the minimal separators of a graph is promising when it comes to developing efficient algorithms for the Győri-Lovász Theorem.

We give a constructive version of the Győri-Lovász Theorem for *chordal* k-connected graphs with a running time in $\mathcal{O}(|V|^2)$. Observe here that this construction works for all values of k. Then we show how this result can be generalized in two directions.

First, we generalize our result to the vertex weighted version of the Győri-Lovász Theorem (as proven independently by Chandran et al. [2], Chen et al. [3] and Hoyer [5]), specifically deriving the following theorem.

Theorem 1. *Let $k \geq 2$ be an integer, $G = (V, E, w)$ a vertex-weighted k-connected chordal graph with $w \colon V \to \mathbb{N}$ and $w_{max} := \max_{u \in V} w(u)$, $t_1, \ldots, t_k \in$*

V distinct vertices, and $w_1, \ldots, w_k \in \mathbb{N}$ with $w_i \geq w(t_i)$ for all $i \in [k]$ and $\sum_{i=1}^{k} w_i = w(V)$. A partition $S_1, \ldots, S_k$ of V, such that $G[S_i]$ is connected, $t_i \in S_i$ and $w_i - w_{max} < w(S_i) < w_i + w_{max}$, for all $i \in [k]$, can be computed in time $\mathcal{O}(|V|^2)$.

We further use this weighted version to derive an approximate version of the Győri-Lovász Theorem for a larger graph class. Specifically we define I_j^i to contain all graphs that occur from two distinct chordless C_j's that have at least i vertices in common. We focus on I_4^2-free combined with HH-free graphs. More specifically, we consider the subclass of k-connected graphs that contain no hole or house as an induced subgraph (see preliminaries for the definitions of structures such as hole, house etc.) and that does not contain two distinct induced C_4 that share more than one vertex. We call this class of graphs HHI_4^2-free . Note that HHI_4^2-free , apart from being a strict superclass of chordal graphs, is also a subclass of HHD-free graphs (that is house, hole, domino-free graphs), a graph class studied and being used in a similar manner as chordal graphs as it is also a class where the minimum fill-in set is proven to be polynomially time solvable [1] (see also [6] for NP-hard problems solved in polynomial time on HHD-free graphs). Taking advantage of the fact that given an HHI_4^2-free graph, the subgraph formed by its induced C_4 has a treelike structure, we are able to derive the following result.

Theorem 2. *Let $k \geq 2$ be an integer, $G = (V, E, w)$ a vertex-weighted k-connected HHI_4^2-free graph with $w \colon V \rightarrow \mathbb{N}$ and $w_{max} := \max_{u \in V} w(u)$, $t_1, \ldots, t_k \in V$ distinct vertices, and $w_1, \ldots, w_k \in \mathbb{N}$ with $w_i \geq w(t_i)$ for all $i \in [k]$ and $\sum_{i=1}^{k} w_i = w(V)$. A partition $S_1, \ldots, S_k$ of V, such that $G[S_i]$ is connected, $t_i \in S_i$ and $w_i - 2w_{max} < w(S_i) < w_i + 2w_{max}$, for all $i \in [k]$, can be computed in time $\mathcal{O}(|V|^4)$.*

Notice that the above theorem implies a polynomial time algorithm with an additive error of 1 for the unweighted case.

2 Preliminaries

All graphs mentioned in this paper are undirected, finite and simple. Given a graph G and a vertex $v \in V(G)$ we denote its *open neighborhood* by $N_G(v) := \{u \in V(G) \mid uv \in E(G)\}$ and by $N_G[v]$ its *closed neighborhood*, which is $N(v) \cup \{v\}$. Similarly we denote by $N_G(S) := \bigcup_{v \in S} N_G(v) \backslash S$ the open neighborhood of a vertex set $S \subseteq V(G)$ and by $N_G[S] := N_G(S) \cup S$ its closed neighborhood. We omit the subscript G when the graph we refer to is clear from the context. A vertex $v \in V(G)$ is *universal to a vertex set* $S \subset V(G)$ if $S \subseteq N(v)$. Let G be a graph and $S \subseteq V(G)$. The *induced subgraph* from S, denoted by $G[S]$, is the graph with vertex set S and all edges of $E(G)$ with both endpoints in S.

A graph G is *chordal* if any cycle of G of size at least 4 has a chord (i.e., an edge linking two non-consecutive vertices of the cycle). A vertex $v \in V(G)$ is called *simplicial* if $N[v]$ induces a clique. Based on the existence of simplicial vertices in chordal graphs, the following notion of vertex ordering was given. Given a graph G, an ordering of its vertices $(v_1, \dots, v_n)$ is called *perfect elimination ordering* (p.e.o.) if v_i is simplicial in $G[\{v_i, v_{i+1}, \dots, v_n\}]$ for all $i \in [n]$. Given such an ordering $\sigma : V(G) \to \{1, \dots, n\}$ and a vertex $v \in V(G)$ we call $\sigma(v)$ the *p.e.o. value of v*. Rose et al. [13] proved that a p.e.o. of any chordal graph can be computed in linear time.

Let $e = \{u, v\}$ be an edge of G. We denote by G/e the graph G', that occurs from G by the contraction of e, that is, by removing u and v from G and replacing it by a new vertex z whose neighborhood is $(N(u) \cup N(v)) \setminus \{u, v\}$.

A graph G is *connected* if there exists a path between any pair of distinct vertices. Moreover, a graph is *k-connected* for some $k \in \mathbb{N}$ if after the removal of any set of at most $k - 1$ distinct vertices G remains connected. Given a graph G and a vertex set $S \subseteq V(G)$, we say that S is a *separator* of G if its removal disconnects G. We call S a *minimal separator* of G if the removal of any subset $S' \subseteq V(G)$ with $|S'| < |S|$ results in a connected graph.

We now define some useful subgraphs, see also Fig. 1 for illustrations. An induced chordless cycle of length at least 5 is called a *hole*. The graph that occurs from an induced chordless C_4 where exactly two of its adjacent vertices have a common neighbor is called a *house*. When referring to the induced C_3 part of a house we call it *roof* while the induced C_4 is called *body*. Two induced C_4 sharing exactly one edge form a *domino*. A graph that contains no hole, house or domino as an induced subgraph is called HHD-free. We call a graph that consists of two C_4 sharing a vertex, and an edge that connects the two neighbors of the common vertex in a way that no other C_4 exists a *double house*.

Lastly, let $G = (V, E)$ be a k-connected graph, let $t_1, \dots, t_k \in V$ be k distinct vertices, and let $n_1, \dots, n_k$ be natural numbers satisfying $\sum_{i=1}^{k} n_i = |V|$. We call $S_1, \dots S_k \subseteq V(G)$ a *GL-Partition of G* if $S_1, \dots S_k$ forms a partition of $V(G)$, such that for all $i \in [k]$ we have that $G[S_i]$ is connected, $t_i \in S_i$ and $|S_i| = n_i$. When there exists an $l \in \mathbb{N}$, such that for such a partition only $n_i - l \leq |S_i| \leq n_i + l$ holds instead of $|S_i| = n_i$, we say that $S_1, \dots, S_k$ is a *GL-Partition of G with deviation l*.

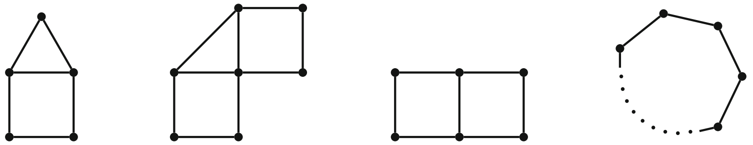


Fig. 1. Specific subgraphs used throughout the paper, from left to right: house, double house, domino and hole example

3 GL-Partition for Chordal Graphs

We present a simple, implementable algorithm with quadratic running time that computes GL-Partitions in chordal graphs. We then show that a slight modification of our algorithm is sufficient to compute a GL-Partition on a vertex weighted graph, thus proving Theorem 1.

3.1 GL-Partition for Unweighted Chordal Graphs

For simplicity, we first prove the restricted version of Theorem 1 to unweighted graphs. We use a p.e.o. to compute a vertex partition, as described formally in Algorithm 1. This algorithm receives as input a k-connected chordal graph $G = (V, E)$, terminal vertices $t_1, \ldots, t_k \in V$, and natural numbers $n_1, \ldots, n_k$ satisfying $\sum_{i=1}^{k} n_i = n$, and outputs connected vertex sets $S_1, \ldots, S_k \subseteq V$ such that $|S_i| = n_i$ and $t_i \in S_i$. In the beginning of the algorithm we initialize each set S_i to contain only the corresponding terminal vertex t_i, and add vertices iteratively to the *non-full sets* (S_i's that have not reached their demanded size). We say a vertex v *is assigned* if it is already part of some S_i and *unassigned* otherwise. At each iteration, the unassigned neighborhood of the union of the previously non-full sets is considered, and the vertex with the minimum p.e.o. value is selected to be added to a non-full set. In case there is more than one non-full set in the neighborhood of this vertex, it is added to the one with lowest priority, where the priority of each set is defined to be the largest p.e.o. value of its vertices so far. The algorithm terminates once all vertices are assigned, in $\mathcal{O}(|V|^2)$ time.

Algorithm 1: `ChordalGL`

```
Input: $k$-connected chordal graph $G = (V, E)$, terminal vertices $t_1, \ldots, t_k \in V$,
       and natural numbers $n_1, \ldots, n_k$ satisfying $\sum_{i=1}^{k} n_i = n$
Output: Connected vertex sets $S_1, \ldots, S_k \subseteq V$ such that $|S_i| = n_i$ and $t_i \in S_i$
$\sigma \leftarrow$ Compute p.e.o. of $G$ as function $\sigma\colon V \to [|V|]$
$S_i \leftarrow \{t_i\}$, for all $i \in [k]$
while $\bigcup_{i\in[k]} S_i \neq V(G)$ do
    $I \leftarrow \{i \in [k] \mid |S_i| < n_i\}$
    $V' \leftarrow N(\bigcup_{i\in I} S_i) \setminus \bigcup_{i\in[k]} S_i$
    $v' \leftarrow arg\,min_{v\in V'} \sigma(v)$
    $J \leftarrow \{i \in I \mid v' \in N(S_i)\}$
    $j' \leftarrow arg\,min_{j\in J} \max(\sigma(S_j))$
    $S_{j'} \leftarrow S_{j'} \cup \{v'\}$
end
return $S_1, \ldots, S_k$
```

For the correctness of Algorithm 1 it is enough to show that the unassigned neighborhood V' of all non-full sets is not empty in each iteration of the while-loop, since this implies that we enlarge a non-full set (in the algorithm denoted by $S_{j'}$) by one vertex (in the algorithm denoted by v') while maintaining the size of all remaining sets. That is, in each iteration we make progress in the sense that $|\bigcup_{i\in[k]} S_i|$ increases while maintaining the invariant $|S_i| \leq n_i$ for all S_i's. Note that $v' \in N(S_{j'})$ which in turn implies that $G[S_i]$ is always connected for all $i \in [k]$. Finally, by $\sum_{i=1}^{k} n_i = n$ and through the way we update I we ensure that the algorithm (or while-loop) terminates as $\bigcup_{i\in[k]} S_i = V$ only if we have $|S_i| = n_i$ for all S_i's.

Towards proving the required Lemmata for the correctness of Algorithm 1 we make the following observation for the p.e.o. of a graph.

Lemma 1. *Let σ be a p.e.o of a graph $G = (V, E)$ and $P = \{v_1, v_2, \ldots, v_k\}$ a vertex set of G that induces a simple path with endpoints v_1 and v_k. Then $\sigma(v_i) > \min\{\sigma(v_1), \sigma(v_k)\}$ for all $i = 2, \ldots, k-1$.*

Lemma 2. *In each iteration of the while-loop in Algorithm 1 we have $V' \neq \varnothing$.*

Proof. We first define the *z-connecting* neighborhood of a vertex v to be the neighbors of v that are included in some induced path connecting v to z.

We prove that every non-full set S_i contains a vertex in its neighborhood $N(S_i)$ that is unassigned, which implies that $V' \neq \varnothing$. Assume for a contradiction that at some iteration of our algorithm there is an non-full set S_i whose neighborhood is already assigned to other sets. Let v be the vertex of S_i of maximum σ value among its vertices and z be the vertex of maximum σ value among the unassigned vertices. Note that $vz \notin E(G)$. Let $\mathcal{P}$ be the set of all simple induced paths of G with endpoints z and v. Consider now the following cases:

1. If $\sigma(z) > \sigma(v)$, we get from Lemma 1 that every internal vertex of each path in $\mathcal{P}$ has higher σ value than v. Note that no vertex of S_i is an internal vertex of some path in $\mathcal{P}$, since all of them have smaller σ value than v by the selection of v. Denote the z-connecting neighborhood of v by C.
   Let a, b be two vertices in C and assume that $a, b \in S_j$ for some j. Assume also that during our algorithm, a is added to S_j before b. Since all vertices of S_i have smaller σ value than both a and b, and a is added to S_j before b, the moment b is added to S_j, S_i has already been formed. Consider now the iteration that this happens. Since $b \in N(v)$, $G[S_i \cup \{b\}]$ is connected. Moreover since $\sigma(a) > \sigma(v)$ and S_i is not full, b should be added to S_i instead of S_j. As a result each set apart from S_i contains at most one such neighbor of v, and hence $|C| < k$.
   Observe that $G \backslash C$ has no induced path connecting z and v which in turn implies that $G \backslash C$ has no $z - v$ path in general. However, this contradicts the k-connectivity of G.
2. If $\sigma(z) < \sigma(v)$, since z is the unassigned vertex of the highest σ value among all unassigned vertices, and by Lemma 1 all vertices in $\mathcal{P}$ have greater σ value than z, all of its v-connecting neighbors in $\mathcal{P}$ are already assigned in some set. Denote the set of v-connecting neighbors of z by C.
   Assume now that there are two vertices of C, a and b, that are contained in some S_j and assume also without loss of generality that a was added to S_j before b. Note that since $\sigma(z) < \sigma(b)$ at each iteration of our algorithm z is considered before b to be added to some set if the induced graph remains connected. As a result, after a is added to S_j, the induced subgraph $G[S_j \cup \{z\}]$ is connected and hence z should be added to S_j before b.

This means that each set contains at most one v-connecting neighbor of z and therefore $|C| < k$. Since $G \backslash C$ has no induced path connecting z and v, there is no z-v-path in $G \backslash C$, which contradicts the k-connectivity.

Corollary 1. *At each iteration of Algorithm 1, unless all vertices are assigned, the neighborhood of each non-full set contains at least one unassigned vertex.*

In the weighted case we use the above corollary of Lemma 2. In particular, it follows from Corollary 1 that as long as we do not declare a set to be full, we ensure that we are able to extend it by a vertex in its neighborhood that is unassigned. Note that in the weighted case we do not know in advance how many vertices are in each part.

3.2 GL-Partition for Weighted Chordal Graphs

With a slight modification of Algorithm 1 we can compute the weighted version of a GL-Partition . In particular, we prove Theorem 1.

The input of our algorithm differs from the unweighted case by having a positive vertex-weighted graph $G = (V, E, w)$ and instead of demanded sizes $n_1, \ldots, n_k$ we have demanded weights $w_1, \ldots, w_k$ for our desired vertex sets $S_1, \ldots, S_k$, where $\sum_{i=1}^{k} w_i = w(V)$. Note also that $w(S_i)$ is not allowed to deviate more than $w_{max} = \max_{v \in V} w(v)$ from w_i, i.e. $w_i - w_{max} < w(S_i) < w_i + w_{max}$.

Again we set each terminal vertex t_i to a corresponding set S_i, and enlarge iteratively the *non-full weighted sets* (S_i's that are not declared as full). One difference to the previous algorithm is that we declare a set S_i as *full weighted set*, if together with the next vertex to be potentially added its weight would exceed w_i. After that, we decide whether to add the vertex with respect to the currently full weighted sets. Similar to Algorithm 1 we interrupt the while-loop if $S_1, \ldots, S_k$ forms a vertex partition of V and the algorithm terminates. However, to ensure that we get a vertex partition in every case, we break the while-loop when only one non-full weighted set is left and assign all remaining unassigned vertices to it.

Observe that we can make use of Corollary 1, since Algorithm 2 follows the same priorities concerning the p.e.o. as Algorithm 1. Basically, it implies that as long we do not declare a set as full weighted set and there are still unassigned vertices then those sets have unassigned vertices in their neighborhood.

We conclude this section by extending the above algorithms to graphs having distance $k/2$ from being chordal. In particular this corollary is based on the observation that an edge added to a graph does not participate in any of the parts those algorithms output if both of its endpoints are terminal vertices.

Corollary 2. *Let G be a k-connected graph which becomes chordal after adding $k/2$ edges. Given this set of edges, a GL-Partition (also its weighted version) can be computed in polynomial time but without fixed terminals.*

Algorithm 2: `WeightedChordalGL`

```
Input: $k$-connected vertex-weighted chordal graph $G(V, E, w)$, terminal vertices
       $t_1, \ldots, t_k \in V$, and positive weights $w_1, \ldots, w_k$ satisfying
       $\sum_{i=1}^{k} w_i = w(V)$
Output: Connected vertex sets $S_1, \ldots, S_k \subseteq V$ such that
        $w_i - w_{\max} < w(S_i) < w_i + w_{\max}$ and $t_i \in S_i$
$\sigma \leftarrow$ Compute p.e.o. of $G$ as function $\sigma\colon V \to [|V|]$
$S_i \leftarrow \{t_i\}$, for all $i \in [k]$
$I \leftarrow \{i \in [k] \mid w(S_i) < w_i\}$
while $|I| \neq 1$ and $\bigcup_{i\in[k]} S_i \neq V(G)$ do
    $V' \leftarrow N(\bigcup_{i\in I} S_i) \setminus \bigcup_{i\in[k]} S_i$
    $v' \leftarrow arg\,min_{v\in V'} \sigma(v)$
    $J \leftarrow \{i \in I \mid v' \in N(S_i)\}$
    $j' \leftarrow arg\,min_{j\in J} \max(\sigma(S_j))$
    if $w(S_{j'}) + w(v') < w_{j'}$ then
        $S_{j'} \leftarrow S_{j'} \cup \{v'\}$
    end
    else
        $I \leftarrow I \setminus \{j'\}$
        if $\sum_{i\in[k]\setminus I}(w_i - w(S_i)) \geq 0$ or $w(S_{j'}) + w(v') = w_{j'}$ then
            $S_{j'} \leftarrow S_{j'} \cup \{v'\}$
        end
    end
end
If $|I| = 1$, assign all vertices $V \setminus \bigcup_{i\in[k]} S_i$ (possibly empty) to $S_j$ with $j \in I$.
```

4 GL-Partition for HHI_4^2-free

This section is dedicated to the proof of Theorem 2. The underlying idea for this result is to carefully contract edges to turn a k-connected HHI_4^2-free graph into a chordal graph that is still k-connected. Note that we indeed have to be very careful here to find a set of contractions, as we need it to satisfy three seemingly contradicting properties: removing all induced C_4, preserving k-connectivity, and contracting at most one edge adjacent to each vertex. The last property is needed to bound the maximum weight of the vertices in the contracted graph. Further, we have to be careful not to contract terminal vertices.

The computation for the unweighted case of the partition for Theorem 2 is given in Algorithm 3 below, which is later extended to the weighted case as well. Note that we can assume that $n_i \geq 2$ since if $n_i = 1$ for some $i \in [k]$ we simply declare the terminal vertex to be the required set and remove it from G. This gives us a $(k-1)$-connected graph and $k-1$ terminal vertices.

Before starting to prove the Lemmata required for the correctness of Algorithm 3 we give a structural insight which is used in almost all proofs of the following Lemmata.

Lemma 3. *Given an HHI$_4^2$-free graph G and an induced C_4, $C \subseteq V(G)$, then any vertex in $V(G) \backslash C$ that is adjacent to two vertices of C is universal to C. Moreover, the set of vertices that are universal to C induces a clique.*

Algorithm 3: HHI$_4^2$-free GL

```
Input: $k$-connected HHI$_4^2$-free graph $G(V, E)$, terminal vertices
       $t_1, \ldots, t_k \in V$, and positive integers $n_1, \ldots, n_k \geq 2$ satisfying
       $\sum_{i=1}^{k} n_i = n$
Output: Connected vertex sets $S_1, \ldots, S_k \subseteq V$ such that
        $n_i - 1 \leq |S_i| \leq n_i + 1$ and $t_i \in S_i$
Add an edge between each pair of non-adjacent terminals that are part of
  an induced $C_4$
$\mathcal{C} \leftarrow$ Set of all induced $C_4$ in $G$.
$G' \leftarrow (\bigcup_{C \in \mathcal{C}} V(C), \bigcup_{C \in \mathcal{C}} E(C))$
$E' \leftarrow \varnothing$
while $\mathcal{C} \neq \varnothing$ do
    Select three vertices $v_1, v_2, v_3$ in $G'$ and the corresponding cycle $C \in \mathcal{C}$
      that satisfies that for all $C' \in \mathcal{C} \backslash \{C\}$ we have $V(C') \cap \{v_1, v_2, v_3\} = \varnothing$.
    Pick a vertex $v$ from $v_1, v_2, v_3$ that is not a terminal vertex and add an
      incident edge of $v$ in $G'[\{v_1, v_2, v_3\}]$ to $E'$.
    Remove the cycle $C$ from $\mathcal{C}$ and the vertices $v_1, v_2, v_3$ from $G'$.
end
Transform $G$ to a weighted graph $G''$ by contracting each edge of $E'$ in $G$,
  assigning to each resulting vertex as weight the number of original
  vertices it corresponds to.
$S_1, \ldots S_k \leftarrow$ Run Algorithm 2 with $G''$, the given set of terminals
  $t_1, \ldots, t_k$, and the size (or weight) demands $n_1, \ldots, n_k$ as input.
Reverse the edge contraction of $E'$ in the sets $S_1, \ldots, S_k$ accordingly.
```

Lemma 4. *Let G be an HHI$_4^2$-free graph. If G contains a double house as a subgraph then at least one of the two C_4 in it has a chord.*

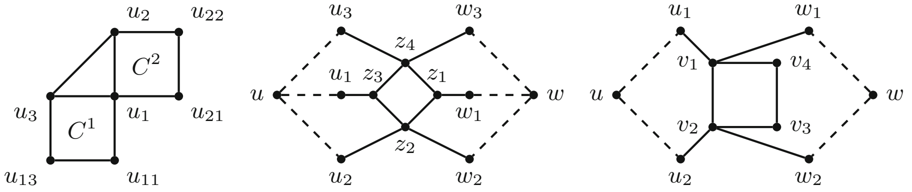


Fig. 2. Illustrations for the vertex namings used in proofs, from left to right: Lemma 4, Lemma 7 and Lemma 8

The following lemma captures the essence of why the algorithm provided in this section cannot be applied also on HHD-free graphs, since it holds for

HHI_4^2-free graphs but not for HHD-free graphs. Think for example of a simple path P of length 5 and a vertex disjoint induced chordless C_4, C. Consider also each vertex of P being universal to C. Observe that this graph is HHD- but not HHI_4^2-free . Every two non adjacent vertices of C together with the endpoints of P create an induced chordless C_4. Adding a chord connecting the two endpoints of P creates a hole and hence the resulting graph is not HHD-free.

Lemma 5. *Let G be an HHI_4^2-free graph and $C = \{v_1, v_2, v_3, v_4\}$ an induced C_4 in G. Then the graph G' created by adding the chord v_1v_3 to G is HHI_4^2-free and has one less induced C_4 than G.*

An essential property of the graph class we work on is being closed under contraction, since our algorithm is based on contracting edges iteratively until the resulting graph becomes chordal. Before proving this property though, although "after an edge contraction a new cycle is created" is intuitively clear, we formally define what it means for a C_4 to be "new".

Definition 1. *Let G be a graph, $uv \in E(G)$ and $G' = G/uv$. Let also w be the vertex of G' that is created by the contraction of uv. We say that an induced cycle C containing w in G' is new if $N_C(w) \nsubseteq N_G(v)$ and $N_C(w) \nsubseteq N_G(u)$.*

Lemma 6. *HHI_4^2-free graphs are closed under contraction of an edge of an induced C_4.*

In order to prove that the contractions of our algorithm do not affect the connectivity, we first study the possible role of vertices on an induced C_4 in minimal separators in HHI_4^2-free graphs.

Lemma 7. *Let G be a k-connected HHI_4^2-free graph for $k \geq 5$. Then no three vertices of an induced C_4 belong in the same minimal separator.*

Proof. Let G be a k-connected HHI_4^2-free graph for $k \geq 5$ and v_1, v_2, v_3, v_4 vertices that induce a C_4, C. Assume that v_1, v_2, v_3 belong in the a same minimal separator S and hence, $(G \backslash \{v_1, v_2, v_3\})$ is only $k-3$ connected. Let also u and w be two distinct vertices belonging in different connected components of $G \backslash S$.

Consider now the chordal graph G' created, by adding v_2v_4 to C and one chord to each other induced C_4 of G. By Lemma 5 this is possible by adding exactly one chord to each induced C_4 of G - in particular each addition does not create new induced C_4. Since G' is chordal each minimal separator induces a clique, and hence v_1, v_2, v_3 cannot be part of the same minimal separator in G' because they do not induce a triangle in G'. Thus $G' \backslash S$ remains connected.

Let P_1 be a $u - w$ path in $G' \backslash S$ that contains a minimal number of added edges. Let $z_1z_3 \in E(P_1)$ be one of the added edges, such that z_3 is closer to u on P_1 than z_1. Note that z_1 and z_3 are part of some induced C_4, $C' = \{z_1, z_2, z_3, z_4\}$ in G. Since z_1z_3 cannot be replaced by neither z_1z_2, z_2z_3, nor z_1z_4, z_4z_3 (otherwise we get a path with strictly less added edges than P_1) it follows that $z_2, z_4 \in S$.

We will use the $u-w$ paths through S in G to reach a contradiction. Since S is a minimal $u-w$ separator in G, there are two internally vertex disjoint $u-w$ paths P_2 and P_3, with $P_2 \cap S = \{z_2\}$ and $P_3 \cap S = \{z_4\}$. Let w_2 be the neighbor of z_2 on P_2 closer to w, w_1 the respective neighbor of z_1 on P_1 and w_3 the respective neighbor of z_4 on P_3. Let also u_1, u_2, u_3 be the corresponding neighbors of these paths closer to u. See the illustration in Fig. 2 for these namings, keeping in mind that it could be $w_1 \in \{w_2, w_3\}$ or $u_1 \in \{u_2, u_3\}$ or also $w_1 = w_2 = w_3 = w$ or $u_1 = u_2 = u_3 = u$.

We claim that, in G, z_3 is adjacent to a vertex on $P_2^{[w_2,w]}$ or $P_3^{[w_3,w]}$. Assume otherwise, and assume that $P_2^{[w_2,w]}$, $P_3^{[w_3,w]}$ are induced paths in G (shortcut them otherwise). If $w_2 = w_3$ then notice that $w_2 = w = w_3$. In order for z_3, z_2, z_4, w not to induce a C_4 with three common vertices to C, z_3 has to be adjacent to w which is on $P_2^{[w_2,w]}$. If $w_2 \neq w_3$ then assume without loss of generality that $w_2 \neq w$. In order to not be a hole, there has to be a chord in the cycle build by $P_2^{[w_2,w]}$, $P_3^{[w_3,w]}$ with z_4, z_3, z_2. By assumption, this chord cannot be from z_3, so it has to involve z_4 or z_2. Since $P_2^{[w_2,w]}$ and $P_3^{[w_3,w]}$ are induced and $w_2 \neq w$, either w_2 is adjacent to z_4, or $w_3 \neq w$ is adjacent to z_2. Both cases create a C_4 that has three vertices in common with C, (w_2, z_2, z_3, z_4, and w_3, z_2, z_3, z_4, resp.) and since $z_4 z_2 \notin E(G)$, the added chord for these C_4 has to be $w_2 z_3$, resp. $w_3 z_3$, leading again to z_3 being adjacent to some vertex on $P_2^{[w_2,w]}$ or $P_3^{[w_3,w]}$.

Thus we conclude that z_3 is adjacent to a vertex x on $P_2^{[w_2,w]}$ or $P_3^{[w_3,w]}$ in G. This however allows to create a path from u to w with (at least) one added edge less than P_1 in G (since P_2, P_3 do not contain any added edges). Specifically, if x is on P_2 we get $P_1' = P_1^{[u,z_3]} x P_2^{[x,w]}$ and if $x \in P_3$, $P_1' = P_1^{[u,z_3]} x P_3^{[x,w]}$.

Since C' was an arbitrary cycle we conclude that v_1, v_2, v_3 cannot be part of the same minimal separator in G.

Lemma 8. *Let G be an HHI_4^2-free k-connected graph and $C = \{v_1, v_2, v_3, v_4\}$ be an induced C_4. The graph $G' = G/v_1v_2$ is still k-connected.*

Now, we finally look specifically at Algorithm 3, and first show that its subroutine creating G'' works correctly.

Lemma 9. *Given an HHI_4^2-free graph G, the vertices selected in line 6 of Algorithm 3 indeed exist as long as an induced C_4 exists.*

Proof. Let G be an HHI_4^2-free graph and $\mathcal{C}$ the set of all induced C_4 in G, consider the bipartite graph T constructed through the following procedure: Its vertices are partitioned into two sets B, and S referred to as *big* and *small* vertices of T, respectively. Each big vertex represents an induced C_4 of $\mathcal{C}$ while each small vertex represents a vertex of G participating in at least two induced C_4. Each small vertex is adjacent to the big vertices which represent a C_4 this vertex participates in. We claim that with this definition T is indeed a tree (actually a forest). Assume now for a contradiction that T contains a cycle and let C be one of the shortest such cycles in T.

First, consider the case that C has length $l \geq 6$. Since T is bipartite, due to its construction, l is even and the vertices of $C = \{s_1, b_1, s_2, b_2, \ldots, s_{l/2}, b_{l/2}\}$ alternate between big and small. We denote by $P_{b_i}^{s_w, s_z}$ a shortest path containing edges from the C_4 represented by the big vertex b_i with endpoints the vertices represented by s_w and s_z. Due to C, the cycle $P_{b_1}^{s_1, s_{l/2}} \ldots P_{b_{l/2}}^{s_{l/2-1}, s_{l/2}}$ exists in G as a subgraph. Note that since we have assumed that C is a minimal length cycle of T it is also chordless. Hence, in order for a hole not to be an induced subgraph of G at least one chord must exist connecting two vertices corresponding to two small ones of T. This however would create a double house as a subgraph with the two C_4 forming it being the two that correspond to big vertices of T. By Lemma 4 this means that one of the C_4 is not induced, a contradiction to the construction of T. Notice also that in the case where $l = 6$ we directly find a double house and reach a contradiction using the same arguments.

Moreover the assumption that $l = 4$, leads us to a contradiction to the fact that two C_4 have at most one vertex in common. Hence, T is a forest and the vertices mentioned in line 6 are the ones belonging only to a cycle represented by one leaf belonging in B.

Lemma 10. *Given an HHI$_4^2$-free graph G, lines 1–10 of Algorithm 3 transforms G into a weighted chordal graph G'', with the same connectivity as G and such that each vertex from G is involved in at most one edge contraction to create G''.*

At last, notice that we can easily alter Algorithm 3 to also work for weighted graphs, with the simple change of setting the weights of a vertex in G'' in line 10 to the sum of the weights of the original vertices it was contracted from. With this alteration, we can conclude now the proof of Theorem 2 with the following.

Lemma 11. *Algorithm 3 works correctly and runs in time $\mathcal{O}(|V|^4)$.*

Proof. By Lemma 10, G'' is a chordal graph with maximum vertex weight $2w_{max}$. Further, observe that we did not merge terminal vertices with each other, thus we can properly run Algorithm 2 on it. By the correctness of this algorithm (Theorem 1), we know that $S_1, \ldots, S_k$ in line 11 is a GL-partition for G'' with deviation $2w_{max}$. Since reversing edge-contraction does not disconnect these sets, the unfolded sets $S_1, \ldots, S_k$ are thus also a GL-partition for G with deviation $2w_{max}$; note here that the only edges we added to create G'' are between terminal vertices, which are in separate sets S_i by definition.

The most time consuming part of Algorithm 3 is the preprocessing to transform the input graph into a weighted chordal graph which requires $\mathcal{O}(|V|^4)$ time in order to find all the induced C_4 (note that the induced C_4 are at most $(n-4)/3$ since they induce a tree).

Moreover, as is the case for chordal graphs, we can sacrifice terminals to enlarge the considered graph class.

Corollary 3. *Let G be a k-connected graph which becomes HHI_4^2-free after adding $k/2$ edges. Then, given those edges, a GL-Partition of G with deviation* 1 *(also its weighted version with deviation $2w_{max} - 1$) can be computed in polynomial time but without fixed terminals.*

References

1. Broersma, H., Dahlhaus, E., Kloks, T.: Algorithms for the treewidth and minimum fill-in of HHD-free graphs. In: International Workshop on Graph-Theoretic Concepts in Computer Science (WG), pp. 109–117 (1997). https://doi.org/10.1007/BFb0024492
2. Chandran, L.S., Cheung, Y.K., Issac, D.: Spanning tree congestion and computation of generalized Györi-Lovász partition. In: International Colloquium on Automata, Languages, and Programming, (ICALP). LIPIcs, vol. 107, pp. 32:1–32:14 (2018). https://doi.org/10.4230/LIPIcs.ICALP.2018.32
3. Chen, J., Kleinberg, R.D., Lovász, L., Rajaraman, R., Sundaram, R., Vetta, A.: (Almost) tight bounds and existence theorems for single-commodity confluent flows. J. ACM **54**(4), 16 (2007). https://doi.org/10.1145/1255443.1255444
4. Győri, E.: On division of graphs to connected subgraphs, combinatorics. In: Colloq. Math. Soc. Janos Bolyai, 1976 (1976)
5. Hoyer, A.: On the independent spanning tree conjectures and related problems. Ph.D. thesis, Georgia Institute of Technology (2019)
6. Jamison, B., Olariu, S.: On the semi-perfect elimination. Adv. Appl. Math. **9**(3), 364–376 (1988)
7. Lovász, L.: A homology theory for spanning tress of a graph. Acta Math. Hungar. **30**(3–4), 241–251 (1977)
8. Lucertini, M., Perl, Y., Simeone, B.: Most uniform path partitioning and its use in image processing. Discrete Appl. Math. **42**(2), 227–256 (1993). https://doi.org/10.1016/0166-218X(93)90048-S
9. Möhring, R.H., Schilling, H., Schütz, B., Wagner, D., Willhalm, T.: Partitioning graphs to speedup Dijkstra's algorithm. ACM J. Exp. Algorithmics **11**, 2–8 (2006). https://doi.org/10.1145/1187436.1216585
10. Nakano, S., Rahman, M.S., Nishizeki, T.: A linear-time algorithm for four-partitioning four-connected planar graphs. Inf. Process. Lett. **62**(6), 315–322 (1997). https://doi.org/10.1016/S0020-0190(97)00083-5
11. Przytycka, T.M.: An important connection between network motifs and parsimony models. In: Apostolico, A., Guerra, C., Istrail, S., Pevzner, P.A., Waterman, M. (eds.) RECOMB 2006. LNCS, vol. 3909, pp. 321–335. Springer, Heidelberg (2006). https://doi.org/10.1007/11732990_27
12. Przytycka, T.M., Davis, G.B., Song, N., Durand, D.: Graph theoretical insights into evolution of multidomain proteins. J. Comput. Biol. **13**(2), 351–363 (2006). https://doi.org/10.1089/cmb.2006.13.351
13. Rose, D.J., Tarjan, R.E., Lueker, G.S.: Algorithmic aspects of vertex elimination on graphs. SIAM J. Comput. **5**(2), 266–283 (1976). https://doi.org/10.1137/0205021
14. Suzuki, H., Takahashi, N., Nishizeki, T., Miyano, H., Ueno, S.: An algorithm for tripartitioning 3-connected graphs. J. Inf. Process. Soc. Japan **31**(5), 584–592 (1990)
15. Suzuki, H., Takahashi, N., Nishizeki, T.: A linear algorithm for bipartition of biconnected graphs. Inf. Process. Lett. **33**(5), 227–231 (1990). https://doi.org/10.1016/0020-0190(90)90189-5

16. Wada, K., Kawaguchi, K.: Efficient algorithms for tripartitioning triconnected graphs and 3-edge-connected graphs. In: van Leeuwen, J. (ed.) WG 1993. LNCS, vol. 790, pp. 132–143. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-57899-4_47
17. Zhou, X., Wang, H., Ding, B., Hu, T., Shang, S.: Balanced connected task allocations for multi-robot systems: an exact flow-based integer program and an approximate tree-based genetic algorithm. Expert Syst. Appl. **116**, 10–20 (2019). https://doi.org/10.1016/j.eswa.2018.09.001

Generating Faster Algorithms for d-Path Vertex Cover

Radovan Červený(✉) and Ondřej Suchý(✉)

Department of Theoretical Computer Science, Faculty of Information Technology,
Czech Technical University in Prague, Prague, Czech Republic
{radovan.cerveny,ondrej.suchy}@fit.cvut.cz

Abstract. Many algorithms which exactly solve hard problems require branching on more or less complex structures in order to do their job. Those who design such algorithms often find themselves doing a meticulous analysis of numerous different cases in order to identify these structures and design suitable branching rules, all done by hand. This process tends to be error prone and often the resulting algorithm may be difficult to implement in practice.

In this work, we aim to automate a part of this process and focus on the simplicity of the resulting implementation.

We showcase our approach on the following problem. For a constant d, the d-Path Vertex Cover problem (d-PVC) is as follows: Given an undirected graph and an integer k, find a subset of at most k vertices of the graph, such that their deletion results in a graph not containing a path on d vertices as a subgraph. We develop a fully automated framework to generate parameterized branching algorithms for the problem and obtain algorithms outperforming those previously known for $3 \leq d \leq 8$, e.g., we show that 5-PVC can be solved in $O(2.7^k * n^{O(1)})$ time.

1 Introduction

The motivation behind this paper is to renew the interest in computer aided design of graph algorithms which was initiated by Gramm et al. [22]. Many parameterized branching algorithms follow roughly the same pattern: 1) perform a meticulous case analysis; 2) based on the analysis, construct branching and reduction rules; 3) argue that once the rules cannot be applied, some specific structure is achieved. Also, depending on how "deeply" you perform the case analysis, you may slightly improve the running time of the algorithm, but bring nothing new to the table.

This paper aims to provide a framework which could help in the first two steps of the pattern at least for some problems. We phrase the framework for a rather general problem which is as follows. For any nonempty finite set of

*The authors acknowledge the support of the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics" and the Grant Agency of the CTU in Prague funded grant No. SGS20/208/OHK3/3T/18.

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 157–171, 2023.
https://doi.org/10.1007/978-3-031-43380-1_12

connected graphs $\mathcal{F}$ we define the problem $\mathcal{F}$-Subgraph Vertex Deletion, $\mathcal{F}$-SVD, where, given a graph $G = (V, E)$ and an integer k, the task is to decide whether there is a subset S of at most k vertices of G such that $G \backslash S$ does not contain any graph from $\mathcal{F}$ as a subgraph (not even as a non-induced one). While we only apply the framework to the problem of d-PVC defined later, the advantage of phrasing the framework for $\mathcal{F}$-SVD is twofold. First, it makes it easier to apply it to other problems. Second, the general notation introduced makes the description less cluttered.

Since the problem is NP-complete for most reasonable choices of $\mathcal{F}$, as follows from the meta-theorem of Lewis and Yannakakis [27], any algorithm solving the problem exactly is expected to have exponential running time. In this paper we aim on the parameterized analysis of the problem, that is, to confine the exponential part of the running time to a specific parameter of the input, presumably much smaller than the input size. In particular, we only use the most standard parameter, which is the desired size of the solution k, also called *the budget*. Algorithms achieving running time $f(k)n^{O(1)}$ are called *parameterized*, *fixed-parameter tractable*, or *FPT*. See Cygan et al. [12] for a broader introduction to parameterized algorithms.

To understand how parameterized branching algorithms typically work, consider the following simple recursive algorithm for $\mathcal{F}$-SVD. We find in the input graph G an occurrence F' of graph F from $\mathcal{F}$. We know that at least one of the vertices of F' must be in any solution. Hence, for each vertex of F' we try adding it to a prospective solution, decreasing the remaining budget by one, and recursing. The recursion is stopped when the budget is exhausted, or there are no more occurrences of graphs from $\mathcal{F}$ in G, i.e., we found a solution. It is easy to analyze that this algorithm has running time[1] $\mathcal{O}^*(d^k)$, where d is the number of vertices of the largest graph in $\mathcal{F}$. Many parameterized branching problems follow a similar scheme, branching into a constant number of alternatives in each step, for each alternative making a recursive call with the budget (or some other parameter) decreased by some constant.

One can improve upon this trivial algorithm by looking at F' together with its surroundings. Working with this larger graph F'' often allows for more efficient branching as now multiple *overlapping* occurences of graphs from $\mathcal{F}$ may appear in F'' instead of just one. Our framework and that of Gramm et al. [22] rely upon this observation, as they iteratively take larger and larger graphs into consideration—similarly to what a human would do, but on a much larger scale.

The fundamental novelty of our framework in comparison to that of Gramm et al. [22] is that we are able to identify which vertices of the graph F'' under consideration can still have outside neighbors and which do not. We call the latter "red". This way we are able to say that if you find an occurrence of F'' in the input graph, you can be sure that the red vertices do not have neighbors in the input graph apart from those that are in F''.

This additional information allows us to eliminate some branches of the constructed branching rules, rapidly improving their efficiency. It also reduces the

[1] The $\mathcal{O}^*()$ notation suppresses all factors polynomial in the input size.

number of graphs we need to consider and also allows us to design better reduction rules to aid our framework.

We apply the general framework to the problem of d-Path Vertex Cover (d-PVC). The problem lies in determining a subset S of vertices of a given graph $G = (V, E)$ of at most a given size k such that $G \backslash S$ does not contain a path on d vertices (even not a non-induced one). It was first introduced by Brešar et al. [2], but its NP-completeness for any $d \geq 2$ follows already from the above-mentioned meta-theorem of Lewis and Yannakakis [27]. The 2-PVC problem corresponds to the well known Vertex Cover problem and the 3-PVC problem is also known as Maximum Dissociation Set or Bounded Degree-One Deletion. The d-PVC problem is motivated by the field of designing secure wireless communication protocols [31] or route planning and speeding up shortest path queries [20].

As mentioned above, d-PVC is directly solvable by a trivial FPT algorithm that runs in $\mathcal{O}^*(d^k)$ time. However, since d-PVC is a special case of d-Hitting Set, it follows from the results of Fomin et al. [17] that for any $d \geq 4$ we have an algorithm solving d-PVC in $\mathcal{O}^*((d - 0.9245)^k)$ time. For $d \geq 6$ algorithms with even better running times are presented in the work of Fernau [15].

In order to find more efficient solutions, the problem has been extensively studied in a setting where d is a small constant. This is in particular the case for the 2-PVC (Vertex Cover) problem [1,3,6,8,11,13,29,30], where the algorithm of Chen, Kanj, and Xia [10] for a long time held the best known running time of $\mathcal{O}^*(1.2738^k)$, but recently Harris and Narayanaswamy [23] claimed the running time of $\mathcal{O}^*(1.25288^k)$. For 3-PVC, Tu [37] used iterative compression to achieve a running time of $\mathcal{O}^*(2^k)$. This was later improved by Katrenič [24] to $\mathcal{O}^*(1.8127^k)$, by Xiao and Kou [40] to $\mathcal{O}^*(1.7485^k)$ by using a branch-and-reduce approach and finally by Tsur [34] to $\mathcal{O}^*(1.713^k)$. For the 4-PVC problem, Tu and Jin [38] again used iterative compression and achieved a running time of $\mathcal{O}^*(3^k)$ and Tsur [35] gave the current best algorithm that runs in $\mathcal{O}^*(2.619^k)$ time. The authors of this paper developed an $\mathcal{O}^*(4^k)$ algorithm for 5-PVC [4]. For $d = 5$, 6, and 7 Tsur [36] discovered algorithms for d-PVC with running times $\mathcal{O}^*(3.945^k)$, $\mathcal{O}^*(4.947^k)$, and $\mathcal{O}^*(5.951^k)$, respectively.

Using our automated framework, we are able to present algorithms with improved running times for some d-PVC problems when parameterized by the size of the solution k. The results are summarized in Table 1.

Further Related Work. The only other approach to generating algorithms with provable worst-case running time upper bounds we are aware of is limited to algorithms for SAT [14,25,26].

Several moderately exponential exact algorithms are known for 2-PVC and 3-PVC [7,39,41].

Full Version of the Paper. Due to space constraints, we omit most technical details from this extended abstract. We refer the kind reader to the full version of the paper [5].

Table 1. Improved running times of some d-PVC problems.

d-PVC	Previously known	Our result	Our # of rules
2-PVC	$\mathcal{O}^*(1.25288^k)$ [23]	$\mathcal{O}^*(1.3294^k)$	9,345,243
3-PVC	$\mathcal{O}^*(1.713^k)$ [34]	$\mathcal{O}^*(1.708^k)$	1,226,384
4-PVC	$\mathcal{O}^*(2.619^k)$ [35]	$\mathcal{O}^*(2.138^k)$	911,193
5-PVC	$\mathcal{O}^*(3.945^k)$ [36]	$\mathcal{O}^*(2.636^k)$	739,542
6-PVC	$\mathcal{O}^*(4.947^k)$ [36]	$\mathcal{O}^*(3.334^k)$	414,247
7-PVC	$\mathcal{O}^*(5.951^k)$ [36]	$\mathcal{O}^*(3.959^k)$	5,916,297
8-PVC	$\mathcal{O}^*(7.0237^k)$ [15]	$\mathcal{O}^*(5.654^k)$	296,044

2 Fundamental Definitions and Basic Observations

In this paper we are going to assume that vertex sets of all graphs are finite subsets of $\mathbb{N}$, the set of all non-negative integers, i.e., we have a set of all graphs. Furthermore, when adding a graph into a set of graphs, we only add the graph if none of the graphs already in the set is isomorphic to it. Similarly, when forming a set of graphs we only add one representative for each isomorphism class. Finally, when subtracting a graph from a set, we remove from the set all graphs isomorphic to it.

For any nonempty finite set of connected graphs $\mathcal{F}$ we define the problem:

$\mathcal{F}$-Subgraph Vertex Deletion, $\mathcal{F}$-SVD	
Input:	A graph $G = (V, E)$, an integer $k \in \mathbb{N}$
Output:	A set $S \subseteq V$, such that $\lvert S\rvert \leq k$ and no subgraph of $G \backslash S$ is isomorphic to a graph in $\mathcal{F}$

We call $\mathcal{F}$ of $\mathcal{F}$-SVD a *bump-inducing* set. We call a graph G *bumpy* if it contains some graph from the bump-inducing set $\mathcal{F}$ as a subgraph. We call a vertex subset S a *solution* (for a graph $G = (V, E)$), if the graph $G \backslash S$ is not bumpy. Since $\mathcal{F}$ is finite, checking if G is bumpy is polynomial in the size of G.

Next we define a variant of a supergraph with a restriction that the original graph has to be an induced subgraph of the supergraph.

Definition 1 (expansion, i-expansion, σ, σ_i, σ^*). *Let H be a connected graph. A graph G is an* expansion *of H, if G is connected, $V(H) \subseteq V(G)$ and $G[V(H)] = H$. It is an i-*expansion *for $i \in \mathbb{N}$ if furthermore $\lvert V(G)\rvert = \lvert V(H)\rvert + i$. For $i \in \mathbb{N}$ let $\sigma_i(H)$ denote the set of all i-expansions of H (note again that we take only one representative for each isomorphism class). As shorthand, we will use $\sigma(H) = \sigma_1(H)$. Let $\sigma^*(H) = \bigcup_{i \in \mathbb{N}} \sigma_i(H)$ denote the set of all expansions of H.*

The following (restricted) variant of a branching rule is the building block of our algorithm.

Definition 2 (Subgraph branching rule). *A* subgraph branching rule *is a triple* $(H, R, \mathcal{B})$*, where* H *is a connected bumpy graph,* $R \subseteq V(H)$ *is a set of* red *vertices (representing the vertices supposed not to have neighbors outside* H*), and* $\mathcal{B} \subseteq \left(2^{V(H)} \setminus \{\emptyset\}\right)$ *is a non-empty set of* branches*.*

Definition 3 (An application of a subgraph branching rule). *We say that a subgraph branching rule* $(H, R, \mathcal{B})$ applies *to graph* G*, if* G *contains an induced subgraph* H' *isomorphic to* H *by isomorphism* $\phi : V(H) \to V(H')$ (witnessing isomorphism) *and for every* $r \in R$ *we have* $N_G(\phi(r)) \subseteq V(H')$*. In other words, the vertices of* H' *corresponding to red vertices only have neighbors inside the subgraph* H'*. If the rule applies and the current instance is* (G, k)*, then the algorithm makes for each* $B \in \mathcal{B}$ *a recursive call with instance* $(G \setminus \phi(B), k - |B|)$*.*

Note that we do not allow $\emptyset \in \mathcal{B}$. Therefore the budget gets reduced and we are making progress in every branch.

Definition 4 (Correctness of a subgraph branching rule). *A subgraph branching rule* $(H, R, \mathcal{B})$ *is* correct*, if for every* G *and every solution* S *for* G *such that* $(H, R, \mathcal{B})$ *applies to* G *and* $\phi : V(H) \to V(H')$ *is the witnessing isomorphism, there exists a solution* S' *for* G *with* $|S'| \leq |S|$ *and a branch* $B \in \mathcal{B}$ *such that* $\phi(B) \subseteq S'$*.*

Definition 5 (Branching factor of a subgraph branching rule). *For any subgraph branching rule* $(H, R, \mathcal{B})$ *let* $bf((H, R, \mathcal{B}))$ *be the branching factor of the branches in* $\mathcal{B}$*, i.e., the unique positive real solution of the equation:* $1 = \sum_{B \in \mathcal{B}} x^{-|B|}$ *(see [19, Chapter 2.1 and Theorem 2.1] for more information on (computing) branching factors).*

Observation 1. *For any connected bumpy graph* H *and any* $R \subseteq V(H)$ *we can always construct at least one correct subgraph branching rule.*

The following definition formalizes a function that, given a graph H and a set of vertices R, computes a set $\mathcal{B}$ of branches such that $(H, R, \mathcal{B})$ is a correct subgraph branching rule.

Definition 6 (Brancher). *A* brancher *is a function which assigns to any connected bumpy graph* H *and* $R \subseteq V(H)$ *a correct branching rule* $\tau(H, R) = (H, R, \mathcal{B})$ *for some non-empty set* $\mathcal{B} \subseteq \left(2^{V(H)} \setminus \{\emptyset\}\right)$*. For a brancher* τ *as a shorthand let* $\tau(H) = \tau(H, \emptyset)$*. For a set of graphs* $\{H_1, H_2, \ldots, H_r\}$ *we will have* $\tau(\{H_1, H_2, \ldots, H_r\}) = \{\tau(H_1), \tau(H_2), \ldots, \tau(H_r)\}$*.*

The above observation shows that at least one brancher exists.

Definition 7. *For a set of subgraph branching rules* $\mathcal{L} = (\varrho_1, \varrho_2, \ldots, \varrho_r)$ *where* $\varrho_i = (H_i, R_i, \mathcal{B}_i)$ *we will denote* $\Psi(\mathcal{L}) = \max\{|V(H_i)| \mid (H_i, R_i, \mathcal{B}_i) \in \mathcal{L}\}$ *the maximum number of vertices among the graphs of the subgraph branching rules in* $\mathcal{L}$*.*

The framework makes possible to introduce a number of handmade reduction[2] or branching rules, denoted as $\mathcal{A}$, to help the generating algorithm steer it away from some difficult corner cases. Typically, their purpose is to ensure some substructures no longer appear in the input graph.

Next we define the crucial property of a set of subgraph branching rules which forms a base for the proof of correctness of the generated algorithm.

Definition 8. *A set of subgraph branching rules $\mathcal{L} = (\varrho_1, \varrho_2, \ldots, \varrho_r)$ is called* exhaustive with respect to $\mathcal{A}$ *if every rule ϱ_i is correct and for every connected bumpy graph G to which no handmade rule in $\mathcal{A}$ applies and which has at least $\Psi(\mathcal{L})$ vertices there is a subgraph branching rule ϱ_i in $\mathcal{L}$ that applies to G. If the set is exhaustive with respect to $\emptyset$, that is, even without any handmade rules, we will omit the "with respect to $\mathcal{A}$" clause.*

In the process of generating the algorithm, we aim to maintain an exhaustive set of subgraph branching rules at all times.

The following observation identifies our starting set of graphs.

Observation 2. *Let $\mathcal{F}$ be the bump-inducing set of some $\mathcal{F}$-*SVD *problem. Let $f = \max_{H \in \mathcal{F}} |V(H)|$. Let $L = \{F_1, F_2, \ldots, F_r\}$ be the set of all connected bumpy graphs with f vertices. Let τ be a brancher. Then the set of subgraph branching rules $\mathcal{L} = \tau(L)$ is exhaustive.*

3 The Output Algorithm and Its Correctness

Our goal will be to obtain a set $\mathcal{L}$ of subgraph branching rules with good branching factors which is exhaustive with respect to $\mathcal{A}$. This section summarizes how we use the set to design an algorithm for $\mathcal{F}$-SVD once we obtain such a set. We call the algorithm $(\mathcal{A}, \mathcal{L})$-Algorithm for $\mathcal{F}$-SVD and its pseudocode is available in Algorithm 1.

The algorithm first applies some trivial stopping conditions (lines 3 to 5). Then it applies the rules from $\mathcal{A}$ (lines 6 to 7). Next, if every connected component is small, it finds a solution for each of them separately by a brute force (lines 8 to 12). Finally, it takes a component which is large enough and finds a subgraph branching rule from $\mathcal{L}$ that applies to the component and applies it by making the appropriate recursive calls (lines 13 to 18).

The following theorem states that this algorithm is indeed correct.

Theorem 1. *Let $\mathcal{A}$ be a list of handmade rules and $\mathcal{L}$ be a set of subgraph branching rules. If $\mathcal{L}$ is exhaustive with respect to $\mathcal{A}$, all rules in $\mathcal{A}$ are correct and can be applied in polynomial time, and each branching rule in $\mathcal{A} \cup \mathcal{L}$ has branching factor at most β, then the $(\mathcal{A}, \mathcal{L})$-Algorithm for $\mathcal{F}$-*SVD *is correct and runs in $\mathcal{O}^*(\beta^k)$ time.*

[2] Roughly speaking, a reduction rule is a polynomial-time procedure that replaces the input instance with another one, preserving the answer.

Implementation Considerations. We want to emphasize, that the effort needed to implement the algorithm does not grow with the number of generated rules in $\mathcal{L}$ as the code that implements the mechanic on line 14 remains the same regardless of the number of rules. Further, the generated list $\mathcal{L}$ is given encoded in a machine-readable format, which further simplifies the implementation.

Algorithm 1. $(\mathcal{A}, \mathcal{L})$-Algorithm for $\mathcal{F}$-SVD

```
1: Let $\mathcal{A}$ be a list of handmade rules and $\mathcal{L}$ be a set of subgraph branching rules.
2: function $SolveRecursively(G, k)$
3:     if $k < 0$ then Return NO.
4:     if $G$ is not bumpy then Return YES.
5:     if $k = 0$ then Return NO.
6:     if Some rule from $\mathcal{A}$ can be applied to $G$ then
7:         Find the first rule $\varrho_A$ from $\mathcal{A}$ that can be applied to $G$. Apply $\varrho_A$ to $G$ and return the corresponding answer (might involve recursive calls to $SolveRecursively$).
8:     if Each bumpy connected component of $G$ has less than $\Psi(\mathcal{L})$ vertices then
9:         Find the optimal solution for each component separately by brute-force.
10:        Let the solutions be $S_1, S_2, \ldots, S_c$.
11:        if $\sum_{i=1}^{c} |S_i| \leq k$ then Return YES.
12:        else Return NO.
13:    Let $C$ be the vertices of the bumpy connected component of $G$ with at least $\Psi(\mathcal{L})$ vertices.
14:    Find a branching rule $(H, R, \mathcal{B})$ from the set $\mathcal{L}$ that can be applied to $G[C]$.
15:    Let $\phi$ be the corresponding isomorphism.
16:    for $B \in \mathcal{B}$ do
17:        if $SolveRecursively(G \backslash \phi(B), k - |B|)$ outputs YES then Return YES.
18:    Return NO.
```

4 The Generating Algorithm

In this section we describe the algorithm to generate a suitable list of subgraph branching rules.

For a fixed $\mathcal{F}$-SVD problem the input of the algorithm are the bump-inducing set $\mathcal{F}$, a function $Handled_{\mathcal{A}}$ which can identify the situations handled by the handmade branching and reduction rules in $\mathcal{A}$, and the target branching factor $\beta \in \mathbb{R}$. We assume that the handmade rules in $\mathcal{A}$ are correct in the context of the given $\mathcal{F}$-SVD problem, they can be applied in polynomial time, and that the branching rules have branching factors at most β. The output of the algorithm is an ordered list of subgraph branching rules $\mathcal{L}$, exhaustive with respect to $\mathcal{A}$, such that every rule in $\mathcal{L}$ has branching factor at most β. The algorithm will be called the $(\mathcal{F}, \mathcal{A}, \beta)$-Algorithm and its output satisfies the assumptions of Theorem 1.

4.1 Overview of the Algorithm

The algorithm maintains an ordered list and a set of connected bumpy graphs named L_{good} and L_{bad}, respectively. The list L_{good} stores graphs that already give rise to good subgraph branching rules, whereas the set L_{bad} represents the substructures for which the algorithm did not find any effective way to tackle them yet. The algorithm starts with L_{good} empty and L_{bad} being the set from Observation 2.

The algorithm works in rounds and in each round it tries to move as many graphs currently in L_{bad} to L_{good}. Firstly, for each graph in L_{bad}, the algorithm "colors red" the vertices that cannot have any outside neighbors. This process is done by the *Color* function introduced below. Secondly, for each now colored graph in L_{bad}, it checks whether the substructure can be handled by some hand-made rule in $\mathcal{A}$. If it does, the graph is moved from L_{bad} to the end of L_{good}. Otherwise, it designs a subgraph branching rule for it with the smallest branching factor it can achieve and if the branching factor of the produced rule is at most β, it again moves the graph from L_{bad} to the end of L_{good}. In one round, the algorithm repeats the above steps as long as possible. Once no graph from L_{bad} can be moved to L_{good} this way, the algorithm replaces all graphs in L_{bad} by all their 1-expansions and starts a next round. This corresponds to deepening the analysis, i.e., considering larger parts of the input graph at once.

4.2 *Color* function

Let H be a connected graph and F be a set of connected graphs. The vertex $v \in V(H)$ will be colored red, i.e., we put v into R, if all 1-expansions of H, where the vertex v has more neighbors in the 1-expansion of H than in H itself, are already also expansions of some graphs in F.

In our algorithm, H is some graph from L_{bad} and F is the list L_{good} (see Fig. 1 for an example).

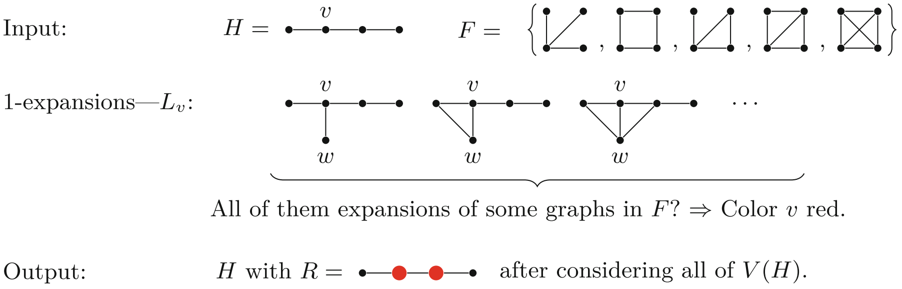

Input:
$H =$
v
$F =$
1-expansions—L_v:
v
w
v
w
v
w
$\cdots$
All of them expansions of some graphs in F? ⇒ Color v red.
Output:
H with $R =$
after considering all of $V(H)$.

Fig. 1. Illustration of the *Color* function.

5 Generating Subgraph Branching Rules

Once we have a graph H together with its red vertices $R \subseteq V(H)$, we want to generate a correct subgraph branching rule for it with as small branching factor as possible.

5.1 Overview of the Approach

We start by brute forcing all the local solutions for the graph H, we keep only those that are inclusion-wise minimal, and we use them to get our initial set of branches $\mathcal{B}_{min}$. It is easy to see, that the resulting subgraph branching rule $(H, R, \mathcal{B}_{min})$ is correct, but not very efficient.

To improve this rule, we employ a function called *DominanceFree* (described below) which uses the red vertices R to filter out some unnecessary branches. Let us label the result as $\mathcal{B}_{df}$.

Finally, we further optimize $\mathcal{B}_{df}$ with the following observation.

Observation 3. *Let $(H, R, \mathcal{B})$ be a correct subgraph branching rule. For any $A \subseteq V(H), A \neq \emptyset$ construct the branches $\mathcal{B}_A = \{B \mid B \in \mathcal{B} \wedge A \nsubseteq B\} \cup \{A\}$. The subgraph branching rule $(H, R, \mathcal{B}_A)$ is correct.*

We greedily improve the branches $\mathcal{B}_{df}$ by repeatedly trying all possible replacements and picking those that minimize the branching factor the most. Let us label the result $\mathcal{B}_{adj}$.

The final generated subgraph branching rule is then $(H, R, \mathcal{B}_{adj})$.

5.2 *DominanceFree* function

The input of the function is a connected bumpy graph H, $R \subseteq V(H)$, and $\mathcal{B}$ such that $(H, R, \mathcal{B})$ is a correct subgraph branching rule.

The point is that if a vertex v has no neighbors outside H (the red vertices), then it might be more beneficial to have a different vertex in the solution instead of v. We call this the dominance between branches.

The basic idea is to take a subset R^* of the red vertices and replace all vertices of the solution in this set by the open neighborhood $N_H(R^*) \backslash R^*$. We only want to do that if this does not increase the size of the solution and if $H[R^*]$ is not bumpy. To increase the power of this notion, we do this in a graph $H' = H \backslash B^{del}$, where B^{del} is a set of vertices shared by both the branches.

Definition 9 (Dominated branch). *Let $(H, R, \mathcal{B})$ be a correct subgraph branching rule. We say that branch $B \in \mathcal{B}$ is* dominated *by branch $B_d \in \mathcal{B}$ if $B_d \neq B$ and there exists a subset $B^{del} \subsetneq B$ such that for $H' = H \backslash B^{del}$, $R' = R \backslash B^{del}$ there exists a subset $R^* \subseteq R', R^* \neq \emptyset$ such that the following holds:*

1. *$H[R^*]$ is not bumpy,*
2. *$|R^* \cap B| \geq |N_{H'}(R^*) \backslash R^*| \geq 1$,*
3. *$B_d \subseteq (B \cup N_{H'}(R^*)) \backslash R^*$.*

Note that if $N_{H'}(R^*) \backslash R^* = \emptyset$, then $B_d \subseteq B$, a case that cannot appear in $\mathcal{B}_{min}$.

Lemma 1. *If $(H, R, \mathcal{B})$ is a correct subgraph branching rule and branch $B \in \mathcal{B}$ is dominated by branch $B_d \in \mathcal{B}$, then $(H, R, \mathcal{B} \backslash \{B\})$ is a correct subgraph branching rule.*

The purpose of the *DominanceFree* function is to remove branches that are dominated by other branches. However, as there might be cycles of dominance, we have to be a little bit more careful. Consider directed graph $G_{\mathcal{B}} = (\mathcal{B}, E_{\mathcal{B}})$ such that $(B_i, B_j) \in E_{\mathcal{B}}$ if and only if B_i is dominated by B_j. Let $C_1, C_2, \ldots, C_c$ be the strongly connected components of $G_{\mathcal{B}}$. By $rep(C_i)$ we denote an arbitrary, but fixed, branch $B \in C_i$. A component C_i is called a *sink* component if there is no other component $C_j, i \neq j$ such that there exists an edge $(B_i, B_j) \in E_{\mathcal{B}}$ where $B_i \in C_i$ and $B_j \in C_j$. The *DominanceFree* function returns the branches $\mathcal{B}_{df} = \{rep(C_i) \mid i \in \{1, 2, \ldots, c\} \wedge C_i \text{ is a sink component}\}$.

6 Applying $(\mathcal{F}, \mathcal{A}, \beta)$-Algorithm to d-PVC

We are now going to show the specifics of applying the $(\mathcal{F}, \mathcal{A}, \beta)$-Algorithm to the d-Path Vertex Cover problem. It is easy to see that d-PVC equals $\mathcal{F}$-SVD for $\mathcal{F} = \{P_d\}$.

6.1 Handmade Rules

For the $(\mathcal{F}, \mathcal{A}, \beta)$-Algorithm to work for interesting values of β, we provide two handmade polynomial time reduction rules to $\mathcal{A}$ that are correct for d-PVC.

Reduction Rule 1 (Red component reduction for d-PVC.) Let (G, k) be an instance of d-PVC. Let $v \in V(G)$ be a vertex such that there are at least two P_d-free connected components C_1, C_2 in $G \backslash v$. If there is a P_d in $G[\{v\} \cup V(C_1) \cup V(C_2)]$, reduce (G, k) to instance $(G \backslash (\{v\} \cup V(C_1) \cup V(C_2)), k-1)$ which corresponds to taking v into a solution. Otherwise, let P_i^1 be the longest path in $G[\{v\} \cup V(C_1)]$ starting in v and let P_j^2 be the longest path in $G[\{v\} \cup V(C_2)]$ starting in v. Assume, without loss of generality, that $i \leq j$. Then, reduce the instance (G, k) to $(G \backslash V(C_1), k)$.

Reduction Rule 2 (Red star reduction for d-PVC, $d \geq 4$) Let (G, k) be the instance of d-PVC. Suppose there exists a subset $C \subseteq V(G)$, $|C| \leq \lfloor \frac{d}{2} \rfloor - 1$ for which there is a subset $L \subseteq V(G)$ such that $\forall v \in L, N(v) = C$ and $|L| \geq 2|C|$. Let $x \in L$. Then reduce instance (G, k) to instance $(G \backslash \{x\}, k)$.

We now discuss how to incorporate these reduction rules into the $(\mathcal{F}, \mathcal{A}, \beta)$-Algorithm. Note that as a part of $\mathcal{A}$, if the rule applies, we would make a call of *SolveRecursively*$(G \backslash (\{v\} \cup V(C_1) \cup V(C_2)), k-1)$, *SolveRecursively*$(G \backslash V(C_1), k)$, or *SolveRecursively*$(G \backslash \{x\}, k)$, respectively, and return the answer obtained. The following two lemmata describe the function $Handled_{\mathcal{A}}$.

Lemma 2. *For the case of the red component reduction rule, let H be a connected bumpy graph and $R \subseteq V(H)$ be its red vertices. If there is a vertex $v \in V(H)$ for which there are at least two d-path free connected components C_1, C_2 in $H \backslash v$ with $V(C_1), V(C_2) \subseteq R$ then the pair H, R is handled by the red component reduction rule, i.e., whenever any subgraph branching rule $(H, R, \mathcal{B})$ would apply to a graph G, the red component reduction rule would also apply to G.*

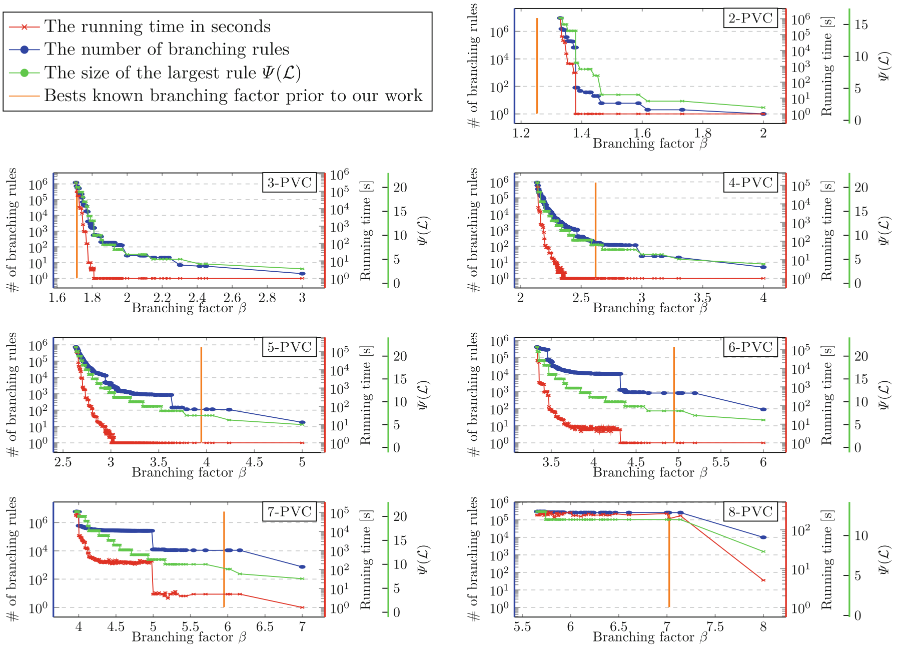
Lemma 3. *For the case of d-PVC, $d \geq 4$. Let H be a connected bumpy graph and $R \subseteq V(H)$ be its red vertices. If there is a subset $C \subseteq V(H), |C| \leq \lfloor \frac{d}{2} \rfloor - 1$ for which there is a subset $L \subseteq R$ such that $\forall v \in L, N(v) = C$ and $2|C| \leq |L|$, then the pair H, R is handled by the red star reduction rule.*

6.2 Obtained Results

With careful implementation the $(\mathcal{F}, \mathcal{A}, \beta)$-Algorithm together with our handmade reduction rules is able to achieve the results as summarized in Table 1. Note that $\mathcal{F}$ is fixed to $\{P_d\}$, $\mathcal{A}$ is as described in the previous subsection and the only parameter that varies is β. The question is then whether the algorithm finishes with the given β or not. The table contains, for each d, the least values of β for which our implementation of the algorithm finished. The full source code of the implementation is available at https://github.com/generating-algorithms/generating-dpvc. We also provide a separate repository https://github.com/generating-algorithms/generating-dpvc-data with annotated descriptions of the obtained algorithms. These are basically logs of the successful computation paths taken by the algorithm and are, to some extent, verifiable by hand. Sadly, we were not able to improve the running time of 2-PVC, but we do not know whether it is a limitation of the algorithm itself or a limitation of time, space, and resources.

To better understand the behavior of the generating algorithm, we provide plots of the number of branching rules and time it takes to achieve target branching factor. The runs depicted in the plots were performed on a virtual computer with 255 CPU cores and 128 GB of RAM.

The main point we would like to emphasize is that for the cases of d-PVC, $d \geq 4$, the first algorithms outperforming the state of the art were found in a matter of seconds and minutes.



7 Future Research Directions

We provided a framework to generate parameterized branching algorithms tailored for specific vertex deletion problems. In comparison, the framework of Gramm et al. [22] is also suited for problems where the task is to either delete or even add *edges* to the graph. We wonder whether some of our ideas can be translated to the edge setting.

While there are rather few studies on computer generated algorithms with provable worst-case running time upper bounds, there are quite some papers that use computer aided *analysis* of algorithms. In particular, the *Measure & Conquer* approach, introduced by Fomin et al. [18], is popular especially for moderately exponential algorithms [28,32,33,41]. Here the idea is to use simple rules, while measuring the progress not only based on the number of vertices resolved, but also on how favorably the remaining graph is structured, e.g., how many vertices of rather low degree are present. The hope is to capture that some unfavorable branching significantly improves the structure so that a favorable branching appears subsequently. To accomplish this, the analysis of a single rule is often split into many cases, based, e.g., on the degrees of the vertices involved. The computer is then used to optimize the values assigned to favorable structures so as to prove the lowest possible worst-case running time upper bound. Other approaches trying to amortize between the rules with bad branching factors and those with good branching factors include *branching potential* [21] or *labeled search trees* [9]. See also Fernau and Raible [16] for an older survey of the topic.

It may seem interesting to combine the automated generation framework with a computer assisted analysis of the algorithm. However, first, it seems that the computer assisted analysis still requires a non-trivial amount of human intervention, e.g., in design of the measure and cases to be distinguished by the computer. Therefore it seems to be limited to algorithms with few branching rules and does not scale to thousands of rules. Second, the favorable structure we gain, if it can be captured in an automated manner at all, is then exploited in the immediate neighborhood of the finished branching to gain the advantage. Hence, we might possibly as well create a single branching rule encompassing both the structures and "amortize within the rule". Of course, many variants of such a rule would be necessary. This is the approach already prevalent in our framework. However, the sizes of the rules necessary might be beyond the reach of our implementation. The question is whether some transfer of "branching potential" or some other kind of advantage can be explicitly included in the construction of the rules in order to enable this advanced analysis.

Finally, an obvious open question is whether there are, e.g., some handmade rules that would help our algorithm generate a faster algorithm for Vertex Cover (2-PVC). The fastest known algorithms of Chen, Kanj, and Xia [10] and Harris and Narayanaswamy [23] are rather complex to both analyze (both from the running time and correctness perspective) and implement. We made some experiments with the *struction* and *vertex-domination* rules from [10], but these did not seem to improve the performance of the generating algorithm.

References

1. Balasubramanian, R., Fellows, M.R., Raman, V.: An improved fixed-parameter algorithm for vertex cover. Inf. Process. Lett. **65**(3), 163–168 (1998). https://doi.org/10.1016/S0020-0190(97)00213-5
2. Brešar, B., Kardoš, F., Katrenič, J., Semanišin, G.: Minimum k-path vertex cover. Discret. Appl. Math. **159**(12), 1189–1195 (2011). https://doi.org/10.1016/j.dam.2011.04.008
3. Buss, J.F., Goldsmith, J.: Nondeterminism within P. SIAM J. Comput. 22(3), 560–572 (1993). https://doi.org/10.1137/0222038
4. Červený, R., Suchý, O.: Faster FPT algorithm for 5-path vertex cover. In: Rossmanith, P., Heggernes, P., Katoen, J. (eds.) 44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26–30, 2019, Aachen, Germany. LIPIcs, vol. 138, pp. 32:1–32:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019). https://doi.org/10.4230/LIPIcs.MFCS.2019.32
5. Červený, R., Suchý, O.: Generating faster algorithms for d-path vertex cover (2021). arxiv.org/abs/2111.05896
6. Chandran, L.S., Grandoni, F.: Refined memorization for vertex cover. Inf. Process. Lett. **93**(3), 123–131 (2005). https://doi.org/10.1016/j.ipl.2004.10.003
7. Chang, M., Chen, L., Hung, L., Liu, Y., Rossmanith, P., Sikdar, S.: Moderately exponential time algorithms for the maximum bounded-degree-1 set problem. Discret. Appl. Math. **251**, 114–125 (2018). https://doi.org/10.1016/j.dam.2018.05.032
8. Chen, J., Kanj, I.A., Jia, W.: Vertex cover: Further observations and further improvements. J. Algorithms **41**(2), 280–301 (2001). https://doi.org/10.1006/jagm.2001.1186
9. Chen, J., Kanj, I.A., Xia, G.: Labeled search trees and amortized analysis: improved upper bounds for NP-hard problems. Algorithmica **43**(4), 245–273 (2005). https://doi.org/10.1007/s00453-004-1145-7
10. Chen, J., Kanj, I.A., Xia, G.: Improved upper bounds for vertex cover. Theor. Comput. Sci. **411**(40–42), 3736–3756 (2010). https://doi.org/10.1016/j.tcs.2010.06.026
11. Chen, J., Liu, L., Jia, W.: Improvement on vertex cover for low-degree graphs. Networks **35**(4), 253–259 (2000)
12. Cygan, M., et al.: Parameterized Algorithms. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3
13. Downey, R.G., Fellows, M.R.: Fixed parameter tractability and completeness. In: Ambos-Spies, K., Homer, S., Schöning, U. (eds.) Complexity Theory: Current Research, Dagstuhl Workshop, February 2–8, 1992, pp. 191–225. Cambridge University Press (1992)
14. Fedin, S.S., Kulikov, A.S.: Automated proofs of upper bounds on the running time of splitting algorithms. In: Downey, R., Fellows, M., Dehne, F. (eds.) IWPEC 2004. LNCS, vol. 3162, pp. 248–259. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28639-4_22
15. Fernau, H.: Parameterized algorithmics for d-Hitting Set. Int. J. Comput. Math. **87**(14), 3157–3174 (2010). https://doi.org/10.1080/00207160903176868
16. Fernau, H., Raible, D.: Searching trees: an essay. In: Chen, J., Cooper, S.B. (eds.) TAMC 2009. LNCS, vol. 5532, pp. 59–70. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02017-9_9
17. Fomin, F.V., Gaspers, S., Kratsch, D., Liedloff, M., Saurabh, S.: Iterative compression and exact algorithms. Theor. Comput. Sci. **411**(7–9), 1045–1053 (2010). https://doi.org/10.1016/j.tcs.2009.11.012

18. Fomin, F.V., Grandoni, F., Kratsch, D.: A measure & conquer approach for the analysis of exact algorithms. J. ACM **56**(5), 25:1–25:32 (2009). https://doi.org/10.1145/1552285.1552286
19. Fomin, F.V., Kratsch, D.: Exact Exponential Algorithms. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16533-7
20. Funke, S., Nusser, A., Storandt, S.: On k-path covers and their applications. VLDB J. **25**(1), 103–123 (2016). https://doi.org/10.1007/s00778-015-0392-3
21. Gaspers, S.: Exponential Time Algorithms - Structures, Measures, and Bounds. VDM Verlag Dr. Mueller e.K. (2010). https://www.cse.unsw.edu.au/sergeg/SergeBookETA2010_print.pdf
22. Gramm, J., Guo, J., Hüffner, F., Niedermeier, R.: Automated generation of search tree algorithms for hard graph modification problems. Algorithmica **39**(4), 321–347 (2004). https://doi.org/10.1007/s00453-004-1090-5
23. Harris, D.G., Narayanaswamy, N.S.: A faster algorithm for vertex cover parameterized by solution size. CoRR abs/2205.08022 (2022), https://arxiv.org/abs/2205.08022
24. Katrenič, J.: A faster FPT algorithm for 3-path vertex cover. Inf. Process. Lett. **116**(4), 273–278 (2016). https://doi.org/10.1016/j.ipl.2015.12.002
25. Kojevnikov, A., Kulikov, A.S.: A new approach to proving upper bounds for MAX-2-SAT. In: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22–26, 2006, pp. 11–17. ACM Press (2006). http://dl.acm.org/citation.cfm?id=1109557.1109559
26. Kulikov, A.S.: Automated generation of simplification rules for SAT and MAXSAT. In: Bacchus, F., Walsh, T. (eds.) SAT 2005. LNCS, vol. 3569, pp. 430–436. Springer, Heidelberg (2005). https://doi.org/10.1007/11499107_35
27. Lewis, J.M., Yannakakis, M.: The node-deletion problem for hereditary properties is NP-complete. J. Comput. Syst. Sci. **20**(2), 219–230 (1980). https://doi.org/10.1016/0022-0000(80)90060-4
28. Lokshtanov, Daniel, Saurabh, Saket, Suchý, Ondřej: Solving MULTICUT faster than 2^n. In: Schulz, Andreas S.., Wagner, Dorothea (eds.) ESA 2014. LNCS, vol. 8737, pp. 666–676. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44777-2_55
29. Niedermeier, R., Rossmanith, P.: Upper bounds for vertex cover further improved. In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, pp. 561–570. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-49116-3_53
30. Niedermeier, R., Rossmanith, P.: An efficient fixed-parameter algorithm for 3-hitting set. J. Discrete Algorithms **1**(1), 89–102 (2003). https://doi.org/10.1016/S1570-8667(03)00009-1
31. Novotný, M.: Design and analysis of a generalized canvas protocol. In: Samarati, P., Tunstall, M., Posegga, J., Markantonakis, K., Sauveron, D. (eds.) WISTP 2010. LNCS, vol. 6033, pp. 106–121. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12368-9_8
32. van Rooij, J.M.M., Bodlaender, H.L.: Exact algorithms for dominating set. Discret. Appl. Math. **159**(17), 2147–2164 (2011). https://doi.org/10.1016/j.dam.2011.07.001
33. van Rooij, J.M.M., Bodlaender, H.L.: Exact algorithms for edge domination. Algorithmica **64**(4), 535–563 (2012). https://doi.org/10.1007/s00453-011-9546-x
34. Tsur, D.: Parameterized algorithm for 3-path vertex cover. Theor. Comput. Sci. **783**, 1–8 (2019). https://doi.org/10.1016/j.tcs.2019.03.013
35. Tsur, D.: An $O^*(2.619^k)$ algorithm for 4-path vertex cover. Discret. Appl. Math. **291**, 1–14 (2021). https://doi.org/10.1016/j.dam.2020.11.019

36. Tsur, D.: Faster parameterized algorithms for two vertex deletion problems. Theor. Comput. Sci. 940(Part), 112–123 (2023). https://doi.org/10.1016/j.tcs.2022.10.044
37. Tu, J.: A fixed-parameter algorithm for the vertex cover P_3 problem. Inf. Process. Lett. **115**(2), 96–99 (2015). https://doi.org/10.1016/j.ipl.2014.06.018
38. Tu, J., Jin, Z.: An FPT algorithm for the vertex cover P_4 problem. Discret. Appl. Math. **200**, 186–190 (2016). https://doi.org/10.1016/j.dam.2015.06.032
39. Xiao, M., Kou, S.: Exact algorithms for the maximum dissociation set and minimum 3-path vertex cover problems. Theor. Comput. Sci. **657**, 86–97 (2017). https://doi.org/10.1016/j.tcs.2016.04.043
40. Xiao, M., Kou, S.: Kernelization and parameterized algorithms for 3-path vertex cover. In: Proc. TAMC 2017, pp. 654–668 (2017). https://doi.org/10.1007/978-3-319-55911-7_47
41. Xiao, M., Nagamochi, H.: Exact algorithms for maximum independent set. Inf. Comput. **255**, 126–146 (2017). https://doi.org/10.1016/j.ic.2017.06.001

A New Width Parameter of Graphs Based on Edge Cuts: α-Edge-Crossing Width

Yeonsu Chang[1], O-joung Kwon[1,2](✉), and Myounghwan Lee[1]

[1] Department of Mathematics, Hanyang University, Seoul, South Korea
{yeonsu,ojoungkwon,sycuel}@hanyang.ac.kr
[2] Discrete Mathematics Group, Institute for Basic Science (IBS),
Daejeon, South Korea

Abstract. We introduce graph width parameters, called α-edge-crossing width and edge-crossing width. These are defined in terms of the number of edges crossing a bag of a tree-cut decomposition. They are motivated by edge-cut width, recently introduced by Brand et al. (WG 2022). We show that edge-crossing width is equivalent to the known parameter tree-partition-width. On the other hand, α-edge-crossing width is a new parameter; tree-cut width and α-edge-crossing width are incomparable, and they both lie between tree-partition-width and edge-cut width.

We provide an algorithm that, for a given n-vertex graph G and integers k and α, in time $2^{O((\alpha+k)\log(\alpha+k))}n^2$ either outputs a tree-cut decomposition certifying that the α-edge-crossing width of G is at most $2\alpha^2+5k$ or confirms that the α-edge-crossing width of G is more than k. As applications, for every fixed α, we obtain FPT algorithms for the List Coloring and Precoloring Extension problems parameterized by α-edge-crossing width. They were known to be W[1]-hard parameterized by tree-partition-width, and FPT parameterized by edge-cut width, and we close the complexity gap between these two parameters.

Keywords: α-edge-crossing width · List Coloring · FPT algorithm

1 Introduction

Tree-width is one of the basic parameters in structural and algorithmic graph theory, which measures how well a graph accommodates a decomposition into a tree-like structure. It has an important role in the graph minor theory developed by Robertson and Seymour [14–16]. For algorithmic aspects, there are various

A full version of the paper is available at https://arxiv.org/abs/2302.04624.

Y. Chang, O. Kwon, and M. Lee are supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science and ICT (No. NRF-2021K2A9A2A11101617 and RS-2023-00211670). O. Kwon is also supported by Institute for Basic Science (IBS-R029-C1).

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 172–186, 2023.
https://doi.org/10.1007/978-3-031-43380-1_13

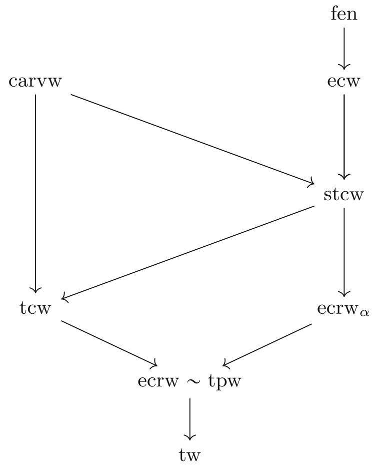


Fig. 1. The hierarchy of the mentioned width parameters. For two width parameters A and B, $A \rightarrow B$ means that every graph class of bounded A has bounded B, but there is a graph class of bounded B and unbounded A. Also, $A \sim B$ means that two parameters A and B are equivalent. fen, carvw, ecw, tcw, stcw, ecrw$_\alpha$, ecrw, tpw, and tw denote feedback edge set number, carving-width, edge-cut width, tree-cut width, slim tree-cut width, α-edge-crossing width, edge-crossing width, tree-partition-width, and tree-width, respectively.

fundamental problems that are NP-hard on general graphs, but fixed parameter tractable (FPT) parameterized by tree-width, that is, that can be solved in time $f(k)n^{O(1)}$ on n-vertex graphs of tree-width k for some computable function f. However, various problems are still W[1]-hard parameterized by tree-width. For example, List Coloring is W[1]-hard parameterized by tree-width [6].

Recently, edge counterparts of tree-width have been considered. One of such parameters is the *tree-cut width* of a graph introduced by Wollan [17]. Similar to the relationship between tree-width and graph minors, Wollan established a relationship between tree-cut width and weak immersions, and discussed structural properties. Since tree-cut width is a weaker parameter than tree-width, one could expect that some problems that are W[1]-hard parameterized by tree-width, are fixed parameter tractable parameterized by tree-cut width. But still several problems, including List Coloring, remain W[1]-hard parameterized by tree-cut width [3,7,8,10,11].

This motivates Brand et al. [2] to consider a more restricted parameter called the *edge-cut width* of a graph. For the edge-cut width of a graph G, a maximal spanning forest F of G is considered as a decomposition tree. For each vertex v of F, the *local feedback edge set* of v is the number of edges $e \in E(G) \backslash E(F)$ where the unique cycle of the graph obtained from F by adding e contains v, and the *edge-cut width* of F is the maximum local feedback edge set plus one over all vertices of G. The edge-cut width of G is the minimum edge-cut width among all maximal spanning forests of G. The edge-cut width with respect to a maximal spanning forest was also considered by Bodlaender [1] to bound the tree-width of certain graphs, with a different name called *vertex remember number*. Brand et al. showed that the tree-cut width of a graph is at most its edge-cut width. Furthermore, they showed that several problems including List Coloring are fixed parameter tractable parameterized by edge-cut width.

A natural question is to find a width parameter f such that graph classes of bounded f strictly generalize graph classes of bounded edge-cut width, and also List Coloring admits a fixed parameter tractable algorithm parameterized by f. This motivates us to define a new parameter called *α-edge-crossing width*. By relaxing the condition, we also define a parameter called *edge-crossing width*, but it turns out that this parameter is equivalent to tree-partition-width [5]. Recently, Ganian and Korchemna [9] introduced slim tree-cut width which also generalizes edge-cut width. See Fig. 1 for the hierarchy of new parameters and known parameters.

We define the *α-edge-crossing width* and *edge-crossing width* of a graph. For a graph G, a pair $\mathcal{T} = (T, \mathcal{X})$ of a tree T and a collection $\mathcal{X} = \{X_t \subseteq V(G) : t \in V(T)\}$ of disjoint sets of vertices in G, called bags (allowing empty bags), with the property $\bigcup_{t \in V(T)} X_t = V(G)$ is called the *tree-cut decomposition* of G. For a node $t \in V(T)$, let $T_1, T_2, \cdots, T_m$ be the connected components of $T - t$, and let $\text{cross}_{\mathcal{T}}(t)$ be the number of edges incident with two distinct sets in $\{\bigcup_{t \in V(T_i)} X_t : 1 \leqslant i \leqslant m\}$. We say that such an edge crosses X_t. The *crossing number* of $\mathcal{T}$ is $\max_{t \in V(T)} \text{cross}_{\mathcal{T}}(t)$, and the *thickness* of $\mathcal{T}$ is $\max_{t \in V(T)} |X_t|$. For a positive integer α, the *α-edge-crossing width* of a graph G, denoted by $\text{ecrw}_\alpha(G)$, is the minimum crossing number over all tree-cut decompositions of G whose thicknesses are at most α. The *edge-crossing width* of $\mathcal{T}$ is the maximum of the crossing number and the thickness of $\mathcal{T}$. The *edge-crossing width* of G, denoted by $\text{ecrw}(G)$, is the minimum *edge-crossing width* over all tree-cut decompositions of G.

It is not difficult to see that the 1-edge-crossing width of a graph is at most its edge-cut width minus one, as we can take the completion of its optimal maximal spanning forest for edge-cut width into a tree as a tree-cut decomposition with small crossing number.

We provide an FPT approximation algorithm for α-edge-crossing width. We adapt an idea for obtaining an FPT approximation algorithm for tree-cut width due to Kim et al. [12].

Theorem 1. *Given an n-vertex graph G and two positive integers α and k, one can in time $2^{\mathcal{O}((\alpha+k)\log(\alpha+k))}n^2$ either*

- *output a tree-cut decomposition of G with thickness at most α and crossing number at most $2\alpha^2 + 5k$, or*
- *correctly report that $\text{ecrw}_\alpha(G) > k$.*

As applications of α-edge-crossing width, we show that List Coloring and Precoloring Extension are FPT parameterized by α-edge-crossing width. They were known to be W[1]-hard parameterized by tree-cut width (and so by tree-partition-width) [7], and FPT parameterized by edge-cut width [2] and by slim tree-cut width [9]. We close the complexity gap between these parameters.

Theorem 2. *For a fixed positive integer α, the* List Coloring *and* Precoloring Extension *problems are FPT parameterized by α-edge-crossing width.*

We remark that the Edge-Disjoint Paths problem is one of the W[1]-hard problems parameterized by tree-width that motivates to study width parameters based on edge cuts. Ganian and Ordyniak [10] recently proved that Edge-Disjoint Paths is NP-hard on graphs admitting a vertex cover of size 3. This implies that for every $\alpha \geqslant 3$, this problem is NP-hard on graphs of α-edge-crossing width 0.

This paper is organized as follows. In Sect. 2, we give basic definitions and notations. In Sect. 3, we establish the relationship between width parameters as presented in Fig. 1. We present an FPT approximation algorithm for α-edge-crossing width in Sect. 4 and discuss algorithmic applications in Sect. 5. We conclude and present some open problems in Sect. 6.

Proofs of statements marked with "$\star$" are deferred to the full version.

2 Preliminaries

For a set X and a positive integer n, we call $\binom{X}{n}$ the set of all subsets of X of size exactly n. Let $\mathbb{N}$ be the set of all non-negative integers, and for a positive integer n, let $[n] = \{1, 2, \cdots, n\}$.

For a graph G, we denote by $V(G)$ and $E(G)$ the vertex set and the edge set of G, respectively. Let G be a graph. For a set S of vertices in G, let $G[S]$ denote the subgraph of G induced by S, and let $G - S$ denote the subgraph of G obtained by removing all the vertices in S. For $v \in V(G)$, let $G - v := G - \{v\}$. For an edge e of G, let $G - e$ denote the graph obtained from G by deleting e. The set of neighbors of a vertex v is denoted by $N_G(v)$, and the *degree* of v is the size of $N_G(v)$. For two disjoint sets S_1, S_2 of vertices in G, we denote by $\delta_G(S_1, S_2)$ the set of edges incident with both S_1 and S_2 in G.

An edge e of a connected graph G is a *cut edge* if $G - e$ is disconnected. A connected graph is *2-edge-connected* if it has no cut edges.

For two graphs G and H, we say that G is a *subdivision* of H if G can be obtained from H by subsequently subdividing edges.

A *tree-decomposition* of a graph G is a pair of a tree T and a family of sets $\{B_t\}_{t \in V(T)}$ of vertices in G such that (1) $V(G) = \bigcup_{t \in V(T)} B_t$, (2) for every edge uv of G, there exists a node t of T such that $u, v \in B_t$, and (3) for every vertex v of G, the set $\{t \in V(T) : v \in B_t\}$ induces a subtree of T. The *width* of a tree-decomposition is $\max_{t \in V(T)} |B_t| - 1$, and the *tree-width* of a graph, denoted by $\operatorname{tw}(G)$, is the maximum width over all its tree-decompositions.

A tree-decomposition $(T, \{B_t\}_{t \in V(T)})$ is called *rooted* if T is a rooted tree. A rooted tree-decomposition $(T, \{B_t\}_{t \in V(T)})$ with a root r is called *nice* if (1) for a non-root leaf t of T, $|B_t| = 1$, and (2) if a node t is not a non-root leaf of T, then it is one of the following;

- (Forget node) t has one child t' and $B_t = B_{t'} \backslash \{v\}$ for some $v \in B_{t'}$.
- (Introduce node) t has one child t' and $B_t = B_{t'} \cup \{v\}$ for some $v \in V(G) \backslash B_{t'}$.
- (Join node) t has exactly two children t_1 and t_2 and $B_t = B_{t_1} = B_{t_2}$.

Theorem 3 (Korhonen [13]). *There is an algorithm running in $2^{\mathcal{O}(w)}n$ time, that given an n-vertex graph G and an integer w, either outputs a tree-decomposition of G of width at most $2w + 1$ or reports that the tree-width of G is more than w.*

By applying the following lemma, we can find a nice tree-decomposition.

Lemma 1 (folklore; see Lemma 7.4 in [4]). *Given a tree-decomposition of an n-vertex graph G of width w, one can construct a nice tree-decomposition $(T, \mathcal{B})$ of width w with $|V(T)| = \mathcal{O}(wn)$ in $\mathcal{O}(w^2 \cdot \max(|V(T)|, n))$ time.*

3 Relationships Between Width Parameters

We compare width parameters as presented in Fig. 1 in the full version. Among the relations, we prove the following inequality. This will be mainly used in our approximation algorithm for α-edge-crossing width in Sect. 4.

Lemma 2. *For every graph G and every positive integer α, $\mathrm{tw}(G) \leqslant 5\,\mathrm{ecrw}(G) - 1$ and $\mathrm{tw}(G) \leqslant 3\,\mathrm{ecrw}_\alpha(G) + 2\alpha - 1$.*

Proof. We show the first inequality. Let $k = \mathrm{ecrw}(G)$. Let $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ be a tree-cut decomposition of G of edge-crossing width $\mathrm{ecrw}(G)$. We consider $\mathcal{T}$ as a rooted tree-cut decomposition with root node r. Let $\sigma : V(G) \to V(T)$ be the function where v is contained in $X_{\sigma(v)}$.

We construct a rooted tree-decomposition $(T, \{B_t\}_{t \in V(T)})$ as follows. For each node t of T, let F_t be the set of edges ab of G satisfying that either

- (type 1) the path between $\sigma(a)$ and $\sigma(b)$ in T contains t as an internal node, or
- (type 2) the path between $\sigma(a)$ and $\sigma(b)$ in T has length at least 1, and contains t as an end node, and the subtree of T rooted at t does not contain the end node of this path other than t.

Let B_t be the union of X_t and the set of vertices of G incident with an edge in F_t.

Since $\mathrm{cross}_{\mathcal{T}}(t) \leqslant k$, the number of the edges of type 1 is at most k. So, because of this type, we put at most $2k$ vertices into B_t. If t is the root node, then there is no edge of type 2. Assume that t is not the root node, and let t' be its parent. For type 2, $\sigma(a) = t$ or $\sigma(b) = t$, and either the vertex in $\{a, b\}$ that is not contained in X_t is contained in $B_{t'}$ or ab crosses $X_{t'}$. Since the number of edges ab of type 2 crossing $X_{t'}$ is at most k, we may add at most k vertices other than $X_t \cup X_{t'}$. As $|X_t \cup X_{t'}| \leqslant 2k$, in total, we have that $|B_t| \leqslant (2k) + (2k) + k = 5k$.

We now verify that $(T, \{B_t\}_{t \in V(T)})$ is a tree-decomposition. Since $X_t \subseteq B_t$ for each $t \in V(T)$, every vertex of G appears in some bag. Let $ab \in E(G)$ and assume that there is no bag of $\mathcal{T}$ containing both a and b. If the path between $\sigma(a)$ and $\sigma(b)$ in T contains some node t of T as an internal node, then by the construction, B_t contains both a and b. Assume that there is no node in T that is an internal node of the path between $\sigma(a)$ and $\sigma(b)$ in T. This means that $\sigma(a)$ is adjacent to $\sigma(b)$ in T. By symmetry, we assume that $\sigma(a)$ is the parent of $\sigma(b)$. Then ab is an edge of type 2 for the node $t = \sigma(b)$, and thus, $\{a, b\} \subseteq B_{\sigma(b)}$. Thus, $(T, \{B_t\}_{t \in V(T)})$ satisfies the second condition.

Lastly, to see that $(T, \{B_t\}_{t \in V(T)})$ satisfies the third condition, let $a \in V(G)$. For every vertex $b \in V(G)$ adjacent to a in G, let P_{ab} be the path between $\sigma(a)$ and $\sigma(b)$ in T. We add a to B_x for all $x \in V(P_{ab} - \{a, b\})$. This is the only procedure to add a to some B_x. Since $B_{\sigma(a)}$ contains a, the subtree of T induced by the union of all t where $a \in B_t$ is connected. This implies that $(T, \{B_t\})_{t \in V(T)}$ satisfies the third condition.

It is straightforward to verify the second inequality with the same argument.

We show that $\mathrm{stcw} \preccurlyeq \mathrm{ecrw}_\alpha$ and $\mathrm{ecrw}_\alpha \not\preccurlyeq \mathrm{stcw}$. We also show that $\mathrm{ecrw}_\alpha \not\preccurlyeq \mathrm{tcw}$ and $\mathrm{tcw} \not\preccurlyeq \mathrm{ecrw}_\alpha$. For positive integers k and n, let $S_{k,n}$ be the graph obtained from $K_{1,n}$ by replacing each edge with k internally vertex-disjoint paths of length 2.

Lemma 3 (Ganian and Korchemna [9]). *The set $\{S_{2,n} : n \in \mathbb{N}\}$ has unbounded slim tree-cut width.*

Lemma 4 ($\star$). *(1) $\{S_{3,n} : n \in \mathbb{N}\}$ has 1-edge-crossing width at most* 2.
(2) $\{S_{3,n} : n \in \mathbb{N}\}$ has unbounded tree-cut width.

Lemma 5. *For every positive integer α,* $\mathrm{ecrw}_\alpha \not\preccurlyeq \mathrm{stcw}$ *and* $\mathrm{ecrw}_\alpha \not\preccurlyeq \mathrm{tcw}$.

Proof. Note that $S_{2,n}$ is isomorphic to an induced subgraph of $S_{3,n}$. So, by (1) of Lemma 4, $S_{2,n}$ has 1-edge crossing width at most 2. On the other hand, Lemma 3 shows that $\{S_{2,n} : n \in \mathbb{N}\}$ has unbounded slim tree-cut width. This shows that $\mathrm{ecrw}_1 \not\preccurlyeq \mathrm{stcw}$. Since $\mathrm{ecrw}_1 \preccurlyeq \mathrm{ecrw}_\alpha$, we have $\mathrm{ecrw}_\alpha \not\preccurlyeq \mathrm{stcw}$.

By (1) and (2) of Lemma 4, $\{S_{3,n} : n \in \mathbb{N}\}$ has 1-edge crossing width at most 2, but unbounded tree-cut width. Therefore, $\mathrm{ecrw}_1 \not\preccurlyeq \mathrm{tcw}$. Since $\mathrm{ecrw}_1 \preccurlyeq \mathrm{ecrw}_\alpha$, we have $\mathrm{ecrw}_\alpha \not\preccurlyeq \mathrm{tcw}$.

We now show that $\mathrm{tcw} \not\preccurlyeq \mathrm{ecrw}_\alpha$. For all positive integers k and n, we construct a graph G_k^n as follows. Let $A := \{a_i : i \in [n]\}$, $B = \binom{A}{2}$, and $B_k = \{(W, \ell) : W \in B \text{ and } \ell \in [k]\}$. Let G_k^n be the graph such that $V(G_k^n) = A \cup B_k$, and for $a \in A$ and $(W, \ell) \in B_k$, a is adjacent to (W, ℓ) in $E(G_k^n)$ if and only if $a \in W$.

Lemma 6. *For every positive integer α,* $\mathrm{tcw} \not\preccurlyeq \mathrm{ecrw}_\alpha$.

Proof. Observe that $\{G_k^{\alpha+1} : k \in \mathbb{N}\}$ has unbounded α-edge-crossing width.

We claim that for every k, $G_k^{\alpha+1}$ has tree-cut width at most $\alpha+1$. Let (A, B_k) be the bipartition of $G_k^{\alpha+1}$ given by the definition. Let T be a star with center t

and leaves $t_1, \dots, t_{k\binom{\alpha+1}{2}}$. Let $X_t = A$ and each X_{t_i} consists of a vertex of B_k. Let $\mathcal{T} = (T, \{X_v\}_{v \in V(T)})$. Note that the 3-center of H_t has only vertices of A. Thus, it has at most $\alpha + 1$ vertices. Also, for every edge e of T, $\text{adh}_{\mathcal{T}}(e) \leqslant 2$. So, $\mathcal{T}$ is a tree-cut decomposition of tree-cut width at most $\alpha + 1$.

To show $\text{stcw} \leqslant \text{ecrw}_\alpha$, we use another equivalent parameter called *super edge-cut width* introduced by Ganian and Korchemna [9]. The *super edge-cut width* $\text{sec}(G)$ of a graph G is defined as the minimum edge-cut width of (H, T) over all supergraphs H of G and maximal spanning forests T of H.

Lemma 7 ($\star$). *For every positive integer α,* $\text{sec} \leqslant \text{ecrw}_\alpha$.

4 An FPT Approximation Algorithm for α-Edge-Crossing Width

We present an FPT approximation algorithm for α-edge-crossing width. We similarly follow the strategy to obtain a 2-approximation algorithm for tree-cut width designed by Kim et al. [12]. We formulate a new problem called Constrained Star-Cut Decomposition, which corresponds to decomposing a large leaf bag in a tree-cut decomposition, and we want to apply this subalgorithm recursively. By Lemma 2, we can assume that a given graph admits a tree-decomposition of bounded width, and we design a dynamic programming to solve Constrained Star-Cut Decomposition on graphs of bounded tree-width.

For a weight function $\gamma : V(G) \to \mathbb{N}$ and a non-empty vertex subset $S \subseteq V(G)$, we define $\gamma(S) := \sum_{v \in S} \gamma(v)$ and $\gamma(\varnothing) := 0$.

Constrained Star-Cut Decomposition
Input : A graph G, two positive integers α, k, and a weight function $\gamma : V(G) \to \mathbb{N}$
Question : Determine whether there is a tree-cut decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ of G such that

- T is a star with center t_c and it has at least one leaf,
- $|X_{t_c}| \leqslant \alpha$ and $\text{cross}_{\mathcal{T}}(t_c) \leqslant k$,
- for each leaf t of T, $\gamma(X_t) \leqslant \alpha^2 + 2k$ and $|\delta_G(X_t, X_{t_c})| \leqslant \alpha^2 + k$, and
- there is no leaf q of T such that $X_q = V(G)$.

The following lemma explains how we will adapt an algorithm for Constraint Star-Cut Decomposition.

Lemma 8. *Let G be a graph, let α, k be positive integers, and let S be a set of vertices in G. Assume that $|S| \geqslant \alpha + 1$ and $|\delta_G(S, V(G) \backslash S)| \leqslant 2\alpha^2 + 4k$. For each vertex $v \in S$, let $\gamma_S(v) = |\delta_G(\{v\}, V(G) \backslash S)|$.*

If $\text{ecrw}_\alpha(G) \leqslant k$, then $(G[S], \alpha, k, \gamma_S)$ is a Yes-instance of Constraint Star-Cut Decomposition.

Proof. Let $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ be a tree-cut decomposition of G of thicknesses at most α and crossing number at most k. For an edge $e = uv$ of T, let $T_{e,u}$ and $T_{e,v}$ be two subtrees of $T - uv$ which contain u and v, respectively.

We want to identify a node t_c of T that will correspond to the central node of the resulting star decomposition. First, we define an extension γ on $V(G)$ of the weight function γ_S on S as $\gamma(v) = \gamma_S(v)$ if $v \in S$ and $\gamma(v) = 0$, otherwise. We orient an edge $e = xy \in E(T)$ from x to y if the edge e satisfies at least one of the following rules; (Rule 1) $S \cap (\bigcup_{t \in V(T_{e,x})} X_t) = \varnothing$ and (Rule 2) $\gamma(\bigcup_{t \in V(T_{e,y})} X_t) > \alpha^2 + 2k$. Note that an edge may have no direction.

Claim ($\star$). Every edge has at most one direction.

So, T has a node whose incident edges have no direction or a direction to the node. Take such a node as a central node t_c. Let $T_1, \ldots, T_m$ be the connected components of $T - t_c$ such that for every $i \in [m]$, $\bigcup_{t \in V(T_i)} X_t \cap S \neq \emptyset$. Note that there is at least one such component, because $|S| > \alpha$ and $|X_{t_c}| \leqslant \alpha$.

Let $(T', \{X'_t\}_{t \in V(T')})$ be a tree-cut decomposition of $G[S]$ such that (1) T' is a star with the central node t_c and leaves $t_1, \ldots, t_m$, (2) $X'_{t_c} = X_{t_c} \cap S$, and (3) for every $i \in [m]$, $X'_{t_i} = \left(\bigcup_{t \in V(T_i)} X_t\right) \cap S$. We claim that $(T', \{X'_t\}_{t \in V(T')})$ satisfies the conditions of answer of Constrained Star-Cut Decomposition. By the construction of decomposition, the first condition holds. Since $X'_{t_c} = X_{t_c} \cap S$ and the crossing number of t_c in $(T, \{X_t\}_{t \in V(T)})$ is at most k, the second condition also holds.

Let t_i be a leaf of T' and let $V_i = \left(\bigcup_{t \in V(T_i)} X_t\right)$. By Rule 2 of the orientation, we have that $\gamma(X'_{t_i}) \leqslant \alpha^2 + 2k$. Let t' be the node in T_i that is adjacent to t_c in T. Since $|X_{t_c}| \leqslant \alpha$ and $|X_{t'}| \leqslant \alpha$, we have $|\delta_G(X_{t_c}, X_{t'})| \leqslant \alpha^2$. Furthermore, because $\text{cross}_{\mathcal{T}}(t') \leqslant k$, we have $|\delta_G(X_{t_c}, V_i \backslash X_{t'})| \leqslant k$. Thus, we have $|\delta_{G[S]}(X'_{t_c}, X'_{t_i})| \leqslant |\delta_G(X_{t_c}, V_i)| \leqslant \alpha^2 + k$.

Lastly, we claim that there is no leaf q of T' such that $X'_q = S$. Suppose that there is a leaf q of T' such that $X'_q = S$. This means that there is no other leaf in T' and $X'_{t_c} = \emptyset$. Let T^* be the connected component of $T - t_c$ for which $\bigcup_{t \in V(T^*)} X_t \cap S = X'_q$. Then for other connected component T^{**} of $T - t_c$, $\bigcup_{t \in V(T^{**})} X_t \cap S = \emptyset$, and therefore the edge of T between T^{**} and the node t_c is oriented towards t_c. Then the edge of T between T^* and t_c should be oriented towards T^*, because $X_{t_c} \cap S = \emptyset$. This contradicts the choice of t_c.

This proves the lemma.

We now devise an algorithm for Constraint Star-Cut Decomposition on graphs of bounded tree-width.

Lemma 9. *Let (G, α, k, γ) be an instance of* Constrained Star-Cut Decomposition *and let $(T, \{B_t\}_{t \in V(T)})$ be a nice tree-decomposition of width at most w. In $2^{\mathcal{O}((k+w)\log(w(\alpha+k)))}|V(T)|$ time, one can either output a solution of (G, α, k, γ), or correctly report that (G, α, k, γ) is a No-instance.*

Proof. Let r be the root of T. We design a dynamic programming to compute a solution of Constrained Star-Cut Decomposition. For each node t of T, let A_t be the union of all bags $B_{t'}$ where t' is a descendant of t in T.

Let $Z \subseteq V(G)$ be a set. A pair $(\mathcal{X}, \mathcal{P})$ of a sequence $\mathcal{X} = (X_0, X_1, \ldots, X_{2k}, Y)$ and a partition $\mathcal{P}$ of Y is *legitimate* with respect to Z if

- $X_0, X_1, \ldots, X_{2k}, Y$ are pairwise disjoint subsets of Z that are possibly empty,
- $(\bigcup_{i \in \{0,1,\ldots,2k\}} X_i) \cup Y = Z$,
- $|X_0| \leqslant \alpha$,
- for each $i \in [2k]$, $|\delta_G(X_i, X_0)| \leqslant \alpha^2 + k$ and $\gamma(X_i) \leqslant \alpha^2 + 2k$,
- $\sum_{\{i,j\} \in \binom{[2k]}{2}} |\delta_G(X_i, X_j)| \leqslant k$,
- for each $i \in [2k]$ and each $P \in \mathcal{P}$, $|\delta_G(X_0, P)| \leqslant \alpha^2 + k$, and $|\delta_G(X_i, P)| = 0$,
- for any distinct sets $P_i, P_j \in \mathcal{P}$, $|\delta_G(P_i, P_j)| = 0$.

Claim $(\star)$. (G, α, k, γ) is a Yes-instance if and only if there is a legitimate pair $(\mathcal{X}, \mathcal{P})$ with respect to $V(G)$ with $\mathcal{X} = (X_0, X_1, \ldots, X_{2k}, Y)$ such that any set of $X_1, \ldots, X_{2k}$ or a set of $\mathcal{P}$ is not the whole set $V(G)$.

For a legitimate pair $(\mathcal{X}, \mathcal{P})$ with respect to Z and $Z' \subseteq Z$, let $\mathcal{X}|_{Z'} = (X_0 \cap Z', X_1 \cap Z', \ldots, X_{2k} \cap Z', Y \cap Z')$ and $\mathcal{P}|_{Z'} = \{P \cap Z' : P \in \mathcal{P}, P \cap Z' \neq \emptyset\}$. We can see that $(\mathcal{X}|_{Z'}, \mathcal{P}|_{Z'})$ is legitimate with respect to Z', because the constraints are the number of vertices in a set, the number of edges between two sets, and the sum of γ-values. Based on this fact, we will recursively store information about all legitimate pairs with respect to A_t for nodes t.

Let $t \in V(T)$. A tuple $(I, \mathcal{Q}, C_1, C_2, D_1, D_2, a, b)$ is a *valid tuple* at t if

- $I : B_t \to \{0, 1, \ldots, 2k, 2k+1\}$,
- $\mathcal{Q}$ is a partition of $I^{-1}(2k+1)$,
- $C_1 : [2k] \to \{0, 1 \ldots, \alpha^2 + 2k\}$,
- $C_2 : \mathcal{Q} \to \{0, 1 \ldots, \alpha^2 + 2k\}$,
- $D_1 : [2k] \to \{0, 1 \ldots, \alpha^2 + k\}$,
- $D_2 : \mathcal{Q} \to \{0, 1 \ldots, \alpha^2 + k\}$, and
- a, b are two integers with $0 \leqslant a \leqslant \alpha$ and $0 \leqslant b \leqslant k$.

A valid tuple $(I, \mathcal{Q}, C_1, C_2, D_1, D_2, a, b)$ at a node t represents a legitimate pair $(\mathcal{X}, \mathcal{P}) = ((X_0, X_1, \ldots, X_{2k}, Y), \mathcal{P})$ with respect to A_t if

- for every $v \in B_t$, $I(v) = i$ if and only if $v \in \begin{cases} X_i & \text{if } 0 \leqslant i \leqslant 2k \\ Y & \text{if } i = 2k+1 \end{cases}$
- $\mathcal{Q} = \mathcal{P}|_{B_t}$,
- for each $i \in [2k]$, $C_1(i) = \gamma(X_i)$,
- for each $Q \in \mathcal{Q}$, $C_2(Q) = \gamma(Q)$,
- for each $i \in [2k]$, $D_1(i) = |\delta_G(X_i \cap A_t, X_0 \cap A_t)|$,
- for each $Q \in \mathcal{Q}$, $D_2(Q) = |\delta_G(Q \cap A_t, X_0 \cap A_t)|$,
- $a = |X_0 \cap A_t|$,
- $b = \sum_{\{i,j\} \subseteq \binom{[2k]}{2}} |\delta_G(X_i, X_j)|$.

We say that a valid tuple is a *record* at t if it represents some legitimate pair with respect to A_t. Let $\mathcal{R}(t)$ be the set of all records at t. It is not difficult to verify that there is a legitimate pair with respect to A_t if and only if there is a record at t. So, $\mathcal{R}(r) \neq \varnothing$ if and only if (G, α, k, γ) is a Yes-instance.

It is known that there is a constant d such that the number of partitions of a set of m elements is at most dm^m. We define a function

$$\zeta(x) = (2k+2)^x(dx^x)(\alpha^2+2k+1)^{2(x-1)+4k}(\alpha+1)(k+1).$$

Observe that if B_t has size q, then the number of all possible valid tuples at t is at most $\zeta(q)$. Note that $\zeta(q) = 2^{\mathcal{O}((q+k)\log(q(\alpha+k)))}$. We describe how to store all records in $\mathcal{R}(t)$ for each node t of T. Due to the space constraint, we only deal with an introduction node.

(Case. t is an introduce node with child t' such that $B_t = B_{t'} \cup \{v\}$.)
We construct a set $\mathcal{R}^*$ from $\mathcal{R}(t')$ as follows. Let

$$\mathcal{J}' = (I', \mathcal{Q}', C_1', C_2', D_1', D_2', a', b') \in \mathcal{R}(t').$$

For every $i \in \{0, 1, \ldots, 2k, 2k+1\}$ and $Q^* \in \mathcal{Q}' \cup \{\varnothing\}$ when $i = 2k+1$, we construct a new tuple $\mathcal{J} = (I, \mathcal{Q}, C_1, C_2, D_1, D_2, a, b)$ as follows:

- $I(w) = \begin{cases} I'(w) \text{ if } w \in B_{t'} \\ i \quad\;\;\; \text{if } w = v \end{cases}$,
- $\mathcal{Q} = \begin{cases} \mathcal{Q}' & \text{if } 0 \leqslant I(v) \leqslant 2k \\ (\mathcal{Q}' \backslash \{Q^*\}) \cup \{Q^* \cup \{v\}\} & \text{if } I(v) = 2k+1 \end{cases}$,
- for every $j \in [2k]$, $C_1(j) = C_1'(j) + \gamma(I^{-1}(j))$,
- for every $Q \in \mathcal{Q}$, $C_2(Q) = \begin{cases} C_2'(Q) & \text{if } Q \neq Q^* \cup \{v\} \\ C_2'(Q^*) + \gamma(v) & \text{if } Q = Q^* \cup \{v\} \end{cases}$,
- for every $j \in [2k]$, $D_1(j) = D_1'(j) + |\delta_G(I^{-1}(j_2), X_0 \cap B_{t'})|$,
- for every $Q \in \mathcal{Q}$, $D_2(Q) = \begin{cases} D_2'(Q) & \text{if } Q \neq Q^* \cup \{v\} \\ D_2'(Q^*) + |\delta_G(\{v\}, X_0 \cap B_{t'})| & \text{if } Q = Q^* \cup \{v\} \end{cases}$,
- $a = \begin{cases} a' + 1 & \text{if } I(v) = 0 \\ a' & \text{otherwise} \end{cases}$,
- $b = \begin{cases} b' + |\delta_G\big(\{v\}, I^{-1}([2k] \backslash \{I(v)\})\big)| & \text{if } 1 \leqslant I(v) \leqslant 2k \\ b' & \text{otherwise} \end{cases}$.

We add this tuple to $\mathcal{R}^*$ whenever it is valid and

- if $1 \leqslant I(v) \leqslant 2k$, then there is no edge between v and $I^{-1}(2k+1)$ in G,
- if $I(v) = 2k+1$, then there is no edge between v and $I^{-1}(\{1, \ldots, 2k\})$ in G and there is no edge between v and $I^{-1}(2k+1) \backslash (Q^* \cup \{v\})$ in G.

We claim that $\mathcal{R}^* = \mathcal{R}(t)$. First we show that $\mathcal{R}^* \subseteq \mathcal{R}(t)$. Let $\mathcal{J}$ be a valid tuple constructed as above from $\mathcal{J}' \in \mathcal{R}(t')$. We have to show that $\mathcal{J}$ represents some legitimate pair with respect to A_t. Since $\mathcal{J}' \in \mathcal{R}(t')$, it represents a legitimate pair $(\mathcal{X}' = (X_0', \ldots, X_{2k}', Y'), \mathcal{P}')$ with respect to $A_{t'}$.

If $0 \leqslant i \leqslant 2k$, then we obtain $\mathcal{X}$ from $\mathcal{X}'$ by replacing X_i' with $X_i' \cup \{v\}$ and set $\mathcal{P} = \mathcal{P}'$. If $i = 2k+1$, then we obtain $\mathcal{X}$ from $\mathcal{X}'$ by replacing Y' with

$Y' \cup \{v\}$ and adding v to the part $P^* \in \mathcal{P}'$ with $P^* \cap B_t = Q^*$ when $Q^* \in \mathcal{Q}'$ or adding a single part $\{v\}$ when $Q^* = \emptyset$, to obtain a new partition $\mathcal{P}$ of $Y' \cup \{v\}$. Then it is straightforward to verify that $(\mathcal{X}, \mathcal{P})$ is a legitimate pair with respect to A_t and $\mathcal{J}$ represents it. This shows that $\mathcal{J} \in \mathcal{R}(t)$.

To show $\mathcal{R}(t) \subseteq \mathcal{R}^*$, suppose $\mathcal{J} = (I, \mathcal{Q}, C_1, C_2, D_1, D_2, a, b) \in \mathcal{R}(t)$. Then there is a legitimate pair $(\mathcal{X}, \mathcal{P})$ represented by $\mathcal{J}$. Since a pair $(\mathcal{X}|_{B_{t'}}, \mathcal{P}|_{B_{t'}})$ is legitimate, there is a record $\mathcal{J}' \in \mathcal{R}(t')$ which represents the pair. By the construction, $\mathcal{J}$ is computed from $\mathcal{J}'$. Hence $\mathcal{R}(t) \subseteq \mathcal{R}^*$.

We take one record in $\mathcal{R}(t')$ and construct a new tuple as explained above. After then, we check its validity. Note that $|\mathcal{R}(t')| \leqslant \zeta(w)$. Checking the validity takes time $\mathcal{O}(w+k)$. Hence, $\mathcal{R}(t)$ is computed in $2^{\mathcal{O}((w+k)\log(w(\alpha+k)))}$ time.

For other nodes t, we can also compute $\mathcal{R}(t)$ in time $2^{\mathcal{O}((w+k)\log(w(\alpha+k)))}$. Overall, the algorithm runs in $2^{\mathcal{O}((w+k)\log(w(\alpha+k)))}|V(T)|$ time.

Theorem 4. *Given an n-vertex graph G and two positive integers α and k, one can in time $2^{\mathcal{O}((\alpha+k)\log(\alpha+k))}n^2$ either*

- *output a tree-cut decomposition of G with thickness at most α and crossing number at most $2\alpha^2 + 5k$, or*
- *correctly report that $\mathrm{ecrw}_\alpha(G) > k$.*

Proof. We recursively apply the algorithm for Constrained Star-Cut Decomposition as follows. At the beginning, we consider a trivial tree-decomposition with one bag containing all the vertices. In the recursive steps, we assume that we have a tree-cut decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ such that

(i) for every internal node t of T, $|X_t| \leqslant \alpha$ and $\mathrm{cross}_{\mathcal{T}}(t) \leqslant 2\alpha^2 + 5k$,
(ii) for every leaf node t of T, $|\delta_G(X_t, V(G)\backslash X_t)| \leqslant 2\alpha^2 + 4k$.

If all leaf bags have size at most α, then this decomposition has thickness at most α and crossing number at most $2\alpha^2 + 5k$. Thus, we may assume that there is a leaf bag X_ℓ having at least $\alpha + 1$ vertices.

We apply Theorem 3 for $G[X_\ell]$ with $w = 3k + 2\alpha - 1$. Then in time $2^{\mathcal{O}(k+\alpha)}n$, either we have a tree-decomposition of width at most $2(3k + 2\alpha - 1) + 1 = 6k + 4\alpha - 1$ or we report that $\mathrm{tw}(G) \geqslant \mathrm{tw}(G[X_\ell]) > 3k + 2\alpha - 1$. In the latter case, by Lemma 2, we have $\mathrm{ecrw}_\alpha(G) > k$. Thus, we may assume that we have a tree-decomposition of $G[X_\ell]$ of width at most $6k + 4\alpha - 1$. By applying Lemma 1, we can find a nice tree-decomposition $(F, \{B_t\}_{t \in V(F)})$ of $G[X_\ell]$ of width at most $6k + 4\alpha - 1$ with $|V(F)| = \mathcal{O}((k+\alpha)n)$.

We define γ on X_ℓ so that $\gamma(v) = |\delta_G(\{v\}, V(G)\backslash X_\ell)|$. We run the algorithm in Lemma 9 for the instance $(G[X_\ell], \alpha, k, \gamma)$. Then in time $2^{\mathcal{O}((\alpha+k)\log(\alpha+k))}|V(F)|$, one can either output a solution of $(G[X_\ell], \alpha, k, \gamma)$, or correctly report that $(G[X_\ell], \alpha, k, \gamma)$ is a No-instance. In the latter case, we have $\mathrm{ecrw}_\alpha(G) > k$, by Lemma 8. In the former case, let $\mathcal{T}^* = (T^*, \{Y_t\}_{t \in V(T^*)})$ be the outcome, where q_c is the center of T^* and $q_1, \ldots, q_m$ are the leaves of T^*. Then we modify the tree-cut decomposition $\mathcal{T}$ by replacing X_ℓ with Y_{q_c} and then attaching bags Y_{q_i} to Y_{q_c}, where corresponding nodes are q_c and $q_1, \dots, q_m$. Let $\mathcal{T}'$ be the resulting tree-cut decomposition.

Observe that $|Y_{q_c}| \leqslant \alpha$ and $\mathrm{cross}_{\mathcal{T}^*}(q_c) \leqslant k$. So, we have $\mathrm{cross}_{\mathcal{T}'}(q_c) \leqslant k + (2\alpha^2 + 4k) = 2\alpha^2 + 5k$. Also, for each $i \in [m]$, we have

$$\begin{aligned}|\delta_G(Y_{q_i}, V(G) \backslash Y_{q_i})| &\leqslant |\delta_G(Y_{q_i}, V(G) \backslash X_\ell)| + |\delta_G(Y_{q_i}, Y_{q_c})| + \mathrm{cross}_{\mathcal{T}^*}(q_c)\\ &\leqslant (\alpha^2 + 2k) + (\alpha^2 + k) + k = 2\alpha^2 + 4k.\end{aligned}$$

Therefore, we obtain a refined tree-cut decomposition with properties (i) and (ii). Note that by the last condition of the solution for Constraint Star-Cut Decomposition, new leaf bags have size less than X_ℓ. Thus, the algorithm will terminate in at most n recursive steps. When this procedure terminates, we either obtain a tree-cut decomposition of G of thickness at most α and crossing number at most $2\alpha^2 + 5k$ or, conclude that $\mathrm{ecrw}_\alpha(G) > k$.

The total running time is $(2^{\mathcal{O}((\alpha+k)\log(\alpha+k))}n) \cdot n = 2^{\mathcal{O}((\alpha+k)\log(\alpha+k))}n^2$.

5 Algorithmic Applications on Coloring Problems

We show that List Coloring and Precoloring Extension are fixed parameter tractable parameterized by α-edge-crossing width for every fixed α.

A vertex-coloring $f : V(G) \to \mathbb{N}$ on a graph G is said to be *proper* if $f(u) \neq f(v)$ for all edges $uv \in E(G)$. For a given set $\{L(v) \subseteq \mathbb{N} : v \in V(G)\}$, a coloring $c : V(G) \to \mathbb{N}$ is called an *L-coloring* if $c(v) \in L(v)$ for all $v \in V(G)$.

List Coloring
Input : A graph G and a set of lists $\mathcal{L} = \{L(v) \subseteq \mathbb{N} : v \in V(G)\}$
Question : Does G admit a proper L-coloring $c : V(G) \to \bigcup \mathcal{L}$?

Assume that a tree-cut decomposition of the input graph G of thickness at most α and crossing number w is given. In the dynamic programming, we need to store colorings on $w + \alpha$ boundaried vertices. Using Lemma 10 the number of colorings to store can be reduced to $g(w + \alpha)$ for some function g. For $V \in \mathbb{N}^q$, $W \in \mathbb{N}^t$, $B \subseteq [q] \times [t]$, we say that (V, W) is *B-compatible* if $V[i] \neq W[j]$ for all $(i, j) \in B$. When we have a vertex partition (X, Y) of a graph G, possible colorings on boundaried vertices in X and Y will be related to vectors V and W.

Lemma 10 ($\star$). *Let q and t be positive integers, and let $B \subseteq [q] \times [t]$. For every set $\mathcal{P}$ of distinct vectors in $\mathbb{N}^q$, there is a subset $\mathcal{P}^*$ of $\mathcal{P}$ of size at most $q!2^{\frac{q(q+1)}{2}}t^{q-1}(t + 1)$ satisfying that for every $W \in \mathbb{N}^t$, if there is $V \in \mathcal{P}$ where (V, W) is B-compatible, then there is $V^* \in \mathcal{P}^*$ where (V^*, W) is B-compatible. Furthermore, such a set $\mathcal{P}^*$ can be computed in time $\mathcal{O}(|\mathcal{P}|2^{q^2}t^{q+2})$.*

Let G be a graph. For disjoint sets S, T of vertices in G and functions $g : S \to \mathbb{N}$ and $h : T \to \mathbb{N}$, we say that (S, g) is *compatible* with (T, h) if for every edge vw with $v \in S$ and $w \in T$, $g(v) \neq h(w)$. If g and h are clear from the context, we simply say that S and T are compatible.

Proof. (of Theorem 2). We describe the algorithm for connected graphs. If a given graph is disconnected, then we can apply the algorithm for each component. We assume that G is connected. We may assume that each list $L(v)$ has size at most the degree of v; otherwise, we can freely color v after coloring $G-v$.

Let $\mathrm{ecrw}_\alpha(G) = k$. Using the algorithm in Theorem 4, we obtain a tree-cut decomposition $\mathcal{T} = (T, \mathcal{X} = \{X_t\}_{t \in V(T)})$ of the input graph G of thickness at most α and crossing number $w \leqslant 2\alpha^2 + 5k$. We consider it as a rooted decomposition by choosing a root node r with $X_r \neq \emptyset$.

For every node $t \in V(T)$, we denote by T_t the subtree of T rooted at t, and let $G_t = G[\bigcup_{v \in V(T_t)} X_v]$. For every $t \in V(T)$, let $\partial(t)$ be the graph H where $E(H)$ is the set of edges incident with both $V(G_t)$ and $V(G) \backslash V(G_t)$, and $V(H)$ is the set of vertices in G incident with an edge in $E(H)$. Let $\hat{\partial}(t) := (V(\partial(t)) \cap V(G_t)) \cup X_t$. Note that $|\hat{\partial}(t)| \leqslant \alpha + w$ for any node t of T.

Let $t \in V(T)$. A coloring g on $\hat{\partial}(t)$ is *valid at* t if there is a proper L-coloring f of G_t for which $f|_{\hat{\partial}(t)} = g$. Clearly, the problem is a Yes-instance if and only if there is a valid coloring at the root node.

Let $\zeta = \max\{(\alpha + w + 1)! 2^{\frac{(\alpha+w)(\alpha+w+1)}{2}} (\alpha + w)^{\alpha+w-1}, (w + 2\alpha - 1)^\alpha\}$.

For each node $t \in V(T)$, let $Q[t]$ be the set of all valid colorings at t. We will recursively construct a subset $Q^*[t] \subseteq Q[t]$ of size at most ζ such that

> ($\Box$) for every proper L-coloring h on $G - V(G_t)$, if there is a valid coloring $g \in Q[t]$ compatible with h, then there is $g^* \in Q^*[t]$ compatible with h.

We describe how to construct $Q^*[t]$ depending on whether t is a non-root leaf. This is easy when t is a leaf. Let t_p be the parent of t when t is not the root.

We classify the children of t into two types. Let A_1 be the set of all children p of t such that $\partial(p) \backslash \hat{\partial}(p) \subseteq X_t$, and let A_2 be the set of all other children of t. Note that $|A_2| \leqslant 2w$ because $(T, \mathcal{X})$ has crossing number at most w. Let $C[t]$ be the set of all proper L-colorings on X_t. Clearly, $|C[t]| \leqslant n^\alpha$.

(Step 1.) We first find the set $C'[t]$ of all proper L-colorings f such that for each $x \in A_1$, there exists $g_x \in Q^*[x]$ that is compatible with f. This can be checked by recursively choosing $x \in A_1$, and comparing each coloring in $C[t]$ with a coloring in $Q^*[x]$, and then remaining one that has a compatible coloring in $Q^*[x]$. The whole procedure runs in time $\mathcal{O}(|A_1| \cdot |Q^*[x]| \cdot n^\alpha \cdot (\alpha + w)^2) = \mathcal{O}(\zeta \cdot n^{\alpha+1} \cdot (\alpha + w)^2)$, because each $Q^*[x]$ has the size at most ζ and $|A_1| \leqslant n$.

(Step 2.) Next, we compute the set $I[t]$ of all tuples U in $\prod_{x \in A_2} Q^*[x]$ such that for all distinct $x, y \in A_2$, $U(x)$ and $U(y)$ are compatible, where $U(x)$ denotes the coordinate of U that comes from $Q^*[x]$. Since $|A_2| \leqslant 2w$, we have $|\prod_{x \in A_2} Q^*[x]| \leqslant \zeta^{2w}$. The set $I[t]$ can be computed in time $\mathcal{O}(\zeta^{2w} \cdot w^2 \cdot (\alpha + w)^2)$.

(Step 3.) Lastly, we construct $Q'[t]$ from $I[t]$ and $C'[t]$ as follows. For every $U \in I[t]$ and every $g \in C'[t]$ where $U(x)$ and g are compatible for all $x \in A_2$, we obtain a new function g' on $\hat{\partial}(t)$ such that $g'(v) = (U(x))(v)$ if $v \in \hat{\partial}(x)$ for some $x \in A_2$, and $g'(v) = g(v)$ if $v \in X_t$, and add it to $Q'[t]$. This can be done in time $\mathcal{O}(|I[t]| \cdot |C'[t]| \cdot (\alpha(\alpha + w))^{2w}) = \mathcal{O}(\zeta^{2w} \cdot n^\alpha \cdot (\alpha(\alpha + w))^{2w})$. This stores valid colorings at t and the size of $Q'[t]$ is at most $|I[t]| \times |C'[t]|$. Using Lemma 10, we find a subset $Q^*[t]$ of $Q'[t]$ of size at most ζ. This can be computed in time

$\mathcal{O}(|Q'[t]| \cdot 2^{(\alpha+w)^2} \cdot (\alpha+w)^{\alpha+w+2}) = \mathcal{O}(\zeta^{2w} \cdot n^{\alpha} \cdot 2^{(\alpha+w)^2} \cdot (\alpha+w)^{\alpha+w+2})$. The total running time for this case is $\mathcal{O}(\zeta^{2w} \cdot n^{\alpha+1} \cdot 2^{(\alpha+w)^2} \cdot (\alpha+w)^{\alpha+4w+2})$.

For the correctness, we prove that ($\star$) $Q^*[t]$ satisfies the property ($\square$).

As $|V(T)| = \mathcal{O}(n)$, the algorithm runs in time $\mathcal{O}(\zeta^{2w} \cdot n^{\alpha+2} \cdot 2^{(\alpha+w)^2} \cdot (\alpha+w)^{\alpha+4w+2}) = 2^{\mathcal{O}((\alpha^2+k)^3)} n^{\alpha+2}$.

Precoloring Extension asks whether a precoloring on a vertex set S can be extended to a k-coloring of G. By making a list for a vertex S to the assigned color, and making a list for a neighbor of $v \in S$ which avoids the assigned color of v, we can reduce to List Coloring.

6 Conclusion

In this paper, we introduced a width parameter called α-edge-crossing width, which lies between edge-cut width and tree-partition-width, and which is incomparable with tree-cut width. We showed that List Coloring and Precoloring Extension are FPT parameterized by α-edge-crossing width. It would be interesting to find more problems that are W[1]-hard parameterized by tree-partition-width, but FPT by α-edge-crossing width for any fixed α. There are six more problems that are known to admit FPT algorithms parameterized by slim tree-cut width, but W[1]-hard parameterized by tree-cut width [9], and these problems are candidates for the next research.

We also introduced edge-crossing width, that is equivalent to tree-partition-width. However, our proof is based on the characterization of graphs of bounded tree-partition-width due to Ding and Oporowski [5], and finding an elementary upper bound of tree-partition-width in terms of edge-crossing width is an interesting problem. More specifically, we ask whether there is a constant c such that for every graph G, $\mathrm{tpw}(G) \leqslant c \cdot \mathrm{ecrw}(G)$.

References

1. Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. Theor. Comput. Sci. **209**(1–2), 1–45 (1998). https://doi.org/10.1016/S0304-3975(97)00228-4
2. Brand, C., Ceylan, E., Hatschka, C., Ganian, R., Korchemna, V.: Edge-cut width: an algorithmically driven analogue of treewidth based on edge cuts (2022). WG2022 accepted. arXiv:2202.13661
3. Bredereck, R., Heeger, K., Knop, D., Niedermeier, R.: Parameterized complexity of stable roommates with ties and incomplete lists through the lens of graph parameters. Inf. Comput. **289**(part A) (2022). Paper No. 104943, 41. https://doi.org/10.1016/j.ic.2022.104943
4. Cygan, M., et al.: Parameterized Algorithms, 1st edn. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-319-21275-3
5. Ding, G., Oporowski, B.: On tree-partitions of graphs. Discrete Math. **149**(1), 45–58 (1996). https://doi.org/10.1016/0012-365X(94)00337-I. www.sciencedirect.com/science/article/pii/0012365X9400337I

6. Fellows, M.R., et al.: On the complexity of some colorful problems parameterized by treewidth. Inf. Comput. **209**(2), 143–153 (2011). https://doi.org/10.1016/j.ic.2010.11.026
7. Ganian, R., Kim, E.J., Szeider, S.: Algorithmic applications of tree-cut width. SIAM J. Discrete Math. **36**(4), 2635–2666 (2022). https://doi.org/10.1137/20M137478X
8. Ganian, R., Korchemna, V.: The complexity of Bayesian network learning: revisiting the superstructure. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems, vol. 34, pp. 430–442. Curran Associates, Inc. (2021). www.proceedings.neurips.cc/paper/2021/file/040a99f23e8960763e680041c601acab-Paper.pdf
9. Ganian, R., Korchemna, V.: Slim tree-cut width (2022). arXiv:2206.15091
10. Ganian, R., Ordyniak, S.: The power of cut-based parameters for computing edge-disjoint paths. Algorithmica **83**(2), 726–752 (2021). https://doi.org/10.1007/s00453-020-00772-w
11. Gözüpek, D., Özkan, S., Paul, C., Sau, I., Shalom, M.: Parameterized complexity of the MINCCA problem on graphs of bounded decomposability. Theor. Comput. Sci. **690**, 91–103 (2017). https://doi.org/10.1016/j.tcs.2017.06.013
12. Kim, E.J., Oum, S.I., Paul, C., Sau, I., Thilikos, D.M.: An FPT 2-approximation for tree-cut decomposition. Algorithmica **80**(1), 116–135 (2018). https://doi.org/10.1007/s00453-016-0245-5
13. Korhonen, T.: A single-exponential time 2-approximation algorithm for treewidth. In: 2021 IEEE 62nd Annual Symposium on Foundations of Computer Science–FOCS 2021, Los Alamitos, CA, pp. 184–192. IEEE Computer Society (2022)
14. Robertson, N., Seymour, P.D.: Graph minors. V. Excluding a planar graph. J. Comb. Theory Ser. B **41**(1), 92–114 (1986). https://doi.org/10.1016/0095-8956(86)90030-4
15. Robertson, N., Seymour, P.D.: Graph minors. XX. Wagner's conjecture. J. Comb. Theory Ser. B **92**(2), 325–357 (2004). https://doi.org/10.1016/j.jctb.2004.08.001
16. Robertson, N., Seymour, P.: Graph minors. X. Obstructions to tree-decomposition. J. Comb. Theory Ser. B **52**(2), 153–190 (1991)
17. Wollan, P.: The structure of graphs not admitting a fixed immersion. J. Comb. Theory Ser. B **110**, 47–66 (2015). https://doi.org/10.1016/j.jctb.2014.07.003

Snakes and Ladders: A Treewidth Story

Steven Chaplick, Steven Kelk(✉), Ruben Meuwese, Matúš Mihalák, and Georgios Stamoulis

Department of Advanced Computing Sciences, Maastricht University, Maastricht, The Netherlands
{s.chaplick,steven.kelk,r.meuwese,matus.mihalak, georgios.stamoulis}@maastrichtuniversity.nl

Abstract. Let G be an undirected graph. We say that G contains a ladder of length k if the $2 \times (k+1)$ grid graph is an induced subgraph of G that is only connected to the rest of G via its four cornerpoints. We prove that if all the ladders contained in G are reduced to length 4, the treewidth remains unchanged (and that this bound is tight). Our result indicates that, when computing the treewidth of a graph, long ladders can simply be reduced, and that minimal forbidden minors for bounded treewidth graphs cannot contain long ladders. Our result also settles an open problem from algorithmic phylogenetics: the common chain reduction rule, used to simplify the comparison of two evolutionary trees, is treewidth-preserving in the display graph of the two trees.

Keywords: Treewidth · Reduction rules · Phylogenetics

1 Introduction

This is a story about treewidth, but it starts in the world of biology. A phylogenetic tree on a set of leaf labels X is a binary tree representing the evolution of X. These are studied extensively in computational biology [16]. Given two such trees a natural aim is to quantify their topological dissimilarity [12]. Many such dissimilarity measures have been devised and they are often NP-hard to compute, stimulating the application of techniques from parameterized complexity [7]. Recently there has been a growing focus on treewidth. This is because, if one takes two phylogenetic trees on X and identifies leaves with the same label, we obtain an auxiliary graph structure known as the *display graph* [6]. Crucially, the treewidth of this graph is often bounded by a function of the dissimilarity measure that we wish to compute [13]. This has led to the use of Courcelle's Theorem within phylogenetics (see e.g. [11,13]) and explicit dynamic programs running over tree decompositions; see [10] and references therein. In [14] the spin-off question was posed: is the treewidth of the display graph actually a meaningful measure of phylogenetic dissimilarity *in itself* - as opposed to purely being a

R. Meuwese was supported by the Dutch Research Council (NWO) KLEIN 1 grant *Deep kernelization for phylogenetic discordance*, project number OCENW.KLEIN.305.

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 187–200, 2023.
https://doi.org/10.1007/978-3-031-43380-1_14

route to efficient algorithms? A closely-related question was whether parameter-preserving reduction rules, applied to two phylogenetic trees to shrink them in size, also preserve the treewidth of the display graph? The well-known *subtree reduction rule* is certainly treewidth preserving [14]. However, the question remained whether the *common chain reduction rule* [2] is treewidth-preserving. A common chain is, informally, a sequence of leaf labels $x_1, \ldots, x_k$ that has the same order in both trees. Concretely, the question arose [14]: is it possible to truncate a common chain to *constant length* such that the treewidth of the display graph is preserved? Common chains form ladder-like structures in the display graph, i.e., this question is about how far ladders can be reduced in length without causing the treewidth to decrease.

In this article we answer this question affirmatively, and more generally. Namely, we do not restrict ourselves to display graphs, but consider arbitrary graphs. A *ladder L of length $k \geq 1$* of a graph G is a $2 \times (k + 1)$ grid graph such that L induces (only) itself and that L is only connected to the rest of the graph by its four cornerpoints. First, we prove that a ladder L can be reduced to length 4 without causing the treewidth to decrease, and that this is best possible: reducing to length 3 sometimes causes the treewidth to decrease. We also show that if $tw(G) \geq 4$ then reduction to length 3 is safe and, again, best possible. These tight examples are also shown to exist for higher treewidths. Returning to phylogenetics, and thus when G is a display graph, we leverage the extra structure in these graphs to show that common chains can be reduced to 4 leaf labels (and thus the underlying ladder to length 3) without altering the treewidth: this result is thus slightly stronger than on general G.

Our proofs are based on first principles: we directly modify a tree decomposition to get what we need. In doing so we come across the problem that, unless otherwise brought under control, the set of bags that contain a given ladder vertex of G can wind and twist through the tree decomposition in very pathological ways. Getting these *snakes* under control is where much of the hard work and creativity lies, and is the inspiration for the title of this paper.

From a graph-theoretic perspective our results have the following significance. First, it is standard folklore that shortening paths (i.e. suppressing vertices of degree 2) is treewidth-preserving, but there is seemingly little in the literature about shortening recursive structures that are slightly more complex than paths, such as ladders. (Note that Sanders [15] did consider ladders, but only for recognizing graphs of treewidth at most 4, and in such a way that the reduction destroys the ladder topology). Second, our results imply a new safe reduction rule for the computation of treewidth; a survey of other reduction rules for treewidth can be found in [1]. Third, we were unable to find sufficiently precise machinery, characterisations of treewidth or restricted classes of tree decomposition in the literature that would facilitate our results. Perhaps most closely related to our ladders are the more general *protrusions*: low treewidth subgraphs that "hang" from a small boundary [9, Ch. 15-16]. There are general (algorithmic) results [5] wherein one can safely cut out a protrusion and replace it with a graph of parameter-proportional size instead – these are based on a problem having *finite* *integer index* [4]. Such techniques might plausibly be used to prove that there is *some* constant to which ladders might safely be shortened, but our tight bounds seem out of their reach. Finally, the results imply that minimal forbidden minors for bounded treewidth cannot have long ladders.

Due to space limitations a number of proofs have been deferred to an appendix, which can be found in the arXiv version of this article [8].

2 Preliminaries

We follow [14] for notation. A *tree decomposition* of an undirected graph $G = (V, E)$ is a pair $(\mathcal{B}, \mathbb{T})$ where $\mathcal{B} = \{B_1, \ldots, B_q\}$, $B_i \subseteq V(G)$, is a multiset of *bags* and $\mathbb{T}$ is a tree whose q nodes are in bijection with $\mathcal{B}$, and

(tw1) $\cup_{i=1}^{q} B_i = V(G)$;
(tw2) $\forall e = \{u, v\} \in E(G), \exists B_i \in \mathcal{B}$ s.t. $\{u, v\} \subseteq B_i$;
(tw3) $\forall v \in V(G)$, all the bags B_i that contain v form a connected subtree of $\mathbb{T}$.

The *width* of $(\mathcal{B}, \mathbb{T})$ is equal to $\max_{i=1}^{q} |B_i| - 1$. The *treewidth* of G, denoted $tw(G)$, is the smallest width among all tree decompositions of G. Given a tree decomposition $\mathbb{T}$ of a graph G, we denote by $V(\mathbb{T})$ the (multi)set of its bags and by $E(\mathbb{T})$ the set of its edges. Property (tw3) is also known as *running intersection property*. Without loss of generality, we consider only connected graphs G.

Note that subdividing an edge $\{u, v\}$ of G with a new degree-2 vertex uv does not change the treewidth of G. In the other direction, suppression of degree-2 vertices is also treewidth preserving *unless* it causes the only cycle in a graph to disappear (e.g. if G is a triangle); unlike [14] we will never encounter this boundary case. An equivalent definition of treewidth is based on chordal graphs. Recall that a graph G is chordal if every induced cycle in G has exactly three vertices. The treewidth of G is the minimum, ranging over *all* chordal completions $c(G)$ of G (we add edges until G becomes chordal), of the size of the maximum clique in $c(G)$ minus one. Under this definition, each bag of a tree decomposition of G naturally corresponds to a maximal clique in a chordal completion of G [3].

We say that a graph H is a *minor* of another graph G if H can be obtained from G by deleting edges and vertices and by contracting edges.

A *ladder L of length* $k \geq 1$ is a $2 \times (k + 1)$ grid graph. A *square* of L is a set of vertices of L that induce a 4-cycle in L. We call the endpoints of L, i.e., the degree-2 vertices of L, the *cornerpoints* of L. We say that a graph G *contains* L if the following holds (see Fig. 1 for illustration):

1. The subgraph induced by vertices of L is L itself.
2. Only cornerpoints of L can be incident to an edge with an endpoint outside L.

Observe that a ladder of length k is a minor of the ladder of length $(k + 1)$. Treewidth is non-increasing under the action of taking minors, so reducing the length of a ladder in a graph cannot increase the treewidth of the graph.

Suppose G contains a ladder L. We say that L *disconnects* G if L contains a square $\{u, v, w, x\}$ such that the two horizontal edges of the square (following

Fig. 1, these are the edges $\{u, w\}$ and $\{v, x\}$) form an edge cut of the entire graph G. Note that a square of L has this property if and only if all squares of L do. Also, if we reduce the length of a ladder L to obtain a shorter ladder L', L' disconnects G if and only if L does. We recall a number of results from Section 5.2 of [14]; these will form the starting point for our work.

Lemma 1 ([14]). *Suppose G contains a disconnecting ladder L. The ladder L can be increased arbitrarily in length without increasing the treewidth of G.*

For the more general case, the following weaker result is known.

Lemma 2 ([14]). *Suppose G has $tw(G) \geq 3$ and contains a ladder. If the ladder is increased arbitrarily in length, the treewidth of G increases by at most one.*

We now make the following (new) observation; the proof is in the appendix.

Observation 1. *Suppose G contains a ladder L of length 2 or longer. If L is not disconnecting, then $tw(G) \geq 3$.*

We can leverage Observation 1 to reformulate Lemma 2 without the $tw(G) \geq 3$ assumption. However it then only applies to ladders of size at least two.

Lemma 3. *Suppose G contains a ladder L with length at least 2. If L is increased arbitrarily in length, the treewidth of the graph increases by at most one.*

If we start from a sufficiently long ladder, can the ladder be increased in length without increasing the treewidth? Past research has the following partial result.

Theorem 1 ([14]). *Let G be a graph with $tw(G) = k$. There is a value $f(k)$ such that if G contains a ladder of length $f(k)$ or longer, the ladder can be increased in length arbitrarily without altering (in particular: increasing) the treewidth.*

Ideally we would like a single, universal value *that does not depend on k*. In this article we will show that such a single, universal constant does exist.

3 Results

We first consider graphs of treewidth at least 4; we later remove this restriction.

Theorem 2. *Let G be a graph with $tw(G) \geq 4$. If G has a ladder L of length 3 or higher, the ladder can be lengthened arbitrarily without changing the treewidth.*

Proof. Due to Lemma 1 we can assume that L is not disconnecting. Our general strategy is to show that if G contains the ladder L shown in Fig. 1, we can insert an extra 'rung' in the ladder without increasing the treewidth, thus obtaining a ladder with one extra square (see Fig. 2). The extension of the ladder by one square can then be iterated to obtain an arbitrary length ladder.

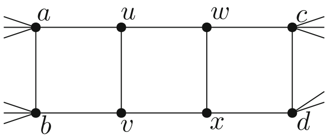


Fig. 1. A ladder L of length 3 with corner points a, b, c, d.

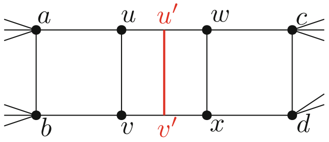


Fig. 2. Inserting a new edge $\{u', v'\}$ into ladder L results in ladder L' of length 4.

Let L be the ladder shown in Fig. 1, and assume that G contains L. Let $(\mathcal{B}, \mathbb{T})$ be a minimum-width tree decomposition for G. We proceed with a case analysis. The cases are cumulative: we will assume that earlier cases do not hold.

Case 1. Suppose that $\mathcal{B}$ contains a bag B such that all four vertices from one of the squares of L are in B. Let $\{u, v, w, x\}$, say, be the square of L contained in bag B, where the position of the vertices is as in Fig. 1. We prolong the ladder as in Fig. 2 and create a valid tree decomposition for the new graph as follows: we introduce a new size-5 bag $B' = \{u', u, v, w, x\}$ which we attach pendant to B in the tree decomposition, and a new size-5 bag $B'' = \{u', v', v, w, x\}$ which we attach pendant to B'. Observe that this is a valid tree decomposition for the new graph. Due to the fact that $tw(G) \geq 4$, the treewidth does not increase, and the statement follows. Note that in this construction B'' contains all four of $\{u', w, v', x\}$, which is a square of the new ladder, so the construction can be applied iteratively many times as desired to produce a ladder of arbitrary length.

Case 2. Suppose that $\mathcal{B}$ contains a bag B such that $|B \cap \{a, u, w, c\}| \geq 2$ and $|B \cap \{b, v, x, d\}| \geq 2$. Let h_1, h_2 be two distinct vertices from $B \cap \{a, u, w, c\}$ and l_1, l_2 be two distinct vertices from $B \cap \{b, v, x, d\}$.

Observe that it is possible to partition the sequence a, u, w, c into two disjoint intervals H_1, H_2, and the sequence b, v, x, d into two disjoint intervals L_1, L_2 such that $h_1 \in H_1$, $h_2 \in H_2$, $l_1 \in L_1$ and $l_2 \in L_2$. If we contract the edges and vertices in each of H_1, H_2, L_1, L_2 we obtain a new graph G' which is a minor of G. Note that G' is similar to G except that the ladder now has two fewer squares – the three original squares have been replaced by a square whose corners correspond to H_1, H_2, L_1, L_2. This square might contain a diagonal but we simply delete this. We have $tw(G') \leq tw(G)$ because treewidth is non-increasing under taking minors. Now, by projecting the contraction operations onto $(\mathcal{B}, \mathbb{T})$ in the usual way[1], we obtain a tree decomposition $(\mathcal{B}', \mathbb{T}')$ for G' such that the width of $\mathbb{T}'$ is less than or equal to the width of $\mathbb{T}$. The bag in $(\mathcal{B}', \mathbb{T}')$ corresponding to B, let us call this B', contains all four vertices H_1, H_2, L_1, L_2. Clearly, $\mathbb{T}'$ is a valid tree decomposition for G'. We distinguish two subcases.

[1] In every bag of the decomposition vertices from H_1 all receive the vertex label H_1, and similarly for the other subsets H_2, L_1, L_2.

1. If $\mathbb{T}'$ has width at least 4, we can repeatedly apply the Case 1 transformation to B' to produce an arbitrarily long ladder without raising the width of $\mathbb{T}'$. The resulting decomposition will thus have width no larger than $\mathbb{T}$.
2. Suppose $\mathbb{T}'$ has width strictly less than 4, and thus strictly less than the width of $\mathbb{T}$. The width of $\mathbb{T}'$ is at least 3 because of the bag containing H_1, H_2, L_1, L_2. Case 1 introduces size-5 bags and can thus raise the width of the decomposition by at most 1. Hence we again obtain a decomposition whose width is no larger than $\mathbb{T}$ for a graph with an arbitrarily long ladder.

This concludes Case 2. Moving on, any chordalization of G must add the diagonal $\{w, v\}$ and/or the diagonal $\{u, x\}$. Hence we can assume that there is a bag containing $\{u, w, v\}$ and another bag containing $\{v, w, x\}$. (If the other diagonal is added we can simply flip the labelling of the vertices in the horizontal axis i.e. $a \Leftrightarrow b, u \Leftrightarrow v$ and so on). As Case 1 does not hold we can assume that the bag containing $\{u, w, v\}$ is distinct from the bag containing $\{v, w, x\}$.

For the benefit of later cases we impose extra structure on our choice of minimum-width tree decomposition of G. The *distance* of decomposition $(\mathcal{B}, \mathbb{T})$ is the minimum, ranging over all pairs of bags B_1, B_2 such that B_1 contains $\{u, w, v\}$ and B_2 contains $\{v, w, x\}$, of the length of the path in $\mathbb{T}$ from B_1 to B_2.

We henceforth let $(\mathcal{B}, \mathbb{T})$ be a minimum-width tree decomposition of G such that, ranging over all minimum-width tree decompositions, the distance is minimized. Clearly such a tree decomposition exists.

Let B_1, B_2 be two bags from $\mathcal{B}$ with $\{u, w, v\} \subseteq B_1$, $\{v, w, x\} \subseteq B_2$ which achieve this minimum distance. Let P be the path of bags from B_1 to B_2, including B_1 and B_2. We assume that P is oriented left to right, with B_1 at the left end and B_2 on the right. As Case 2 does not hold, we obtain the following.

Observation 2. *B_1 does not contain b, x or d, and B_2 does not contain a, u, c.*

Case 3. B_1 and B_2 are adjacent in P. Although this could be subsumed into a later case it introduces important machinery; we therefore treat it separately.

Subcase 3.1: Suppose $a \in B_1$ (or, completely symmetrically, $d \in B_2$). Note that in this case all the edges in G incident to u are covered by B_1. Hence, we can safely delete u from all bags except B_1. Next, we create a new bag $B^* = \{a, u, w, v\}$ and attach it pendant to B_1, and finally we replace u with x in B_1. It can be easily verified that this is a valid tree decomposition for G and that the width is not increased, so it is still a minimum-width tree decomposition. However, B_1 is now a candidate for Case 2, and we are done. Note that replacing u with x in B_1 is only possible because B_1 is next to B_2 in P.

Subcase 3.2: Suppose Subcase 3.1 does not hold. Then $a \notin B_1$ (and, symmetrically, $d \notin B_2$). Putting all earlier insights together, we see $a, b, x, d \notin B_1$ and $a, u, c, d \notin B_2$. Observe that a, which is not in B_2, is not in any bag to the right of B_2. If it was, then the fact that some bag contains the edge $\{a, u\}$, and the running intersection property, entails that B_2 would contain at least one of a and u, neither of which is permitted. Hence, if a appears in bags other than B_1, they are all in the left part of the decomposition. Completely symmetrically, if d is in bags other than B_2, they are all in the right part of the decomposition. Because of this, b can only appear on the left of the decomposition (because the edge $\{a, b\}$ has to be covered) and c can only be on the right of the decomposition (because of the edge $\{c, d\}$). Summarising, B_1 (respectively, B_2) does not contain a or b (respectively, c or d) and all bags containing a or b (respectively, c or d) are in the left (respectively, right) part of the decomposition. Note that $c \notin B_1$. This is because edge $\{c, d\}$ has to be in some bag, and this must necessarily be to the right of B_2: but then running intersection puts at least one of c, d in B_2, contradiction. Symmetrically, $b \notin B_2$. So $a, b, c, d, x \notin B_1$ and $a, b, c, d, u \notin B_2$.

We now describe a construction that we will use extensively: *reeling in (the snakes) a and b*. Observe that, due to coverage of the edge $\{a, u\}$, and running intersection, there is a simple path of bags p_{ua} starting at B_1 that all contain u such that the endpoint of the path also contains a. The path will necessarily be entirely on the left of the decomposition. Due to coverage of the edge $\{b, v\}$ there is an analogously-defined simple path p_{vb}. (Note that p_{ua} and p_{vb} both exit B_1 via the same bag B'. If they exited via different bags, coverage of the edge $\{a, b\}$ would force at least one of a, b to be in B_1, yielding a contradiction). Now, in the bags along p_{ua}, except B_1, we relabel u to be a, and in the bags along p_{vb}, except B_1, we relabel v to be b. This is no longer necessarily a valid tree decomposition, because coverage of the edges $\{u, a\}$ and $\{v, b\}$ is no longer guaranteed, but we shall address this in due course. Next we delete the vertices u, w, v from all bags on the left of the decomposition, except B_1; they will not be needed. (The only reason that w would be in a bag on the left, would be to meet c, since B_1 and B_2 already cover the edges $\{u, w\}$ and $\{w, x\}$. But then, due to coverage of the edge $\{c, d\}$ and the fact that d only appears on the right of the decomposition, running intersection would put at least one of c, d in B_1, contradiction.) Observe that B' contains $\{a, b\}$. We replace B_1 with 5 copies of itself, and place these bags in a path such that the leftmost copy is adjacent to B', the rightmost copy is adjacent to B_2, and all other bags that were originally adjacent to B_1 can (arbitrarily) be made adjacent to the leftmost copy. In the 5 copied bags we replace $\{u, w, v\}$ respectively with: $\{a, u', b\}$, $\{u', b, v'\}$, $\{u', u, v'\}$, $\{v', u, v\}$ and $\{u, w, v\}$. It can be verified that this is a valid tree decomposition for G', and our construction did not inflate the treewidth - we either deleted vertices from bags or relabelled vertices that were already in bags - so we are done. The operation can easily be telescoped, if desired, to achieve an arbitrarily long ladder.

Case 4. P contains at least one bag other than B_1 and B_2.

Observation 3. *All bags in P contain v, w, by the running intersection property.*

We partition the bags of the decomposition into (i) B_1, (ii) bags *left* of B_1, (iii) B_2, (iv) bags *right* of B_2, (v) all other bags (which we call the *interior*).

Recall that $b, d, x \notin B_1$, $a, c, u \notin B_2$ (because Case 2 does not hold). The proofs of the following two observations are in the appendix.

Observation 4. *No bag in the interior contains u or x. B_1 does not contain x, and no bag on the left contains x. Symmetrically, B_2 does not contain u, and no bag on the right contains u.*

Observation 5. *At least one of the following is true: $a \in B_1$, a is in a bag on the left. Symmetrically, at least one of the following is true: $d \in B_2$, d is in a bag on the right.*

Now, suppose w is somewhere on the left. We will show that then either w can be deleted from the bags on the left, or Case 2 holds. A symmetrical analysis will hold if v is somewhere on the right. Specifically, the only possible reason for w to be on the left would be to cover the edge $\{w, c\}$ – all other edges incident to w are already covered by B_1 and B_2. If no bags on the left contain c, we can simply delete w from all bags on the left. On the other hand, if some bag on the left contains c, then $c \in B_1$, because: $d \notin B_1$, the need to cover the edge $\{c, d\}$, the presence of d on the other 'side' of the decomposition (Observation 5), and running intersection. So we have that $c, u, w, v \in B_1$. This bag already covers all edges incident to w, except possibly the edge $\{w, x\}$. To address this, we replace w everywhere in the tree decomposition with x - this is a legal tree decomposition because some bag contains $\{w, x\}$ - and then add a bag $B' = \{u, w, x, c\}$ pendant to B. This new bag serves to cover all edges incident to w. But B_1 now contains u, v, c, x, so Case 2 applies, and we are done! Hence, we can assume that w is nowhere on the left, and, symmetrically, that v is nowhere on the right. In fact, the above argument can, independently of w, be used to trigger Case 2 whenever $c \in B_1$ or $b \in B_2$. So at this stage of the proof we know: $b, c, d, x \notin B_1$ (and c is not on the left) and $a, b, c, u \notin B_2$ (and b is not on the right).

Subcase 4.1: Suppose $a \notin B_1$. Then, a must only be on the left. It cannot be in the interior (or on the right) because the edge $\{u, a\}$ must be covered, $a \notin B_1$, $u \in B_1$, and u is not in the interior (Observation 4). Because a is on the left, and because some bag must contain the edge $\{a, b\}$, b must also be on the left. In fact b is only on the left. The presence of b both on the left and in the interior (or on the right) would force b into B_1 by running intersection, contradicting the fact that $b \notin B_1$. So a, b are only on the left. We are now in a situation similar to Subcase 3.2. We use the same *reeling in a and b* construction and we are done.

Subcase 4.2: Suppose $a \in B_1$. Note that here u has all its incident edges covered by B_1, so u can be deleted from all other bags. Recall that $b \notin B_1$. Due to edge $\{b, v\}$ some bag must contain both v and b. Suppose there is such a bag on the left. We attach a new bag $\{a, u, w, v\}$ pendant to B_1 and delete u from B_1. We put x in B_1 and to ensure running intersection we replace v with x in all bags anywhere to the right of B_1. This is safe, because in the part of the decomposition right of B_1, v only needs to meet x (and not b, because v meets b on the left). Thus, B_1 now contains $\{a, v, w, x\}$ and Case 2 can be applied.

Hence, we conclude that $\{v, b\}$ is not in a bag on the left. Because of this v can safely be deleted from all bags on the left. That is because any path p_{vb} that starts at B_1 and finishes at a bag containing b must go via the interior. In fact, such a path must avoid B_2, and is thus entirely contained in the interior.

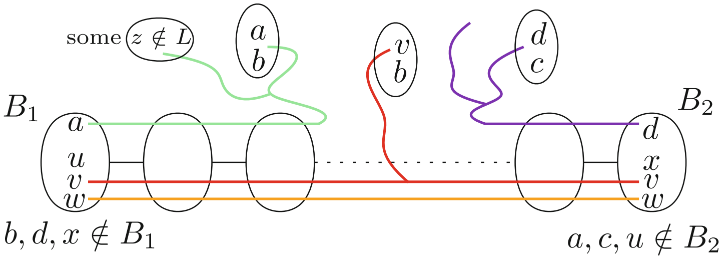


Fig. 3. Path p_{ab} goes via the interior, but it cannot be relabelled to b because it is used by other paths p_{az} to some neighbour z of a that does *not* lie on the ladder.

It avoids B_2 because $a, b \notin B_2$ and $\{v, b\}$ cannot be in a bag to the right: if it was, coverage of edge $\{a, b\}$, the fact that $a \in B_1$ and running intersection would mean that at least one of a and b is in B_2, yielding a contradiction.

The only case that remains is $a \in B_1$, $\{v, b\}$ is not in a bag on the left and thus p_{vb} is in the interior. By symmetry, we assume that $d \in B_2$, $\{w, c\}$ is not in a bag on the right and thus p_{wc} is in the interior. Consider any path p_{ab} starting at B_1, defined in the now familiar way. Note that no bag on the left of B_1 can contain b. This is because $\{v, b\}$ is in a bag in the interior: hence if b was also on the left, b would then by running intersection be in B_1 and we would be in an earlier case. This means that p_{ab} must go via the interior. Suppose the following operation gives a valid tree decomposition: delete u from B_1, attach a new bag $B^* = \{a, u, w, v\}$ pendant to B_1, and relabel all occurrences of a along the path p_{ab} (except in B_1) with b. Then we are done, because we are back in Case 2. A symmetrical situation holds for the path p_{dc}.

Assume therefore that this transformation does not give a valid tree decomposition. This is the most complicated case to deal with. It is depicted in Fig. 3. The issue here is that the path p_{ab} (respectively, p_{dc}) necessarily goes via the interior, but cannot be relabelled with b (respectively, c) because the path is also part of p_{az} (respectively, p_{dz}) where z is some non-ladder vertex that is adjacent to a (respectively d). We deal with this as follows. We argue that some bag in the decomposition *must* contain a, b, v (and possibly other vertices). Suppose this is not the case. By standard chordalization arguments, every chordalization adds at least one diagonal edge to every square of the ladder. If $\{a, b, v\}$ are not together in a bag, then this is because the corresponding chordalization did not add the diagonal $\{a, v\}$ to square $\{a, u, b, v\}$. Hence, the chordalization must have added the diagonal $\{u, b\}$. This would in turn mean that some bag contains $\{a, u, b\}$. Such a bag must be on the interior, because this is the only place that b can be found. However, no bags in the interior contain u – contradiction.

Hence, some bag B' indeed contains $\{a, b, v\}$. Again, because b is only on the interior, B' must be in the interior. There could be multiple such bags, but this does not harm us. Let $B_{v\text{-done}}$ be the rightmost bag on the path P that is part of a path, starting from B_1, from a to some bag B' containing $\{a, b, v\}$. Let $B_{a\text{-done}}$ be the rightmost bag on P that contains a. Note that $B_{v\text{-done}}$ contains

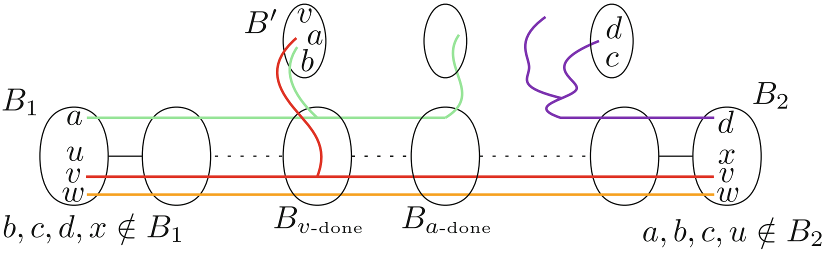


Fig. 4. The bags B', $B_{v\text{-done}}$ and $B_{a\text{-done}}$ illustrated. Note that $B_{a\text{-done}}$ cannot be the penultimate bag on the path P from B_1 to B_2, due to the presence of d in that bag.

a (because of running intersection: $a \in B_1$ and $a \in B'$) and v, w (because it lies on P). We also have $a, v, w \in B_{a\text{-done}}$. By construction, $B_{v\text{-done}}$ is either equal to $B_{a\text{-done}}$ or left of it. This is important because it means that the only reason v might need to be in bags to the right of $B_{a\text{-done}}$ is to reach a bag containing x (i.e. to cover the edge $\{v, x\}$) – all other edges are already covered elsewhere in the decomposition; in particular, edge $\{v, b\}$ is covered by the bag B' containing $\{a, b, v\}$. See Fig. 4 for clarification.

Recall that none of the bags on the path P contain both a and d. (If they did, there would be a bag containing $\{a, d, v, w\}$ and we would be in Case 2, done.) We also know that some path p_{dc} goes via the interior and thus that the penultimate bag on P (i.e. the one before B_2) thus definitely contains d. (To clarify: $c \notin B_2$, $d \in B_2$, the edge $\{c, d\}$ must be covered, and c is only in the interior). Combining these insights tells us that this penultimate bag definitely does *not* contain a, and hence $B_{a\text{-done}}$ is *not* equal to the penultimate bag; it is further left. This fact is crucial. Consider $B_{a\text{-done}}$ and the bag immediately to its right on P. Between these two bags we insert a copy of $B_{a\text{-done}}$, call it B_r, remove a from B_r (i.e. forget it), and add the element x to it instead. Finally, we switch v to x in all bags on P right of B_r, including B_2 itself, and delete v from all bags in the tree decomposition that are anywhere to the right of B_r; there is no point having them there. It requires some careful checking but this is a valid (minimum-width) tree decomposition. Moreover, B_r contains w, v, x. The fact that $B_{a\text{-done}}$ was not the penultimate bag of P, means that the length of the path from B_1 to B_r is strictly less than the length of the path from B_1 to B_2: contradiction on the assumption that these were the closest bags containing $\{u, w, v\}$ and $\{w, v, x\}$ respectively. We are done. ■

We now deal with the situation when the $tw(G) \geq 4$ assumption is removed.

Lemma 4. *If G has a ladder L of length 5 or longer, the ladder can be increased in length arbitrarily without altering (in particular: increasing) the treewidth. This holds irrespective of the treewidth of G.*

Proof. Let L be a ladder of length 5 or longer. We can assume that L is not disconnecting and $tw(G) \leq 3$. We select the three most central squares and label these as in Fig. 1. These are flanked on both sides by at least one other square. Hence, a, b, c, d each has exactly one neighbour outside the 3 squares, let us call these a', b', c', d' respectively, where $\{a', b'\}$ is an edge and $\{c', d'\}$ is an edge. Now, $tw(G) = 3$ because L is not disconnecting. The only part of the proof of Theorem 2 that does not work for $tw(G) = 3$ is Case 1 and (indirectly) Case 2 because these create size-5 bags. We show that neither case can hold.

Consider Case 1. Let B be a bag containing one of the three most central squares S of the ladder (these are the only squares to which Case 1 is ever applied). A *small* tree decomposition is one where no bag is a subset of another. If a tree decomposition is not small, then by running intersection it must contain two adjacent bags $B^{\dagger}, B^{\ddagger}$ such that $B^{\dagger} \subseteq B^{\ddagger}$. The two bags can then be safely merged into $B^{\ddagger}$. By repeating this a small tree decomposition can be obtained without raising the width of the original minimum-width decomposition. Furthermore, some bag B will still exist containing S. If B has five or more vertices we immediately have $tw(G) \geq 4$ and we are done. Otherwise, let B' be any bag adjacent to B; such a bag must exist because G has more than 4 vertices. Due to the smallness of the decomposition we have $B \not\subseteq B'$ and $B' \not\subseteq B$. Hence, $B \cap B' \subset B$ and $B \cap B' \subset B'$. A *separator* is a subset of vertices whose deletion disconnects the graph. Now, $B \cap B'$ is by construction, and the definition of tree decompositions a separator of G. However, due to our use of the three central squares, S is not a separator, and no subset of it is a separator either; the inclusion of a', b', c', d' and the edges $\{a', b'\}$ and $\{c', d\}$, alongside the fact that L is not disconnecting, ensure this. This yields a contradiction. Hence Case 1 implies $tw(G) \geq 4$ i.e. it cannot happen when $tw(G) = 3$.

We are left with Case 2. This case replaces the three centremost squares with a single square, and deletes any diagonals that this single square might have, to obtain a new graph G'. We have $tw(G') \leq tw(G)$, by minors. Note that $tw(G') \geq 3$ because the shorter ladder in G' (which has length at least 3) is still disconnecting. Hence, $tw(G') = tw(G) = 3$. The decomposition $\mathbb{T}'$ of G' obtained by projecting the contraction operations onto the tree decomposition $\mathbb{T}$ of G, is a valid tree decomposition (as argued in Case 2) with the property that the width of $\mathbb{T}'$ is less than or equal to the width of $\mathbb{T}$. $\mathbb{T}'$ cannot have width less than 3, so it must have width 3. Hence it is a tree decomposition of G' in which all bags have at most four vertices. We then transform $\mathbb{T}'$ into a small tree composition: this does not raise the width of the decomposition, and every bag prior to the transformation either survives or is absorbed into another. Consider the bag B' containing H_1, H_2, L_1, L_2. The presence of $a', b', c,', d'$ in G' and the fact that the ladder in G' is not disconnecting, means that H_1, H_2, L_1, L_2 is not a separator for G', and neither is any subset of those four vertices. But the intersection of B' with any neighbouring bag *must* be a separator. Hence B' must contain a fifth vertex, contradiction. So Case 2 cannot happen when $tw(G) = 3$. ■

We can, however, still do better. The following proof is in the appendix.

Theorem 3. *If G has a ladder L of length 4 or longer, the ladder can be increased in length arbitrarily without altering (in particular: increasing) the treewidth. This holds irrespective of the treewidth of G.*

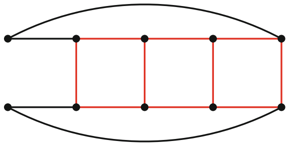


Fig. 5. A graph of treewidth 3 that contains a ladder with 3 squares, shown in red. Increasing the length of the ladder by 1 square increases the treewidth to 4. (Color figure online)

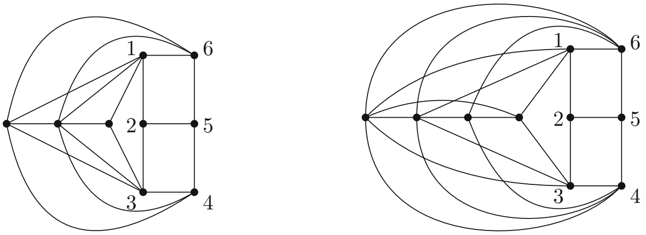


Fig. 6. These graphs have treewidth 4 (left) and 5 (right) and each one has a length 2 ladder, induced by $\{1, 2, 3, 4, 5, 6\}$. In each case increasing the length of the ladder to length 3 increases the treewidth of the graph by one.

Tightness. The constant 4 in the statement of Theorem 3 is equal to the constant obtained for the 'bottleneck' case $tw(G) = 3$. An improved constant 3 for this case is not possible, as Fig. 5 shows. It is natural to ask whether, when $tw(G) \geq 4$, we can start from ladders of length 2, rather than 3. This is also not possible. See Fig. 6. In fact, we have examples up to treewidth 20. These can be found at https://github.com/skelk2001/snakes_and_ladders. We conjecture that this holds for all treewidths, but defer this to future work.

Implications for Phylogenetics. A phylogenetic tree is a binary tree whose leaves are bijectively labelled by a set X. The *display graph* of T_1, T_2 on X is obtained by identifying leaves with the same label. In [14] it was asked whether *common chains* could be truncated to constant length without lowering the treewidth of the display graph. Theorem 3 establishes that the answer is *yes*. In fact, due to the restricted structure of display graphs, we can prove a stronger result: truncation to 4 leaves (i.e. 3 squares) is safe, and this is best possible. Theorem 4 summarizes this. We provide full details in the appendix.

Theorem 4. *Let T_1, T_2 be two unrooted binary phylogenetic trees on the same set of taxa X, where $|X| \geq 4$ and $T_1 \neq T_2$. Then exhaustive application of the subtree reduction and the common chain reduction (where common chains are reduced to 4 leaf labels) does not alter the treewidth of the display graph. This is best possible, because there exist tree pairs where truncation of common chains to length 3 does reduce the treewidth of the display graph (see Fig 7).*

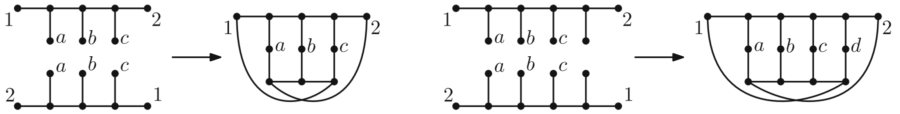


Fig. 7. Lengthening the common chain $\{a, b, c\}$ to $\{a, b, c, d\}$ in these phylogenetic trees causes the treewidth of the display graph to increase. Equivalently: shortening common chains to 3 leaf labels is not guaranteed to preserve treewidth in the display graph.

4 Future Work

A number of interesting open questions remain. First, can the results in this paper be made constructive? That is, given a minimum-width tree decomposition for a graph with a short ladder, but not necessarily the special distance-minimizing one assumed in our proofs, can we manipulate it to obtain a tree decomposition of the same width after the ladder is lengthened? Second, can we prove that for every starting treewidth, not just $tw(G) \leq 20$, extending a ladder of length 2 sometimes causes the treewidth to increase? Third, can our results be (elegantly) generalized to more complex recursive structures than ladders? This requires careful analysis of which parts of our proof are specific to ladders. Finally, if our results are generalized to more general structures, the question arises of how to implement these results as reduction rules: this requires efficient algorithmic recognition of "long" structures in order to produce the "short" variant.

Acknowledgements. We thank Hans Bodlaender and Bart Jansen for insightful feedback. We also thank the members of our department for useful discussions.

References

1. Abu-Khzam, F.N., Lamm, S., Mnich, M., Noe, A., Schulz, C., Strash, D.: Recent advances in practical data reduction. In: Bast, H., Korzen, C., Meyer, U., Penschuck, M. (eds.) Algorithms for Big Data. LNCS, vol. 13201, pp. 97–133. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-21534-6_6
2. Allen, B., Steel, M.: Subtree transfer operations and their induced metrics on evolutionary trees. Ann. Comb. **5**, 1–15 (2001)
3. Blair, J., Peyton, B.: Graph Theory and Sparse Matrix Computation, chap. An Introduction to Chordal Graphs and Clique Trees. In: George, A., Gilbert, J.R., Liu, J.W.H. (eds.) Graph Theory and Sparse Matrix Computation. The IMA Volumes in Mathematics and its Applications, vol. 56, pp. 1–29. Springer, New York (1993). https://doi.org/10.1007/978-1-4613-8369-7_1
4. Bodlaender, H.L., van Antwerpen-de Fluiter, B.: Reduction algorithms for graphs of small treewidth. Inf. Comput. **167**(2), 86–119 (2001)
5. Bodlaender, H.L., Fomin, F.V., Lokshtanov, D., Penninkx, E., Saurabh, S., Thilikos, D.M.: (meta) kernelization. J. ACM **63**(5), 44:1–44:69 (2016)
6. Bryant, D., Lagergren, J.: Compatibility of unrooted phylogenetic trees is FPT. Theoret. Comput. Sci. **351**(3), 296–302 (2006)

7. Bulteau, L., Weller, M.: Parameterized algorithms in bioinformatics: an overview. Algorithms **12**(12), 256 (2019)
8. Chaplick, S., Kelk, S., Meuwese, R., Mihalak, M., Stamoulis, G.: Snakes and ladders: a treewidth story. arXiv preprint arXiv:2302.10662 (2023)
9. Fomin, F.V., Lokshtanov, D., Saurabh, S., Zehavi, M.: Kernelization: Theory of Parameterized Preprocessing. Cambridge University Press, Cambridge (2019)
10. van Iersel, L., Jones, M., Weller, M.: Embedding phylogenetic trees in networks of low treewidth. In: Chechik, S., Navarro, G., Rotenberg, E., Herman, G. (eds.) 30th Annual European Symposium on Algorithms, ESA 2022, September 5–9, 2022, Berlin/Potsdam, Germany. LIPIcs, vol. 244, pp. 69:1–69:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). arXiv preprint arXiv:2207.00574
11. Janssen, R., Jones, M., Kelk, S., Stamoulis, G., Wu, T.: Treewidth of display graphs: bounds, brambles and applications. J. Graph Algorithms Appl. **23**(4), 715–743 (2019)
12. John, K.S.: The shape of phylogenetic treespace. Syst. Biol. **66**(1), e83 (2017)
13. Kelk, S., van Iersel, L., Scornavacca, C., Weller, M.: Phylogenetic incongruence through the lens of monadic second order logic. J. Graph Algorithms Appl. **20**(2), 189–215 (2016)
14. Kelk, S., Stamoulis, G., Wu, T.: Treewidth distance on phylogenetic trees. Theoret. Comput. Sci. **731**, 99–117 (2018)
15. Sanders, D.: On linear recognition of tree-width at most four. SIAM J. Discret. Math. **9**(1), 101–117 (1996)
16. Steel, M.: Phylogeny: Discrete and Random Processes in Evolution. SIAM (2016)

Parameterized Results on Acyclic Matchings with Implications for Related Problems

Juhi Chaudhary(✉) and Meirav Zehavi

Department of Computer Science, Ben-Gurion University of the Negev,
Beersheba, Israel
juhic@post.bgu.ac.il, meiravze@bgu.ac.il

Abstract. A matching M in a graph G is an *acyclic matching* if the subgraph of G induced by the endpoints of the edges of M is a forest. Given a graph G and a positive integer ℓ, Acyclic Matching asks whether G has an acyclic matching of *size* (i.e., the number of edges) at least ℓ. In this paper, we first prove that assuming $\mathsf{W[1]} \nsubseteq \mathsf{FPT}$, there does not exist any FPT-approximation algorithm for Acyclic Matching that approximates it within a constant factor when parameterized by ℓ. Our reduction is general in the sense that it also asserts FPT-inapproximability for Induced Matching and Uniquely Restricted Matching. We also consider three below-guarantee parameters for Acyclic Matching, viz. $\frac{n}{2} - \ell$, $\mathsf{MM(G)} - \ell$, and $\mathsf{IS(G)} - \ell$, where n is the number of vertices in G, $\mathsf{MM(G)}$ is the *matching number* of G, and $\mathsf{IS(G)}$ is the *independence number* of G. We note that the result concerning the below-guarantee parameter $\frac{n}{2} - \ell$ is the most technical part of our paper. Also, we show that Acyclic Matching does not exhibit a polynomial kernel with respect to vertex cover number (or vertex deletion distance to clique) plus the size of the matching unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

Keywords: Acyclic Matching · Parameterized Algorithms · Kernelization Lower Bounds · Induced Matching · Uniquely Restricted Matching

1 Introduction

Matchings form a central topic in graph theory and combinatorial optimization [31]. In addition to their theoretical fruitfulness, matchings have various practical applications, such as assigning new physicians to hospitals, students to high schools, clients to server clusters, kidney donors to recipients [32], and so on. Moreover, matchings can be associated with the concept of *edge colorings* [3, 6, 40], and they are a useful tool for finding optimal solutions or bounds in

The authors are supported by the European Research Council (ERC) project titled PARAPATH (101039913).

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 201–216, 2023.
https://doi.org/10.1007/978-3-031-43380-1_15

competitive optimization games on graphs [2,21]. Matchings have also found applications in Parameterized Complexity and Approximation Algorithms. For example, *Crown Decomposition*, a construction widely used for kernelization, is based on classical matching theorems [12], and matchings are one of the oldest tools in designing approximation algorithms for graph problems (e.g., consider the classical 2-approximation algorithm for Vertex Cover [1]).

A matching M is a *$\mathcal{P}$-matching* if $G[V_M]$ (the subgraph induced by the endpoints of edges in M) has the property $\mathcal{P}$, where $\mathcal{P}$ is some graph property. The problem of deciding whether a graph admits a $\mathcal{P}$-matching of a given size (number of edges) has been investigated for several graph properties [17,20,22,36,38,39]. If the property $\mathcal{P}$ is that of being a graph, a disjoint union of edges, a forest, or having a unique perfect matching, then a $\mathcal{P}$-matching is a *matching* [33], an *induced matching* [39], an *acyclic matching* [20], and a *uniquely restricted matching* [22], respectively. In this paper, we focus on the parameterized complexity of acyclic matchings, but also discuss implications for the cases of induced matchings and uniquely restricted matchings. In particular, we study the parameterized complexity of these problems.

Problem Definitions and Related Works. Given a graph G and a positive integer ℓ, Acyclic Matching asks whether G has an acyclic matching of size at least ℓ. Goddard et al. [20] introduced the concept of acyclic matching, and since then, it has gained significant popularity [3,4,18,36,38]. Acyclic Matching is known to be NP-complete for perfect elimination bipartite graphs [38], star-convex bipartite graphs [36], and dually chordal graphs [36]. On the positive side, Acyclic Matching is known to be polynomial-time solvable for chordal graphs [4] and bipartite permutation graphs [38]. Fürst and Rautenbach [18] characterized the graphs for which every maximum matching is acyclic and gave linear-time algorithms to compute a maximum acyclic matching in graph classes such as P_4-free graphs and $2P_3$-free graphs. With respect to approximation, Panda and Chaudhary [36] showed that Acyclic Matching is hard to approximate within factor $n^{1-\epsilon}$ for every $\epsilon > 0$ unless $\mathsf{P} = \mathsf{NP}$. Apart from that, Baste et al. [4] showed that finding a maximum cardinality 1-degenerate matching in a graph G is equivalent to finding a maximum acyclic matching in G.

From the viewpoint of Parameterized Complexity, Hajebi and Javadi [24] studied Acyclic Matching. See Table 1 for a tabular view of known parameterized results concerning acyclic matchings.

Recall that a matching M is an induced matching if $G[V_M]$ is a disjoint union of $K_2's$. In fact, note that every induced matching is an acyclic matching, but the converse need not be true. Given a graph G and $\ell \in \mathbb{N}$, Induced Matching asks whether G has an induced matching of size at least ℓ. Induced Matching has been studied extensively due to its various applications and relations with other graph problems [7,11,15,25–27,29,34,37,39,42]. Given a graph G and a positive integer ℓ, Induced Matching Below Triviality (IMBT) asks whether G has an induced matching of size at least ℓ with the parameter $k = \frac{n}{2} - \ell$, where $n = |V(G)|$. IMBT has been studied in the literature,

Table 1. Overview of parameterized results for Acyclic Matching. Here, c_4 and tw denote the number of cycles with length four and the treewidth of the input graph, respectively.

	Parameter	**Result**
1	size of the matching	W[1]-hard on bipartite graphs [24]
2	size of the matching	FPT on line graphs [24]
3	size of the matching	FPT on every proper minor-closed class of graphs [24]
4	size of the matching plus c_4	FPT [24]
5	tw	FPT [9,24]

albeit under different names: Moser and Thilikos [35] gave an algorithm to solve IMBT in $\mathcal{O}(9^k \cdot n^{\mathcal{O}(1)})$ time. Subsequently, Xiao and Kou [41] developed an algorithm running in $\mathcal{O}(3.1845^k \cdot n^{\mathcal{O}(1)})$ time. Induced Matching with respect to some new *below guarantee* parameters have also been studied recently [28].

Next, recall that a matching M is a uniquely restricted matching if $G[V_M]$ has exactly one perfect matching. Note by definition that an acyclic matching is always a uniquely restricted matching, but the converse need not be true. Given a graph G and a positive integer ℓ, Uniquely Restricted Matching asks whether G has a uniquely restricted matching of size at least ℓ. The concept of uniquely restricted matching, motivated by a problem in Linear Algebra, was introduced by Golumbic et al. [22]. Some more results related to uniquely restricted matchings can be found in [5,6,8,17,20,22].

Our Contributions and Methods. Proofs of the results marked with a (∗) are omitted due to lack of space and can be found in the full version (see [10]).

In Sect. 3, we show that it is very unlikely that there exists any FPT-approximation algorithm for Acyclic Matching that approximates it to a constant factor. Our simple reduction also asserts the FPT-inapproximability of Induced Matching and Uniquely Restricted Matching. In particular, we have the following theorem.

Theorem 1. *Assuming* W[1] $\nsubseteq$ FPT*, there is no* FPT *algorithm that approximates any of the following to any constant when the parameter is the size of the matching:* (1) Acyclic Matching*;* (2) Induced matching*;* (3) Uniquely Restricted Matching.

If $\mathcal{R} \in$ {acyclic, induced, uniquely restricted}, then the $\mathcal{R}$ *matching number* of G is the maximum cardinality of an $\mathcal{R}$ matching among all $\mathcal{R}$ matchings in G. We denote by AM(G), IM(G), and URM(G), the *acyclic matching number*, the *induced matching number*, and the *uniquely restricted matching number*, respectively, of G. Also, we denote by IS(G) the *independence number* of G.

In order to prove Theorem 1, we exploit the relationship of IS(G) with the following: AM(G), IM(G), and URM(G). While the relationship of AM(G) and IM(G) with IS(G) is straightforward, extra efforts are needed to state a similar relation between URM(G) and IS(G), which may be of independent interest as well. Also, we note that Induced Matching is already known to be W[1]-hard, even for bipartite graphs [34]. However, for Uniquely Restricted Matching, Theorem 1 is the first to establish the W[1]-hardness of the problem.

In Sect. 4, we consider below-guarantee parameters for Acyclic Matching, i.e., parameterizations of the form $\mathsf{UB} - \ell$ for an upper bound UB on the size of any acyclic matching in G. Since the inception of below-guarantee (and above-guarantee) parameters, there has been significant progress in the area concerning graph problems parameterized by such parameters (see a recent survey paper by Gutin and Mnich [23]). It is easy to observe that in a graph G, the size of any acyclic matching is at most $\frac{n}{2}$, where $n = |V(G)|$. This observation yields the definition of Acyclic Matching Below Triviality, which, given a graph G with $|V(G)| = n$ and a positive integer ℓ, asks whether G has an acyclic matching of size at least ℓ. Here, the parameter is $k = n - 2\ell$.

Note that for ease of exposition, we are using the parameter $n - 2\ell$ instead of $\frac{n}{2} - \ell$. Next, we have the following theorem.

Theorem 2. *There exists a randomized algorithm that solves* AMBT *with success probability at least* $1 - \frac{1}{e}$ *in* $10^k \cdot n^{\mathcal{O}(1)}$ *time.*

The proof of Theorem 2 is the most technical part of our paper. The initial intuition in proving Theorem 2 was to take a cue from the existing literature on similar problems (which are quite a few) like Induced Matching Below Triviality. However, induced matchings have many nice properties, which do not hold for acyclic matchings in general, and thus make it more difficult to characterize edges or vertices that should necessarily belong to an optimal solution of Acyclic Matching Below Triviality. However, there is one nice property about Acyclic Matching, and it is that it is closely related to the Feedback Vertex Set problem as follows. Given an instance (G, ℓ) of Acyclic Matching, where $n = |V(G)|$, G has an acyclic matching of size ℓ if and only if there exists a (not necessarily minimal) feedback vertex set, say, X, of size $n - 2\ell$ such that $G - X$ has a perfect matching. We use randomization techniques to find a (specific) feedback vertex set of the input graph and then check whether the remaining graph (which is a forest) has a matching of the desired size or not. Note that the classical randomized algorithm for Feedback Vertex Set (which is a classroom problem now) [12] cannot be applied "as it is" here. In other words, since our ultimate goal is to find a matching, we cannot get rid of the vertices or edges of the input graph by applying some reduction rules "as they are". Instead, we can do something meaningful if we store the information about everything we delete or modify, and maintain some specific property (called Property R by us) in our graph. We also use our own lemma (Lemma 7) to pick a vertex in our desired feedback vertex set with high probability. Then, using an algorithm for Max Weight Matching (see Sect. 2), we compute an acyclic matching of size at least ℓ, if such a matching exists.

In light of Theorem 2, for AMBT, we ask if Acyclic Matching is FPT for natural parameters smaller than $\frac{n}{2} - \ell$. Here, an obvious upper bound on $\mathsf{AM(G)}$ is the matching number of G. Thus, we consider the below-guarantee parameter

$\mathsf{MM(G)} - \ell$, where $\mathsf{MM(G)}$ denotes the matching number of G, which yields Acyclic Matching Below Maximum Matching (AMBMM). Given a graph G and a positive integer ℓ, AMBMM asks whether G has an acyclic matching of size at least ℓ. Note that the parameter in AMBMM is $k = \mathsf{MM(G)} - \ell$.

In [18], Fürst and Rautenbach showed that deciding whether a given bipartite graph of maximum degree at most 4 has a maximum matching that is also an acyclic matching is NP-hard. Thus, for $k = 0$, the AMBMM problem is NP-hard, and we have the following result.

Corollary 1. AMBMM *is* para-NP-hard *even for bipartite graphs of maximum degree at most 4.*

Next, consider the following lemma.

Lemma 1. ($*$) *If G has an acyclic matching M of size ℓ, then G has an independent set of size at least ℓ. Moreover, given M, the independent set is computable in polynomial time.*

By Lemma 1, for any graph G, $\mathsf{IS}(G) \geq \mathsf{AM}(G)$, which yields the Acyclic Matching Below Independent Set (AMBIS) problem. Given a graph G and a positive integer ℓ, AMBIS asks whether G has an acyclic matching of size at least ℓ. Note that the parameter in AMBIS is $k = \mathsf{IS(G)} - \ell$.

In [36], Panda and Chaudhary showed that Acyclic Matching is hard to approximate within factor $n^{1-\epsilon}$ for any $\epsilon > 0$ unless $\mathsf{P} = \mathsf{NP}$ by giving a polynomial-time reduction from Independent Set. We notice that with a more careful analysis of the proof, the reduction given in [36] can be used to show that AMBIS is NP-hard for $k = 0$. Therefore, we have the following result.

Corollary 2. AMBIS *is* para-NP-hard*.*

We note that Hajebi and Javadi [24] showed that Acyclic Matching parameterized by treewidth (tw) is FPT by using Courcelle's theorem. Since $\mathsf{tw}(G) \leq \mathsf{vc}(G)$[1], this result immediately implies that Acyclic Matching is FPT with respect to the parameter vc. We complement this result by showing that it is unlikely for Acyclic Matching to admit a polynomial kernel when parameterized not only by vc, but also when parameterized jointly by vc plus the size of the acyclic matching. In particular, we have the following.

Theorem 3. Acyclic Matching *does not admit a polynomial kernel when parameterized by vertex cover number plus the size of the matching unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$*.*

Parameterization by the size of a *modulator* (a set of vertices in a graph whose deletion results in a graph that belongs to a well-known and easy-to-handle graph class) is another natural choice of investigation. We observe that with only a minor modification in our construction (in the proof of Theorem 3), we derive the following result.

[1] We denote by $\mathsf{vc}(G)$, the *vertex cover number* of a graph G.

Theorem 4. Acyclic Matching *does not admit a polynomial kernel when parameterized by the vertex deletion distance to clique plus the size of the matching unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

Due to space constraints, the section concerning negative kernelization results has been deferred to the full version of the paper (see [10]).

2 Preliminaries

For $k \in \mathbb{N}$, let $[k] = \{1, 2, \ldots, k\}$. We consider only simple and undirected graphs unless stated otherwise. For a graph G, let $V(G)$ denote its vertex set, and $E(G)$ denote its edge set. For a graph G, the subgraph of G induced by $S \subseteq V(G)$ is denoted by $G[S]$, where $G[S] = (S, E_S)$ and $E_S = \{xy \in E(G) : x, y \in S\}$. Given a matching M, a vertex $v \in V(G)$ is *M-saturated* if v is incident on an edge of M. Given a graph G and a matching M, we use the notation V_M to denote the set of M-saturated vertices. The *matching number* of G is the maximum cardinality of a matching among all matchings in G, and we denote it by $\mathsf{MM}(G)$. The edges in a matching M are *matched edges*. A matching that saturates all the vertices of a graph is a *perfect matching*. If $uv \in M$, then v is the *M-mate* of u and vice versa. Given a weighted graph G with a weight function $\mathsf{w} : E(G) \rightarrow \mathbb{R}$ and a weight $\mathsf{W} \in \mathbb{R}$, Max Weight Matching asks whether G has a matching with weight at least W in G, and can be solved in $\mathcal{O}(m\sqrt{n}\log(N))$ time, where $m = |E(G)|$, $n = |V(G)|$, and the weights are integers lying in $[0, N]$ [14].

The *degree* of a vertex v, denoted by $d_G(v)$, is $|N_G(v)|$. When there is no ambiguity, we do not use the subscript G. The minimum degree of graph G is denoted by $\delta(G)$. We use the notation $\widehat{d}(u, v)$ to represent the distance between two vertices u and v in a graph G. We denote by $\mathsf{IS}(G)$, the *independence number* of a graph G. Given a (multi)graph G and a positive integer ℓ, Feedback Vertex Set asks whether there exists a feedback vertex set in G of size at most ℓ. A *factor* of a graph G is a spanning subgraph of G (a subgraph with vertex set $V(G)$). A *k-factor* of a graph is a k-regular subgraph of order n. In particular, a 1-*factor* is a perfect matching. Let K_n denote a *complete graph* with n vertices. The size of a clique modulator of minimum size is known as the *vertex deletion distance to a clique*. For a graph G and a set $X \subseteq V(G)$, we use $G - X$ to denote $G[V(G) \backslash X]$.

Standard notions in Parameterized Complexity not explicitly defined here can be found in [12,13]. In the framework of Parameterized Complexity, each instance of a problem Π is associated with a positive integer *parameter k*. A parameterized problem Π is *fixed-parameter tractable* (FPT) if there is an algorithm that, given an instance (I, k) of Π, solves it in time $f(k) \cdot |I|^{\mathcal{O}(1)}$, for some computable function $f(\cdot)$. Due to space constraints, the definition of FPT-*approximation algorithm* has been deferred to the full version (see [10]).

3 FPT-Inapproximation Results

To obtain our result, we need the following proposition.

Proposition 1 **([30]).** *Assuming* $\mathsf{W}[1] \nsubseteq \mathsf{FPT}$*, there is no* FPT*-algorithm that approximates* Clique *to any constant.*

From Proposition 1, we derive the following corollary.

Corollary 3. *Assuming* $\mathsf{W}[1] \nsubseteq \mathsf{FPT}$*, there is no* FPT*-algorithm that approximates* Independent Set *to any constant.*

Before presenting our reduction from Independent Set, we establish the relationship of $\mathsf{IS(G)}$ with the following: $\mathsf{AM(G)}$, $\mathsf{IM(G)}$, and $\mathsf{URM(G)}$, which will be critical for the arguments in the proof of our main theorem in this section.

Relation of $\mathsf{IS(G)}$ with $\mathsf{AM(G)}$, $\mathsf{IM(G)}$ and $\mathsf{URM(G)}$.

Lemma 1. ($*$) *If G has an acyclic matching M of size ℓ, then G has an independent set of size at least ℓ. Moreover, given M, the independent set is computable in polynomial time.*

Since every induced matching is also an acyclic matching, the next lemma directly follows from Lemma 1.

Lemma 2. *If G has an induced matching M of size ℓ, then G has an independent set of size at least ℓ. Moreover, given M, the independent set is computable in polynomial time.*

Proving a similar lemma for uniquely restricted matching is more complicated. To this end, we need the following notation. Given a graph G, an *even cycle* (i.e., a cycle with an even number of edges) in G is said to be an *alternating cycle* with respect to a matching M if every second edge of the cycle belongs to M. The following proposition characterizes uniquely restricted matchings in terms of alternating cycles.

Proposition 2 **([22]).** *Let G be a graph. A matching M in G is uniquely restricted if and only if there is no alternating cycle with respect to M in G.*

Our proof will also identify some bridges based on the following proposition.

Proposition 3 **([19]).** *A graph with a unique* 1*-factor has a bridge that is matched.*

Lemma 3. *If G has a uniquely restricted matching M of size ℓ, then G has an independent set of size at least $\frac{\ell+1}{2}$. Moreover, given M, the independent set is computable in polynomial time.*

Proof. Let M be a uniquely restricted matching in G of size ℓ. By the definition of a uniquely restricted matching, $G[V_M]$ has a unique perfect matching (1-factor). Let H and I be two sets. Initialize $H = G[V_M]$ and $I = \emptyset$. Next, we design an iterative algorithm, say, Algorithm Find, to compute an independent set in H with the help of Proposition 3. Algorithm Find does the following:

While H has a connected component of size at least four, go to 1.

1. Pick a bridge, say, e, in H that belongs to M, and go to 2. (The existence of e follows from Proposition 3.)
2. If e is a pendant edge, then remove e along with its endpoints from H, and store the pendant vertex incident on e to I. Else, go to 3.
3. Remove e along with its endpoints from H.

After recursively applying 1–3, Algorithm Find arbitrarily picks exactly one vertex from each of the remaining connected components and adds them to I.

Now, it remains to show that I is an independent set of H (and therefore also of G) of size at least $\frac{\ell+1}{2}$. For this purpose, we note that Algorithm Find gives rise to a recursive formula (defined below).

Let R_h denote a lower bound on the maximum size of an independent set in H of a maximum matching of size h. First, observe that if H has a matching of size one, then it is clear that H has an independent set of size at least 1 (we can pick one of the endpoints of the matched edge). Thus, $R_1 = 1$. Now, we define how to compute R_h recursively.

$$R_h = \min\{R_{h-1} + 1, \min_{1 \leq i \leq h-2} R_i + R_{h-i-1}\}. \tag{1}$$

The first term in (1) corresponds to the case where the matched bridge is a pendant edge. On the other hand, the second term in (1) corresponds to the case where the matched bridge is not a pendant edge. In this case, all the connected components have at least one of the matched edges. Next, we claim the following,

$$R_h \geq \frac{h+1}{2}. \tag{2}$$

We prove our claim by applying induction on the maximum size of a matching in H. Recall that $R_1 = 1$. Next, by the induction hypothesis, assume that (2) is true for all $k < h$. Note that since $h-1 < h$, (2) is true for $k = h-1$, i.e., $R_{h-1} \geq \frac{h}{2}$. To prove that (2) is true for $k = h$, we first assume that the first term, i.e., $R_{h-1}+1$ gives the minimum in (1). In this case, $R_h \geq \frac{h}{2} + 1 = \frac{h+2}{2} \geq \frac{h+1}{2}$. Next, assume that the second term gives the minimum in (1) for some $i', 1 \leq i' \leq h-2$. In this case, note that $i' < h$ and $h - i' - 1 \leq h - 2 < h$, and thus, by the induction hypothesis, (2) holds for both $R_{i'}$ and $R_{h-i'-1}$. Therefore, $R_h \geq \frac{i'+1}{2} + \frac{h-i'}{2} = \frac{h+1}{2}$. □

Remark 1. Throughout this section, let Restricted $\in$ {Acyclic, Induced, Uniquely Restricted} and $\mathcal{R} \in$ {acyclic, induced, uniquely restricted}.

By Lemmas 1–3, we have the following corollary.

Corollary 4. *If a graph G has an $\mathcal{R}$ matching M of size ℓ, then G has an independent set of size at least $\frac{\ell+1}{2}$. Moreover, given M, the independent set is computable in polynomial time.*

Now, consider the following construction.

Construction. Given a graph G, where $V(G) = \{v_1, \ldots, v_n\}$, we construct a graph $H = \mathsf{reduce}(G)$ as follows. Let G^1 and G^2 be two copies of G. Let $V^1 = \{v_1^1, \ldots, v_n^1\}$ and $V^2 = \{v_1^2, \ldots, v_n^2\}$ denote the vertex sets of G^1 and G^2, respectively. Let $V(H) = V(G^1) \cup V(G^2)$ and $E(H) = E(G^1) \cup E(G^2) \cup \{v_i^1 v_i^2 : i \in [n]\}$. Let us call the edges between G^1 and G^2 *vertical edges.*

Lemma 4. ($*$) *Let G and H be as defined above. If H has $\mathcal{R}$ matching of the form $M = \{v_1^1 v_1^2, \ldots, v_p^1 v_p^2\}$, then $I_M = \{v_1, \ldots, v_p\}$ is an independent set of G.*

Hardness of Approximation Proof. To prove Theorem 1, we first suppose that Restricted Matching can be approximated within a ratio of $\alpha > 1$, where $\alpha \in \mathbb{R}^+$ is a constant, by some FPT-approximation algorithm, say, Algorithm $\mathcal{A}$. By the definition of $\mathcal{A}$, the following is true:

i) If H does not have an $\mathcal{R}$ matching of size ℓ, then the output of $\mathcal{A}$ is arbitrary (indicating that (H, ℓ) is a No-instance).
ii) If H has an $\mathcal{R}$ matching of size ℓ, then $\mathcal{A}$ returns an $\mathcal{R}$ matching, say, X, such that $|X| \geq \frac{\ell}{\alpha}$ and $|X| \leq \mathsf{opt(H)}$, where $\mathsf{opt(H)}$ denotes the optimal size of an $\mathcal{R}$ matching in H.

Next, we propose an FPT-approximation algorithm, say, Algorithm $\mathcal{B}$, to compute an FPT-approximate solution for Independent Set as follows. Given an instance (G, ℓ) of Independent Set, Algorithm $\mathcal{B}$ first constructs an instance (H, ℓ) of Restricted Matching, where $H = \mathsf{reduce}(G)$. Algorithm $\mathcal{B}$ then solves (H, ℓ) by using Algorithm $\mathcal{A}$. If Algorithm $\mathcal{A}$ returns an $\mathcal{R}$ matching X, then Algorithm $\mathcal{B}$ returns an independent set of size at least $\frac{|X|}{8} (\geq \frac{\ell}{8\alpha})$. Else, the output is arbitrary.

Now, it remains to show that Algorithm $\mathcal{B}$ is an FPT-approximation algorithm for Independent Set with an approximation factor of $\beta > 1$, where $\beta \in \mathbb{R}^+$, which we will show with the help of the following two lemmas.

Lemma 5. ($*$) *Algorithm $\mathcal{B}$ approximates* Independent Set *within a constant factor $\beta > 1$, where $\beta \in \mathbb{R}^+$.*

Lemma 6. ($*$) *Algorithm $\mathcal{B}$ runs in* FPT *time.*

By Corollary 4 and Lemmas 5 and 6, we have Theorem 1.

4 FPT Algorithm for AMBT

In this section, we prove that AMBT is FPT by giving a randomized algorithm that runs in time $10^k \cdot n^{\mathcal{O}(1)}$, where $n = |V(G)|$ and $k = n - 2\ell$.

First, we define some terminology that is crucial for proceeding further in this section. A graph G *has property* R if $\delta(G) \geq 2$ and no two adjacent vertices of G have degree exactly 2. A path P is a *maximal degree-2 path* in G if: (i) it has at least two vertices, (ii) the degree of each vertex in P (including the endpoints) is exactly 2, and (iii) it is not contained in any other degree-2 path.

If we replace a maximal degree-2 path P with a single vertex, say, v_P, of degree exactly 2 (in G), then we call this operation Path-Replacement(P,v_P) (note that the neighbors of v_P are the neighbors of the endpoints of P that do not belong to P). Furthermore, we call the newly introduced vertex (that replaces a maximal degree-2 path in G) *virtual vertex*. Note that if both endpoints of P have a common neighbor, then this gives rise to multiple edges in G. Next, if there exists a cycle, say, C, of length $p \geq 2$ such that the degree of each vertex in C is exactly 2 (in G), then the Path-Replacement operation also identifies such cycles and replaces each of them with a virtual vertex having a self-loop, and the corresponding maximal degree-2 path, in this case, consists of all the vertices of C. Therefore, it is required for us to consider AMBT in the more general setting of multigraphs, where the graph obtained after applying the Path-Replacement operation may contain multiple edges and self-loops. We also note that multiple edges and self-loops are cycles.

We first present a lemma (Lemma 7) that is crucial to prove Theorem 2.

Lemma 7. (∗) *Let G be a graph on n vertices with the property* R. *Then, for every feedback vertex set X of G, more than $\frac{|E(G)|}{5}$ of the edges of G have at least one endpoint in X.*

Now, consider Algorithm 1.

Observe that the task of Algorithm 1 is first to modify an input graph G to a graph that has property R. By abuse of notation, we call this modified graph G. Since G is non-empty and G has property R, then G definitely has a cycle, and by Lemma 7, with probability at least $\frac{1}{10}$, we pick one vertex, say v, that belongs to a specific feedback vertex set of G. We store this vertex in a set $\widehat{X}$. After removing v from G, we also decrease k by 1. We again repeat the process until either the graph becomes empty or k becomes non-positive while there are still some cycles left in the graph; we return No in the latter case, and the sets A, Z, and $\widehat{X}$ in the former case.

Given an input graph G and a positive integer k, let $\widehat{X}$ be a virtual feedback vertex set returned by Algorithm 1. Note that the set $\widehat{X}$ contains a combination of virtual vertices and the vertices from the set $V(G)$. Let $\widehat{V} \subseteq \widehat{X}$ be the set of virtual vertices. If $v_P \in \widehat{V}$, then there exists some maximal degree-2 path P such that $(v_P, P) \in A$. If all the vertices in P are from $V(G)$, then we say that the set of vertices in P is *safe* for v_P. On the other hand, if the path P contains some virtual vertices, then note that there exist maximal degree-2 paths corresponding to these virtual vertices as well. In this case, we recursively replace the virtual vertices present in P with their corresponding maximal degree-2 paths until we obtain a set that contains vertices from $V(G)$ only, and we say that these vertices are safe for v_P. The process of obtaining a set of safe vertices corresponding to virtual vertices is shown in Fig. 1. Note that, for the graph shown in Fig. 1 (i), if $\widehat{X} = \{v_{P_3}, v_{P_4}, v_{P_6}\}$ is a virtual feedback vertex set returned by Algorithm 1 corresponding to $X = \{i, k, a\}$, then the safe set corresponding to v_{P_3} is $\{i, j\}$, corresponding to v_{P_4} is $\{k, l\}$, and corresponding to v_{P_6} is $\{a, b, c, d, e, f, g, h\}$.

Algorithm 1

```
Input: A graph $G$ and a positive integer $k$;
Output: A set $\widehat{X}$ of size at most $k$, a set $Z$, and a set $A$ or No;
Initialize $Z \leftarrow \emptyset$, $A \leftarrow \emptyset$, $\widehat{X} \leftarrow \emptyset$;
while ($V(G) \neq \emptyset$) do
    while ($\delta(G) \leq 1$) do
        Pick a vertex $v \in V(G)$ such that $d(v) \leq 1$;
        $Z \leftarrow Z \cup \{v\}$;
        $V(G) \leftarrow V(G) \setminus \{v\}$;
    while (there exists a maximal degree-2 path $P$ in $G$) do
        Path-Replacement($P$,$v_P$);
        $A \leftarrow A \cup \{(P, v_P)\}$;
    if ($k > 0$ and $G$ has a cycle) then
        if ($G$ has a self-loop at some $v$) then
            $\widehat{X} \leftarrow \widehat{X} \cup \{v\}$;
            $V(G) \leftarrow V(G) \setminus \{v\}$;
            $k \leftarrow k - 1$;
        else
            Pick an edge $e \in E(G)$;
            Pick an endpoint $v$ of $e$;
            $\widehat{X} \leftarrow \widehat{X} \cup \{v\}$;
            $V(G) \leftarrow V(G) \setminus \{v\}$;
            $k \leftarrow k - 1$;
    else if ($k \leq 0$ and $G$ has a cycle) then
        return No;
return $\widehat{X}, Z, A$;
```

Remark 2. Throughout this section, if $\widehat{X}$ is a virtual feedback vertex set returned by Algorithm 1, then let $\widehat{V} \subseteq \widehat{X}$ denote the set of virtual vertices.

Definition 1. *Let $\widehat{X}$ be a virtual feedback vertex set returned by Algorithm 1 if given as input a graph G and a positive integer k. A set $X \subseteq V(G)$ is* compatible *with $\widehat{X}$ if the following hold: (1) For every $v_P \in \widehat{V}$, X contains at least one vertex from the set of safe vertices corresponding to v_P; (2) For every $v \in \widehat{X} \setminus \widehat{V}$, v belongs to X; (3) $|X| \leq k$.*

Lemma 8. (∗) *Given a graph G and a positive integer k, if $\widehat{X}$ is a virtual feedback vertex set returned by Algorithm 1, then any set $X \subseteq V(G)$ compatible with $\widehat{X}$ is a feedback vertex set of G.*

Lemma 9. (∗) *Let G be a graph and $k \in \mathbb{N}$. Then, for any feedback vertex set X of G of size at most k, with probability at least 10^{-k}, Algorithm 1 returns a virtual feedback vertex set $\widehat{X}$ such that X is compatible with $\widehat{X}$.*

Remark 3. Let Algorithm $\mathcal{A}$ be any algorithm that solves Max Weight Matching in polynomial time.

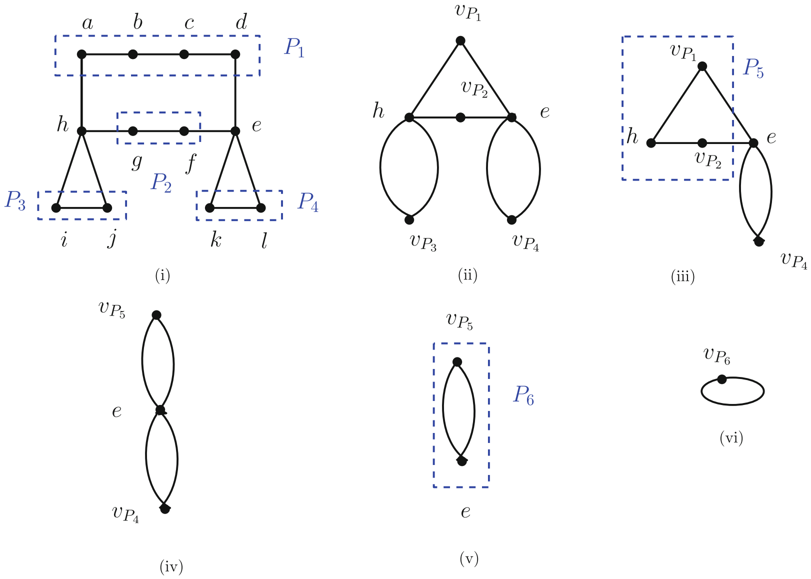


Fig. 1. After applying Path-Replacement to paths P_1-P_4 in (i), we obtain the graph in (ii), which has property R. We assume that Algorithm 1 picks v_{P_3} in $\widehat{X}$. After removing v_{P_3}, we obtain the graph shown in (iii) that has a maximal degree-2 path P_5. After applying Path-Replacement to P_5, we obtain the graph in (iv). We assume that Algorithm 1 picks v_{P_4} in $\widehat{X}$. After removing v_{P_4}, we obtain the graph shown in (v). Note that the Path-Replacement operation identifies the cycle shown in (v) as a maximal degree-2 path and replaces it with v_{P_6} with a self-loop, as shown in (vi). Algorithm 1 then picks v_{P_6} in $\widehat{X}$.

Next, consider Algorithm 2.

Lemma 10. (∗) *Let G, ℓ, $\widehat{X}$, G_W, M_W, and M be as defined in Algorithm 2. If M_W is of weight at least $\ell + |\widehat{V}| \cdot c$, then M is an acyclic matching in G of size at least ℓ.*

Lemma 11. (∗) *Let (G, ℓ) be a Yes-instance of* Acyclic Matching *with $n = |V(G)|$. Then, with probability at least $10^{2\ell - n}$, Algorithm 2 returns an acyclic matching M in G of size at least ℓ.*

Lemma 12. (∗) *Let (G, ℓ) be a No-instance of* Acyclic Matching *with $n = |V(G)|$. Then, with probability 1, Algorithm 2 returns* No.

We can improve the success probability of Algorithm 1 and thus Algorithm 2, by repeating it, say, t times, and returning a No only if we are not able to find a virtual feedback vertex set of size at most k in each of the repetitions. Clearly, due to Lemma 12, given a No-instance, even after repeating the procedure t times, we will necessarily get No as an answer. However, given a Yes-instance,

Algorithm 2

```
Input: An instance $(G, \ell)$ of Acyclic Matching with $n = |V(G)|$;
Output: An acyclic matching $M$ in $G$ of size at least $\ell$ or No;
Call Algorithm 1 with input $(G, n - 2\ell)$;
if (Algorithm 1 returns No) then
    return No;
else if (Algorithm 1 returns a virtual feedback vertex set $\widehat{X}$) then
    Initialize $G_W = G$;
    For every $v_P \in \widehat{V}$, add a vertex $w_P$ to $G_W$ and make it adjacent to every vertex
    in the safe set corresponding to $v_P$. Call all edges introduced here new edges.
    Remove all $v \in \widehat{X} \backslash \widehat{V}$ from $G_W$;
    Assign weight $c = |E(G)| + 1$ to all new edges of $G_W$ and weight 1 to all the
    remaining edges of $G_W$;
    Call Algorithm $\mathcal{A}$ with input $(G_W, \ell + |\widehat{V}| \cdot c)$;
    if (Algorithm $\mathcal{A}$ returns a matching $M_W$ of weight at least $\ell + |\widehat{V}| \cdot c$) then
        $M = \{e \in M_W :$ weight of $e$ is $1\}$;
        return $M$;
    else
        return No;
```

we return a No only if all t repetitions return an incorrect No, which, by Lemma 11, has probability at most

$$(1 - 10^{-k})^t \leq (e^{-10^{-k}})^t \leq \frac{1}{e^{10^{-k}t}}. \tag{3}$$

Note that we are using the identity $1 + x \leq e^x$ in (3). In order to obtain a constant failure probability, we take $t = 10^k$. By taking $t = 10^k$, the success probability becomes at least $1 - \frac{1}{e}$. Thus, we have the following theorem.

Theorem 1. *There exists a randomized algorithm that solves* AMBT *with success probability at least $1 - \frac{1}{e}$ in $10^k \cdot n^{\mathcal{O}(1)}$ time.*

5 Conclusion and Future Research

Based on the results given by Moser and Sikdar [34], we note that both Acyclic Matching and Uniquely Restricted Matching admit quadratic kernels (with respect to the maximum degree of the input graph) for bounded degree graphs. In fact, for Uniquely Restricted Matching, the quadratic kernel can be further improved to a linear kernel. The following is true for any maximum uniquely restricted matching.

Property $\widehat{\mathsf{P}}$: If M is a maximum uniquely restricted matching in a graph G, then for each vertex $v \in V(G)$, there exists a vertex $u \in V_M$ such that $\widehat{d}(u, v) \leq 1$.

One possible direction for future research is to seek a below-guarantee parameter smaller than the parameter $\frac{n}{2} - \ell$, so that Acyclic Matching remains

FPT. Also, it would be interesting to see if the running time in Theorem 2 can be substantially improved. Apart from that, we strongly believe that the arguments presented in this work (in Sect. 4) will be useful for other future works concerning problems where one seeks a solution that, among other properties, satisfies that it is itself, or its complement, a feedback vertex set, or, much more generally, an alpha-cover (see [16]). A more detailed conclusion can be found in the full version (see [10]).

References

1. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation: Combinatorial Optimization Problems and their Approximability properties. Springer, Heidelberg (2012)
2. Bachstein, A., Goddard, W., Lehmacher, C.: The generalized matcher game. Discret. Appl. Math. **284**, 444–453 (2020)
3. Baste, J., Fürst, M., Rautenbach, D.: Approximating maximum acyclic matchings by greedy and local search strategies. In: Kim, D., Uma, R.N., Cai, Z., Lee, D.H. (eds.) COCOON 2020. LNCS, vol. 12273, pp. 542–553. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58150-3_44
4. Baste, J., Rautenbach, D.: Degenerate matchings and edge colorings. Discret. Appl. Math. **239**, 38–44 (2018)
5. Baste, J., Rautenbach, D., Sau, I.: Uniquely restricted matchings and edge colorings. In: Bodlaender, H.L., Woeginger, G.J. (eds.) WG 2017. LNCS, vol. 10520, pp. 100–112. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68705-6_8
6. Baste, J., Rautenbach, D., Sau, I.: Upper bounds on the uniquely restricted chromatic index. J. Graph Theory **91**(3), 251–258 (2019)
7. Cameron, K.: Induced matchings. Discret. Appl. Math. **24**(1–3), 97–102 (1989)
8. Chaudhary, J., Panda, B.S.: On the complexity of minimum maximal uniquely restricted matching. Theoret. Comput. Sci. **882**, 15–28 (2021)
9. Chaudhary, J., Zehavi, M.: $\mathcal{P}$-matchings parameterized by treewidth. In: Paulusma, D., Ries, B. (eds.) WG 2023. LNCS, vol. 14093, pp. 217–231. Springer, Cham (2023)
10. Chaudhary, J., Zehavi, M.: Parameterized results on acyclic matchings with implications for related problems. arXiv preprint arXiv:2307.05446v1 (2023)
11. Cooley, O., Draganic, N., Kang, M., Sudakov, B.: Large induced matchings in random graphs. SIAM J. Discret. Math. **35**(1), 267–280 (2021)
12. Cygan, M., et al.: Parameterized Algorithms, vol. 5. Springer, Cham (2015)
13. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity, vol. 4. Springer, London (2013)
14. Duan, R., Su, H.-H.: A scaling algorithm for maximum weight matching in bipartite graphs. In Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms. SIAM, pp. 1413–1424 (2012)
15. Erman, R., Kowalik, Ł, Krnc, M., Waleń, T.: Improved induced matchings in sparse graphs. Discret. Appl. Math. **158**(18), 1994–2003 (2010)
16. Fomin, F.V., Lokshtanov, D., Misra, N., Saurabh, S.: Planar f-deletion: approximation, kernelization and optimal FPT algorithms. In: 2012 53rd Annual Symposium on Foundations of Computer Science. IEEE, pp. 470–479 (2012)
17. Francis, M.C., Jacob, D., Jana, S.: Uniquely restricted matchings in interval graphs. SIAM J. Discret. Math. **32**(1), 148–172 (2018)

18. Fürst, M., Rautenbach, D.: On some hard and some tractable cases of the maximum acyclic matching problem. Ann. Oper. Res. **279**, 291–300 (2019)
19. Gabow, H.N.: Algorithmic proofs of two relations between connectivity and the 1-factors of a graph. Discret. Math. **26**(1), 33–40 (1979)
20. Goddard, W., Hedetniemi, S.M., Hedetniemi, S.T., Laskar, R.: Generalized subgraph-restricted matchings in graphs. Discret. Math. **293**(1–3), 129–138 (2005)
21. Goddard, W., Henning, M.A.: The matcher game played in graphs. Discret. Appl. Math. **237**, 82–88 (2018)
22. Golumbic, M.C., Hirst, T., Lewenstein, M.: Uniquely restricted matchings. Algorithmica **31**, 139–154 (2001)
23. Gutin, G., Mnich, M.: A survey on graph problems parameterized above and below guaranteed values. arXiv preprint arXiv:2207.12278 (2022)
24. Hajebi, S., Javadi, R.: On the parameterized complexity of the acyclic matching problem. Theoret. Comput. Sci. **958**, 113862 (2023)
25. Kanj, I., Pelsmajer, M.J., Schaefer, M., Xia, G.: On the induced matching problem. J. Comput. Syst. Sci. **77**(6), 1058–1070 (2011)
26. Klemz, B., Rote, G.: Linear-time algorithms for maximum-weight induced matchings and minimum chain covers in convex bipartite graphs. Algorithmica **84**(4), 1064–1080 (2022)
27. Ko, C., Shepherd, F.B.: Bipartite domination and simultaneous matroid covers. SIAM J. Discret. Math. **16**(4), 517–523 (2003)
28. Koana, T.: Induced matching below guarantees: average paves the way for fixed-parameter tractability. In: Proceedings of the 40th International Symposium on Theoretical Aspects of Computer Science (STACS), pp. 39:1–39:21 (2023)
29. Kowalik, L., Luzar, B., Skrekovski, R.: An improved bound on the largest induced forests for triangle-free planar graphs. Discrete Math. Theor. Comput. Sci. **12**(1), 87–100 (2010)
30. Lin, B.: Constant approximating k-clique is w [1]-hard. In: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, pp. 1749–1756 (2021)
31. Lovász, L., Plummer, M.D.: Matching Theory. Annals of Discrete Mathematics, vol. 29. North-Holland, Amsterdam (1986)
32. Manlove, D.: Algorithmics of Matching Under Preferences, vol. 2. World Scientific, Singapore (2013)
33. Micali, S., Vazirani, V.V.: An o($\sqrt{|V|}|e|$) algoithm for finding maximum matching in general graphs. In 21st Annual Symposium on Foundations of Computer Science (sfcs 1980). IEEE, pp. 17–27 (1980)
34. Moser, H., Sikdar, S.: The parameterized complexity of the induced matching problem. Discret. Appl. Math. **157**(4), 715–727 (2009)
35. Moser, H., Thilikos, D.M.: Parameterized complexity of finding regular induced subgraphs. J. Discrete Algorithms **7**(2), 181–190 (2009)
36. Panda, B.S., Chaudhary, J.: Acyclic matching in some subclasses of graphs. Theoret. Comput. Sci. **943**, 36–49 (2023)
37. Panda, B.S., Pandey, A., Chaudhary, J., Dane, P., Kashyap, M.: Maximum weight induced matching in some subclasses of bipartite graphs. J. Comb. Optim. **40**(3), 713–732 (2020)
38. Panda, B.S., Pradhan, D.: Acyclic matchings in subclasses of bipartite graphs. Discrete Math. Algorithms Appl. **4**(04), 1250050 (2012)
39. Stockmeyer, L.J., Vazirani, V.V.: Np-completeness of some generalizations of the maximum matching problem. Inf. Process. Lett. **15**(1), 14–19 (1982)
40. Vizing, V.G.: On an estimate of the chromatic class of a p-graph. Diskret Analiz **3**, 25–30 (1964)

41. Xiao, M., Kou, S.: Parameterized algorithms and kernels for almost induced matching. Theoret. Comput. Sci. **846**, 103–113 (2020)
42. Zito, M.: Induced matchings in regular graphs and trees. In: Widmayer, P., Neyer, G., Eidenbenz, S. (eds.) WG 1999. LNCS, vol. 1665, pp. 89–101. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-46784-X_10

$\mathcal{P}$-Matchings Parameterized by Treewidth

Juhi Chaudhary(✉) and Meirav Zehavi

Department of Computer Science, Ben-Gurion University of the Negev,
BeerSheba, Israel
juhic@post.bgu.ac.il, meiravze@bgu.ac.il

Abstract. A *matching* is a subset of edges in a graph G that do not share an endpoint. A matching M is a $\mathcal{P}$*-matching* if the subgraph of G induced by the endpoints of the edges of M satisfies property $\mathcal{P}$. For example, if the property $\mathcal{P}$ is that of being a matching, being acyclic, or being disconnected, then we obtain an *induced matching*, an *acyclic matching*, and a *disconnected matching*, respectively. In this paper, we analyze the problems of the computation of these matchings from the viewpoint of Parameterized Complexity with respect to the parameter *treewidth*.

Keywords: Matching · Treewidth · Parameterized Algorithms · Exponential Time Hypothesis

1 Introduction

Matching in graphs is a central topic of Graph Theory and Combinatorial Optimization [28]. Matchings possess both theoretical significance and practical applications, such as the assignment of new physicians to hospitals, students to high schools, clients to server clusters, kidney donors to recipients [29], and so on. Additionally, the field of competitive optimization games on graphs has witnessed substantial growth in recent years, where matching serves as a valuable tool for determining optimal solutions or bounds in such games [1,19]. The study of matchings is closely related to the concept of *edge colorings* as well [2,5,38], and the minimum number of matchings into which the edge set of a graph G can be partitioned is known as the *chromatic index* of G [38].

Given a graph G, Maximum Matching is the problem of finding a matching of maximum size (number of edges) in G. A matching M is said to be a $\mathcal{P}$*-matching* if $G[V_M]$ (the subgraph of G induced by the endpoints of edges in M) has property $\mathcal{P}$, where $\mathcal{P}$ is some graph property. The problem of deciding whether a graph admits a $\mathcal{P}$-matching of a given size has been investigated for many different properties [4,16,18,20,34,37]. If the property $\mathcal{P}$ is that of being a graph, a disjoint union of $K_2's$, a forest, a connected graph, a disconnected graph, or having a unique perfect matching, then a $\mathcal{P}$-matching is a *matching* [30], an

The authors are supported by the European Research Council (ERC) project titled PARAPATH (101039913).

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 217–231, 2023.
https://doi.org/10.1007/978-3-031-43380-1_16

induced matching [37], an *acyclic matching* [18], a *connected matching*[1] [18], a *disconnected matching*[2] [18], and a *uniquely restricted matching* [20], respectively. Notably, only the optimization problem corresponding to matching [30] and connected matching [18] are polynomial-time solvable for a general graph, while the decision problems corresponding to other above-mentioned variants of matching are NP-complete [18,20,37].

Given a graph G and a positive integer ℓ, the Induced Matching problem asks whether G has an induced matching of size at least ℓ. The concept of induced matching was introduced by Stockmeyer and Vazirani as the "risk-free" marriage problem in 1982 [37]. Since then, this concept, and the corresponding Induced Matching problem, have been studied extensively due to their wide range of applications and connections to other graph problems [8,12,23,25,31,37]. Similarly, the Acyclic Matching problem considers a graph G and a positive integer ℓ, and asks whether G contains an acyclic matching of size at least ℓ. Goddard et al. [18] introduced the concept of acyclic matching, and since then, it has gained significant popularity in the literature [2,3,17,34]. For a fixed $c \in \mathbb{N}$, a matching M is *c-disconnected* if $G[V_M]$ has at least c connected components. In the c-Disconnected Matching problem, given a graph G and a positive integer ℓ, we seek to determine if G contains a c-disconnected matching of size at least ℓ. In the c-Disconnected Matching problem, if c is a part of the input, then the problem that arises is known as the Disconnected Matching problem. Goddard et al. [18] introduced the concept of disconnected matching and asked about the complexity of determining the maximum size of a matching whose vertex set induces a disconnected graph, which is a restricted version of c-disconnected matching studied in this paper.

The parameter considered in this paper is *treewidth*, a structural parameter that indicates how much a graph resembles a tree. Robertson and Seymour introduced the notion of treewidth in their celebrated work on graph minors [35], and since then, over 4260 papers on google scholar consider treewidth as a parameter in the context of Parameterized Complexity. In practice also, graphs of bounded treewidth appear in many different contexts; for example, many probabilistic networks appear to have small treewidth [7]. Thus, concerning the problems studied in this paper, after the *solution size*, treewidth is one of the most natural parameters. In fact, many of the problems investigated in this paper have already been analyzed with respect to treewidth as the parameter.

Related Work. In what follows, we present a brief survey of algorithmic results concerning the variants of matchings discussed in this paper.

Induced Matching: From the viewpoint of Parameterized Complexity, in [31], Moser and Sikdar showed that Induced Matching is fixed-parameter tractable (FPT) when parameterized by treewidth by developing an $\mathcal{O}(4^{\mathrm{tw}} \cdot n)$-time dynamic programming algorithm. In the same paper, when the parameter

[1] This name is also used for a different problem where we are asked to find a matching M such that every pair of edges in M has a common edge [9].

[2] In this paper, we are using a different (more general) definition for disconnected matching than the one mentioned in [18].

is the size of the matching ℓ, Induced Matching was shown to be FPT for line graphs, planar graphs, bounded-degree graphs, and graphs of girth at least 6 that include C_4-free graphs[3]. On the other hand, for the same parameter, that is, ℓ, the problem is W[1]-hard for bipartite graphs [31]. Song [36] showed that given a Hamiltonian cycle in a Hamiltonian bipartite graph, Induced Matching is W[1]-hard with respect to ℓ and cannot be solved in time $n^{o(\sqrt{\ell})}$ unless $\mathsf{W}[1] = \mathsf{FPT}$, where $n = |V(G)|$. Induced Matching with respect to *below guarantee* parameterizations have also been studied [26,32,39].

Acyclic Matching: From the viewpoint of Parameterized Complexity, Hajebi and Javadi [22] showed that Acyclic Matching is FPT when parameterized by treewidth using Courcelle's theorem. Furthermore, they showed that the problem is W[1]-hard on bipartite graphs when parameterized ℓ. However, under the same parameter, the authors showed that the problem is FPT for line graphs, C_4-free graphs, and every proper minor-closed class of graphs. In the same paper, Acyclic Matching was shown to be FPT when parameterized by the size of the matching plus the number of cycles of length four in the given graph. Some more results concerning acyclic matchings can be found in [2,11,17,34].

c-Disconnected Matching and Disconnected Matching: For every fixed integer $c \geq 2$, c-Disconnected Matching is known to be NP-complete even for bounded diameter bipartite graphs [21]. On the other hand, for $c = 1$, c-Disconnected Matching is the same as Maximum Matching, which is known to be polynomial-time solvable [30]. Regarding disconnected matchings, Disconnected Matching is NP-complete for chordal graphs [21] and polynomial-time solvable for interval graphs [21]. Also, since Induced Matching is a special case of Disconnected Matching, we note that Disconnected Matching is NP-hard for every graph class for which Induced Matching is NP-hard [21].

From the viewpoint of Parameterized Complexity, Gomes et al. [21] proved that for graphs with a polynomial number of minimal separators, Disconnected Matching parameterized by the number of connected components, belongs to XP. Furthermore, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, Disconnected Matching does not admit a polynomial kernel when parameterized by vertex cover (vc) plus ℓ nor when parameterized by the vertex deletion distance to clique plus ℓ. In the same paper, the authors also proved that Disconnected Matching is FPT when parameterized by treewidth (tw). They used the standard dynamic programming technique, and the running time of their algorithm is $\mathcal{O}(8^{\mathrm{tw}} \cdot \eta_{\mathrm{tw}+1}^{3} \cdot n^2)$, where η_i is the i-th Bell number (the *Bell number* η_i counts the number of different ways to partition a set that has exactly i elements).

Our Contribution and Methods. In this paper, we consider the parameter to be the treewidth of the input graph, and as is customary in the field, we suppose that the input also consists of a tree decomposition $\mathcal{T} = (\mathbb{T}, \{\mathcal{B}_x\}_{x \in V(\mathbb{T})})$ of width tw of the input graph. Proofs of the results marked with a $(*)$ are omitted and can be found in the full version (see [10]).

[3] Here, C_n denotes a cycle on n vertices.

First, we present a $3^{\text{tw}} \cdot \text{tw}^{\mathcal{O}(1)} \cdot n$ time algorithm for Induced Matching in graphs with n vertices, improving upon the $\mathcal{O}(4^{\text{tw}} \cdot n)$ time bound by Moser and Sikdar [31]. For this purpose, we use a nice tree decomposition that satisfies the "deferred edge property" (defined in Sect. 2) and the fast subset convolution (see Sect. 2) for the join nodes. Due to space constraints, this section has been deferred to the full version of the paper (see [10]).

Theorem 1. Induced Matching *can be solved in* $3^{\text{tw}} \cdot \text{tw}^{\mathcal{O}(1)} \cdot n$ *time by a deterministic algorithm.*

In Sect. 3, we present a $6^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ time algorithm for Acyclic Matching in graphs with n vertices, improving the result by Hajebi and Javadi [22], who proved that Acyclic Matching parameterized by tw is FPT. They used Courcelle's theorem, which is purely theoretical, and thus the hidden parameter dependency in the running time is huge (a tower of exponents). To develop our algorithm, we use the *Cut & Count* method introduced by Cygan et al. [13] in addition to the fast subset convolution. The Cut & Count method allows us to deal with connectivity-type problems through randomization; here, randomization arises from the usage of the *Isolation Lemma* (see Sect. 2).

Theorem 2. Acyclic Matching *can be solved in* $6^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ *time by a randomized algorithm. The algorithm cannot give false positives and may give false negatives with probability at most* $\frac{1}{3}$*.*

In Sect. 4, we present a $(3c)^{\text{tw}} \cdot \text{tw}^{\mathcal{O}(1)} \cdot n$ time algorithm for c-Disconnected Matching in graphs with n vertices. We use dynamic programming along with the fast subset convolution for the join nodes. This resolves an open question by Gomes et al. [21], who asked whether c-Disconnected Matching can be solved in a single exponential time with vertex cover (vc) as the parameter. Since for any graph G, $\text{tw}(G) \leq \text{vc}(\text{G})$, we answer their question in the affirmative.

Theorem 3. *For a fixed positive integer* $c \geq 2$*,* c-Disconnected Matching *can be solved in* $(3c)^{\text{tw}} \cdot \text{tw}^{\mathcal{O}(1)} \cdot n$ *time by a deterministic algorithm.*

In Sect. 5, we present a lower bound for the time complexity of Disconnected Matching, proving that for any choice of a constant c, an $\mathcal{O}(c^{\text{tw}} \cdot n)$-time algorithm for Disconnected Matching is unlikely. In fact, we prove that even an $\mathcal{O}(c^{\text{pw}} \cdot n)$-time algorithm is not possible, where pw is the pathwidth of the graph which is bounded from below by the treewidth.

Theorem 4. *Assuming the Exponential Time Hypothesis, there is no* $2^{o(\text{pw} \log \text{pw})} \cdot n^{\mathcal{O}(1)}$*-time algorithm for* Disconnected Matching*.*

2 Preliminaries

Graph-Theoretic Notations and Definitions. For a graph G, let $V(G)$ denote its vertex set and $E(G)$ denote its edge set. Given a matching M, a vertex $v \in V(G)$ is *M-saturated* if v is incident on an edge of M. Given a graph G and a matching M, let V_M denote the set of M-saturated vertices and $G[V_M]$ denote the subgraph of G induced by V_M. A matching that saturates all the vertices of a graph is a *perfect matching.* If $uv \in M$, then v is the *M-mate* of u, and vice versa. Standard graph-theoretic terms not defined here can be found in [15].

A *cut* of a set $X \subseteq V(G)$ is a pair (X_l, X_r) with $X_l \cap X_r = \emptyset$ and $X_l \cup X_r = X$, where X is an arbitrary subset of $V(G)$. When X is immaterial, we do not mention it explicitly. A cut (X_l, X_r) is *consistent* in a subgraph H of G if $u \in X_l$ and $v \in X_r$ implies $uv \notin E(H)$. For a graph G, let *cc*(G) denote the number of connected components of G. For a graph G, a *coloring* on a set $X \subseteq V(G)$ is a function $f : X \rightarrow S$, where S is any set and the elements of S are called *colors.*

Algebraic Definitions. For a set X, let 2^X denote the set of all subsets of X. For a positive integer k, let $[k]$ denote the set $\{1, \ldots, k\}$. In the set $[k] \times [k]$, a *row* is a set $\{i\} \times [k]$ and a *column* is a set $[k] \times \{i\}$ for some $i \in [k]$. For two integers, a and b, we use $a \equiv b$ to indicate that a is even if and only if b is even. If $\mathsf{w} : U \rightarrow \{1, \ldots, N\}$, then for $S \subseteq U$, $\mathsf{w}(S) = \sum_{e \in S} \mathsf{w}(e)$. For definitions of *ring* and *semiring*, we refer the readers to any elementary book on abstract algebra. Given an integer $n > 1$, called a *modulus*, two integers a and b are *congruent modulo* n if there is an integer k such that $a - b = kn$. Note that two integers are said to be *congruent modulo* 2 if they have the same parity (that is, either both are odd or both are even). For a set S, we use the notation $|S|_2$ to denote the number of elements in set S congruent modulo 2.

Subset Convolution

Definition 1. *Let B be a finite set and $\mathbb{R}$ be a semiring. Then, the* subset convolution *of two functions $f, g : 2^B \rightarrow \mathbb{R}$ is the function $(f * g) : 2^B \rightarrow \mathbb{R}$ such that for every $Y \subseteq B$,*

$$(f * g)(Y) = \sum_{X \subseteq Y} f(X)g(Y \setminus X). \tag{1}$$

Equivalently, (1) can be written as $(f * g)(Y) = \sum_{\substack{A \cup B = Y \\ A \cap B = \emptyset}} f(A)g(B)$.

Given f and g, a direct evaluation of $f * g$ for all $X \subseteq Y$ requires $\Omega(3^n)$ semiring operations, where $n = |B|$. However, we have the following results.

Proposition 5 ([6,13])**.** *For two functions $f, g : 2^B \rightarrow \mathbb{Z}$, where $n = |B|$ and $\mathbb{Z}$ is a ring, given all the 2^n values of f and g in the input, all the 2^n values of the subset convolution of $f * g$ can be computed in $\mathcal{O}(2^n \cdot n^3)$ arithmetic operations.*

Proposition 6 ([13])**.** *For two functions $f, g : 2^B \rightarrow \{-P, \ldots, P\}$, where $n = |B|$, given all the 2^n values of f and g in the input, all the 2^n values of the subset convolution of $f * g$ over the integer max-sum semiring can be computed in time $2^n \cdot n^{\mathcal{O}(1)} \cdot \mathcal{O}(P \log P \log \log P)$.*

Treewidth. Due to space constraints, the definitions of tree decomposition, nice tree decomposition, path decomposition, treewidth, and pathwidth have been deferred to the full version (see [10]). We denote the treewidth and pathwidth of a graph by tw and pw, respectively. Standard notions in Parameterized Complexity not explicitly defined here can be found in [13].

For our problems, we want the standard nice tree decomposition to satisfy an additional property, and that is, among the vertices present in the bag of a join node, no edges have been introduced yet. To achieve this, we use another known type of node, an *introduce edge node*, which is defined as follows:

Introduce Edge Node: x has exactly one child y, and x is labeled with an edge $uv \in E(G)$ such that $u, v \in \mathcal{B}_x$ and $\mathcal{B}_x = \mathcal{B}_y$. We say that uv is *introduced* at x.

The use of introduce edge nodes enables us to add edges one by one in our nice tree decomposition. We additionally require that every edge is introduced exactly once. For every vertex $v \in V(G)$, there exists a unique highest node $t(v)$ such that $v \in B_{t(v)}$. Further, without loss of generality, assume $t(v)$ is an ancestor of $t(u)$. Our nice tree decomposition will insert the introduce edge bags (introducing edges of the form uv) between $t(u)$ and its parent in an arbitrary order. If a nice tree decomposition having introduce edge nodes satisfies these additional conditions, then we say that it exhibits the *deferred edge* property. Note that, given a tree decomposition of a graph G, where $n = |V(G)|$, a nice tree decomposition satisfying the *deferred edge* property of equal width, at most $\mathcal{O}(\text{tw} \cdot n)$ nodes, and at most $\text{tw} \cdot n$ edges can be found in $\text{tw}^{\mathcal{O}(1)} \cdot n$ time [13,24]. Furthermore, for each node x of the tree decomposition, let V_x be the union of all the bags present in the subtree of $\mathbb{T}$ rooted at x, including $\mathcal{B}_x$. Then, for each node x of the tree decomposition, define the subgraph G_x of G as follows:

$G_x = (V_x, E_x = \{e : e \text{ is introduced in the subtree of } \mathbb{T} \text{ rooted at } x\})$.

Now, consider the following two definitions.

Definition 2 (Valid Coloring). *Given a node x of $\mathbb{T}$, a coloring $d : \mathcal{B}_x \rightarrow \{0,1,2\}$ is* valid *on $\mathcal{B}_x$ if there exists a coloring $\widehat{d} : V_x \rightarrow \{0,1,2\}$ in G_x, called a* valid extension *of d, such that the following hold: (i) $\widehat{d}$ restricted to $\mathcal{B}_x$ is exactly d, (ii) The subgraph induced by the vertices colored 1 under $\widehat{d}$ has a perfect matching, (iii) Vertices colored 2 under $\widehat{d}$ must all belong to $\mathcal{B}_x$.*

Definition 3 (Correct Coloring). *Given a graph G and a set $X \subseteq V(G)$, two colorings $f_1, f_2 : X \rightarrow \{0,1,2\}$ are* correct *for a coloring $f : X \rightarrow \{0,1,2\}$ if the following conditions hold: (i) $f(v) = 0$ if and only if $f_1(v) = f_2(v) = 0$, (ii) $f(v) = 1$ if and only if $(f_1(v), f_2(v)) \in \{(1,2),(2,1)\}$, (iii) $f(v) = 2$ if and only if $f_1(v) = f_2(v) = 2$.*

In Sects. 3 and 4, we use different colors to represent the possible states of a vertex in a bag $\mathcal{B}_x$ of $\mathcal{T}$ with respect to a matching M as follows:

- **white(0):** A vertex colored 0 is not saturated by M.
- **black(1):** A vertex colored 1 is saturated by M, and the edge between the vertex and its M-mate has also been introduced in G_x.

- **gray(2):** A vertex colored 2 is saturated by M, and either its M-mate has not yet been introduced in G_x, or the edge between the vertex and its M-mate has not yet been introduced in G_x.

Definition 4 (Monte Carlo Algorithms with False Negatives). *An algorithm is* Monte Carlo with false negatives *if when queried about the existence of an object: If it answers yes, then it is true, and if it answers no, then it is correct with probability at least* $\frac{2}{3}$ *(here, the constant* $\frac{2}{3}$ *is chosen arbitrarily).*

Cut & Count Method. The Cut & Count method was introduced by Cygan et al. [14]. It is a tool for designing algorithms with a single exponential running time for problems with certain connectivity requirements. The method is mainly divided into the following two parts:

The Cut part: Let $\mathcal{S}$ denote the set of feasible solutions. Here, we relax the connectivity requirement by considering a set $\mathcal{R} \supseteq \mathcal{S}$ of possibly connected candidate solutions. Furthermore, we consider a set $\mathcal{C}$ of pairs (X, C), where $X \in \mathcal{R}$ and C is a consistent cut of X.

The Count part: Here, we compute the cardinality of $|\mathcal{C}|_2$ using a subprocedure. Non-connected candidate solutions $X \in \mathcal{R} \setminus \mathcal{S}$ cancel since they are consistent with an even number of cuts and the connected candidates $x \in \mathcal{S}$ remain.

More information on the Cut & Count method can be found in [13,14].

Isolation Lemma. Let U be a universe. A function $\mathsf{w} : U \to \mathbb{Z}$ isolates a set family $\mathcal{F} \subseteq 2^U$ if there is a unique $S' \in \mathcal{F}$ with $\mathsf{w}(S') = \min_{S \in \mathcal{F}} \mathsf{w}(S)$. Let $\mathcal{F} \subseteq 2^U$ be a set family over a universe U with $|\mathcal{F}| > 0$. For each $u \in U$, choose a weight $\mathsf{w}(u) \in \{1, 2, \ldots, N\}$ uniformly and independently at random. Then, *isolation lemma* states that $\mathsf{prob}(\mathsf{w}$ *isolates* $\mathcal{F}) \geq 1 - \frac{|U|}{N}$ [33].

3 Algorithm for Acyclic Matching

We use the Cut & Count technique along with a concept called *markers* (see [14]). Given that the Acyclic Matching problem does not impose an explicit connectivity requirement, we can proceed by selecting the (presumed) forest obtained after choosing the vertices saturated by an acyclic matching M and using the following result:

Proposition 7 ([14]). *A graph G with n vertices and m edges is a forest if and only if it has at most $n - m$ connected components.*

Our solution set contains pairs (X, P), where $X \subseteq V(G)$ is a set of M-saturated vertices and $P \subseteq V(G)$ is a set of marked vertices (markers) such that each connected component in $G[X]$ contains at least one marked vertex. Markers will be helpful in bounding the number of connected components in $G[X]$ by $n' - m'$, where $n' = |X|$ and m' is the number of edges in $G[X]$

(so that Proposition 7 can be applied). Since we will use the Isolation lemma, we will be assigning random weights to the vertices of X. Furthermore, note that two pairs from our solution set with different sets of marked vertices are necessarily considered to be two different solutions. For this reason, we assign random weights both to the vertices of X and vertices of P.

Throughout this section, as the universe, we take the set $U = V(G) \times \{\mathbf{F}, \mathbf{P}\}$, where $V(G) \times \{\mathbf{F}\}$ is used to assign weights to vertices of the chosen forest and $V(G) \times \{\mathbf{P}\}$ is used to assign weights to vertices chosen as markers. Also, throughout this section, we assume that we are given a weight function $\mathsf{w} : U \to \{1, 2, \ldots, N\}$, where $N = 3|U| = 6|V(G)|$.

Let us first consider the Cut part and define the objects we will count.

Definition 5. *Let G be a graph with n vertices and m edges. For integers $0 \leq A \leq n, 0 \leq B \leq m, 0 \leq C \leq n$, and $0 \leq W \leq 2Nn$, we define the following:*

1. $\mathcal{R}_W^{A,B,C} = \{(X, P) : X \subseteq V(G) \wedge |X| = A \wedge G[X]$ *contains exactly B edges* $\wedge$ $G[X]$ *has a perfect matching* $\wedge$ $P \subseteq X \wedge |P| = C \wedge \mathsf{w}(X \times \{\boldsymbol{F}\}) + \mathsf{w}(P \times \{\boldsymbol{P}\}) = W\}$.
2. $\mathcal{S}_W^{A,B,C} = \{(X, P) \in \mathcal{R}_W^{A,B,C} : G[X]$ *is a forest containing at least one marker from the set P in each connected component*$\}$.
3. $\mathcal{C}_W^{A,B,C} = \{((X, P), (X_l, X_r)) : (X, P) \in \mathcal{R}_W^{A,B,C} \wedge P \subseteq X_l \wedge (X_l, X_r)$ *is a consistent cut of* $G[X]\}$.

We call $\mathcal{R} = \bigcup_{A,B,C,W} \mathcal{R}_W^{A,B,C}$ the family of *candidate solutions*, $\mathcal{S} = \bigcup_{A,B,C,W} \mathcal{S}_W^{A,B,C}$ the family of *solutions*, and $\mathcal{C} = \bigcup_{A,B,C,W} \mathcal{C}_W^{A,B,C}$ the family of *cuts*.

Let us now define the count part.

Lemma 8 (∗). *Let $G, A, B, C, W, \mathcal{C}_W^{A,B,C}$, and $\mathcal{S}_W^{A,B,C}$ be as defined in Definition 5. Then, for every A, B, C, W satisfying $C \leq A - B$, $\left|\mathcal{C}_W^{A,B,C}\right|_2 \equiv \left|\mathcal{S}_W^{A,B,C}\right|$.*

Remark 1. Condition $C \leq A - B$ is necessary for Lemma 8 as otherwise (if $A - B < C$), it is not possible to bound the number of connected components in $G[X]$ by $A - B$. As a result, the elements of $\mathcal{S}_W^{A,B,C}$ could not be identified, and Proposition 7 could not be applied.

By Isolation Lemma [33], we have the following lemma.

Lemma 9. *Let G and $\mathcal{S}_W^{A,B,C}$ be as defined in Definition 5. For each $u \in U$, where U is the universe, choose a weight $\mathsf{w}(u) \in \{1, 2, \ldots, 3|U|\}$ uniformly and independently at random. For some A, B, C, W satisfying $C \leq A - B$, if $|\mathcal{S}_W^{A,B,C}| > 0$, then $\mathsf{prob}(\mathsf{w}$ isolates $\mathcal{S}_W^{A,B,C}) \geq \frac{2}{3}$.*

The following observation helps us in proving Theorem 2.

Observation 10 (∗). *G admits an acyclic matching of size $\frac{\ell}{2}$ if and only if there exist integers B and W such that the set $\mathcal{S}_W^{\ell,B,\ell-B}$ is nonempty.*

Now we describe a procedure that, given a nice tree decomposition $\mathcal{T}$ with the deferred edge property, a weight function $\mathsf{w} : U \to \{1, 2, \ldots, N\}$, and integers A, B, C, W as defined in Definition 5 and satisfying $C \leq A - B$, computes $\left|\mathcal{C}_W^{A,B,C}\right|_2$ using dynamic programming. For this purpose, consider the following.

Definition 6. *For every bag $\mathcal{B}_x$ of the tree decomposition $\mathcal{T}$, for every integer $0 \leq a \leq n, 0 \leq b < n, 0 \leq c \leq n, 0 \leq w \leq 12n^2$, for every coloring $d : \mathcal{B}_x \to \{0, 1, 2\}$, for every coloring $s : \mathcal{B}_x \to \{0, l, r\}$, we define the following:*

1. *$\mathcal{R}_x[a, b, c, d, w] = \{(X, P) : d$ is a valid coloring of $\mathcal{B}_x \wedge X$ is the set of vertices colored 1 or 2 under some valid extension $\widehat{d}$ of d in $G_x \wedge |X| = a \wedge |E_x \cap E(G[X])| = b \wedge P \subseteq X \setminus \mathcal{B}_x \wedge |P| = c \wedge \mathsf{w}(X \times \{\boldsymbol{F}\}) + \mathsf{w}(P \times \{\boldsymbol{P}\}) = w\}$.*
2. *$\mathcal{C}_x[a, b, c, d, w] = \{((X, P), (X_l, X_r)) : (X, P) \in \mathcal{R}_x[a, b, c, d, w] \ \wedge P \subseteq X_l \ \wedge (X, (X_l, X_r))$ is a consistently cut subgraph of $G_x\}$.*
3. *$\widetilde{\mathcal{A}}_x[a, b, c, d, w, s] = |\{((X, P), (X_l, X_r)) \in \mathcal{C}_x[a, b, c, d, w] : (s(v) = l \Rightarrow v \in X_l) \ \wedge (s(v) = r \Rightarrow v \in X_r) \ \wedge (s(v) = 0 \Rightarrow \ v \notin X)\}|$.*

Remark 2. In Definition 6, we assume $b < n$ because otherwise, an induced subgraph containing b edges is definitely not a forest.

The intuition behind Definition 6 is that the set $\mathcal{R}_x[a, b, c, d, w]$ contains all pairs (X, P) that could potentially be extended to a candidate solution from $\mathcal{R}$ (with cardinality and weight restrictions as prescribed by a, b, c, and w), and the set $\mathcal{C}_x[a, b, c, d, w]$ contains all consistently cut subgraphs of G_x that could potentially be extended to elements of $\mathcal{C}$ (with cardinality and weight restrictions as prescribed by a, b, c, and w). The number $\widetilde{\mathcal{A}}_x[a, b, c, d, w, s]$ counts precisely those elements of $\mathcal{C}_x[a, b, c, d, w]$ for which $s(v)$ describes whether for every $v \in \mathcal{B}_x$, v lies in X_l, X_r, or outside X depending on whether $s(v)$ is l, r, or 0, respectively.

We have a table $\mathcal{A}$ with an entry $\mathcal{A}_x[a, b, c, d, w, s]$ for each bag $\mathcal{B}_x$ of $\mathbb{T}$, for integers $0 \leq a \leq n, 0 \leq b < n, 0 \leq c \leq n, 0 \leq w \leq 12n^2$, for every coloring $d : \mathcal{B}_x \to \{0, 1, 2\}$, and for every coloring $s : \mathcal{B}_x \to \{0, l, r\}$. We say that s and d are compatible if for every $v \in \mathcal{B}_x$, the following hold: $d(v) = 0$ if and only if $s(v) = 0$. Note that we have at most $\mathcal{O}(\text{tw} \cdot n)$ many choices for x, at most n choices for a, b, and c, at most $12n^2$ choices for w, and at most 5^{tw} many compatible choices for d and s. Whenever s is not compatible with d, we do not store the entry $\mathcal{A}_x[a, b, c, d, w, s]$ and assume that the access to such an entry returns 0. Therefore, the size of table $\mathcal{A}$ is bounded by $\mathcal{O}(5^{\text{tw}} \cdot \text{tw} \cdot n)$. We will show how to compute the table $\mathcal{A}$ so that the following will be satisfied.

Lemma 11. *If d is valid and d is compatible with s, then $\mathcal{A}_x[a, b, c, d, w, s]$ stores the value $\widetilde{\mathcal{A}}_x[a, b, c, d, w, s]$. Else, the entry $\mathcal{A}_x[a, b, c, d, w, s]$ stores the value 0.*

By Lemma 8, we seek values $\left|\mathcal{C}_W^{A,B,C}\right|_2$. By Observation 10, Definition 6, and Lemma 11, it suffices to compute values $\mathcal{A}_r[\ell, B, \ell - B, \emptyset, W, \emptyset]$ for all B and W, where r is the root of the decomposition $\mathcal{T}$. (Note that $\mathcal{A}_r[\ell, B, \ell - B, \emptyset, W, \emptyset] = \left|\mathcal{C}_W^{\ell,B,\ell-B}\right|$, and we will calculate the modulo 2 separately.) Further, to achieve the time complexity we aim to achieve, we decide whether to mark a vertex or not in its forget bag. Our algorithm computes $\mathcal{A}_x[a, b, c, d, w, s]$ for all bags $\mathcal{B}_x \in \mathcal{T}$ in a bottom-up manner for all integers $0 \leq a \leq n, 0 \leq b < n, 0 \leq c \leq n, 0 \leq w \leq 12n^2$, and for all compatible colorings $d : \mathcal{B}_x \rightarrow \{0, 1, 2\}$ and $s : \mathcal{B}_x \rightarrow \{0, l, r\}$. Further details are deferred to the full version (see [10]).

We note that, by the naive method, the evaluation of all leaf nodes, introduce vertex and edge nodes, and forget nodes can be done in $5^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ time, but the evaluation of all join nodes altogether can be done in $7^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ time. However, the fast subset convolution can be used to handle join nodes more efficiently, and therefore the evaluation of all join nodes altogether can be done in $6^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ time, and hence we have Theorem 2.

4 Algorithm for c-Disconnected Matching

We first define the following notion.

Definition 7 (Fine Coloring). *Given a node x of $\mathbb{T}$ and a fixed integer $c \geq 2$, a coloring $f : \mathcal{B}_x \rightarrow \{0, 1, \ldots, c\}$ is a* fine coloring *if there exists a coloring $\widehat{f} : V_x \rightarrow \{0, 1, \ldots, c\}$ in G_x, a* fine extension *of f, such that the following hold: (i) $\widehat{f}$ restricted to $\mathcal{B}_x$ is f, (ii) If $uv \in E_x$, $\widehat{f}(u) \neq 0$, $\widehat{f}(v) \neq 0$, then $\widehat{f}(u) = \widehat{f}(v)$.*

Note that point (ii) in Definition 7 implies that whenever two vertices in G_x have an edge between them, then they should get the same color under a fine extension except possibly when either of them is colored 0.

Before we begin the formal description of the algorithm, let us briefly discuss the idea that yields us a single exponential running time for the c-Disconnected Matching problem rather than a slightly exponential running time[4] (which is common for most of the naive dynamic programming algorithms for connectivity type problems). We will use Definition 7 to partition the vertices of V_x into color classes (at most c). Note that we do not require in Definition 7 that $G_x[\widehat{f}^{-1}(i)]$ for any $i \in [c]$ is a connected graph. This is the crux of our efficiency. Specifically, this means that we do not keep track of the precise connected components of $G[V_M]$ in G_x for a matching M, yet Definition 7 is sufficient for us.

Now, let us discuss our ideas more formally. We have a table $\mathcal{A}$ with an entry $\mathcal{A}_x[d, f, \widehat{c}]$ for each bag $\mathcal{B}_x$, for every coloring $d : \mathcal{B}_x \rightarrow \{0, 1, 2\}$, for every coloring $f : \mathcal{B}_x \rightarrow \{0, 1, \ldots, c\}$, and for every set $\widehat{c} \subseteq \{1, 2, \ldots, c\}$. We say that d and f are *compatible* if for every $v \in \mathcal{B}_x$, the following hold: $d(v) = 0$ if and only if $f(v) = 0$. We say that f and $\widehat{c}$ are compatible if for any $v \in \mathcal{B}_x$, $f(v) \in \widehat{c}$. Note that we have at most $\mathcal{O}(\text{tw} \cdot n)$ many choices for x, at most $(2c+1)^{\text{tw}}$ many choices for compatible d and f, and at most 2^c choices for $\widehat{c}$. Furthermore, whenever f is not compatible with d or $\widehat{c}$, we do not store the entry $\mathcal{A}_x[d, f, \widehat{c}]$ and assume that the access to such an entry returns $-\infty$. Therefore, the size of table $\mathcal{A}$ is bounded by $\mathcal{O}((2c+1)^{\text{tw}} \cdot \text{tw} \cdot n)$. The following definition specifies the value each entry $\mathcal{A}_x[d, f, \widehat{c}]$ of $\mathcal{A}$ is supposed to store.

[4] That is, running time $2^{\mathcal{O}(\text{tw})} \cdot n^{\mathcal{O}(1)}$ rather than $\text{tw}^{\mathcal{O}(\text{tw})} \cdot n^{\mathcal{O}(1)}$..

Definition 8. *If d is valid, f is fine, f is compatible with d and $\widehat{c}$, and there exists a fine extension $\widehat{f}$ of f such that $\widehat{c}$ equals the set of distinct non-zero colors assigned by $\widehat{f}$, then the entry $\mathcal{A}_x[d, f, \widehat{c}]$ stores the maximum number of vertices that are colored 1 or 2 under some valid extension $\widehat{d}$ of d in G_x such that for every $v \in V_x$, $\widehat{d}(v) = 0$ if and only if $\widehat{f}(v) = 0$. Otherwise, the entry $\mathcal{A}_x[d, f, \widehat{c}]$ stores the value $-\infty$.*

Since the root of $\mathbb{T}$ is an empty node, note that the maximum number of vertices saturated by any c-disconnected matching is exactly $\mathcal{A}_r[\emptyset, \emptyset, \{1, 2, \ldots, c\}]$, where r is the root of $\mathbb{T}$. We now provide recursive formulas to compute the entries of table $\mathcal{A}$.

Leaf Node: For a leaf node x, we have that $\mathcal{B}_x = \emptyset$. Hence there is only one possible coloring on $\mathcal{B}_x$, that is, the empty coloring (for both d and f). Since f and G_x are empty, the only compatible choice for $\widehat{c}$ is $\{\}$, and we have $\mathcal{A}_x[\emptyset, \emptyset, \{\}] = 0$.

Introduce Vertex Node: Suppose that x is an introduce vertex node with child node y such that $\mathcal{B}_x = \mathcal{B}_y \cup \{v\}$ for some $v \notin \mathcal{B}_y$. For every coloring $d : \mathcal{B}_x \to \{0, 1, 2\}$, every set $\widehat{c} \subseteq \{1, 2, \ldots, c\}$, and every coloring $f : \mathcal{B}_x \to \{0, 1, \ldots, c\}$ such that f is compatible with d and $\widehat{c}$, we have the following recursive formula:

$$\mathcal{A}_x[d, f, \widehat{c}] = \begin{cases} \mathcal{A}_y[d|_{\mathcal{B}_y}, f|_{\mathcal{B}_y}, \widehat{c}] & \text{if } d(v) = 0, \\ -\infty & \text{if } d(v) = 1, \\ \max\{\mathcal{A}_y[d|_{\mathcal{B}_y}, f|_{\mathcal{B}_y}, \widehat{c} \setminus \{f(v)\}] + 1, & \\ \mathcal{A}_y[d|_{\mathcal{B}_y}, f|_{\mathcal{B}_y}, \widehat{c}] + 1\} & \text{if } d(v) = 2. \end{cases}$$

Introduce Edge Node: Suppose that x is an introduce edge node that introduces an edge uv, and let y be the child of x. For every coloring $d : \mathcal{B}_x \to \{0, 1, 2\}$, every set $\widehat{c} \subseteq \{1, 2, \ldots, c\}$, and every coloring $f : \mathcal{B}_x \to \{0, 1, \ldots, c\}$ such that f is compatible with d and $\widehat{c}$, we consider the following cases:

If at least one of $d(u)$ or $d(v)$ is 0, then

$$\mathcal{A}_x[d, f, \widehat{c}] = \mathcal{A}_y[d, f, \widehat{c}].$$

Else, if at least one of $d(u)$ or $d(v)$ is 2, and $f(u) = f(v)$, then

$$\mathcal{A}_x[d, f, \widehat{c}] = \mathcal{A}_y[d, f, \widehat{c}].$$

Else, if both $d(u)$ and $d(v)$ are 1, and $f(u) = f(v)$, then

$$\mathcal{A}_x[d, f, \widehat{c}] = \max\{\mathcal{A}_y[d, f, \widehat{c}], \mathcal{A}_y[d_{\{u,v\} \to 2}, f, \widehat{c}]\}.$$

Else, $\mathcal{A}_x[d, f, \widehat{c}] = -\infty$.

Forget Node: Suppose that x is a forget vertex node with a child y such that $\mathcal{B}_x = \mathcal{B}_y \setminus \{u\}$ for some $u \in \mathcal{B}_y$. For every coloring $d : \mathcal{B}_x \to \{0, 1, 2\}$, every

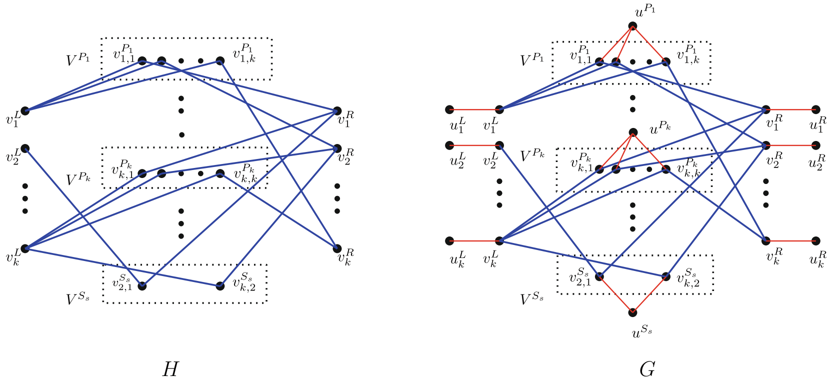


Fig. 1. An illustration of the construction of G from H. Here, we assume that $S_s = \{(2,1),(k,2)\}$.

set $\widehat{c} \subseteq \{1,2,\dots,c\}$, and every coloring $f : \mathcal{B}_x \rightarrow \{0,1,\dots,c\}$ such that f is compatible with d and $\widehat{c}$, we have

$$\mathcal{A}_x[d,f,\widehat{c}] = \max\{\mathcal{A}_y[d_{u\rightarrow 0}, f_{u\rightarrow 0}, \widehat{c}], \max_{\overline{c}\in\widehat{c}}\{\mathcal{A}_y[d_{u\rightarrow 1}, f_{u\rightarrow \overline{c}}, \widehat{c}]\}\}.$$

Join Node: Let x be a join node with children y_1 and y_2. For every coloring $d : \mathcal{B}_x \rightarrow \{0,1,2\}$, every set $\widehat{c} \subseteq \{1,2,\dots,c\}$, and for every coloring $f : \mathcal{B}_x \rightarrow \{0,1,\dots,c\}$ such that f is compatible with d and $\widehat{c}$, we have

$$\mathcal{A}_x[d,f,\widehat{c}] = \max_{d_1,d_2}\{\max_{\substack{\widehat{c}_{y_1},\widehat{c}_{y_2}\\ \widehat{c}_{y_1}\cup\widehat{c}_{y_2}=\widehat{c}}}\{\mathcal{A}_{y_1}[d_1,f,\widehat{c}_{y_1}]+\mathcal{A}_{y_2}[d_2,f,\widehat{c}_{y_2}]-|d^{-1}(1)|-|d^{-1}(2)|\}\},$$

where $d_1 : \mathcal{B}_{y_1} \rightarrow \{0,1,2\}$, $d_2 : \mathcal{B}_{y_2} \rightarrow \{0,1,2\}$, $\widehat{c}_{y_1}, \widehat{c}_{y_2} \subseteq \{1,2,\dots,c\}$ such that f is compatible with d_1, d_2, $\widehat{c}_{y_1}$, and $\widehat{c}_{y_2}$, and d_1, d_2 are correct for d.

Further details are deferred to the full version (see [10]). We note that, by the naive method, the evaluation of all leaf nodes, introduce vertex and edge nodes, and forget nodes can be done in $(2c+1)^{\text{tw}} \cdot \text{tw}^{\mathcal{O}(1)} \cdot n$ time, but the evaluation of all join nodes altogether can be done in $(4c+2)^{\text{tw}} \cdot \text{tw}^{\mathcal{O}(1)} \cdot n$ time. However, the fast subset convolution can be used to handle join nodes more efficiently, and therefore the evaluation of all join nodes altogether can be done in $(3c)^{\text{tw}} \cdot \text{tw}^{\mathcal{O}(1)} \cdot n$ time, and hence we have Theorem 3.

5 Lower Bound for Disconnected Matching

We give a reduction from $k \times k$ Hitting Set. The input of $k \times k$ Hitting Set consists of a family of sets $S_1, \dots, S_m \subseteq [k] \times [k]$ such that each set contains at most one element from each row of $[k] \times [k]$, and the question is to determine if there exists a set $\hat{S}$ containing exactly one element from each row such that $\hat{S} \cap S_i \neq \emptyset$ for every $i \in [m]$? Here, the parameter is k.

Proposition 12 **([27]).** *Assuming Exponential Time Hypothesis, there is no $2^{o(k \log k)} \cdot n^{\mathcal{O}(1)}$-time algorithm for $k \times k$* Hitting Set.

Our reduction is inspired by the reduction given by Cygan et al. in [14] to prove that there is no $2^{o(\mathrm{pw} \log \mathrm{pw})} \cdot n^{\mathcal{O}(1)}$-time algorithm for Maximally Disconnected Dominating Set unless the Exponential Time Hypothesis fails. Given an instance $(k, S_1, \ldots, S_m)$ of $k \times k$ Hitting Set, we construct an equivalent instance $(G, 3k+m, k)$ of Disconnected Matching in polynomial time. First, we define a simple gadget that will be used in our construction.

Definition 9 (Star Gadget). *By adding a* star gadget *to a vertex set $X \subseteq V(G)$, we mean the following construction: We introduce a new vertex of degree $|X|$ and connect it to all vertices in X.*

If we attach a star gadget to multiple vertex disjoint subsets of H, then we have the following lemma.

Lemma 13 **($*$).** *Let H be a graph and let G be the graph constructed from H by adding star gadgets to vertex disjoint subsets $X_1, \ldots, X_l$ of $V(H)$. Assume we are given a path decomposition $\widetilde{\mathcal{P}}$ of H of width* pw *with the following property: For each X_i, $i \in [l]$, there exists a bag in $\widetilde{\mathcal{P}}$ that contains X_i. Then, in polynomial time, we can construct a path decomposition of G of width at most* $\mathrm{pw} + 1$.

Now, consider the following construction (see Fig. 1 for an illustration).

Construction $\mathcal{D}$. Let $P_i = \{i\} \times [k]$ be a set containing all elements in the i-th row in the set $[k] \times [k]$. We define $\mathcal{S} = \{S_s : s \in [m]\} \cup \{P_i : i \in [k]\}$. Note that for each $X \in \mathcal{S}$, we have $|X| \leq k$, as each S_s, $s \in [m]$ contains at most one element from each row and $|P_i| = k$ for each $i \in [k]$.

First, let us define a graph H. We start by introducing vertices v_i^L for each $i \in [k]$ and vertices v_j^R for each $j \in [k]$. Then, for each set $X \in \mathcal{S}$, we introduce vertices $v_{i,j}^X$ for every $(i,j) \in X$. Let $V^X = \{v_{i,j}^X : (i,j) \in X\}$. We also introduce the edge set $\{v_i^L v_{i,j}^X\} \cup \{v_{i,j}^X v_j^R\}$ for each $X \in \mathcal{S}$ and $i, j \in [k]$. This ends the construction of H. Now, we construct a graph G from the graph H as follows: For each $i \in [k]$ and $j \in [k]$, we attach star gadgets to vertices v_i^L and v_j^R. Furthermore, for each $X \in \mathcal{S}$, we attach star gadgets to X. For each $i \in [k]$ (resp. $j \in [k]$), let u_i^L (resp. u_j^R) denote the unique vertex in the star gadget corresponding to v_i^L (resp. v_j^R). For each $X \in \mathcal{S}$, let u^X denote the unique vertex in the star gadget corresponding to X. Let $E^X = \{v_{i,j}^X u^X : (i,j) \in X\}$.

We now provide a pathwidth bound on G.

Lemma 14 **($*$).** *Let H and G be as defined in Construction $\mathcal{D}$. Then, the pathwidth of G is at most $3k$.*

Lemma 15 **($*$).** *Let G be as defined in Construction $\mathcal{D}$. If the initial $k \times k$* Hitting Set *instance is a Yes-instance, then there exists a matching M in G such that $|M| = 3k + m$ and $G[V_M]$ has exactly k connected components.*

Lemma 16 (∗). *Let G be as defined in Construction $\mathcal{D}$. If there exists a matching M in G such that $|M| \geq 3k + m$ and $G[V_M]$ has at least k connected components, then the initial $k \times k$* Hitting Set *instance is a Yes-instance.*

By Proposition 12 and Lemmas 14, 15 and 16, we have Theorem 4.

6 Conclusion

A detailed conclusion (where we briefly discuss the SETH lower bounds) can be found in the full version (see [10]).

References

1. Bachstein, A., Goddard, W., Lehmacher, C.: The generalized matcher game. Discret. Appl. Math. **284**, 444–453 (2020)
2. Baste, J., Fürst, M., Rautenbach, D.: Approximating maximum acyclic matchings by greedy and local search strategies. In: Kim, D., Uma, R.N., Cai, Z., Lee, D.H. (eds.) COCOON 2020. LNCS, vol. 12273, pp. 542–553. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58150-3_44
3. Baste, J., Rautenbach, D.: Degenerate matchings and edge colorings. Discret. Appl. Math. **239**, 38–44 (2018)
4. Baste, J., Rautenbach, D., Sau, I.: Uniquely restricted matchings and edge colorings. In: Bodlaender, H.L., Woeginger, G.J. (eds.) WG 2017. LNCS, vol. 10520, pp. 100–112. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68705-6_8
5. Baste, J., Rautenbach, D., Sau, I.: Upper bounds on the uniquely restricted chromatic index. J. Graph Theory **91**(3), 251–258 (2019)
6. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Fourier meets möbius: fast subset convolution. In: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing, pp. 67–74 (2007)
7. Bodlaender, H.L., et al.: A tourist guide through treewidth (1992)
8. Cameron, K.: Induced matchings. Discret. Appl. Math. **24**(1–3), 97–102 (1989)
9. Cameron, K.: Connected matchings. In: Jünger, M., Reinelt, G., Rinaldi, G. (eds.) Combinatorial Optimization—Eureka, You Shrink! LNCS, vol. 2570, pp. 34–38. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36478-1_5
10. Chaudhary, J., Zehavi, M.: $\mathcal{P}$-matchings parameterized by treewidth. arXiv preprint arXiv:2307.09333v1 (2023)
11. Chaudhary, J., Zehavi, M.: Parameterized results on acyclic matchings with implications for related problems. In: Paulusma, D., Ries, B. (eds.) WG 2023. LNCS, vol. 14093, pp. 201–216. Springer, Heidelberg (2023)
12. Cooley, O., Draganic, N., Kang, M., Sudakov, B.: Large induced matchings in random graphs. SIAM J. Discret. Math. **35**(1), 267–280 (2021)
13. Cygan, M., et al.: Parameterized Algorithms, vol. 5. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-319-21275-3
14. Cygan, M., Nederlof, J., Pilipczuk, M., Pilipczuk, M., Van Rooij, J.M., Wojtaszczyk, J.O.: Solving connectivity problems parameterized by treewidth in single exponential time. ACM Trans. Algorithms (TALG) **18**(2), 1–31 (2022)
15. Diestel, R.: Graph Theory. Graduate Text GTM, vol. 173. Springer, Heidelberg (2012)

16. Francis, M.C., Jacob, D., Jana, S.: Uniquely restricted matchings in interval graphs. SIAM J. Discret. Math. **32**(1), 148–172 (2018)
17. Fürst, M., Rautenbach, D.: On some hard and some tractable cases of the maximum acyclic matching problem. Ann. Oper. Res. **279**, 291–300 (2019)
18. Goddard, W., Hedetniemi, S.M., Hedetniemi, S.T., Laskar, R.: Generalized subgraph-restricted matchings in graphs. Discret. Math. **293**(1–3), 129–138 (2005)
19. Goddard, W., Henning, M.A.: The matcher game played in graphs. Discret. Appl. Math. **237**, 82–88 (2018)
20. Golumbic, M.C., Hirst, T., Lewenstein, M.: Uniquely restricted matchings. Algorithmica **31**, 139–154 (2001)
21. Gomes, G.C., Masquio, B.P., Pinto, P.E., dos Santos, V.F., Szwarcfiter, J.L.: Disconnected matchings. Theoret. Comput. Sci. **956**, 113821 (2023)
22. Hajebi, S., Javadi, R.: On the parameterized complexity of the acyclic matching problem. Theoret. Comput. Sci. **958**, 113862 (2023)
23. Klemz, B., Rote, G.: Linear-time algorithms for maximum-weight induced matchings and minimum chain covers in convex bipartite graphs. Algorithmica **84**(4), 1064–1080 (2022)
24. Kloks, T.: Treewidth: Computations and Approximations. Springer, Heidelberg (1994). https://doi.org/10.1007/BFb0045375
25. Ko, C., Shepherd, F.B.: Bipartite domination and simultaneous matroid covers. SIAM J. Discret. Math. **16**(4), 517–523 (2003)
26. Koana, T. Induced matching below guarantees: average paves the way for fixed-parameter tractability. arXiv preprint arXiv:2212.13962 (2022)
27. Lokshtanov, D., Marx, D., Saurabh, S.: Slightly superexponential parameterized problems. SIAM J. Comput. **47**(3), 675–702 (2018)
28. Lovász, L., Plummer, M.D.: Matching Theory. Annals of Discrete Mathematics, vol. 29. North-Holland, Amsterdam (1986)
29. Manlove, D.: Algorithmics of Matching Under Preferences, vol. 2. World Scientific, Singapore (2013)
30. Micali, S., Vazirani, V.V.: An o($\sqrt{|V|}|e|$) algorithm for finding maximum matching in general graphs. In: 21st Annual Symposium on Foundations of Computer Science (SFCS 1980), pp. 17–27. IEEE (1980)
31. Moser, H., Sikdar, S.: The parameterized complexity of the induced matching problem. Discret. Appl. Math. **157**(4), 715–727 (2009)
32. Moser, H., Thilikos, D.M.: Parameterized complexity of finding regular induced subgraphs. J. Discret. Algorithms **7**(2), 181–190 (2009)
33. Mulmuley, K., Vazirani, U.V., Vazirani, V.V.: Matching is as easy as matrix inversion. In Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, pp. 345–354 (1987)
34. Panda, B., Chaudhary, J.: Acyclic matching in some subclasses of graphs. Theoret. Comput. Sci. **943**, 36–49 (2023)
35. Robertson, N., Seymour, P.D.: Graph minors. II. Algorithmic aspects of tree-width. J. Algorithms **7**(3), 309–322 (1986)
36. Song, Y.: On the induced matching problem in Hamiltonian bipartite graphs. Geor. Math. J. **28**(6), 957–970 (2021)
37. Stockmeyer, L.J., Vazirani, V.V.: Np-completeness of some generalizations of the maximum matching problem. Inf. Process. Lett. **15**(1), 14–19 (1982)
38. Vizing, V.G.: On an estimate of the chromatic class of a P-graph. Diskret analiz **3**, 25–30 (1964)
39. Xiao, M., Kou, S.: Parameterized algorithms and kernels for almost induced matching. Theoret. Comput. Sci. **846**, 103–113 (2020)

Algorithms and Hardness for Metric Dimension on Digraphs

Antoine Dailly[1](✉), Florent Foucaud[1], and Anni Hakanen[1,2]

[1] Université Clermont Auvergne, CNRS, Clermont Auvergne INP, Mines Saint-Étienne, LIMOS, 63000 Clermont-Ferrand, France
{antoine.dailly,florent.foucaud,anni.hakanen}@uca.fr

[2] Department of Mathematics and Statistics, University of Turku, 20014 Turku, Finland

Abstract. In the Metric Dimension problem, one asks for a minimum-size set R of vertices such that for any pair of vertices of the graph, there is a vertex from R whose two distances to the vertices of the pair are distinct. This problem has mainly been studied on undirected graphs and has gained a lot of attention in the recent years. We focus on directed graphs, and show how to solve the problem in linear-time on digraphs whose underlying undirected graph (ignoring multiple edges) is a tree. This (nontrivially) extends a previous algorithm for oriented trees. We then extend the method to unicyclic digraphs (understood as the digraphs whose underlying undirected multigraph has a unique cycle). We also give a fixed-parameter-tractable algorithm for digraphs when parameterized by the directed modular-width, extending a known result for undirected graphs. Finally, we show that Metric Dimension is NP-hard even on planar triangle-free acyclic digraphs of maximum degree 6.

1 Introduction

The metric dimension of a (di)graph G is the smallest size of a set of vertices that distinguishes all vertices of G by their vectors of distances from the vertices of the set. This concept was introduced in the 1970s by Harary and Melter [14] and by Slater [30] independently. Due to its interesting nature and numerous applications (such as robot navigation [18], detection in sensor networks [30] or image processing [22], to name a few), it has enjoyed a lot of attention. It also has been studied in the more general setting of metric spaces [3], and is generally part of the rich area of identification problems of graphs and other discrete structures [20].

More formally, let us denote by $\mathrm{dist}(x, y)$ the distance from x to y in a digraph. Here, the distance $\mathrm{dist}(x, y)$ is taken as the length of a shortest directed

Research funded by the French government IDEX-ISITE initiative 16-IDEX-0001 (CAP 20–25) and by the ANR project GRALMECO (ANR-21-CE48-0004).

A. Hakanen—Research supported by the Jenny and Antti Wihuri Foundation and partially by Academy of Finland grant number 338797.

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 232–245, 2023.
https://doi.org/10.1007/978-3-031-43380-1_17

path from x to y; if no such path exists, $\mathrm{dist}(x, y)$ is infinite, and we say that y is not *reachable* from x. We say that a set S is a *resolving set* of a digraph G if for any pair of distinct vertices v, w from G, there is a vertex x in S with $\mathrm{dist}(x, v) \neq \mathrm{dist}(x, w)$. Furthermore, we require that every vertex of G is reachable from at least one vertex of S. The *metric dimension* of G is the smallest size of a resolving set of G, and a minimum-size resolving set of G is called a *metric basis* of G.[1]

We denote by Metric Dimension the computational version of the problem: given a (di)graph G, determine its metric dimension.

For undirected graphs, Metric Dimension has been extensively studied, and its non-local nature makes it highly nontrivial from an algorithmic point of view. On the hardness side, Metric Dimension was shown to be NP-hard for planar graphs of bounded degree [7], split, bipartite and line graphs [9], unit disk graphs [17], interval and permutation graphs of diameter 2 [11], and graphs of pathwidth 24 [19]. On the positive side, it can easily be solved in linear time on trees [4,14,18,30]. More involved polynomial-time algorithms exist for unicyclic graphs [29] and, more generally, graphs of bounded cyclomatic number [9]; outerplanar graphs [7]; cographs [9]; chain graphs [10]; cactus-block graphs [16]; bipartite distance-hereditary graphs [24]. There are fixed parameter tractable (FPT) algorithms for the undirected graph parameters max leaf number [8], tree-depth [13], modular-width [2] and distance to cluster [12], but FPT algorithms are highly unlikely to exist for parameters solution size [15] and feedback vertex set [12].

Due to the interest for Metric Dimension on undirected graphs, it is natural to ask what can be said in the context of digraphs. The metric dimension of digraphs was first studied in [5] under a somewhat restrictive definition; for our definitions, we follow the recent paper [1], in which the algorithmic aspects of Metric Dimension on digraphs have been addressed. We call *oriented graph* a digraph without directed 2-cycles. A *directed acyclic digraph* (DAG for short) has no directed cycles at all. The *underlying multigraph* of a digraph is the one obtained by ignoring the arc orientations; its *underlying graph* is obtained from it by ignoring multiple edges. In a digraph, a *strongly connected component* is a subgraph where every vertex is reachable from all other vertices. Note that for the Metric Dimension problem, undirected graphs can be seen as a special type of digraphs where each arc has a symmetric arc.

The NP-hardness of Metric Dimension was proven for oriented graphs in [27] and, more recently, for bipartite DAGs of maximum degree 8 and maximum distance 4 [1] (the *maximum distance* being the length of a longest directed path without shortcuts). A linear-time algorithm for Metric Dimension on oriented trees was given in [1].

[1] The definition that we use has been called *strong metric dimension* in [1], as opposed to *weak metric dimension*, where one single vertex may be unreachable from any resolving set vertex. The former definition seems more natural to us. However, the term *strong metric dimension* is already used for a different concept, see [25]. Thus, to prevent confusion, we avoid the prefix *strong* in this paper.

Our Results. We generalize the linear-time algorithm for Metric Dimension on oriented trees from [1] to all digraphs whose underlying graph is a tree. In other words, here we allow 2-cycles. This makes a significant difference with oriented trees, and as a result our algorithm is nontrivial. We then extend the used methods to solve Metric Dimension in linear time for unicyclic digraphs (digraphs with a unique cycle). Then, we prove that Metric Dimension can be solved in time $f(t)n^{O(1)}$ for digraphs of order n and modular-width t (a parameter recently introduced for digraphs in [31]). This extends the same result for undirected graphs from [2], and is the first FPT algorithm for Metric Dimension on digraphs. Finally, we complement the hardness result from [1] by showing that Metric Dimension is NP-hard even for planar triangle-free DAGs of maximum degree 6 and maximum distance 4. For results marked with ($*$), we omit the full proof due to space constraints; those can be found in [6].

2 Digraphs Whose Underlying Graph is a Tree

For the sake of convenience, we call *di-tree* a digraph whose underlying graph is a tree. Trees are often the first non-trivial class to study for a graph problem. Metric Dimension is no exception to this, having been studied in the first papers for the undirected [4,14,18,30] and the oriented [1] cases. In the undirected case, a minimum-size resolving set can be found by taking, for each vertex of degree at least 3 spanning k legs, the endpoint of $k-1$ of its legs (a *leg* is an induced path spanning from a vertex of degree at least 3, having its inner vertices of degree 2, and ending in a leaf). In the case of oriented trees, taking all the sources (a *source* is a vertex with no in-neighbour) and $k-1$ vertices in each set of k in-twins yields a metric basis (two vertices are *in-twins* if they have the same in-neighbourhood). Our algorithm, being on di-trees (which include both undirected trees and oriented trees), will reuse those strategies, but we will need to refine them in order to obtain a metric basis. The first refinement is of the notion of in-twins:

Definition 1. *A strongly connected component E of a di-tree is an* escalator *if it satisfies the following conditions:*

1. *its underlying graph is a path with vertices $e_1, \dots, e_k$ ($k \geq 2$);*
2. *there is a unique vertex $y \notin E$ such that the arc $\overrightarrow{ye_1}$ (resp. $\overrightarrow{ye_k}$) exists;*
3. *there can be any number (possibly, zero) of vertices $z \notin E$ such that the arc $\overrightarrow{e_kz}$ (resp. $\overrightarrow{e_1z}$) exists; for every $i \in \{1, \dots, k-1\}$ (resp. $i \in \{2, \dots, k\}$), no arc $\overrightarrow{e_iz}$ with $z \notin E$ exists.*

Definition 2. *In a di-tree, a set of vertices $A = \{a_1, \dots, a_k\}$ is a set of* almost-in-twins *if there is a vertex x such that:*

1. *for every $i \in \{1, \dots, k\}$, the arc $\overrightarrow{xa_i}$ exists and the arc $\overrightarrow{a_ix}$ does not exist;*
2. *for every $i \in \{1, \dots, k\}$, either a_i is a trivial strongly connected component and $N^-(a_i) = \{x\}$, or a_i is the endpoint of an escalator and $N^-(a_i) = \{x, y\}$ where y is its neighbour in the escalator.*

Note that regular in-twins are also almost-in-twins. The second refinement is the following (for a given vertex x in a strongly connected component with C as an underlying graph, we call $d_C(x)$ the degree of x in C):

Definition 3. *Given the underlying graph C of a strongly connected component of a di-tree and a set D of vertices, we call a set S of vertices inducing a path of order at least 2 in C a* special leg *if it verifies the four following properties:*

1. *S has a unique vertex v such that $v \in D$ or $d_C(v) \geq 3$;*
2. *S has a unique vertex w such that $d_C(w) = 1$, furthermore $w \notin D$: w is called the* endpoint *of S;*
3. *all of the other vertices x of S verify $d_C(x) = 2$ and $x \notin D$;*
4. *at least one of the vertices $y \in S \setminus \{w\}$ has an out-arc $\overrightarrow{yz}$ with $z \notin C$.*

Note that several special legs can span from the same vertex, from which regular legs can also span. Algorithm 1, illustrated in Fig. 1, computes a metric basis of a di-tree.

Explanation of Algorithm 1. The algorithm will compute a metric basis $\mathcal{B}$ of a di-tree T in linear-time. The first thing we do is to add every source in T to $\mathcal{B}$ (line 1). Then, for every set of almost-in-twins, we add all of them but one to $\mathcal{B}$ (lines 2–3). Those two first steps, depicted in Fig. 1a, are the ones used to compute the metric basis of an orientation of a tree [1], and as such they are still necessary for managing the non-strongly connected components of the di-tree. Note that we are specifically managing sets of *almost-in-twins*, which include sets of in-twins, since it is necessary to resolve the specific case of escalators. The rest of the algorithm consists in managing the strongly connected components.

For each strongly connected component having C as an underlying graph, we first identify each vertex x of C that has an in-arc coming from **outside** C. Indeed, since x is the "last" vertex of a path coming from outside C, there are vertices of $\mathcal{B}$ "behind" this in-arc (or they can themselves be a vertex in $\mathcal{B}$), which we will call $\mathcal{B}_x$. However, the vertices in $\mathcal{B}_x$ can be "projected" on x since, T being a di-tree, x is on every shortest path from the vertices of $\mathcal{B}$ "behind"the in-arc to the vertices of C. Hence, we will mark x as a **dummy vertex** (lines 5–7, depicted in Figure 1b): we will consider that it is in $\mathcal{B}$ for the rest of this step, and acts as a representative of the set $\mathcal{B}_x$ with respect to C.

We then have to manage some specific cases whenever C is a path (lines 8–17). Indeed, the last two steps of the algorithm do not always work under some conditions. Those specific conditions are highlighted in the proof.

The last two steps are then applied. First, we have to consider the **special legs** defined in Definition 3. The idea behind those special legs is the following: for every out-arc $\overrightarrow{yz}$ with y in the special leg and z outside of C, any vertex in the metric basis "before" the start of the special leg will not distinguish z and the next neighbour of y in the special leg. Hence, we have to add at least one vertex to $\mathcal{B}$ for each special leg, and we choose the endpoint of the special leg (lines 18–19, depicted in Fig. 1c). Finally, we apply the well-known algorithm for

Algorithm 1: An algorithm computing the metric basis of a di-tree.

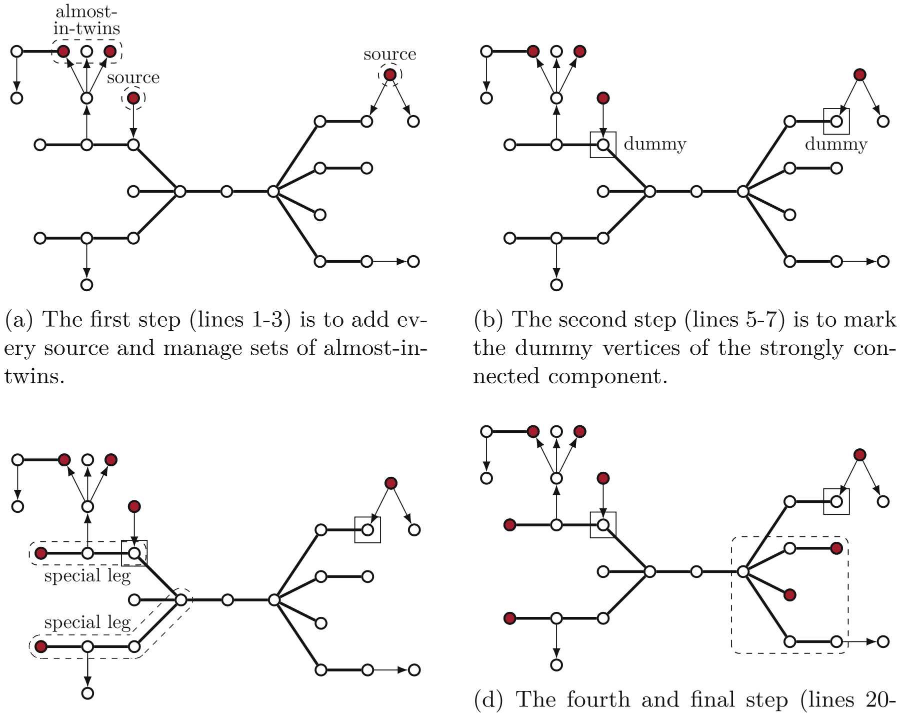
```
   Input  : A di-tree $T$.
   Output: A metric basis $\mathcal{B}$ of $T$.
1  $\mathcal{B} \leftarrow$ Every source of $T$
2  foreach set $I$ of almost-in-twins do
3      Add $|I| - 1$ vertices of $I$ to $\mathcal{B}$
4  foreach strongly connected component with $C$ as an underlying graph do
5      $D \leftarrow \emptyset$
6      foreach arc $\overrightarrow{uv}$ with $v \in C$ and $u \notin C$ do
7          Add $v$ to $D$
8      if $C$ is a path with endpoints $x$ and $y$ then
9          if there is no vertex in $C \cap D$ then
10             if there is no out-arc from $C$ to outside of $C$ then
11                 Add $x$ to $\mathcal{B}$
12             else if there is an out-arc from $x$ (resp. $y$) to outside of $C$ and no
               other out-arc from $C$ to outside of $C$ then
13                 Add $y$ (resp. $x$) to $\mathcal{B}$
14             else
15                 Add $x$ and $y$ to $\mathcal{B}$
16         else if there is exactly one vertex $w$ in $C \cap D$, $w$ is neither $x$ nor $y$, and
           there is no out-arc from $w$ to outside of $C$ then
17             Add $x$ to $\mathcal{B}$
18     foreach special leg $L$ of $C$ do
19         Add the endpoint of $L$ to $\mathcal{B}$
20     foreach vertex of degree $\geq 3$ in $C$ from which span $k \geq 2$ legs of $C$ that do
       not have a vertex in $\mathcal{B}$ or in $D$ do
21         Add the endpoint of $k - 1$ such legs to $\mathcal{B}$
22 return $\mathcal{B}$
```

computing the metric basis of a tree to the remaining parts of C (lines 20–21, depicted in Fig. 1d). The special legs and the legs containing a dummy vertex, being already resolved, are not considered in this part.

Theorem 4 (∗). *Algorithm 1 computes a metric basis of a di-tree in linear time.*

3 Orientations of Unicyclic Graphs

A unicyclic graph U is constituted of a cycle C with vertices $c_1, \ldots, c_n$, and each vertex c_i is the root of a tree T_i (we can have T_i be simply the isolated c_i itself). The metric dimension of an undirected unicyclic graph has been studied in [26,28,29]. In [26], Poisson and Zhang proved bounds for the metric dimension



(a) The first step (lines 1-3) is to add every source and manage sets of almost-in-twins.

(b) The second step (lines 5-7) is to mark the dummy vertices of the strongly connected component.

(c) The third step (lines 18-19) is to manage all the special legs.

(d) The fourth and final step (lines 20-21) is to manage the remaining legs with a common ancestor.

Fig. 1. Illustration of Algorithm 1. For the sake of simplicity, there are only two strongly connected components, for which we only represent the underlying graph with bolded edges, so every bolded edge is a 2-cycle. One of the two strongly connected components is a simple path that does not require any action. Vertices in the metric basis are colored in red. (Color figure online)

of a unicyclic graph in terms of the metric dimension of a tree we obtain by removing one edge from the cycle. Sedlar and Škrekovski showed more recently that the metric dimension of a unicyclic graph is one of two values in [28], and then the exact value of the metric dimension based on the structure of the graph in [29]. In this section, we will show that one can compute a metric basis of an orientation of a unicyclic graph in linear time. The algorithm mostly consists in using sources and in-twins, with a few specific edge cases to consider.

In this section, an *induced directed path* $\overrightarrow{P}$ is the orientation of an induced path with only one source and one sink which are its two endpoints. It is said to be *spanning from u* if u is its source endpoint, and its *length* is its number of edges. We also need the following definition:

Definition 5. *Let $\overrightarrow{U}$ be the orientation of a unicyclic graph. Given an orientation of a cycle $\overrightarrow{C}$ of even length $n = 2k$ with two sources, if its sources are c_i and c_{i+2}, its sinks are c_{i+1} and c_{i+1+k}, and there are, in $\overrightarrow{C} \setminus \{c_i, c_{i+2}\}$, neither in-twins nor in-arcs coming from outside of $\overrightarrow{C}$, we call an induced directed path $\overrightarrow{P}$ an* concerning path *if it verifies the three following properties:*

1. *$\overrightarrow{P}$ spans from c_{i+1};*
2. *$\overrightarrow{P}$ has length $k-2$;*
3. *$\overrightarrow{P}$ has no in-arc coming from outside of $\overrightarrow{P} \cup \overrightarrow{C}$;*

Furthermore, if, for every vertex in $\overrightarrow{P}$ belonging to a nonempty set I of in-twins, every vertex in I belongs to a concerning path, then, we call $\overrightarrow{P}$ an unfixable path.

A path that is a concerning path, but not an unfixable path, will be called a fixable path.

*Finally, a vertex might belong both to an unfixable path and to a fixable path; in this case, the fixable path takes precedence (*i.e.*, we will consider that the vertex belongs to the fixable path).*

Algorithm 2: An algorithm computing the metric basis of an orientation of a unicyclic graph.

```
  Input  : An orientation $\overrightarrow{U}$ of a unicyclic graph $U$.
  Output: A metric basis $\mathcal{B}$ of $\overrightarrow{U}$.
1 Add to $\mathcal{B}$ every source of $\overrightarrow{U}$
2 Apply the special cases in Algorithm 3
3 foreach set $I$ of in-twins in $\overrightarrow{U}$ that are not already in $\mathcal{B}$ do
4     if all the vertices of $I$ are in concerning paths then
5         Add $|I|-1$ vertices of $I$ to $\mathcal{B}$, prioritizing vertices in unfixable paths
6     else
7         Add $|I|-1$ vertices of $I$ to $\mathcal{B}$, prioritizing vertices in the cycle $\overrightarrow{C}$ or in
          concerning paths, if there are any
8 return $\mathcal{B}$
```

Explanation of Algorithm 2. The algorithm will compute a metric basis $\mathcal{B}$ of an orientation $\overrightarrow{U}$ of a unicyclic graph U in linear-time. The first thing we do is to add every source in $\overrightarrow{U}$ to $\mathcal{B}$ (line 1). We will also manage the sets of in-twins in $\overrightarrow{U}$ (lines 3–7), which we need to do after taking care of some special cases that might influence the choice of in-twins. When we have the choice, we prioritize taking in-twins that are in the cycle to guarantee reachability of vertices in the cycle. Note that those two sets (along with the right priority) are enough in most cases.

Algorithm 3: Special cases of Algorithm 2.

```
1  if the cycle $\overrightarrow{C}$ has no sink, there is no in-arc coming from outside of $C$, and no
   vertex of $C$ is in a set of in-twin then
2      Add $c_1$ to $\mathcal{B}$
3  if the cycle $\overrightarrow{C}$ has no sink, there is exactly one in-arc $\overrightarrow{uc_i}$ with $u \notin \overrightarrow{C}$, no vertex
   $c_j$ with $j \neq i$ is an in-twin or has in-arc coming from outside of $\overrightarrow{C}$, and $u$ has
   an out-neighbour $v$ with $N^-(v) = \{u\}$ then
4      Add $c_i$ to $\mathcal{B}$
5  if the cycle $\overrightarrow{C}$ has exactly one source $c_i$ then
6      if the one sink is either $c_{i-1}$ or $c_{i+1}$, and no vertex $c_j$ with $j \neq i$ is an
       in-twin or has an in-arc coming from outside of $\overrightarrow{C}$ then
7          Add $c_{i-1}$ to $\mathcal{B}$
8      else if the one sink is $c_{i+k}$ with $k > 1$, $|\overrightarrow{C}| \geq 2k$, $c_{i+k-1}$ (resp. $c_{i+k+1}$) has
       an out-neighbour $v$ such that $N^-(v) = \{c_{i+k-1}\}$ (resp. $N^-(v) = \{c_{i+k+1}\}$),
       no vertex in $\{c_{i-1}, c_{i-2}, \ldots, c_{i+k}\}$ (resp. $\{c_{i+1}, c_{i+2}, \ldots, c_{i+k}\}$) has an
       in-arc, and no vertex in $\{c_{i-2}, c_{i-3}, \ldots, c_{i+k+1}\}$ (resp.
       $\{c_{i+2}, c_{i+3}, \ldots, c_{i+k-1}\}$) is an in-twin then
9          Add $c_{i-1}$ (resp. $c_{i+1}$) to $\mathcal{B}$
10     else if the one sink is $c_{i+k}$ with $k > 1$, $|\overrightarrow{C}| = 2k$, $c_{i+k-1}$ has an
       out-neighbour $v_-$ such that $N^-(v_-) = \{c_{i+k-1}\}$, $c_{i+k+1}$ has an
       out-neighbour $v_+$ such that $N^-(v_+) = \{c_{i+k+1}\}$, no vertex in $\overrightarrow{C}$ except $c_i$
       has an in-arc, no vertex in $\overrightarrow{C} \setminus \{c_i, c_{i-1}, c_{i+1}\}$ is an in-twin, and $c_{i-1}$ and
       $c_{i+1}$ are not in a set $I$ of in-twins verifying $|I| \geq 3$ then
11         Add $c_{i+k}$ to $\mathcal{B}$
12 if the cycle $\overrightarrow{C}$ has exactly two sources $c_i$ and $c_{i+2}$, $|\overrightarrow{C}| = 2k$ with $k > 2$, the two
   sinks are $c_{i+1}$ and $c_{i+1+k}$, no vertex from $\overrightarrow{C}$ except $c_i$ and $c_{i+2}$ is an in-twin or
   has an in-arc coming from outside of $\overrightarrow{C}$, there is at least one unfixable path, and
   there is no fixable path then
13     Add $c_{i+1}$ to $\mathcal{B}$
```

We then have to manage six specific cases (line 2). **Those special cases are handled in Algorithm 3**. The first two special cases occur when the cycle has no sink. First, if the cycle has no sink, no in-twin, and no arc coming from outside, then, we have to add one vertex of the cycle to $\mathcal{B}$ in order to maintain reachability (lines 1–2). Then, if the cycle has no sink, only one in-arc $\overrightarrow{uc_i}$ is coming from outside of it, and there is a vertex v with $N^-(v) = \{u\}$, then, we have to add either c_i or v to $\mathcal{B}$ in order to resolve them (lines 3–4).

The next three special cases occur when the cycle has one sink. First, if there is only one sink in the cycle, it is an out-neighbour of the source, and no vertex from the cycle apart from the source is an in-twin or has an in-arc coming from outside of the cycle, then we need to add one of the out-neighbours of the source in the cycle to $\mathcal{B}$ in order to resolve them (lines 6–7).

Then, there are two specific cases when the cycle has one sink, both based on the same principle. Both happen when the source is c_i, the sink is c_{i+k}, it has no in-arc, and the cycle contains at least $2k$ vertices. In the fourth special case (lines 8–9), the vertex c_{i+k-1} has an out-neighbour v verifying $N^-(v) = \{c_{i+k-1}\}$. We can see that, if no vertex in the other path from c_i to c_{i+k} (the path going through $c_{i-1}, c_{i-2}, \ldots, c_{i+k+1}$) is in $\mathcal{B}$, then, v and c_{i+k} will not be resolved. Those vertices can be added to $\mathcal{B}$ if they have an in-arc or if they are an in-twin (they will have priority). However, note that c_{i-1} might be an in-twin of c_{i+1}, in which case it should be added to $\mathcal{B}$, resolving the conflict. Hence, if none of $c_{i-1}, c_{i-2}, \ldots, c_{i+k+1}$ has an in-arc or is an in-twin, then, we can add c_{i-1} to $\mathcal{B}$ in order to resolve v and c_{i+k}. Note that, in this case, in comparison to just the sources and the resolution of sets of in-twins, we add one more vertex to $\mathcal{B}$ if c_{i-1} is the only in-twin of c_{i+1}. The same reasoning can be made with the symmetric case.

The fifth special case (lines 10–11) occurs when the cycle contains exactly $2k$ vertices and both c_{i+k-1} and c_{i+k+1} have an out-neighbour (respectively v_- and v_+) with in-degree 1: the pairs of vertices (v_-, c_{i+k}) and (v_+, c_{i+k}) might not be resolved. We can see that any in-arc or in-twin along a path from c_i to c_{i+k} will resolve c_{i+k} and the v pendant on the other path (thus either fully resolving those two pairs, or bringing us back to the previous special case), except if c_{i-1} and c_{i+1} are the only in-twins in the cycle and if they do not have another in-twin. Hence, if no vertex from the cycle except c_i has an in-arc, no vertex from the cycle except c_i, c_{i-1} and c_{i+1} is an in-twin, and c_{i-1} and c_{i+1} do not have another in-twin, then, we need to add at least one more vertex to $\mathcal{B}$ in order to resolve the two pairs of vertices, and adding c_{i+k} does exactly that.

Finally, the sixth special case is more complex (lines 12–13 and consideration in the choice of in-twins). Assume that the cycle $\overrightarrow{C}$ is of even length n, has neither in-twin nor in-arc coming from outside (except the sources), and that there are two sinks in the $\overrightarrow{C}$: one at distance 1 from the sources, and the other at the opposite end of $\overrightarrow{C}$. Now, if the first sink has spanning concerning paths, then, the second cycle and the endpoints of those concerning paths might not be resolved, since they are at the same distance $(\frac{n}{2} - 1)$ of both sources of $\overrightarrow{C}$. Thus, we need to apply a strategy in order to resolve those vertices while trying to not add a supplementary vertex to $\mathcal{B}$. This is done by considering the two kinds of concerning paths, and having a priority in the selection of in-twins.

All the other cases of the cycle are already resolved through the sources and in-twins steps.

Theorem 6 (∗). *Algorithm 2 computes a metric basis of an orientation of a unicyclic graph in linear time.*

4 Modular Width

In a digraph G, a set $X \subseteq V(G)$ is a *module* if every vertex not in X 'sees' all vertices of X in the same way. More precisely, for each $v \in V(G) \setminus X$ one of the following holds: (i) $(v, x), (x, v) \in E(G)$ for all $x \in X$, (ii) $(v, x), (x, v) \notin E(G)$ for all $x \in X$, (iii) $(v, x) \in E(G)$ and $(x, v) \notin E(G)$ for all $x \in X$, (iv) $(v, x) \notin E(G)$ and $(x, v) \in E(G)$ for all $x \in X$. The singleton sets, $\emptyset$, and $V(G)$ are trivially modules of G. We call the singleton sets the *trivial modules* of G.

The graph $G[X]$ where X is a module of G is called a *factor* of G. A family $\mathcal{X} = \{X_1, \ldots, X_s\}$ is a *factorization* of G if $\mathcal{X}$ is a partition of $V(G)$, and each X_i is a module of G. If X and Y are two non-intersecting modules, then the relationship between $x \in X$ and $y \in Y$ is one of (i)-(iv) and always the same no matter which vertices x and y are exactly. Thus, given a factorization $\mathcal{X}$, we can identify each module with a vertex, and connect them to each other according to the arcs between the modules. More formally, we define the *quotient* $G/\mathcal{X}$ with respect to the factorization $\mathcal{X}$ as the graph with the vertex set $\mathcal{X} = \{X_1, \ldots, X_s\}$ and $(X_i, X_j) \in E(G/\mathcal{X})$ if and only if $(x_i, x_j) \in E(G)$ where $x_i \in X_i$ and $x_j \in X_j$. A quotient depicts the connections of the different modules of a factorization to each other while omitting the internal structure of the factors. Each factor itself can be factorized further (as long as it is nontrivial, i.e. not a single vertex). By factorizing the graph G and its factors until no further factorization can be done, we obtain a *modular decomposition* of G. The *width* of a decomposition is the maximum number of sets in a factorization (or equivalently, the maximum number of vertices in a quotient) in the decomposition. The *modular width* of G is defined as the minimum width over all possible modular decompositions of G, and we denote it by $\mathrm{mw}(G)$. An optimal modular decomposition of a digraph can be computed in linear time [21]. METRIC DIMENSION for undirected graphs was shown to be fixed parameter tractable when parameterized by modular width by Belmonte et al. [2]. We will generalize their algorithm to directed graphs.

The following result lists several useful observations.

Proposition 7 $(*)$**.** *Let $\mathcal{X} = \{X_1, \ldots, X_s\}$ be a factorization of G, and let $W \subseteq V(G)$ be a resolving set of G.*

(i) For all $x, y \in X_i$ and $z \in X_j$, $i \neq j$, we have $\mathrm{dist}_G(x, z) = \mathrm{dist}_G(y, z)$ and $\mathrm{dist}_G(z, x) = \mathrm{dist}_G(z, y)$.
(ii) For all $x \in X_i$ and $y \in X_j$, $i \neq j$, we have $\mathrm{dist}_G(x, y) = \mathrm{dist}_{G/\mathcal{X}}(X_i, X_j)$.
(iii) For all $x, y \in V(G)$ we have either $\mathrm{dist}_G(x, y) \leq \mathrm{mw}(G)$ or $\mathrm{dist}_G(x, y) = \infty$.
(iv) The set $\{X_i \in \mathcal{X} \mid W \cap X_i \neq \emptyset\}$ is a resolving set of the quotient $G/\mathcal{X}$.
(v) For all distinct $x, y \in X_i$, where $X_i \in \mathcal{X}$ is nontrivial, we have $\mathrm{dist}_G(w, x) \neq \mathrm{dist}_G(w, y)$ for some $w \in W \cap X_i$.
(vi) Let $w_1, w_2 \in X_i$. If $\mathrm{dist}_G(w_1, x) \neq \mathrm{dist}_G(w_2, x)$, then $x \in X_i$ and $\mathrm{dist}_G(w_1, x) \neq \mathrm{dist}_G(w_1, y)$ or $\mathrm{dist}_G(w_2, x) \neq \mathrm{dist}_G(w_2, y)$ for each $y \notin X_i$.

The basic idea of our algorithm (and that of [2]) is to compute metric bases that satisfy certain conditions for the factors and combine these local solutions into a global solution. We know that nontrivial modules must contain elements of a resolving set, as modules must be resolved locally (Proposition 7 (i)). While combining the local solutions of nontrivial modules, we need to make sure that a vertex $x \in X_i$, where X_i is nontrivial, is resolved from all $y \notin X_i$. If x and y are resolved as described in Proposition 7 (vi), then we need to do nothing special.

However, if $x \in X_i$ is such that $\mathrm{dist}_G(w,x) = d$ for all $w \in W_i$ and a fixed $d \in \{1,\dots,\mathrm{mw}(G),\infty\}$, there might exist a vertex $y \notin X_i$ such that W_i does not resolve x and y. We call such a vertex x *d-constant* (with respect to W_i). We need to keep track of d-constant vertices and make sure they are resolved when we combine the local solutions. There are at most $\mathrm{mw}(G)+1$ d-constant vertices in each factor due to Proposition 7 (iii). We need to also make sure vertices in different modules that contain no elements of the solution set are resolved. To do this, we might need to include some vertices from the trivial modules in addition to the vertices we have included from the nontrivial modules.

In the algorithm presented in [2], the problems described above are dealt with by computing values $w(H,p,q)$ for every factor H, where $w(H,p,q)$ is the minimum cardinality of a resolving set of H (with respect to the distance in G) where some vertex is 1-constant iff $p = \mathit{true}$ and some vertex is 2-constant iff $q = \mathit{true}$ (for undirected graphs these are the only two relevant cases). The same values are then computed for the larger graph by combining different solutions of the factors and taking their minimum. Our generalization of this algorithm is along the same lines as the original, however, we have more boolean values to keep track of. One difference to the techniques of the original algorithm is that we do not use the auxiliary graphs Belmonte et al. use. These auxiliary graphs were needed to simulate the distances of the vertices of a factor in G as opposed to only within the factor. In our approach, we simply use the distances in G.

Theorem 8 (∗). *The metric dimension of a digraph G with $\mathrm{mw}(G) \le t$ can be computed in time $\mathcal{O}(t^5 2^{t^2} n + n^3 + m)$ where $n = |V(G)|$ and $m = |E(G)|$.*

Proof (sketch). Let us consider one level of an optimal modular decomposition of G. Let H be a factor somewhere in the decomposition, and let $\mathcal{X} = \{X_1,\dots,X_s\}$ be the factorization of H according to the modular decomposition. For the graph H (and its nontrivial factors $H[X_i]$) we denote by $w(H,\mathbf{p})$ the minimum cardinality of a set $W \subseteq V(H)$ such that

(i) W resolves $V(H)$ in G,
(ii) $\mathbf{p} = (p_1,\dots,p_{\mathrm{mw}(G)},p_\infty)$ where $p_d = \mathit{true}$ if and only if H contains a d-constant vertex with respect to W.

If such a set does not exist, then $w(H,\mathbf{p}) = \infty$. In order to compute the values $w(H,\mathbf{p})$, we next introduce the auxiliary values $\omega(\mathbf{p},I,P)$. The values $w(H[X_i],\mathbf{p})$ are assumed to be known for all $\mathbf{p}$ and nontrivial modules X_i. Let the factorization $\mathcal{X}$ be labeled so that the modules X_i are trivial for $i \in \{1,\dots,h\}$ and nontrivial for $i \in \{h+1,\dots,s\}$. Let $I \subseteq \{1,\dots,h\}$ and $P = \left(\mathbf{p}^{h+1},\dots\mathbf{p}^s\right)^T$.

We define $\omega(\mathbf{p},I,P) = |I| + \sum_{i=h+1}^{s} w(H[X_i],\mathbf{p}^i)$ if the conditions (a)–(d) hold. In what follows, a representative of a module X_i is denoted by x_i.

(a) The set $Z = \{X_i \in \mathcal{X} \,|\, i \in I \cup \{h+1,\dots,s\}\}$ resolves the quotient $H/\mathcal{X}$ with respect to the distances in G.
(b) For $d \in \{1,\dots,\mathrm{mw}(G),\infty\}$ and $i \in \{h+1,\dots,s\}$, if $p^i_d = \mathit{true}$, then for each trivial module $X_j = \{x_j\}$ where $j \notin I$ we have $\mathrm{dist}_G(x_i,x_j) \neq d$ or there exists $X_k \in Z \setminus \{X_i\}$ such that $\mathrm{dist}_G(x_k,x_i) \neq \mathrm{dist}_G(x_k,x_j)$.

(c) For $d_1, d_2 \in \{1, \ldots, \mathrm{mw}(G), \infty\}$ and distinct $i, j \in \{h+1, \ldots, s\}$, if $p^i_{d_1} = p^j_{d_2} = true$, then $\mathrm{dist}_G(x_i, x_j) \neq d_1$, or $\mathrm{dist}_G(x_j, x_i) \neq d_2$, or there exists $X_k \in Z \setminus \{X_i, X_j\}$ such that $\mathrm{dist}_G(x_k, x_i) \neq \mathrm{dist}_G(x_k, x_j)$.

(d) For all $d \in \{1, \ldots, \mathrm{mw}(G), \infty\}$, we have $p_d = true$ (in $\mathbf{p}$) if and only if for some $i \in \{1, \ldots, h\} \setminus I$ we have $\mathrm{dist}_G(x_j, x_i) = d$ for all $X_j \in Z$, or for some $i \in \{h+1, \ldots, s\}$ we have $p^i_d = true$ and $\mathrm{dist}_G(x_j, x_i) = d$ for all $X_j \in Z \setminus \{X_i\}$.

If these conditions cannot be met, then we set $\omega(\mathbf{p}, I, P) = \infty$.

When we know all the values $\omega(\mathbf{p}, I, P)$, we can easily calculate $w(H, \mathbf{p})$ since $w(H, \mathbf{p}) = \min_{I,P} \omega(\mathbf{p}, I, P)$ (proof omitted due to lack of space).

To conclude the proof, we note that $w(G, \mathbf{p})$ is the minimum cardinality of a resolving set that gives some vertices the specific distance combinations according to $\mathbf{p}$. Thus, the metric dimension of G is $\min w(G, \mathbf{p})$ where the minimum is taken over all $\mathbf{p}$ such that $p_\infty = false$. □

5 NP-Hardness for Restricted DAGs

We now complement the hardness result from [1], which was for bipartite DAGs of maximum degree 8 and maximum distance 4.

Theorem 9 (∗). Metric Dimension *is NP-complete, even on planar triangle-free DAGs of maximum degree 6 and maximum distance 4.*

Proof (sketch). We reduce from Vertex Cover on 2-connected planar cubic graphs, which is known to be NP-complete [23, Theorem 4.1].

Given a 2-connected planar cubic graph G, we construct a DAG G' as follows. First of all, note that by Petersen's theorem, G contains a perfect matching $M \subset E(G)$, that can be constructed in polynomial time. A planar embedding of G can also be constructed in polynomial time, so we fix one. We let $V(G') = V(G) \bigcup_{e=uv \in E(G)} \{a_e, b_e, c_e, d^u_e, d^v_e\} \bigcup_{e=uv \in M} \{f_e, g_e, h_e\}$. For every edge $e = uv$ of G, we add the arcs $\{\overrightarrow{a_e b_e}, \overrightarrow{b_e c_e}, \overrightarrow{c_e d^u_e}, \overrightarrow{c_e d^v_e}, \overrightarrow{u d^u_e}, \overrightarrow{v d^v_e}\}$. For every edge $e = uv$ of the perfect matching M of G, assuming the neighbours of u (in the clockwise cyclic order with respect to the planar embedding of G) are v, x, y and those of v are u, s, t, we arbitrarily fix one side of the edge uv to place the vertices f_e, g_e and h_e (say, on the side that is close to the edges ux and vt). We add the arcs $\{\overrightarrow{f_e g_e}, \overrightarrow{g_e c_e}, \overrightarrow{g_e h_e}, \overrightarrow{h_e u}, \overrightarrow{h_e v}, \overrightarrow{c_e c_{uy}}, \overrightarrow{c_e c_{vs}}, \overrightarrow{h_e c_{ux}}, \overrightarrow{h_e c_{vt}}\}$.

Using the embedding of G, G' can also be drawn in a planar way, it has maximum degree 6 (the vertices of type c_e are of degree 6), has no triangles, and no shortest directed path of length 4.

We claim that G has a vertex cover of size at most k if and only if G' has metric dimension at most $k + |E(G)| + |M|$ (proof omitted due to lack of space). □

6 Conclusion

Metric Dimension can be solved in polynomial time on outerplanar graphs, using an involved algorithm [7]. Can one generalize our algorithms for trees and unicyclic graphs to solve Metric Dimension for directed (or at least, oriented) outerplanar graphs in polynomial time? Extending our algorithm to cactus graphs already seems nontrivial.

Is Metric Dimension NP-hard on planar bipartite subcubic DAGs?

Also, it would be interesting to see which hardness results known for Metric Dimension of undirected graphs also hold for DAGs, or for oriented graphs.

References

1. Araujo, J., et al.: On finding the best and worst orientations for the metric dimension. Algorithmica 1–41 (2023)
2. Belmonte, R., Fomin, F.V., Golovach, P.A., Ramanujan, M.S.: Metric dimension of bounded tree-length graphs. SIAM J. Discret. Math. **31**(2), 1217–1243 (2017)
3. Blumenthal, L.M.: Theory and Applications of Distance Geometry. Oxford University Press, Oxford (1953)
4. Chartrand, G., Eroh, L., Johnson, M.A., Oellermann, O.R.: Resolvability in graphs and the metric dimension of a graph. Discret. Appl. Math. **105**(1), 99–113 (2000)
5. Chartrand, G., Raines, M., Zhang, P.: The directed distance dimension of oriented graphs. Math. Bohem. **125**, 155–168 (2000)
6. Dailly, A., Foucaud, F., Hakanen, A.: Algorithms and hardness for metric dimension on digraphs. arXiv preprint arXiv:2307.09389 (2023)
7. Díaz, J., Pottonen, O., Serna, M.J., van Leeuwen, E.J.: Complexity of metric dimension on planar graphs. J. Comput. Syst. Sci. **83**(1), 132–158 (2017)
8. Eppstein, D.: Metric dimension parameterized by max leaf number. J. Graph Algorithms Appl. **19**(1), 313–323 (2015)
9. Epstein, L., Levin, A., Woeginger, G.J.: The (weighted) metric dimension of graphs: hard and easy cases. Algorithmica **72**(4), 1130–1171 (2015)
10. Fernau, H., Heggernes, P., van't Hof, P., Meister, D., Saei, R.: Computing the metric dimension for chain graphs. Inf. Process. Lett. **115**(9), 671–676 (2015)
11. Foucaud, F., Mertzios, G.B., Naserasr, R., Parreau, A., Valicov, P.: Identification, location-domination and metric dimension on interval and permutation graphs. II. Algorithms and complexity. Algorithmica **78**(3), 914–944 (2017)
12. Galby, E., Khazaliya, L., Inerney, F.M., Sharma, R., Tale, P.: Metric dimension parameterized by feedback vertex set and other structural parameters. In: 47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022, 22–26 August 2022, Vienna, Austria. LIPIcs, vol. 241, pp. 51:1–51:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022)
13. Gima, T., Hanaka, T., Kiyomi, M., Kobayashi, Y., Otachi, Y.: Exploring the gap between treedepth and vertex cover through vertex integrity. Theoret. Comput. Sci. **918**, 60–76 (2022)
14. Harary, F., Melter, R.A.: On the metric dimension of a graph. Ars Combin. **2**, 191–195 (1976)
15. Hartung, S., Nichterlein, A.: On the parameterized and approximation hardness of metric dimension. In: Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5–7 June 2013, pp. 266–276. IEEE Computer Society (2013)

16. Hoffmann, S., Elterman, A., Wanke, E.: A linear time algorithm for metric dimension of cactus block graphs. Theoret. Comput. Sci. **630**, 43–62 (2016)
17. Hoffmann, S., Wanke, E.: Metric Dimension for gabriel unit disk graphs Is NP-complete. In: Bar-Noy, A., Halldórsson, M.M. (eds.) ALGOSENSORS 2012. LNCS, vol. 7718, pp. 90–92. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36092-3_10
18. Khuller, S., Raghavachari, B., Rosenfeld, A.: Landmarks in graphs. Discret. Appl. Math. **70**(3), 217–229 (1996)
19. Li, S., Pilipczuk, M.: Hardness of metric dimension in graphs of constant treewidth. Algorithmica **84**(11), 3110–3155 (2022)
20. Lobstein, A.: Watching systems, identifying, locating-dominating and discriminating codes in graphs: a bibliography (2022). https://www.lri.fr/~lobstein/debutBIBidetlocdom.pdf
21. McConnell, R.M., de Montgolfier, F.: Linear-time modular decomposition of directed graphs. Discret. Appl. Math. **145**(2), 198–209 (2005)
22. Melter, R.A., Tomescu, I.: Metric bases in digital geometry. Comput. Vision Graph. Image Process. **25**(1), 113–121 (1984)
23. Mohar, B.: Face covers and the genus problem for apex graphs. J. Combin. Theory Ser. B **82**(1), 102–117 (2001)
24. Moscarini, M.: Computing a metric basis of a bipartite distance-hereditary graph. Theoret. Comput. Sci. **900**, 20–24 (2022)
25. Oellermann, O.R., Peters-Fransen, J.: The strong metric dimension of graphs and digraphs. Discret. Appl. Math. **155**(3), 356–364 (2007)
26. Poisson, C., Zhang, P.: The metric dimension of unicyclic graphs. J. Combin. Math. Combin. Comput. **40**, 17–32 (2002)
27. Rajan, B., Rajasingh, I., Cynthia, J.A., Manuel, P.: Metric dimension of directed graphs. Int. J. Comput. Math. **91**(7), 1397–1406 (2014)
28. Sedlar, J., Škrekovski, R.: Bounds on metric dimensions of graphs with edge disjoint cycles. Appl. Math. Comput. **396**, 125908 (2021)
29. Sedlar, J., Škrekovski, R.: Vertex and edge metric dimensions of unicyclic graphs. Discret. Appl. Math. **314**, 81–92 (2022)
30. Slater, P.J.: Leaves of trees. Congressius Numer. **14**, 549–559 (1975)
31. Steiner, R., Wiederrecht, S.: Parameterized algorithms for directed modular width. In: Changat, M., Das, S. (eds.) CALDAM 2020. LNCS, vol. 12016, pp. 415–426. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-39219-2_33

Degreewidth: A New Parameter for Solving Problems on Tournaments

Tom Davot[1(✉)], Lucas Isenmann[2], Sanjukta Roy[3], and Jocelyn Thiebaut[4]

[1] Université de Technologie de Compiègne, CNRS, Heudiasyc, Compiègne, France
tom.davot@hds.utc.fr

[2] Université de Montpellier, Montpellier, France
lucas.isenmann@umontpellier.fr

[3] Pennsylvania State University, State College, USA
sanjukta@psu.edu

[4] Faculty of Information Technology, Czech Technical University in Prague, Prague, Czech Republic
jocelyn.thiebaut@cvut.cz

Abstract. In the paper, we define a new parameter for tournaments called degreewidth which can be seen as a measure of how far is the tournament from being acyclic. The degreewidth of a tournament T denoted by $\Delta(T)$ is the minimum value k for which we can find an ordering $\langle v_1, \dots, v_n \rangle$ of the vertices of T such that every vertex is incident to at most k backward arcs (*i.e.* an arc (v_i, v_j) such that $j < i$). Thus, a tournament is acyclic if and only if its degreewidth is zero. Additionally, the class of sparse tournaments defined by Bessy *et al.* [ESA 2017] is exactly the class of tournaments with degreewidth one.

We study computational complexity of finding degreewidth. We show it is NP-hard and complement this result with a 3-approximation algorithm. We provide a $O(n^3)$-time algorithm to decide if a tournament is sparse, where n is its number of vertices.

Finally, we study classical graph problems Dominating Set and Feedback Vertex Set parameterized by degreewidth. We show the former is fixed-parameter tractable whereas the latter is NP-hard even on sparse tournaments. Additionally, we show polynomial time algorithm for Feedback Arc Set on sparse tournaments.

Keywords: Tournaments · NP-hardness · graph-parameter · feedback arc set · approximation algorithm · parameterized algorithms

1 Introduction

A tournament is a directed graph such that there is exactly one arc between each pair of vertices. Tournaments form a very rich subclass of digraphs which

Sanjukta Roy was affiliated to Faculty of Information Technology, Czech Technical University in Prague when majority of this work was done. Jocelyn Thiebaut was supported by the CTU Global postdoc fellowship program.

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 246–260, 2023.
https://doi.org/10.1007/978-3-031-43380-1_18

has been widely studied both from structural and algorithmic point of view [4]. Unlike for complete graphs, a number of classical problems remain difficult in tournaments and therefore interesting to study. These problems include Dominating Set [14], Winner Determination [22], or maximum cycle packing problems. For example, Dominating Set is W[2]-hard on tournaments with respect to solution size [14]. However, many of these problems become easy on acyclic tournaments (*i.e.* without directed cycle). Therefore, a natural question that arises is whether these problems are easy to solve on tournaments that are close to being acyclic. The phenomenon of a tournament being "close to acyclic" can be captured by minimum size of a *feedback arc set* (fas). A fas is a collection of arcs that, when removed from the digraph (or, equivalently, reversed) makes it acyclic. This parameter has been widely studied, for numerous applications in many fields, such as circuit design [19], or artificial intelligence [5,13]. However, the problem of finding a minimum fas on tournaments (the problem is then called *FAST* for Feedback Arc Set in Tournaments), remained opened for over a decade before being proven NP-complete [3,10]. From the approximability point of view, van Zuylen and Williamson [25] provided a 2-approximation of FAST, and Kenyon-Mathieu and Schudy [21] a PTAS algorithm. On the parameterized-complexity side, Feige [15] as well as Karpinski and Schudy [20] independently proved an $2^{O(\sqrt{k})} + n^{O(1)}$ running-time algorithm. Another way to define FAST is to consider the problem of finding an ordering of the vertices $\langle v_1, \ldots, v_n \rangle$ minimising the number of arcs (v_i, v_j) with $j < i$; such arcs are called *backward arcs*. Then, it is easy to see that a tournament is acyclic if and only if it admits an ordering with no backward arcs. Several parameters exploiting an ordering with specific properties have been studied in this sense [18] such as the cutwidth. Given an ordering of vertices, for each prefix of the ordering we associate a cut defined as the set of backward arcs with head in the prefix and tail outside of it. Then cutwidth is the minimum value, among all the orderings, of the maximum size of any possible cut w.r.t the ordering (a formal definition is introduced in next section). It is well-known that computing cutwidth is NP-complete [17], and has an $O(\log^2(n))$-approximation on general graphs [23]. Specifically on tournaments, one can compute an optimal ordering for the cutwidth by sorting the degrees according to the in-degrees [16].

In this paper, we propose a new parameter called *degreewidth* using the concept of backward arcs in an ordering of vertices. Degreewidth of a tournament is the minimum value, among all the orderings, of the maximum number of backward arcs incident to a vertex. Hence, an acyclic tournament is a tournament with degreewidth zero. Furthermore, one can notice that tournaments with degreewidth at most one are the same as the *sparse tournaments* introduced in [8,24]. A tournament is *sparse* if there exists an ordering of vertices such that the backward arcs form a matching. It is known that computing a maximum sized arc-disjoint packing of triangles and computing a maximum sized arc-disjoint packing of cycles can be done in polynomial time [7] on sparse tournaments.

To the best of our knowledge this paper is the first to study the parameter degreewidth. As we will see in the next part, although having similarities with the cutwidth, this new parameter differs in certain aspects. We first study structural and computational aspects of degreewidth. Then, we show how it can be used to solve efficiently some classical problems on tournaments.

Our Contributions and Organization of the Paper. Next section provides the formal definition of degreewidth and some preliminary observations. In Sect. 3, we first study the degreewidth of a special class of tournaments, called regular tournaments, of order $2k+1$ and prove they have degreewidth k. We then prove that it is NP-hard to compute the degreewidth in general tournaments. We finally give a 3-approximation algorithm to compute this parameter which is tight in the sense that it cannot produce better than 3-approximation for a class of tournaments.

Then in Sect. 4, we focus on tournaments with degreewidth one, *i.e.*, the sparse tournaments. Note that it is claimed in [8] that there exists a polynomial-time algorithm for finding such ordering, but the only available algorithm appearing in [24, Lemma 35.1, p.97] seems to be incomplete (see discussion Subsect. 4.2). We first define a special class of tournaments that we call U-tournaments. We prove there are only two possible sparse orderings for such tournaments. Then, we give a polynomial time algorithm to decide if a tournament is sparse by carefully decomposing it into U-tournaments.

Finally, in Sect. 5 we study degreewidth as a parameter for some classical graph problems. First, we show an FPT algorithm for Dominating Set w.r.t degreewidth. Then, we focus on tournaments with degreewidth one. We design an algorithm running in time $O(n^3)$ to compute a Feedback Arc Set on tournaments on n vertices with degreewidth one. However, we show that Feedback Vertex Set remains NP-complete on this class of tournaments.

Due to paucity of space the missing proofs are deferred to full version [12].

2 Preliminaries

2.1 Notations

In the following, all the digraphs are simple, that is without self-loop and multiple arcs sharing the same head and tail, and all cycles are directed cycles. The *underlying graph* of a digraph D is an undirected graph obtained by replacing every arc of D by an edge. Furthermore, we use $[n]$ to denote the set $\{1, 2, \dots, n\}$.

A tournament is a digraph where there is exactly one arc between each pair of vertices. It can alternatively be seen as an orientation of the complete graph. Let T be a tournament with vertex set $\{v_1, \dots, v_n\}$. We denote $N^+(v)$ the *out-neighbourhood* of a vertex v, that is the set $\{u \mid (v, u) \in A(T)\}$. Then, T being a tournament, the *in-neighbourhood* of the vertex v denoted $N^-(v)$ corresponds to $V(T) \setminus (N^+(v) \cup \{v\})$. The *out-degree* (resp. *in-degree*) of v denoted $d^+(v)$ (resp. $d^-(v)$) is the size of its out-neighbourhood (resp. in-neighbourhood).

A tournament T of order $2k + 1$ is *regular* if for any vertex v, we have $d^+(v) = d^-(v) = k$. Let X be a subset of $V(T)$. We denote by $T - X$ the subtournament induced by the vertices $V(T) \setminus X$. Furthermore, when X contains only one vertex $\{v\}$ we simply write $T - v$ instead of $T - \{v\}$. We also denote by $T[X]$ the tournament induced by the vertices of X. Finally, we say that $T[X]$ *dominates* T if, for every $x \in X$ and every $y \in V(T) \setminus X$, we have $(x, y) \in A(T)$. For more definitions on directed graphs, please refer to [4].

Given a tournament T, we equip the vertices of T with is a strict total order $\prec_\sigma$. This operation also defines an ordering of the set of vertices denoted by $\sigma := \langle v_1, \dots, v_n\rangle$ such that $v_i \prec_\sigma v_j$ if and only if $i < j$. Given two distinct vertices u and v, if $u \prec_\sigma v$ we say that u is *before* v in σ; otherwise, u is *after* v in σ. Additionally, an arc (u, v) is said to be *forward* (resp. *backward*) if $u \prec_\sigma v$ (resp. $v \prec_\sigma u$). A topological ordering is an ordering without any backward arcs. A tournament that admits a topological ordering does not contain a cycle. Hence, it is said to be *acyclic*.

A *pattern* $p_1 := \langle v_1, \dots, v_k\rangle$ is a sequence of vertices that are consecutive in an ordering. Furthermore, considering a second pattern $p_2 := \langle u_1, \dots, u_{k'}\rangle$ where $\{v_1, \dots, v_k\}$ and $\{u_1, \dots, u_{k'}\}$ are disjoint, the pattern $\langle p_1, p_2\rangle$ is defined by $\langle v_1, \dots, v_k, u_1, \dots, u_{k'}\rangle$.

Degreewidth. Given a tournament T, an ordering σ of its vertices $V(T)$ and a vertex $v \in V(T)$, we denote $d_\sigma(v)$ to be the number of backward arcs incident to v in σ, that is $d_\sigma(v) := |\{u \mid u \prec_\sigma v,\, u \in N^+(v)\} \cup \{u \mid v \prec_\sigma u,\, u \in N^-(v)\}|$. Then, we define the degreewidth of a tournament with respect to the ordering σ, denoted by $\Delta_\sigma(T) := \max\{d_\sigma(v) \mid v \in V(T)\}$. Note that $\Delta_\sigma(T)$ is also the maximum degree of the underlying graph induced by the backward arcs of σ. Finally, we define the degreewidth $\Delta(T)$ of the tournament T as follows.

Definition 1. *The degreewidth of a tournament T, denoted $\Delta(T)$, is defined as $\Delta(T) := \min_{\sigma \in \Sigma(T)} \Delta_\sigma(T)$, where $\Sigma(T)$ is the set of possible orderings for $V(T)$.*

As mentioned before, this new parameter tries to measure how far a tournament is from being acyclic. Indeed, it is easy to see that a tournament T is acyclic if and only if $\Delta(T) = 0$. Additionally, when degreewidth of a tournament is one, it coincides with the notion of sparse tournaments, introduced in [8].

Remark. The definition of degreewidth naturally extends to directed graphs and we hope it will be an exciting parameter for problems on directed graphs. However, in this article we study this as a parameter for tournaments which is well-studied in various domains [2,9,22]. Moreover, degreewidth also gives a succinct representation of a tournament. Informally, sparse graphs[1] are graphs with a low density of edges. Hence, it may be surprising to talk about sparsity in tournaments. However, if a tournament on n vertices admits an ordering σ where the backward arcs form a matching, then it can be encoded by σ and the set of backward arcs (at most $n/2$). Thus, the size of the encoding for such tournament is $O(n)$, instead of $O(n^2)$. For a tournament with degreewidth k, the same reasoning implies that it can be encoded in $O(kn)$ space.

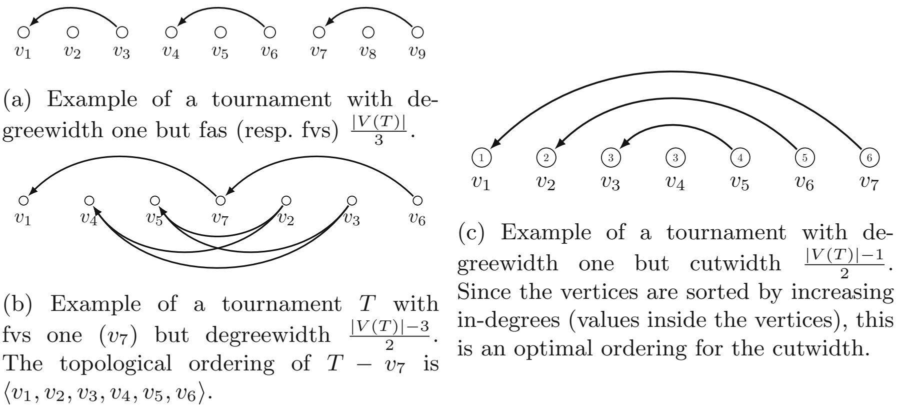
[1] Not to be confused with sparse tournaments that has an arc between every pair of vertices, hence, is not a sparse graph.

2.2 Links to Other Parameters

Feedback Arc/Vertex Set. A *feedback arc set* (fas) is a collection of arcs that, when removed from the digraph (or, equivalently, reversed) makes it acyclic. The size of a minimum fas is considered for measuring how far the digraph is from being acyclic. In this context, degreewidth comes as a promising alternative. Finding a small subset of arcs hitting all substructures (in this case, directed cycles) of a digraph is one of the fundamental problems in graph theory. Note that we can easily bound the degreewidth of a tournament by its minimum fas f.

Observation 1. *For any tournament T, we have $\Delta(T) \leq |f|$.*

Note however that the opposite is not true; it is possible to construct tournaments with small degreewidth but large fas, see Fig. 1(a).


v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9
v_1 v_4 v_5 v_7 v_2 v_3 v_6
1 2 3 3 4 5 6
v_1 v_2 v_3 v_4 v_5 v_6 v_7

(a) Example of a tournament with degreewidth one but fas (resp. fvs) $\frac{|V(T)|}{3}$.

(b) Example of a tournament T with fvs one (v_7) but degreewidth $\frac{|V(T)|-3}{2}$. The topological ordering of $T - v_7$ is $\langle v_1, v_2, v_3, v_4, v_5, v_6 \rangle$.

(c) Example of a tournament with degreewidth one but cutwidth $\frac{|V(T)|-1}{2}$. Since the vertices are sorted by increasing in-degrees (values inside the vertices), this is an optimal ordering for the cutwidth.

Fig. 1. Link between degreewidth and other parameters. All the non-depicted arcs are forward.

Similarly, a *feedback vertex set* (fvs) consists of a collection of vertices that, when removed from the digraph makes it acyclic. However, – unlike the feedback arc set – the link between feedback vertex set and degreewidth seems less clear; we can easily construct tournaments with low degreewidth and large fvs (see Fig. 1(a)) as well as large degreewidth and small fvs (see Fig. 1(b)).

Cutwidth. Let us first recall the definition of the cutwidth of a digraph. Given an ordering $\sigma := \langle v_1, \ldots, v_n \rangle$ of the vertices of a digraph D, we say that a prefix of σ is a sequence of consecutive vertices $\langle v_1, \ldots, v_k \rangle$ for some $k \in [n]$. We associate for each prefix of σ a *cut* defined as the set of backward arcs with head in the prefix and tail outside of it. The *width* of the ordering σ is defined as the size of a maximum cut among all the possible prefixes of σ. The cutwidth of D, $ctw(D)$, is the minimum width among all orderings of the vertex set of D.

Intuitively, the difference between the cutwidth and the degreewidth is that the former focuses on the backward arcs going "above" the intervals between the vertices while the latter focuses on the backward arcs coming from and to the vertices themselves. Observe that for any tournament T, the degreewidth is bounded by a function of the cutwidth. Formally, we have the following

Observation 2. *For any tournament T, we have $\Delta(T) \leq 2ctw(T)$.*

Note however that the opposite is not true; it is possible to construct tournaments with small degreewidth but large cutwidth, see Fig. 1(c). We remark that the graph problems that we study parameterized by degreewidth, namely, minimising fas, fvs, and dominating set are FPT w.r.t cutwidth [1,11].

3 Degreewidth

In this section, we present some structural and algorithmical results for the computation of degreewidth. We first introduce the following lemma that provides a lower bound on the degreewith.

Lemma 1. *Let T be a tournament. Then $\Delta(T) \geq \min_{v \in V(T)} d^-(v)$ and $\Delta(T) \geq \min_{v \in V(T)} d^+(v)$.*

3.1 Degreewidth of Regular Tournaments

Theorem 1. *Let T be a regular tournament of order $2k+1$. Then $\Delta(T) = k$. Furthermore, for any ordering σ, by denoting u and v respectively the first and last vertices in σ, we have $d_\sigma(u) = d_\sigma(v) = k$.*

Note that regular tournaments contain many cycles; therefore it is not surprising that their degreewidth is large. This corroborates the idea that this parameter measures how far a tournament is from being acyclic.

3.2 Computational Complexity

We now show that computing the degreewidth of a tournament is NP-hard by defining a reduction from Balanced 3-SAT(4), proven NP-complete [6] where each clause contains exactly three unique literals and each variable occurs two times positively and two times negatively.

Let φ be a Balanced 3-SAT(4) formula with m clauses $c_1, \ldots, c_m$ and n variables $x_1, \ldots, x_n$. In the construction, we introduce several regular tournaments of size W or $\frac{W+1}{2} + n + m$, where W is value greater than $n^3 + m^3$. Note that $n + m$ is necessarily odd since $4n = 3m$. By taking a value $W = 3 \mod 4$, we ensure that every regular tournament of size W or $\frac{W+1}{2} + n + m$ has an odd number of vertices.

Construction 1. *Let φ be a* Balanced 3-SAT(4) *formula with m clauses $c_1, \ldots, c_m$ clauses and n variables $x_1, \ldots, x_n$ such that n is odd and m is even. Let $W = 3 \mod 4$ be an integer greater than $n^3 + m^3$. We construct a tournament T as follows.*

- *Create two regular tournaments A and D of order $\frac{W+1}{2} + m + n$ such that D dominates A.*
- *Create two regular tournaments B and C of order W such that A dominates $B \cup C$, B dominates C and $B \cup C$ dominates D.*

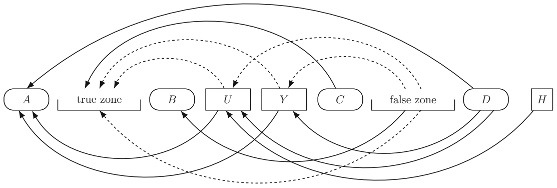


Fig. 2. Example of a nice ordering. A rectangle represents an acyclic tournament, while a rectangle with rounded corners represents a regular tournament. A plain arc between two patterns P and P' represents the fact that there is a backward arc between every pair of vertices $v \in P$ and $v' \in P'$. A dashed arc means some backward arcs may exist between the patterns.

- *Create an acyclic tournament* X *of order* $2n$ *with topological ordering* $\langle v_1, v'_1, \ldots, v_n, v'_n \rangle$ *such that* $A \cup C$ *dominates* X *and* X *dominates* $B \cup D$*.*
- *Create an acyclic tournament* Y *of order* $2m$ *with topological ordering* $\langle q_1, q'_1, \ldots, q_m, q'_m \rangle$ *such that* $B \cup D$ *dominates* Y *and* Y *dominates* $A \cup C$*.*
- *For each clause* c_ℓ *and each variable* x_i *of* φ*,*
  - *if* x_i *occurs positively in* c_ℓ*, then* $\{v_i, v'_i\}$ *dominates* $\{q_\ell, q'_\ell\}$*,*
  - *if* x_i *occurs negatively in* c_ℓ*, then* $\{q_\ell, q'_\ell\}$ *dominates* $\{v_i, v'_i\}$*,*
  - *if* x_i *does not occur in* c_ℓ*, then introduce the paths* (v_i, q_ℓ, v'_i) *and* (v'_i, q'_ℓ, v_i)*.*
- *Introduce an acyclic tournament* $U = \{u_i^p, \bar{u}_i^p \mid i \leq n, p \leq 2\}$ *of order* $4n$ *such that* U *dominates* $A \cup Y \cup C$ *and* $B \cup D$ *dominates* U*. For each variable* x_i*, add the following paths,*
  - *for all variable* $x_k \neq x_i$ *and all* $p \leq 2$*, introduce the paths* (v_k, u_i^p, v'_k) *and* $(v'_k, \bar{u}_i^p, v_k)$*,*
  - *introduce the paths* (v_i, u_i^1, v'_i)*,* (v'_i, u_i^2, v_i)*,* $(v_i, \bar{u}_i^1, v'_i)$ *and* $(v'_i, \bar{u}_i^2, v_i)$*.*
- *Finally, introduce an acyclic tournament* $H = \{h_1, h_2\}$ *with topological ordering* $\langle h_1, h_2 \rangle$ *and such that* $A \cup B \cup C \cup X \cup Y \cup D$ *dominates* H *and* H *dominates* U*.*

We call a vertex of X a *variable vertex* and a vertex of Y a *clause vertex*. Furthermore, we say that the vertices (v_i, v'_i) (resp. (q_ℓ, q'_ℓ)) is a *pair of variable vertices* (resp. *pair of clause vertices*).

Definition 2. *Let* T *be a tournament resulting from Construction 1. An ordering* σ *of* T *is nice if:*

- $\Delta_\sigma(A) = \frac{|A|-1}{2}$*,* $\Delta_\sigma(B) = \frac{|B|-1}{2}$*,* $\Delta_\sigma(C) = \frac{|C|-1}{2}$*, and* $\Delta_\sigma(D) = \frac{|D|-1}{2}$*,*
- σ *respects the topological ordering of* $U \cup Y$*,*
- $A \prec_\sigma B \prec_\sigma U \prec_\sigma Y \prec_\sigma C \prec_\sigma D \prec_\sigma H$*, and*
- *for any variable* x_i*, either* $A \prec_\sigma v_i \prec_\sigma v'_i \prec_\sigma B$ *or* $C \prec_\sigma v_i \prec_\sigma v'_i \prec_\sigma D$*.*

An example of a nice ordering is depicted in Fig. 2. Let σ be a nice ordering, we call the pattern corresponding to the vertices between A and B, the *true zone* and the pattern after the vertices of C the *false zone*. Let (q_ℓ, q'_ℓ) be a pair of clause vertices and let (v_i, v'_i) be a pair of variable vertices such that

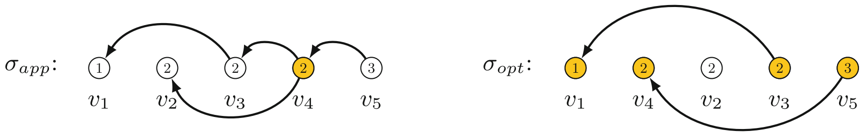


Fig. 3. Example of a tournament where the approximate algorithm can return an ordering σ_{app} (on the left) with degreewidth three while the optimal solution is one in σ_{opt} (on the right). Coloured vertices are the ones incident to the maximum number of backward arcs. all non-depicted arcs are forward arcs.

x_i occurs positively (resp. negatively) in c_ℓ in φ. We say that the pair (v_i, v_i') satisfies (q_ℓ, q_ℓ') if v_i and v_i' both belong to the true zone (resp. false zone). Note that there is no backward arc between $\{q_\ell, q_\ell'\}$ and $\{v_i, v_i'\}$ if and only if (v_i, v_i') satisfies (q_ℓ, q_ℓ'). Notice also that for any pair of variable vertices (v_i, v_i') such that x_i does not appear in c_ℓ and (v_i, v_i') is either in the true zone or in the false zone, then there is exactly two backward arcs between $\{q_\ell, q_\ell'\}$ and $\{v_j, v_j'\}$.

Let φ be an instance of Balanced 3-SAT(4) and T its tournament resulting from Construction 1. We show that φ is satisfiable if and only if there exists an ordering σ of T such that $\Delta_\sigma(T) < W + 2m + 3n + 4$, which yields the following.

Theorem 2. *Given a tournament T and an integer k, it is NP-complete to compute an ordering σ of T such that $\Delta_\sigma(T) \leq k$.*

3.3 An Approximation Algorithm to Compute Degreewidth

In this subsection, we prove that sorting the vertices by increasing in-degree is a tight 3-approximation algorithm to compute the degreewidth of a tournament. Intuitively, the reasons why it returns a solution not too far from the optimal are twofold. Firstly, observe that the only optimal ordering for acyclic tournaments (*i.e.* with degreewidth 0) is an ordering with increasing in-degrees. Secondly, this strategy also gives an optimal solution for cutwidth in tournaments.

Theorem 3. *Ordering the vertices by increasing order of in-degree is a tight 3-approximation algorithm to compute the degreewidth of a tournament (see Fig. 3).*

4 Results on Sparse Tournaments

In this section, we focus on tournaments with degreewidth one, called sparse tournaments. The main result of this section is that unlike in the general case, it is possible to compute in polynomial time a sparse ordering of a tournament (if it exists). We begin with an observation about sparse orderings (if it exists).

Lemma 2. *Let T be a sparse tournament of order $n > 4$ and σ be an ordering of its vertices. If σ is a sparse ordering, then for any vertex v such that $d^-(v) = i$, the only possible positions of v in σ are $\{i, i+1, i+2\} \cap [n]$.*

Note that Lemma 2 gives immediately an exponential running-time algorithm to decide if a tournament is sparse. However, we give in Subsect. 4.2 a polynomial running-time algorithm for this problem. Before that we study a useful subclass of sparse tournaments, we call the U-tournaments.

4.1 *U*-Tournaments

In this subsection, we study one specific type of tournaments called U-tournaments. Informally, they correspond to the acyclic tournaments where we reversed all the arcs of its Hamiltonian path.

Definition 3. *For any integer $n \geq 1$, we define U_n as the tournament on n vertices with $V(U_n) = \{v_1, v_2, \ldots, v_n\}$ and $A(U_n) = \{(v_{i+1}, v_i) \mid \forall i \in [n-1]\} \cup \{(v_i, v_j) \mid 1 \leq i < n, i+1 < j \leq n\}$. We say that a tournament of order n is a U-tournament if it is isomorphic to U_n.*

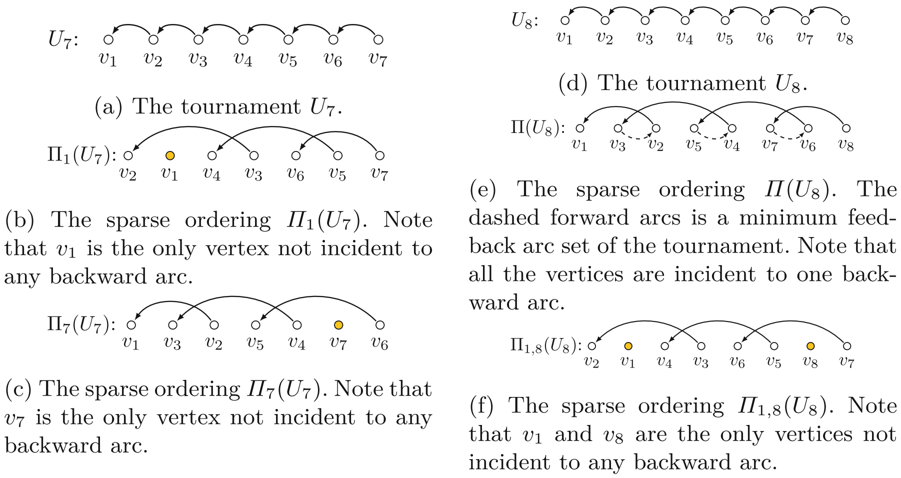
Figures 4(a) and 4(d) depict respectively the tournaments U_7 and U_8. This family of tournaments seems somehow strongly related to sparse tournaments and the following results will be useful later for both the polynomial algorithm to decide if a tournament is sparse and the polynomial algorithm for minimum feedback arc set in sparse tournaments. To do so, we prove that each U-tournament of order $n > 4$ has exactly two sparse orderings of its vertices that we formally define.

Definition 4. *Let $P(k) = \langle v_{k+1}, v_k \rangle$ be a pattern of two vertices of U_n for some integer $k \in [n-1]$. For any integer $n \geq 2$, we define the following special orderings of U_n:*

- *if n is even:*
  - *$\Pi(U_n)$ is the ordering given by $\langle v_1, P(2), P(4), \ldots, P(n-2), v_n \rangle$.*
  - *$\Pi_{1,n}(U_n)$ is the ordering given by $\langle P(1), P(3), \ldots, P(n-2), P(n) \rangle$.*
- *if n is odd:*
  - *$\Pi_1(U_n)$ is the ordering given by $\langle P(1), P(3), \ldots, P(n-2), v_n \rangle$.*
  - *$\Pi_n(U_n)$ is the ordering given by $\langle v_1, P(2), P(4), \ldots, P(n-3), P(n-1) \rangle$.*

Figures 4(b) and 4(c) (and Figs. 4(e) and 4(f)) depict the orderings $\Pi_1(U_7)$ and $\Pi_7(U_7)$ (resp. $\Pi(U_8)$ and $\Pi_{1,8}(U_8)$) of the tournament U_7 (resp. U_8). One can notice that these orderings are sparse and the subscript of Π indicates the vertex (or vertices) without a backward arc incident to it in this ordering. In the following, we prove that when $n > 4$ there are no other sparse orderings of U_n. However, note that there are three possible sparse orderings of U_3 (namely, $\Pi_1(U_3)$ and $\Pi_3(U_3)$ defined previously, as well as $\Pi_2(U_3) := \langle v_3, v_2, v_1 \rangle$) and three sparse orderings of U_4 (namely, $\Pi(U_4)$, $\Pi_{1,4}(U_4)$ as defined before, and $\Pi'(U_4) := \langle v_2, v_4, v_1, v_3 \rangle$).

Theorem 4. *For each integer $n > 4$ there are exactly two sparse orderings of U_n. Specifically, if n is even, these two sparse orderings are $\Pi(U_n)$ and $\Pi_{1,n}(U_n)$; otherwise, the two sparse orderings are $\Pi_1(U_n)$ and $\Pi_n(U_n)$.*


U_7:
v_1 v_2 v_3 v_4 v_5 v_6 v_7
$\Pi_1(U_7)$:
v_2 v_1 v_4 v_3 v_6 v_5 v_7
$\Pi_7(U_7)$:
v_1 v_3 v_2 v_5 v_4 v_7 v_6
U_8:
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8
$\Pi(U_8)$:
v_1 v_3 v_2 v_5 v_4 v_7 v_6 v_8
$\Pi_{1,8}(U_8)$:
v_2 v_1 v_4 v_3 v_6 v_5 v_8 v_7

(a) The tournament U_7.

(b) The sparse ordering $\Pi_1(U_7)$. Note that v_1 is the only vertex not incident to any backward arc.

(c) The sparse ordering $\Pi_7(U_7)$. Note that v_7 is the only vertex not incident to any backward arc.

(d) The tournament U_8.

(e) The sparse ordering $\Pi(U_8)$. The dashed forward arcs is a minimum feedback arc set of the tournament. Note that all the vertices are incident to one backward arc.

(f) The sparse ordering $\Pi_{1,8}(U_8)$. Note that v_1 and v_8 are the only vertices not incident to any backward arc.

Fig. 4. The tournaments U_7 and U_8 and their sparse orderings. The non-depicted arcs are forward arcs.

4.2 A Polynomial Time Algorithm for Sparse Tournaments

We give here a polynomial algorithm to compute a sparse ordering of a tournament (if any). First of all, let us recall a classical algorithm to compute a topological ordering of a tournament (if any): we look for the vertex v with the smallest in-degree; if v has in-degree one or more, we have a certificate that the tournament is not acyclic. Otherwise, we add v at the beginning of the ordering, and we repeat the reasoning on $T - v$, until $V(T)$ is empty.

The idea of the original "proof" in [24, Lemma 35.1, p.97] was similar: considering the set of vertices X of smallest in-degrees, put X at the beginning of the ordering, and remove X from the tournament. However, potential backward arcs from the remaining vertices of $V \setminus X$ to X may have been forgotten. For example, consider a tournament over 9 vertices consisting of a U_5 (with vertex set $\{v_1, \ldots, v_5\}$) that dominates a U_4 (with vertex set $\{u_1, ..., u_4\}$) except for the backward arc (u_4, v_5). It is sparse ($\langle \Pi_5(U_5), \Pi_{1,4}(U_4)\rangle$) but the algorithm returns the (non-sparse) ordering $\langle \Pi_1(U_5), \Pi_{1,4}(U_4)\rangle$ (v_5 is incident to two backward arcs). The problem is that this algorithm is too "local"; it will always prefer the sparse ordering $\Pi_1(U_{2k+1})$ over $\Pi_{2k+1}(U_{2k+1})$, but it may be necessary to take the latter. Therefore, to correct this, we needed a much more involved algorithm, requiring the study of the U-tournaments and the notion of quasi-domination (see Definition 6). Indeed, unlike the algorithm for the topological ordering, we may have to look more carefully how the vertices with low in-degrees are connected to the rest of the digraph. These correspond to the case where there exists a U-sub-tournament of T which either dominates or "quasi-dominates" (see Definition 6) the tournament T. Because of the latter possibility (where a backward arc (a, b) is forced to appear), we need to look for specific sparse orderings, called M-sparse orderings (where a or b should not be end-

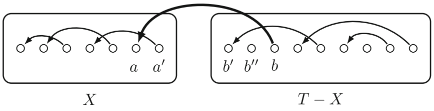


Fig. 5. An example where X (b, a)-quasi-dominates T. Non-depicted arcs are forward. The vertex a' is an out-neighbour of a in X, and b', b'' are in-neighbours of b in $T - X$.

vertices of other backward arcs). As all the sparse orderings for U-tournaments have been described, we can derive a recursive algorithm.

Definition 5. *Let T be a tournament, X be a subset of vertices of T, and M be a subset of X. We say $T[X]$ is M-sparse if there exists an ordering σ of X such that $\Delta_{\sigma(T[X])}(X) \leq 1$ and $d_\sigma(v) = 0$ for all $v \in M$. In that case, σ is said to be an M-sparse ordering of $T[X]$.*

For example, $U_4[\{v_1, v_2, v_3\}]$ is $\{v_2\}$-sparse, because there exists a sparse ordering $\sigma := \langle v_3, v_2, v_1 \rangle$ of $U_4[\{v_1, v_2, v_3\}]$ such that $d_\sigma(v_2) = 0$. We remark that T is sparse if and only if T is $\emptyset$-sparse. In fact, the algorithm described in this section computes a $\emptyset$-sparse ordering of the given tournament (if any).

Definition 6 (see Fig. 5). *Given a tournament T and two of its vertices a and b, we say that a subset of vertices X quasi-dominates T if:*

- *there exists an arc $(b, a) \in A(T)$ such that $a \in X$ and $b \notin X$,*
- *$(u, v) \in A(T)$ for every $(u, v) \in (X \times (V(T) \setminus X)) \setminus \{(a, b)\}$,*
- *$d^-(b) \geq |X| + 1$, and*
- *the vertex a has an out-neighbour in X.*

In this case, we also say X (b, a)-quasi-dominates T.

We can create the algorithm `isUkMsparse` which given $(v_1, \ldots, v_k)$ a U-tournament and M a subset of these vertices, returns a boolean which is True if and only if this tournament is M-sparse. We can also create the algorithm `getUsubtournament` which given T a tournament, and $X = (u_1, \ldots, u_k)$ a list of vertices such that $d^-(u_1) = 1$ and $d^-(u_i) = i - 1$ and $(u_i, u_{i-1}) \in A(T)$ for all $i \in \{2, \ldots, k\}$, returns a U-subtournament dominating or quasi-dominating T. With these two previous algorithms, we can derive Algorithm 3 `isMsparse`.

Algorithm 1: `getUsubtournament`

```
Data: $T$ a tournament, and $X = (u_1, \ldots, u_k)$ a list of vertices such that $d^-(u_1) = 1$ and
      $d^-(u_i) = i - 1$ and $(u_i, u_{i-1}) \in A(T)$ for all $i \in \{2, \ldots, k\}$.
Result: A $U$-subtournament dominating or quasi-dominating $T$.
$w \longleftarrow$ a vertex of $N^-(u_k) \setminus X$;
if $d^-(w) = d^-(u_k)$ then return $X \cup \{w\}$                /* this set dominates $T$ */ ;
else if $d^-(w) = d^-(u_k) + 1$ then return getUsubtournament($T, X \cup \{w\}$) ;
else return $X$                                 /* this set $(w, u_k)$-quasi-dominates $T$ */ ;
```

Algorithm 2: `isUkMsparse`

```
Data: $(v_1, \ldots, v_k)$ a $U_k$ tournament, $M$ a subset of the vertices of $U_k$
Result: True if $U_k$ is $M$-sparse and False otherwise
if $k \leq 2$ then return True ;
else if $k = 3$ then return $|M| \leq 1$ ;
else if $k$ is even then return $|M \setminus \{v_1, v_k\}| = 0$ ;
else if $k$ is odd then return ($v_1 \notin M$ or $v_k \notin M$) and $|M \setminus \{v_1, v_k\}| = 0$ ;
```

Algorithm 3: `isMsparse`

```
Data: $T$ a tournament, $M$ a subset of the vertices of $T$
Result: True if $T$ is $M$-sparse and False otherwise
if $|V(T)| \leq 1$ then return True ;
else if $\min_{v \in V(T)} d^-(v) \geq 2$ then return False ;
else if $\min_{v \in V(T)} d^-(v) = 0$ then
    $v \longleftarrow$ the vertex of in-degree 0;
    return isMsparse($T - v, M \setminus \{v\}$);
else if $|\{v \in V(T) : d^-(v) = 1\}| = 1$ then
    $v, w \longleftarrow$ two vertices such that $d^-(v) = 1$ and $(w, v) \in A(T)$;
    return $v \notin M$ and isMsparse($T - v$, $(M \cup \{w\}) \setminus \{v\}$);
else
    $v, w \longleftarrow$ two vertices of in-degree 1 such that $(w, v) \in A(T)$;
    $X \longleftarrow$ getUsubtournament($T$,$(v, w)$);
    if $X$ dominates $T$ then
        return (isUkMsparse($X$,$M \cap X$) and isMsparse($T - X$, $M \setminus X$));
    else
        $a, b \longleftarrow$ the vertices such that $X$ $(b, a)$-quasi-dominates $T$;
        return (isUkMsparse($X$,$(M \cup \{a\}) \cap X$) and isMsparse($T - X$, $(M \cup \{b\}) \setminus X$));
```

Theorem 5. *Algorithm 3 is correct. Hence, it is possible to decide if a tournament T with n vertices is sparse in $\mathcal{O}(n^3)$ by calling* `isMsparse(`T`,`$\emptyset$`)`.

Observe that we can easily modify Algorithm 3 to obtain a sparse ordering (if exists). Next corollary follows from the above algorithm.

Corollary 1. *The vertex set of a sparse tournament on n vertices can be decomposed into a sequence $U_{n_1}, U_{n_2}, \ldots, U_{n_\ell}$ for some $\ell \leq n$ such that each $T[U_{n_i}]$ dominates or quasi-dominates $T[\bigcup_{i<j\leq\ell} U_{n_j}]$ and $\sum_{i\in[\ell]} n_i = n$.*

5 Degreewidth as a Parameter

5.1 Dominating Set Parameterized by Degreewidth

A set of vertices X of a directed graph G is a *dominating set (DS)* if for each vertex $v \in V(G) \setminus X$, we have $N^+(v) \cap X \neq \emptyset$. Observe that in graphs where degreewidth is zero, DS is of size one. Similarly, for tournaments with degreewidth equals to one, the DS is of size at most two. That is, we have trivial solutions for DS for acyclic and sparse tournaments. This motivates us to look for FPT algorithm parameterized by degreewidth. In the following, we develop an FPT algorithm for Dominating Set using universal families. Before that we observe that size of a dominating is always bounded by the size of degreewidth.

Observation 3. *The size of a minimum dominating set of a tournament T is at most $\Delta(T) + 1$.*

Theorem 6. Dominating Set *is FPT in tournaments with respect to degreewidth.*

5.2 FAST and FVST in Sparse Tournaments

A *forbidden pattern* corresponds to the patterns $\Pi(U_{2k})$ for any $k \geq 1$ as well as $\Pi'(U_4) := \langle v_2, v_4, v_1, v_3 \rangle$. An example of the forbidden pattern $\Pi(U_8)$ is depicted in Fig. 4(e). We say a sparse ordering has *forbidden pattern* if a contiguous subsequence of the ordering is a forbidden pattern. Intuitively, the problem of such patterns is that the set of their backward arcs is not a minimum fas. Hopefully, we can use Theorem 4 in such a way that if the pattern $\Pi(U_{2k})$ appears, we can restructure it into $\Pi_{1,2k}(U_{2k})$.

If a sparse ordering does not contain a forbidden pattern then its set of backward arcs is a fas. Hence, we obtain the following result.

Theorem 7. *FAST is solvable in time $O(n^3)$ in sparse tournaments on n vertices.*

For FVST, we show that the problem is difficult to solve on sparse tournaments.

Theorem 8. *FVST is NP-complete on sparse tournaments.*

6 Conclusion

In this paper, we studied a new parameter for tournaments, called degreewidth. We showed that it is NP-hard to decide if degreewidth is at most k, for some natural number k and we proceeded to design a 3-approximation for the degreewidth. One may ask if there is a PTAS for this problem. Then, we investigated sparse tournaments, *i.e.*, tournaments with degreewidth one and developed a polynomial time algorithm to compute a sparse ordering. Is it possible to generalise this result by providing an FPT algorithm to compute the degreewidth? We also showed that FAST can be solved in polynomial time in sparse tournaments, matching with the known result that Arc-Disjoint Triangles Packing and Arc-Disjoint Cycle Packing are both polynomial in sparse tournaments [7]. Therefore, the question arise: can this parameter be used to provide an FPT algorithm for FAST in the general case?

Furthermore, we showed an FPT algorithm for DS w.r.t degreewidth. Are there other domination problems e.g., perfect code, partial dominating set, or connected dominating set that is FPT w.r.t degreewidth? Lastly, we also can wonder if this parameter is useful for general digraphs.

Acknowledgements. We would like to thank Frédéric Havet for pointing us a counter-example to the polynomial running-time algorithm in [24, Lemma 35.1, p. 97].

References

1. Alber, J., Bodlaender, H.L., Fernau, H., Kloks, T., Niedermeier, R.: Fixed parameter algorithms for dominating set and related problems on planar graphs. Algorithmica **33**(4), 461–493 (2002)

2. Allesina, S., Levine, J.M.: A competitive network theory of species diversity. Proc. Natl. Acad. Sci. **108**(14), 5638–5642 (2011)
3. Alon, N.: Ranking tournaments. SIAM J. Discret. Math. **20**(1), 137–142 (2006). https://doi.org/10.1137/050623905
4. Bang-Jensen, J., Gutin, G.Z.: Digraphs - Theory, Algorithms and Applications. Springer Monographs in Mathematics, 2nd edn. Springer, Heidelberg (2009). https://doi.org/10.1007/978-1-84800-998-1
5. Bar-Yehuda, R., Geiger, D., Naor, J., Roth, R.M.: Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference. SIAM J. Comput. **27**(4), 942–959 (1998). https://doi.org/10.1137/S0097539796305109
6. Berman, P., Karpinski, M., Scott, A.D.: Approximation hardness of short symmetric instances of MAX-3SAT. Electron. Colloquium Comput. Complex. (049) (2003). http://eccc.hpi-web.de/eccc-reports/2003/TR03-049/index.html
7. Bessy, S., et al.: Packing arc-disjoint cycles in tournaments. In: Rossmanith, P., Heggernes, P., Katoen, J. (eds.) 44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, 26–30 August 2019, Aachen, Germany. LIPIcs, vol. 138, pp. 27:1–27:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019). https://doi.org/10.4230/LIPIcs.MFCS.2019.27
8. Bessy, S., Bougeret, M., Thiebaut, J.: Triangle packing in (sparse) tournaments: approximation and kernelization. In: Pruhs, K., Sohler, C. (eds.) 25th Annual European Symposium on Algorithms, ESA 2017, 4–6 September 2017, Vienna, Austria. LIPIcs, vol. 87, pp. 14:1–14:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017). https://doi.org/10.4230/LIPIcs.ESA.2017.14
9. Brandt, F., Fischer, F.: PageRank as a weak tournament solution. In: Deng, X., Graham, F.C. (eds.) WINE 2007. LNCS, vol. 4858, pp. 300–305. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77105-0_30
10. Charbit, P., Thomassé, S., Yeo, A.: The minimum feedback arc set problem is NP-hard for tournaments. Comb. Probab. Comput. **16**(1), 1–4 (2007). https://doi.org/10.1017/S0963548306007887
11. Chen, J., Liu, Y., Lu, S., O'sullivan, B., Razgon, I.: A fixed-parameter algorithm for the directed feedback vertex set problem. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, pp. 177–186 (2008)
12. Davot, T., Isenmann, L., Roy, S., Thiebaut, J.: DegreeWidth: a new parameter for solving problems on tournaments. CoRR abs/2212.06007 (2022). https://doi.org/10.48550/arXiv.2212.06007
13. Dechter, R.: Enhancement schemes for constraint processing: backjumping, learning, and cutset decomposition. Artif. Intell. **41**(3), 273–312 (1990). https://doi.org/10.1016/0004-3702(90)90046-3
14. Downey, R.G., Fellows, M.R.: Parameterized computational feasibility. In: Clote, P., Remmel, J.B. (eds.) Feasible Mathematics II. Progress in Computer Science and Applied Logic, vol. 13, pp. 219–244. Springer, Boston (1995). https://doi.org/10.1007/978-1-4612-2566-9_7
15. Feige, U.: Faster fast (feedback arc set in tournaments). CoRR abs/0911.5094 (2009). http://arxiv.org/abs/0911.5094
16. Fradkin, A.O.: Forbidden structures and algorithms in graphs and digraphs. Ph.D. thesis, USA (2011). aAI3463323
17. Gavril, F.: Some NP-complete problems on graphs. In: Proceedings of the 11th Conference on Information Sciences and Systems. Johns Hopkins University, Baltimore (1977)

18. Gurski, F., Rehs, C.: Comparing linear width parameters for directed graphs. Theory Comput. Syst. **63**(6), 1358–1387 (2019). https://doi.org/10.1007/s00224-019-09919-x
19. Johnson, D.B.: Finding all the elementary circuits of a directed graph. SIAM J. Comput. **4**(1), 77–84 (1975). https://doi.org/10.1137/0204007
20. Karpinski, M., Schudy, W.: Faster algorithms for feedback arc set tournament, Kemeny rank aggregation and betweenness tournament. In: Cheong, O., Chwa, K.-Y., Park, K. (eds.) ISAAC 2010. LNCS, vol. 6506, pp. 3–14. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17517-6_3
21. Kenyon-Mathieu, C., Schudy, W.: How to rank with few errors. In: Johnson, D.S., Feige, U. (eds.) Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, 11–13 June 2007, pp. 95–103. ACM (2007). https://doi.org/10.1145/1250790.1250806
22. Laslier, J.F.: Tournament Solutions and Majority Voting, vol. 7. Springer, Heidelberg (1997)
23. Leighton, T., Rao, S.: Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. J. ACM (JACM) **46**(6), 787–832 (1999)
24. Thiebaut, J.: Algorithmic and structural results on directed cycles in dense digraphs. (Résultats algorithmiques et structurels sur les cycles orientés dans les digraphes denses). Ph.D. thesis, University of Montpellier, France (2019). https://tel.archives-ouvertes.fr/tel-02491420
25. van Zuylen, A., Williamson, D.P.: Deterministic pivoting algorithms for constrained ranking and clustering problems. Math. Oper. Res. **34**(3), 594–620 (2009). https://doi.org/10.1287/moor.1090.0385

Approximating Bin Packing with Conflict Graphs via Maximization Techniques

Ilan Doron-Arad and Hadas Shachnai(✉)

Computer Science Department, Technion, Haifa 3200003, Israel
{idoron-arad,hadas}@cs.technion.ac.il

Abstract. We give a comprehensive study of *bin packing with conflicts* (BPC). The input is a set I of items, sizes $s : I \rightarrow [0, 1]$, and a conflict graph $G = (I, E)$. The goal is to find a partition of I into a minimum number of independent sets, each of total size at most 1. Being a generalization of the notoriously hard graph coloring problem, BPC has been studied mostly on polynomially colorable conflict graphs. An intriguing open question is whether BPC on such graphs admits the same best known approximation guarantees as classic bin packing.

We answer this question negatively, by showing that (in contrast to bin packing) there is no asymptotic polynomial-time approximation scheme (APTAS) for BPC already on seemingly easy graph classes, such as *bipartite* and *split* graphs. We complement this result with improved approximation guarantees for BPC on several prominent graph classes. Most notably, we derive an asymptotic 1.391-approximation for bipartite graphs, a 2.445-approximation for perfect graphs, and a $\left(1 + \frac{2}{e}\right)$-approximation for split graphs. To this end, we introduce a generic framework relying on a novel interpretation of BPC allowing us to solve the problem via *maximization* techniques. Our framework may find use in tackling BPC on other graph classes arising in applications.

1 Introduction

We study the *bin packing with conflicts (BPC)* problem. We are given a set I of n items, sizes $s : I \rightarrow [0, 1]$, and a conflict graph $G = (I, E)$ on the items. A *packing* is a partition $(A_1, \ldots, A_t)$ of I into independent sets called *bins*, such that for all $b \in \{1, \ldots, t\}$ it holds that $s(A_b) = \sum_{\ell \in A_b} s(\ell) \leq 1$. The goal is to find a packing in a minimum number of bins. Let $\mathcal{I} = (I, s, E)$ denote a BPC instance. We note that BPC is a generalization of *bin packing (BP)* (where $E = \emptyset$) as well as the graph coloring problem (where $s(\ell) = 0 \ \forall \ell \in I$).[1] BPC captures many real-world scenarios such as resource clustering in parallel computing [2], examination scheduling [21], database storage [16], and product delivery [3]. As the special case of graph coloring cannot be approximated within a ratio better than $n^{1-\varepsilon}$ [28], most of the research work on BPC has focused on families of

[1] See the formal definitions of *graph coloring* and *independent sets* in Sect. 2.

A full version of the paper is available in [6].

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 261–275, 2023.
https://doi.org/10.1007/978-3-031-43380-1_19

conflict graphs which can be optimally colored in polynomial time [4,5,8,15–17,22,23].

Let OPT $=$ OPT$(\mathcal{I})$ be the value of an optimal solution for an instance $\mathcal{I}$ of a minimization problem $\mathcal{P}$. As in the bin packing problem, we distinguish between *absolute* and *asymptotic* approximation. For $\alpha \geq 1$, we say that $\mathcal{A}$ is an absolute α-approximation algorithm for $\mathcal{P}$ if for any instance $\mathcal{I}$ of $\mathcal{P}$ we have $\mathcal{A}(\mathcal{I})/\text{OPT}(\mathcal{I}) \leq \alpha$, where $\mathcal{A}(\mathcal{I})$ is the value of the solution returned by $\mathcal{A}$. Algorithm $\mathcal{A}$ is an *asymptotic* α-approximation algorithm for $\mathcal{P}$ if for any instance $\mathcal{I}$ it holds that $\mathcal{A}(\mathcal{I}) \leq \alpha\text{OPT}(\mathcal{I}) + o(\text{OPT}(\mathcal{I}))$. An APTAS is a family of algorithms $\{\mathcal{A}_\varepsilon\}$ such that, for every $\varepsilon > 0$, $\mathcal{A}_\varepsilon$ is a polynomial time asymptotic $(1+\varepsilon)$-approximation algorithm for $\mathcal{P}$. An *asymptotic fully polynomial-time approximation scheme (AFPTAS)* is an APTAS $\{\mathcal{A}_\varepsilon\}$ such that $\mathcal{A}_\varepsilon(\mathcal{I})$ runs in time poly$(|\mathcal{I}|, \frac{1}{\varepsilon})$, where $|\mathcal{I}|$ is the encoding length of the instance $\mathcal{I}$.

It is well known that, unless P=NP, BP cannot be approximated within ratio better than $\frac{3}{2}$ [10]. This ratio is achieved by First-Fit Decreasing (FFD) [26].[2] Also, BP admits an AFPTAS [19], and an additive approximation algorithm which packs any instance $\mathcal{I}$ in at most $\text{OPT}(\mathcal{I}) + O(\log(\text{OPT}(\mathcal{I})))$ bins [14]. Despite the wide interest in BPC on polynomially colorable graphs, the intriguing question whether BPC on such graphs admits the same best known approximation guarantees as classic bin packing remained open.

Table 1. Known results for Bin Packing with Conflict Graphs

	Absolute		**Asymptotic**	
	Lower Bound	Upper Bound	Lower Bound	Upper Bound
General graphs	$n^{1-\varepsilon}$ [28]	$O\left(\frac{n(\log\log n)^2}{(\log n)^3}\right)$ [13]	$n^{1-\varepsilon}$ [28]	$O\left(\frac{n(\log\log n)^2}{(\log n)^3}\right)$ [13]
Perfect graphs	·	**2.445** (2.5 [8])	$\mathbf{c > 1}$	**2.445** (2.5 [8])
Chordal graphs	·	$\frac{7}{3}$ [8]	$\mathbf{c > 1}$	$\frac{7}{3}$ [8]
Cluster graphs	·	2 [1]		1 [5]
Cluster complement	·	**3/2**	**3/2**	**3/2**
Split graphs	·	$\mathbf{1 + 2/e}$ (2 [15])	$\mathbf{c > 1}$	$\mathbf{1 + 2/e}$ (2 [15])
Bipartite graphs	·	$\frac{5}{3}$ [15]	$\mathbf{c > 1}$	**1.391** ($\frac{5}{3}$ [15])
Partial k-trees	·	$2 + \varepsilon$ [17]		1 [16]
Trees	·	$\frac{5}{3}$ [15]		·
No conflicts	$\frac{3}{2}$ [10]	$\frac{3}{2}$ [26]		1 [25]

We answer this question negatively, by showing that (in contrast to bin packing) there is no APTAS for BPC even on seemingly easy graph classes, such as *bipartite* and *split* graphs. We complement this result with improved approximation guarantees for BPC on several prominent graph classes. For BPC on bipartite graphs, we obtain an asymptotic 1.391-approximation. We further derive improved bounds of 2.445 for perfect graphs, $\left(1 + \frac{2}{e}\right)$ for split graphs, and $\frac{5}{3}$ for

[2] We give a detailed description of Algorithm FFD in [6].

bipartite graphs.[3] Finally, we obtain a tight $\frac{3}{2}$-asymptotic lower bound and an absolute $\frac{3}{2}$-upper bound for graphs that are the complements of cluster graphs (we call these graphs below *complete multi-partite*).

Table 1 summarizes the known results for BPC on various classes of graphs. New bounds given in this paper are shown in boldface. Entries that are marked with $\cdot$ follow by inference, either by using containment of graph classes (trees are partial k-trees), or since the hardness of BPC on all considered graph classes follows from the hardness of classic BP. Empty entries for lower bounds follow from tight upper bounds.

1.1 Related Work

The BPC problem was introduced by Jansen and Öhring [17]. They presented a general algorithm that initially finds a coloring of the conflict graph, and then packs each color class separately using the First-Fit Decreasing algorithm. This approach yields a 2.7-approximation for BPC on perfect graph. The paper [17] includes also a 2.5-approximation for subclasses of perfect graphs on which the corresponding *precoloring extension problem* can be solved in polynomial time (e.g., interval and chordal graphs). The authors present also a $(2 + \varepsilon)$-approximation algorithm for BPC on cographs and partial k-trees.

Epstein and Levin [8] present better algorithms for BPC on perfect graphs (2.5-approximation), graphs on which the precoloring extension problem can be solved in polynomial time ($\frac{7}{3}$-approximation), and bipartite graphs ($\frac{7}{4}$-approximation). Their techniques include matching between *large* items and a sophisticated use of new item *weights*. Recently, Huang et al. [15] provided fresh insights to previous algorithms, leading to $\frac{5}{3}$-approximation for BPC on bipartite graphs and a 2-approximation on split graphs.

Jansen [16] presented an AFPTAS for BPC on *d-inductive* conflict graphs, where $d \geq 1$ is some constant. This graph family includes trees, grid graphs, planar graphs, and graphs with constant treewidth. For a survey of *exact* algorithms for BPC see, e.g., [15].

1.2 Techniques

There are several known approaches for tackling BPC instances. One celebrated technique introduced by Jansen and Öhring [17] relies on finding initially a minimum coloring of the given conflict graph, and then packing each color class using a bin packing heuristic, such as First-Fit Decreasing. A notable generalization of this approach is the sophisticated integration of *precoloring extension* [8,17], which completes an initial partial coloring of the conflict graph, with no increase to the number of color classes. Another elegant technique is a matching-based algorithm, applied by Epstein and Levin [8] and by Huang et al. [15].

[3] Recently, Huang et al. [15] obtained a $\frac{5}{3}$-approximation for bipartite graphs, simultaneously and independently of our work. We note that the techniques of [15] are different than ours, and their algorithm is more efficient in terms of running time.

The best known algorithms (prior to this work), e.g., for perfect graphs [8] and split graphs [15] are based on the above techniques. While the analyses of these algorithms are tight, the approximation guarantees do not match the existing lower bounds for BPC on these graph classes; thus, obtaining improved approximations requires new techniques.

In this paper we present a novel point of view of BPC involving the solution of a maximization problem as a subroutine. We first find an *initial packing* of a subset $S \subseteq I$ of items, which serves as a baseline packing with *high potential* for adding items (from $I \setminus S$) without increasing the number of bins used. The remaining items are then assigned to extra bins using a simple heuristic. Thus, given a BPC instance, our framework consists of the following main steps.

1. Find an initial packing $\mathcal{A} = (A_1, \ldots, A_m)$ of high potential for $S \subseteq I$.
2. Maximize the total size of items in $\mathcal{A}$ by adding items in $I \setminus S$.
3. Assign the remaining (unpacked) items to extra bins using a greedy approach respecting the conflict graph constraints.

The above generic framework reduces BPC to cleverly finding an initial packing of high potential, and then efficiently approximating the corresponding maximization problem, while exploiting structural properties of the given conflict graph. One may view classic approaches for solving BP (e.g., [20]), as an application of this technique: find an initial packing of high potential containing the *large* items; then add the *small* items using First-Fit. In this setting, the tricky part is to find an initial high potential packing, while adding the small items is trivial. However, in the presence of a conflict graph, solving the induced maximization problem is much more challenging.

Interestingly, we are able to obtain initial packings of high potential for BPC on several conflict graph classes. To solve the maximization problem, we first derive efficient approximation for maximizing the total size of items within a *single* bin. Our algorithm is based on finding a maximum weight independent set of *bounded* total size in the graph, combined with enumeration over items of large sizes. Using the single bin algorithm, the maximization problem is solved via application of the *separable assignment problem (SAP)* [9] framework, adapted to our setting. Combined with a hybrid of several techniques (to efficiently handle different types of instances) this leads to improved bounds for BPC on perfect, split, and bipartite graphs (see Sects. 3, 4, and the full version of the paper [6]). Our framework may find use in tackling BPC on other graph classes arising in applications.

1.3 Organization

In Sect. 2 we give some definitions and preliminary results. Section 3 presents an approximation algorithm for BPC on perfect graphs and an asymptotic approximation on bipartite graphs. In Sect. 4 we give an algorithm for split graphs. We present our hardness results in Sect. 5 and conclude in Sect. 6. Due to space constraints, some of our results and proofs are given in the full version of the paper [6].

2 Preliminaries

For any $k \in \mathbb{R}$, let $[k] = \{1, 2, \ldots, \lfloor k \rfloor\}$. Also, for a function $f : A \rightarrow \mathbb{R}_{\geq 0}$ and a subset of elements $C \subseteq A$, we define $f(C) = \sum_{e \in C} f(e)$.

2.1 Coloring and Independent Sets

Given a graph $G = (V, E)$, an *independent set* in G is a subset of vertices $S \subseteq V$ such that for all $u, v \in S$ it holds that $(u, v) \notin E$. Let $\mathsf{IS}(G)$ be the collection of all independent sets in G. Given weight function $w : V \rightarrow \mathbb{R}_{\geq 0}$, a *maximum independent set w.r.t.* w is an independent set $S \in \mathsf{IS}(G)$ such that $w(S)$ is maximized. A *coloring* of G is a partition $(V_1, \ldots, V_t)$ of V such that $\forall i \in [t] : V_i \in \mathsf{IS}(G)$; we call each subset of vertices V_i *color class* i. Let $\chi(G)$ be the minimum number of colors required for a coloring of G. A graph G is *perfect* if for every induced subgraph G' of G the cardinality of the maximum clique of G' is equal to $\chi(G')$; note that G' is also a perfect graph. The following well known result is due to [12].

Lemma 2.1. *Given a perfect graph $G = (V, E)$, a minimum coloring of G and a maximum weight independent set of G can be computed in polynomial time.*

2.2 Bin Packing with Conflicts

Given a BPC instance $\mathcal{I}$, let $G_{\mathcal{I}} = (I, E)$ denote the conflict graph of $\mathcal{I}$. A *packing* of a subset of items $S \subseteq I$ is a partition $\mathcal{B} = (B_1, \ldots, B_t)$ of S such that, for all $i \in [t]$, B_i is an independent set in $G_{\mathcal{I}}$, and $s(B_i) \leq 1$. Let $\#\mathcal{B}$ be the number of bins (i.e., entries) in $\mathcal{B}$.

In this paper we consider BPC on several well studied classes of perfect graphs and the acronym BPC refers from now on to perfect conflict graphs. For *bin packing with bipartite conflicts (BPB)*, where the conflict graph is bipartite, we assume a bipartition of V is known and given by X_V and Y_V. Recall that $G = (V, E)$ is a split graph if there is a partition K, S of V into a clique and an independent set, respectively. We call this variant of BPC *bin packing with split graph conflicts (BPS).*

The following notation will be useful while enhancing a partial packing by new items. For two packings $\mathcal{B} = (B_1, \ldots, B_t)$ and $\mathcal{C} = (C_1, \ldots, C_r)$, let $\mathcal{B} \oplus \mathcal{C} = (B_1, \ldots, B_t, C_1, \ldots, C_r)$ be the *concatenation* of $\mathcal{B}$ and $\mathcal{C}$; also, for $t = r$ let $\mathcal{B} + \mathcal{C} = (B_1 \cup C_1, \ldots, B_t \cup C_t)$ be the *union* of the two packings; note that the latter is not necessarily a packing. We denote by $\mathsf{items}(\mathcal{B}) = \bigcup_{i \in [t]} B_i$ the set of items in the packing $\mathcal{B}$. Finally, let $\mathcal{I} = (I, s, E)$ be a BPC instance and $T \subseteq I$ a subset of items. Define the BPC instances $\mathcal{I} \cap T = (T, s, E_T)$ and $\mathcal{I} \backslash T = (I \backslash T, s, E_{I \backslash T})$ where for all $X \in \{T, I \backslash T\}$ $E_X = \{(u, v) \in E \mid u, v \in X\}$.

2.3 Bin Packing Algorithms

We use $\mathcal{I} = (I, s)$ to denote a BP instance, where I is a set of n items for some $n \geq 1$, and $s : I \rightarrow [0, 1]$ is the size function. Let $L_{\mathcal{I}} = \{\ell \in I \mid s(\ell) > \frac{1}{2}\}$ be the set of *large* items, $M_{\mathcal{I}} = \{\ell \in I \mid \frac{1}{3} < s(\ell) \leq \frac{1}{2}\}$ the set of *medium* items, and $S_{\mathcal{I}} = \{\ell \in I \mid s(\ell) \leq \frac{1}{3}\}$ the set of *small* items. Our algorithms use as building blocks also algorithms for BP. The results in the next two lemmas are tailored for our purposes. We give the detailed proofs in [6].[4]

Lemma 2.2. *Given a* BP *instance* $\mathcal{I} = (I, s)$*, there is a polynomial-time algorithm* First-Fit Decreasing (FFD) *which returns a packing* $\mathcal{B} = (B_1, \ldots, B_t)$ *of* $\mathcal{I}$ *where* $\#\mathcal{B} \leq (1 + 2 \cdot \max_{\ell \in I} s(\ell)) \cdot s(I) + 1$*. Moreover, it also holds that* $\#\mathcal{B} \leq |L_{\mathcal{I}}| + \frac{3}{2} \cdot s(M_{\mathcal{I}}) + \frac{4}{3} \cdot s(S_{\mathcal{I}}) + 1$*.*

Lemma 2.3. *Given a* BP *instance* $\mathcal{I} = (I, s)$*, there is a polynomial-time algorithm* AsymptoticBP *which returns a packing* $\mathcal{B} = (B_1, \ldots, B_t)$ *of* $\mathcal{I}$ *such that* $t = \mathrm{OPT}(\mathcal{I}) + o(\mathrm{OPT}(\mathcal{I}))$*. Moreover, if* $\mathrm{OPT}(\mathcal{I}) \geq 100$ *then* $t \leq 1.02 \cdot \mathrm{OPT}(\mathcal{I})$*.*

3 Approximations for Perfect and Bipartite Graphs

In this section we consider the bin packing problem with a perfect or bipartite conflict graph. Previous works (e.g., [8,17]) showed the usefulness of the approach based on finding first a minimum coloring of the given conflict graph, and then packing each color class as a separate bin packing instance (using, e.g., algorithm FFD). Indeed, this approach yields efficient approximations for BPC; however, it does reach a certain limit. To enhance the performance of this *coloring based* approach, we design several subroutines. Combined, they cover the problematic cases and lead to improved approximation guarantees (see Table 1).

Our first subroutine is the coloring based approach, with a simple modification to improve the asymptotic performance. For each color class $C_i, i = 1, \ldots, k$ in a minimum coloring of the given conflict graph, we find a packing of C_i using FFD, and another packing using AsymptoticBP (see Lemma 2.3). We choose the packing which has smaller number of bins. Finally, the returned packing is the concatenation of the packings of all color classes. The pseudocode of Algorithm Color_Sets is given in Algorithm 1.

Algorithm 1. Color_Sets($\mathcal{I} = (I, s, E)$)

```
Compute a minimum coloring $\mathcal{C} = (C_1, \ldots, C_k)$ of $G_{\mathcal{I}}$.
Initialize an empty packing $\mathcal{B} \leftarrow ()$.
for $i \in [k]$ do
    Compute $\mathcal{A}_1 \leftarrow$ FFD($(C_i, s)$) and $\mathcal{A}_2 \leftarrow$ AsymptoticBP($(C_i, s)$).
    $\mathcal{B} \leftarrow \mathcal{B} \oplus \arg\min_{\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}} \#\mathcal{A}$.
end for
Return $\mathcal{B}$.
```

For the remainder of this section, fix a BPC instance $\mathcal{I} = (I, s, E)$. The performance guarantees of Algorithm Color_Sets are stated in the next lemma.

[4] For more details on algorithms FFD and AsymptoticBP see, e.g., [27].

Lemma 3.1. *Given a* BPC *instance* $\mathcal{I} = (I, s, E)$*, Algorithm* Color_Sets *returns in polynomial time in* $|\mathcal{I}|$ *a packing* $\mathcal{B}$ *of* $\mathcal{I}$ *such that* $\#\mathcal{B} \leq \chi(G_{\mathcal{I}}) + |L_{\mathcal{I}}| + \frac{3}{2} \cdot s(M_{\mathcal{I}}) + \frac{4}{3} \cdot s(S_{\mathcal{I}})$*. Moreover, if* $\mathcal{I}$ *is a* BPB *instance then* $\#\mathcal{B} \leq \frac{3}{2} \cdot |L_{\mathcal{I}}| + \frac{4}{3} \cdot (\mathrm{OPT}(\mathcal{I}) - |L_{\mathcal{I}}|) + o(\mathrm{OPT}(\mathcal{I}))$*.*

Note that the bounds may not be tight for instances with many large items. Specifically, if $|L_{\mathcal{I}}| \approx \mathrm{OPT}(\mathcal{I})$ then a variant of Algorithm Color_Sets was shown to yield a packing of at least $2.5 \cdot \mathrm{OPT}(\mathcal{I})$ bins [8]. To overcome this, we use an approach based on the simple yet crucial observation that there can be at most one large item in a bin. Therefore, we view the large items as *bins* and assign items to these bins to maximize the total size packed in bins including large items. We formalize the problem initially on a single bin.

Definition 3.2. *In the* bounded independent set problem (BIS) *we are given a graph* $G = (V, E)$*, a weight function* $w : V \rightarrow \mathbb{R}_{\geq 0}$*, and a budget* $\beta \in \mathbb{R}_{\geq 0}$*. The goal is to find an independent set* $S \subseteq V$ *in* G *such that* $w(S)$ *is maximized and* $w(S) \leq \beta$*. Let* $\mathcal{I} = (V, E, w, \beta)$ *be a BIS instance.*

Towards solving BIS, we need the following definitions. For $\alpha \in (0, 1]$, $\mathcal{A}$ is an α-approximation algorithm for a maximization problem $\mathcal{P}$ if, for any instance $\mathcal{I}$ of $\mathcal{P}$, $\mathcal{A}$ outputs a solution of value at least $\alpha \cdot OPT(\mathcal{I})$. A *polynomial-time approximation scheme (PTAS)* for $\mathcal{P}$ is a family of algorithms $\{A_\varepsilon\}$ such that, for any $\varepsilon > 0$, A_ε is a polynomial-time $(1 - \varepsilon)$-approximation algorithm for $\mathcal{P}$. A *fully PTAS (FPTAS)* is a PTAS $\{A_\varepsilon\}$ where, for all $\varepsilon > 0$, A_ε is polynomial also in $\frac{1}{\varepsilon}$. We now describe a PTAS for BIS. Fix a BIS instance $\mathcal{I} = (V, E, w, \beta)$ and $\varepsilon > 0$. As there can be at most ε^{-1} items with weight at least $\varepsilon \cdot \beta$ in some optimal solution OPT for $\mathcal{I}$, we can *guess* this set F of items via enumeration. Then, to add smaller items to F, we define a residual graph G_F of items with weights at most $\varepsilon \cdot \beta$ which are not adjacent to any item in F. Formally, define $G_F = (V_F, E_F)$, where

$$V_F = \{v \in V \backslash F \mid w(v) \leq \varepsilon \cdot \beta, \forall u \in F : (v, u) \notin E\}, \; E_F = \{(u, v) \in E \mid u, v \in V_F\}$$

Now, we find a maximum weight independent set S in G_F. Note that this can be done in polynomial time for perfect and bipartite graphs. If $w(F \cup S) \leq \beta$ then we have an optimal solution; otherwise, we discard iteratively items from S until the remaining items form a feasible solution for $\mathcal{I}$. Since we discard only items with relatively small weights, we lose only an ε-fraction of the weight relative to the optimum. The pseudocode for the scheme is given in Algorithm 2.

Lemma 3.3. *Algorithm 2 is a* PTAS *for* BIS*.*

We note that by a result of [7], unless P=NP, BIS does not admit an *efficient* PTAS, even on bipartite graphs.[5] Thus, our PTAS for this problem is of an independent interest.

[5] An *efficient PTAS* is a PTAS $\{A_\varepsilon\}$ where, for all $\varepsilon > 0$, the running time of A_ε is given by $f(1/\varepsilon)$ times a polynomial of the input size.

Algorithm 2. PTAS$((V,E,w,\beta),\varepsilon)$

```
Initialize $A \leftarrow \emptyset$.
for all independent sets $F \subseteq V$ in $(V,E)$ s.t. $|F| \leq \varepsilon^{-1}, w(F) \leq \beta$ do
    Define the residual graph $G_F = (V_F, E_F)$.
    Find a maximum independent set $S$ of $G_F$ w.r.t. $w$.
    while $w(F \cup S) > \beta$ do
        Choose arbitrary $z \in S$.
        Update $S \leftarrow S \setminus \{z\}$.
    end while
    if $w(A) < w(F \cup S)$ then
        Update $A \leftarrow F \cup S$.
    end if
end for
Return $A$.
```

We now define our maximization problem for multiple bins. We solve a slightly generalized problem in which we have an initial partial packing in t bins. Our goal is to add to these bins (from unpacked items) a subset of items of maximum total size. Formally,

Definition 3.4. *Given a* BPC *instance* $\mathcal{I} = (I, s, E)$*,* $S \subseteq I$*, and a packing* $\mathcal{B} = (B_1, \ldots, B_t)$ *of* S*, define the* maximization problem *of* $\mathcal{I}$ *and* $\mathcal{B}$ *as the problem of finding a packing* $\mathcal{B}+\mathcal{C}$ *of* $S \cup T$*, where* $T \subseteq I \setminus S$ *and* $\mathcal{C} = (C_1, \ldots, C_t)$ *is a packing of* T*, such that* $s(T)$ *is maximized.*

Our solution for BIS is used to obtain a $(1-\frac{1}{e}-\varepsilon)$-approximation for the maximization problem described in Definition 3.4. This is done using the approach of [9] for the more general *separable assignment problem (SAP).*

Lemma 3.5. *Given a* BPC *instance* $\mathcal{I} = (I, s, E)$*,* $S \subseteq I$*, a packing* $\mathcal{B} = (B_1, \ldots, B_t)$ *of* S*, and a constant* $\varepsilon > 0$*, there is an algorithm* MaxSize *which returns in time polynomial in* $|\mathcal{I}|$ *a* $(1-\frac{1}{e}-\varepsilon)$*-approximation for the maximization problem of* $\mathcal{I}$ *and* $\mathcal{B}$*. Moreover, given an* FPTAS *for* BIS *on the graph* (I, E)*, the weight function* s*, and the budget* $\beta = 1$*,* MaxSize *is a* $(1-\frac{1}{e})$*-approximation algorithm for the maximization problem of* $\mathcal{I}$ *and* $\mathcal{B}$*.*

We use the above to obtain a feasible solution for the instance. This is done via a reduction to the maximization problem of the instance with a singleton packing of the large items and packing the remaining items in extra bins. Specifically, in the subroutine MaxSolve, we initially put each item in $L_{\mathcal{I}}$ in a separate bin. Then, additional items from $S_{\mathcal{I}}$ and $M_{\mathcal{I}}$ are added to the bins using Algorithm MaxSize (defined in Lemma 3.5). The remaining items are packed using Algorithm Color_Sets. The pseudocode of the subroutine MaxSolve is given in Algorithm 3.

The proof of Lemma 3.6 uses Lemmas 3.1, 3.3, and 3.5.

Lemma 3.6. *Given a* BPC *instance* $\mathcal{I} = (I, s, E)$ *and an* $\varepsilon > 0$*, Algorithm* MaxSolve *returns in polynomial time in* $|\mathcal{I}|$ *a packing* $\mathcal{C}$ *of* $\mathcal{I}$ *such that there are* $0 \leq x \leq s(M_{\mathcal{I}})$ *and* $0 \leq y \leq s(S_{\mathcal{I}})$ *such that the following holds.*

Algorithm 3. MaxSolve($\mathcal{I} = (I, s, E), \varepsilon$)

```
Define $T \leftarrow (\{\ell\} \mid \ell \in L_{\mathcal{I}})$.
$\mathcal{A} \leftarrow$ MaxSize($\mathcal{I}, L_{\mathcal{I}}, T, \varepsilon$).
$\mathcal{B} \leftarrow$ Color_Sets($\mathcal{I} \setminus$ items($\mathcal{A}$)).
Return $\mathcal{A} \oplus \mathcal{B}$.
```

1. $x + y \leq \text{OPT}(\mathcal{I}) - |L_{\mathcal{I}}| + \left(\frac{1}{e} + \varepsilon\right) \cdot \frac{|L_{\mathcal{I}}|}{2}$.
2. $\#\mathcal{C} \leq \chi(G_{\mathcal{I}}) + |L_{\mathcal{I}}| + \frac{3}{2} \cdot x + \frac{4}{3} \cdot y$.

Lemma 3.6 improves significantly the performance of Algorithm Color_Sets for instances with many large items. However, Algorithm MaxSize may prefer small over medium items; the latter items will be packed by Algorithm Color_Sets (see Algorithm 3). The packing of these medium items may harm the approximation guarantee. Thus, to tackle instances with many medium items, we use a reduction to a maximum matching problem for packing the large and medium items in at most $\text{OPT}(\mathcal{I})$ bins.[6] Then, the remaining items can be packed using Algorithm Color_Sets. The graph used for the following subroutine Matching contains all large and medium items; there is an edge between any two items which can be assigned to the same bin in a packing of the instance $\mathcal{I}$. Formally,

Definition 3.7. *Given a* BPC *instance* $\mathcal{I} = (I, s, E)$*, the* auxiliary graph of $\mathcal{I}$ *is* $H_{\mathcal{I}} = (L_{\mathcal{I}} \cup M_{\mathcal{I}}, E_H)$*, where* $E_H = \{(u, v) \mid u, v \in L_{\mathcal{I}} \cup M_{\mathcal{I}}, s(\{u, v\}) \leq 1, (u, v) \notin E\}$.

Algorithm Matching finds a maximum matching in $H_{\mathcal{I}}$ and outputs a packing of the large and medium items where pairs of items taken to the matching are packed together, and the remaining items are packed in extra bins using Algorithm Color_Sets. The pseudocode of the subroutine Matching is given in Algorithm 4.

Algorithm 4. Matching($\mathcal{I} = (I, s, E)$)

```
Find a maximum matching $\mathcal{M}$ in $H_{\mathcal{I}}$.
$\mathcal{B} \leftarrow (\{u, v\} \mid (u, v) \in \mathcal{M}) \oplus (\{v\} \mid v \in M_{\mathcal{I}} \cup L_{\mathcal{I}}, \forall u \in M_{\mathcal{I}} \cup L_{\mathcal{I}} : (u, v) \notin \mathcal{M})$.
Return $\mathcal{B} \oplus$ Color_Sets($\mathcal{I} \setminus (M_{\mathcal{I}} \cup L_{\mathcal{I}})$).
```

The proof of Lemma 3.8 follows by noting that the cardinality of a maximum matching in $H_{\mathcal{I}}$ in addition to the number of unmatched vertices in $L_{\mathcal{I}} \cup M_{\mathcal{I}}$ is at most $\text{OPT}(\mathcal{I})$.

Lemma 3.8. *Given a* BPC *instance* $\mathcal{I} = (I, s, E)$*, Algorithm* Matching *returns in polynomial time in* $|\mathcal{I}|$ *a packing* $\mathcal{A}$ *of* $\mathcal{I}$ *such that* $\#\mathcal{A} \leq \text{OPT}(\mathcal{I}) + \chi(G_{\mathcal{I}}) + \frac{4}{3} \cdot s(S_{\mathcal{I}})$.

[6] We note that a maximum matching based technique for BPC is used also in [8,15].

We now have the required components for the approximation algorithm for BPC and the asymptotic approximation for BPB. Our algorithm, ApproxBPC, applies all of the above subroutines and returns the packing which uses the smallest number of bins. We use $\varepsilon = 0.0001$ for the error parameter in MaxSolve. The pseudocode of ApproxBPC is given in Algorithm 5.

Algorithm 5. ApproxBPC($\mathcal{I}$)

```
1: Let $\varepsilon = 0.0001$.
2: Compute $\mathcal{A}_1 \leftarrow$ Color_Sets($\mathcal{I}$), $\mathcal{A}_2 \leftarrow$ MaxSolve($\mathcal{I}, \varepsilon$), $\mathcal{A}_3 \leftarrow$ Matching($\mathcal{I}$).
3: Return $\arg\min_{\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}} \#\mathcal{A}$.
```

We give below the main result of this section. The proof follows by the argument that the subroutines Color_Sets, MaxSolve, and Matching handle together most of the difficult cases. Specifically, if the instance contains many large items, then MaxSolve produces the best approximation. If there are many large and medium items, then Matching improves the approximation guarantee. Finally, for any other case, our analysis of the Color_Sets algorithm gives us the desired ratio. We summarize with the next result.

Theorem 3.9. *Algorithm 5 is a* 2.445*-approximation for* BPC *and an asymptotic* 1.391*-approximation for* BPB*.*

4 Split Graphs

In this section we enhance the use of maximization techniques for BPC to obtain an absolute approximation algorithm for BPS. In particular, we improve upon the recent result of Huang et al. [15]. We use as a subroutine the maximization technique as outlined in Lemma 3.5. Specifically, we start by obtaining an FPTAS for the BIS problem on split graphs. For the following, fix a BPS instance $\mathcal{I} = (I, s, E)$. It is well known (see, e.g., [11]) that a partition of the vertices of a split graph into a clique and an independent set can be found in polynomial time. Thus, for simplicity we assume that such a partition of the split graph G is known and given by K_G, S_G. We note that an FPTAS for the BIS problem on split graphs follows from a result of Pferschy and Schauer [24] for *knapsack with conflicts*, since split graphs are a subclass of chordal graphs. We give a simpler FPTAS for our problem in [6].

Lemma 4.1. *There is an algorithm* FPTAS-BIS *that is an* FPTAS *for the* BIS *problem on split graphs.*

Our next goal is to find a suitable initial packing $\mathcal{B}$ to which we apply MaxSize. Clearly, the vertices $K_{G_\mathcal{I}}$ must be assigned to different bins. Therefore, our initial packing contains the vertices of $K_{G_\mathcal{I}}$ distributed to $|K_{G_\mathcal{I}}|$ bins as $\{\{v\} \mid v \in K_{G_\mathcal{I}}\}$. In addition, let $\alpha \in \{0, 1, \ldots, \lceil 2 \cdot s(I) \rceil + 1\}$ be a *guess* of $\text{OPT}(\mathcal{I}) - |K_{G_\mathcal{I}}|$;

then, $(\emptyset)_{i \in [\alpha]}$ is a packing of α bins that do not contain items. Together, the two above packings form the initial packing $\mathcal{B}_\alpha$. Our algorithm uses $\mathsf{MaxSize}$ to add items to the existing bins of $\mathcal{B}_\alpha$ and packs the remaining items using FFD. Note that we do not need an error parameter ε, since we use $\mathsf{MaxSize}$ with an FPTAS (see Lemma 3.5). For simplicity, we assume that $\text{OPT}(\mathcal{I}) \geq 2$ (else we can trivially pack the instance in a single bin) and omit the case where $\text{OPT}(\mathcal{I}) = 1$ from the pseudocode. We give the pseudocode of our algorithm for BPS in Algorithm 6.

Algorithm 6. $\mathsf{Split\text{-}Approx}(\mathcal{I} = (I, s, E))$

```
for $\alpha \in \{0, 1, \ldots, \lceil 2 \cdot s(I) \rceil + 1\}$ do
    Define $\mathcal{B}_\alpha = \{\{v\} \mid v \in K_{G_\mathcal{I}}\} \oplus (\emptyset)_{i \in [\alpha]}$
    $\mathcal{A}_\alpha \leftarrow \mathsf{MaxSize}(\mathcal{I}, K_{G_\mathcal{I}}, \mathcal{B}_\alpha)$.
    $\mathcal{A}^*_\alpha \leftarrow \mathcal{A}_\alpha \oplus \mathsf{FFD}(\mathcal{I} \setminus \mathsf{items}(\mathcal{A}_\alpha))$.
end for
Return $\arg\min_{\alpha \in \{0,1,\ldots,\lceil 2 \cdot s(I) \rceil + 1\}} \#\mathcal{A}^*_\alpha$.
```

By Lemmas 4.1 and 3.5 we have a $\left(1 - \frac{1}{e}\right)$-approximation for the maximization problem of the BPS instance $\mathcal{I}$ and an initial partial packing $\mathcal{B}$. Hence, for a correct guess $\alpha = \text{OPT}(\mathcal{I}) - |K_{G_\mathcal{I}}|$, the remaining items to be packed by FFD are of total size at most $\frac{s(I)}{e}$ and can be packed in $\frac{2 \cdot \text{OPT}(\mathcal{I})}{e}$ bins. Thus, we have

Theorem 4.2. *Algorithm 6 is a $\left(1 + \frac{2}{e}\right)$-approximation for* BPS.

5 Asymptotic Hardness for Bipartite and Split Graphs

In this section we show that there is no APTAS for BPB and BPS, unless $P = NP$. We use a reduction from the *Bounded 3-dimensional matching (B3DM)* problem, that is known to be MAX SNP-complete [18].

For the remainder of this section, let $c > 2$ be some constant. A B3DM instance is a four-tuple $\mathcal{J} = (X, Y, Z, T)$, where X, Y, Z are three disjoint finite sets and $T \subseteq X \times Y \times Z$; also, for each $u \in X \cup Y \cup Z$ there are at most c triples in T to which u belongs. A *solution* for $\mathcal{J}$ is $M \subseteq T$ such that for all $u \in X \cup Y \cup Z$ it holds that u appears in at most one triple of M. The objective is to find a solution M of maximum cardinality. Let $\text{OPT}(\mathcal{J})$ be the value of an optimal solution for $\mathcal{J}$. We use in our reduction a *restricted* instance of B3DM defined as follows.

Definition 5.1. *For $k \in \mathbb{N}$, a* B3DM *instance $\mathcal{J}$ is k-*restricted *if* $\text{OPT}(\mathcal{J}) \geq k$.

In the next lemma we show the hardness of k-restricted B3DM. Intuitively, since B3DM instances $\mathcal{J}$ with $\text{OPT}(\mathcal{J}) \leq k$ are polynomially solvable for a fixed k (e.g., by exhaustive enumeration), it follows that restricted-B3DM must be hard to approximate, by the hardness result of Kann [18].

Lemma 5.2. *There is a constant $\alpha < 1$ such that for any $k \in \mathbb{N}$ there is no α-approximation for the k-restricted* B3DM *problem unless* P=NP.

We give below the main idea of our reduction, showing the asymptotic hardness of BPB and BPS. A more formal description and the proof of Lemma 5.2 are given in [6]. For a sufficiently large $n \in \mathbb{N}$, let $\mathcal{J} = (X, Y, Z, T)$ be an n-restricted instance of B3DM, and let the components of $\mathcal{J}$, together with appropriate indexing, be $U = X \cup Y \cup Z$ and T, where

$$X = \{x_1, \ldots, x_{\tilde{x}}\}, Y = \{y_1, \ldots, y_{\tilde{y}}\}, Z = \{z_1, \ldots, z_{\tilde{z}}\}, T = \{t_1, \ldots, t_{\tilde{t}}\}.$$

We outline our reduction for BPB and later show how it can be modified to yield the hardness result for BPS. Given an n-restricted B3DM instance, we construct a sequence of BPB instances. Each BPB instance contains an item for each element $u \in U$, and an item for each triple $t \in T$. There is an edge (u, t) if $u \in U$ and $t \in T$, and u does not appear in t, i.e., we forbid packing an element u in the same bin with a triple not containing u, for any $u \in U$. Since we do not know the exact value of $\text{OPT}(\mathcal{J})$, we define a family of instances with different number of *filler items*; these items are packed in the optimum of our constructed BPB instance together with elements not taken to the solution for $\mathcal{J}$.

Specifically, for a *guess* $i \in \{n, n+1, \ldots, |T|\}$ of $\text{OPT}(\mathcal{J})$, we define a BPB instance $\mathcal{I}_i = (I_i, s, E)$. The set of items in $\mathcal{I}_i$ is $I_i = U \cup P_i \cup T \cup Q_i$, where P_i, Q_i are a set of $\tilde{t} - i$ (filler) items and a set of $\tilde{x} + \tilde{y} + \tilde{z} - 3 \cdot i$ (filler) items, respectively, such that $P_i \cap U = \emptyset$ and $Q_i \cap U = \emptyset$. The bipartite (conflict) graph of $\mathcal{I}_i$ is $G_i = (I_i, E)$, where $E = E_X \cup E_Y \cup E_Z$ is defined as follows.

$$\begin{aligned} E_X &= \{(x, t) \mid x \in X, t = (x', y, z) \in T, x \neq x'\} \\ E_Y &= \{(y, t) \mid y \in Y, t = (x, y', z) \in T, y \neq y'\} \\ E_Z &= \{(z, t) \mid z \in Z, t = (x, y, z') \in T, z \neq z'\} \end{aligned}$$

Finally, define the sizes of items in $\mathcal{I}_i$ to be

$$\forall u \in U, p \in P_i, q \in Q_i, t \in T: \ s(u) = 0.15, s(p) = 0.45, s(q) = 0.85, s(t) = 0.55.$$

By the above, the only way to pack three items from $x, y, z \in U$ with a triple $t \in T$ is if $(x, y, z) = t$; also, $s(\{x, y, z, t\}) = 1$. For an illustration of the reduction see Fig. 1.

Given a packing $(A_1, \ldots, A_q)$ for the BPB instance $\mathcal{I}_i$, we consider all *useful bins* A_b in the packing, i.e., $A_b = \{x, y, z, t\}$, where $x \in X, y \in Y, z \in Z$ and $t = (x, y, z)$. The triple t from bin A_b is taken to our solution for the original n-restricted B3DM instance $\mathcal{J}$. Note that taking all triples as described above forms a feasible solution for $\mathcal{J}$, since each element is packed only once. Thus, our goal becomes to find a packing for the reduced BPB instance with a maximum number of useful bins. Indeed, since $s(A_b) = 1$ for any useful bin A_b, finding a packing with many useful bins coincides with an efficient approximation for BPB.

For the optimal guess $i^* = \text{OPT}(\mathcal{J})$, it is not hard to see that the optimum for the BPB instance $\mathcal{I}_{i^*}$ satisfies $s(I_{i^*}) = \text{OPT}(\mathcal{I}_{i^*})$; that is, all bins in the optimum

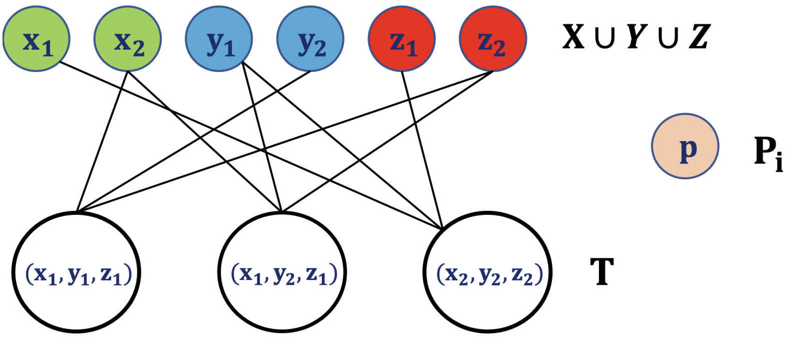


Fig. 1. An illustration of the BPB instance $\mathcal{I}_i = (I_i, s, E)$, where $i = \text{OPT}(\mathcal{J}) = 2$. The optimal solution for $\mathcal{I}_i$ contains the bins $\{x_1, y_1, z_1, (x_1, y_1, z_1)\}, \{x_2, y_2, z_2, (x_2, y_2, z_2)\}$, and $\{p, (x_1, y_2, z_1)\}$; this corresponds to an optimal solution $(x_1, y_1, z_1), (x_2, y_2, z_2)$ for the original B3DM instance. Note that in this example $Q_i = \emptyset$.

are *fully* packed. For a sufficiently large n, and assuming there is an APTAS for BPB, we can find a packing of $\mathcal{I}_{i^*}$ with a large number of bins that are fully packed. A majority of these bins are useful, giving an efficient approximation for the original B3DM instance. A similar reduction to BPS is obtained by adding to the bipartite conflict graph of the BPB instance an edge between any pair of vertices in T; thus, we have a *split* conflict graph. We summarize the above discussion in the next result (the proof is given in [6]).

Theorem 5.3. *There is no* APTAS *for* BPB *and* BPS*, unless* P=NP*.*

6 Discussion

In this work we presented the first theoretical evidence that BPC on polynomially colorable graphs is harder than classic bin packing, even in the special cases of bipartite and split graphs. Furthermore, we introduced a new generic framework for tackling BPC instances, based on a reduction to a maximization problem. Using this framework, we improve the state-of-the-art approximations for BPC on several well studied graph classes.

We note that better bounds for the maximization problems solved within our framework will imply improved approximation guarantees for BPC on perfect, bipartite, and split graphs. It would be interesting to apply our techniques to improve the known results for other graph classes, such as chordal graphs or partial k-trees.

References

1. Adany, R., et al.: All-or-nothing generalized assignment with application to scheduling advertising campaigns. ACM Trans. Algorithms **12**(3), 38:1–38:25 (2016)

2. Beaumont, O., Bonichon, N., Duchon, P., Larchevêque, H.: Distributed approximation algorithm for resource clustering. In: Shvartsman, A.A., Felber, P. (eds.) SIROCCO 2008. LNCS, vol. 5058, pp. 61–73. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69355-0_7
3. Christofides, N.: The vehicle routing problem. Combinatorial optimization (1979)
4. Doron-Arad, I., Kulik, A., Shachnai, H.: An APTAS for bin packing with clique-graph conflicts. In: Lubiw, A., Salavatipour, M. (eds.) WADS 2021. LNCS, vol. 12808, pp. 286–299. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-83508-8_21
5. Doron-Arad, I., Kulik, A., Shachnai, H.: An AFPTAS for bin packing with partition matroid via a new method for LP rounding. In: Proceedings of APPROX (2023)
6. Doron-Arad, I., Shachnai, H.: Approximating bin packing with conflict graphs via maximization techniques. arXiv preprint arXiv:2302.10613 (2023)
7. Doron-Arad, I., Shachnai, H.: Tight bounds for budgeted maximum weight independent set in bipartite and perfect graphs. arXiv preprint arXiv:2307.08592 (2023)
8. Epstein, L., Levin, A.: On bin packing with conflicts. SIAM J. Optim. **19**(3), 1270–1298 (2008)
9. Fleischer, L., Goemans, M.X., Mirrokni, V.S., Sviridenko, M.: Tight approximation algorithms for maximum separable assignment problems. Math. Oper. Res. **36**(3), 416–431 (2011)
10. Garey, M.R., Johnson, D.S.: Computers and intractability. A Guide to the (1979)
11. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Elsevier, Amsterdam (2004)
12. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization, vol. 2. Springer, Berlin (2012)
13. Halldórsson, M.M.: A still better performance guarantee for approximate graph coloring. Inf. Process. Lett. **45**(1), 19–23 (1993)
14. Hoberg, R., Rothvoß, T.: A logarithmic additive integrality gap for bin packing. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 2616–2625. SIAM (2017)
15. Huang, Z., Zhang, A., Dósa, G., Chen, Y., Xiong, C.: Improved approximation algorithms for bin packing with conflicts. Int. J. Found. Comput. Sci. 1–16 (2023)
16. Jansen, K.: An approximation scheme for bin packing with conflicts. J. Comb. Optim. **3**(4), 363–377 (1999)
17. Jansen, K., Öhring, S.R.: Approximation algorithms for time constrained scheduling. Inf. Comput. **132**(2), 85–108 (1997)
18. Kann, V.: Maximum bounded 3-dimensional matching is MAX SNP-complete. Inf. Process. Lett. **37**(1), 27–35 (1991)
19. Karmarkar, N., Karp, R.M.: An efficient approximation scheme for the one-dimensional bin-packing problem. In: 23rd Annual Symposium on Foundations of Computer Science, pp. 312–320. IEEE (1982)
20. de La Vega, W.F., Lueker, G.S.: Bin packing can be solved within 1+ ε in linear time. Combinatorica **1**(4), 349–355 (1981)
21. Laporte, G., Desroches, S.: Examination timetabling by computer. Comput. Oper. Res. **11**(4), 351–360 (1984)
22. McCloskey, B., Shankar, A.: Approaches to bin packing with clique-graph conflicts. University of California, Computer Science Division (2005)
23. Oh, Y., Son, S.: On a constrained bin-packing problem. Technical Report CS-95-14 (1995)
24. Pferschy, U., Schauer, J.: The knapsack problem with conflict graphs. J. Graph Algorithms Appl. **13**(2), 233–249 (2009)

25. Rothvoß, T.: Approximating bin packing within O(log OPT * log log OPT) bins. In: 54th Annual IEEE Symposium on Foundations of Computer Science, pp. 20–29. IEEE Computer Society (2013)
26. Simchi-Levi, D.: New worst-case results for the bin-packing problem. Naval Res. Logist. (NRL) **41**(4), 579–585 (1994)
27. Vazirani, V.V.: Approximation Algorithms. Springer, Berlin, Heidelberg (2001)
28. Zuckerman, D.: Linear degree extractors and the inapproximability of max clique and chromatic number. In: Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing, pp. 681–690 (2006)

α_i-Metric Graphs: Radius, Diameter and all Eccentricities

Feodor F. Dragan[1] and Guillaume Ducoffe[2(✉)]

[1] Computer Science Department, Kent State University, Kent, USA
dragan@cs.kent.edu

[2] National Institute for Research and Development in Informatics and University of Bucharest, Bucharest, Romania
guillaume.ducoffe@ici.ro

Abstract. We extend known results on chordal graphs and distance-hereditary graphs to much larger graph classes by using only a common metric property of these graphs. Specifically, a graph is called α_i-metric ($i \in \mathcal{N}$) if it satisfies the following α_i-metric property for every vertices u, w, v and x: if a shortest path between u and w and a shortest path between x and v share a terminal edge vw, then $d(u,x) \geq d(u,v) + d(v,x) - i$. Roughly, gluing together any two shortest paths along a common terminal edge may not necessarily result in a shortest path but yields a "near-shortest" path with defect at most i. It is known that α_0-metric graphs are exactly ptolemaic graphs, and that chordal graphs and distance-hereditary graphs are α_i-metric for $i = 1$ and $i = 2$, respectively. We show that an additive $O(i)$-approximation of the radius, of the diameter, and in fact of all vertex eccentricities of an α_i-metric graph can be computed in total linear time. Our strongest results are obtained for α_1-metric graphs, for which we prove that a central vertex can be computed in subquadratic time, and even better in linear time for so-called $(\alpha_1, \varDelta)$-metric graphs (a superclass of chordal graphs and of plane triangulations with inner vertices of degree at least 7). The latter answers a question raised in (Dragan, *IPL*, 2020). Our algorithms follow from new results on centers and metric intervals of α_i-metric graphs. In particular, we prove that the diameter of the center is at most $3i + 2$ (at most 3, if $i = 1$). The latter partly answers a question raised in (Yushmanov & Chepoi, *Mathematical Problems in Cybernetics*, 1991).

Keywords: metric graph classes · chordal graphs · α_i-metric · radius · diameter · vertex eccentricity · eccentricity approximating trees · approximation algorithms

1 Introduction

Euclidean spaces have the following nice property: if the geodesic between u and w contains v, and the geodesic between v and x contains w, then their

This work was supported by a grant of the Romanian Ministry of Research, Innovation and Digitalization, CCCDI - UEFISCDI, proect number PN-III-P2-2.1-PED-2021-2142, within PNCDI III.

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 276–290, 2023.
https://doi.org/10.1007/978-3-031-43380-1_20

union must be the geodesic between u and x. In 1991, Chepoi and Yushmanov introduced α_i-metric properties ($i \in \mathcal{N}$), as a way to quantify by how much a graph is close to satisfy this above requirement [39] (see also [10,11] for earlier use of α_1-metric property). All graphs $G = (V, E)$ occurring in this paper are connected, finite, unweighted, undirected, loopless and without multiple edges. The *length* of a path between two vertices u and v is the number of edges in the path. The *distance* $d_G(u, v)$ is the length of a shortest path connecting u and v in G. The *interval* $I_G(u, v)$ between u and v consists of all vertices on shortest (u, v)-paths, that is, it consists of all vertices (metrically) between u and v: $I_G(u, v) = \{x \in V : d_G(u, x) + d_G(x, v) = d_G(u, v)\}$. Let also $I^o_G(u, v) = I_G(u, v) \setminus \{u, v\}$. If no confusion arises, we will omit subindex G.

α_i-metric property: if $v \in I(u, w)$ and $w \in I(v, x)$ are adjacent, then
$$d(u, x) \geq d(u, v) + d(v, x) - i = d(u, v) + 1 + d(w, x) - i.$$

Roughly, gluing together any two shortest paths along a common terminal edge may not necessarily result in a shortest path (unlike in the Euclidean space) but yields a "near-shortest" path with defect at most i. A graph is called α_i-metric if it satisfies the α_i-metric property. α_i-Metric graphs were investigated in [10,11,39]. In particular, it is known that α_0-metric graphs are exactly the distance-hereditary chordal graphs, also known as ptolemaic graphs [32]. Furthermore, α_1-metric graphs contain all chordal graphs [10] and all plane triangulations with inner vertices of degree at least 7 [25]. α_2-Metric graphs contain all distance-hereditary graphs [39] and, even more strongly, all HHD-free graphs [13]. Evidently, every graph is an α_i-metric graph for some i. Chepoi and Yushmanov in [39] also provided a characterization of all α_1-metric graphs: They are exactly the graphs where all disks are convex and the graph W_6^{++} from Fig. 1 is forbidden as an isometric subgraph (see [39] or Theorem 5). This nice characterization was heavily used in [3] in order to characterize δ-hyperbolic graphs with $\delta \leq 1/2$.

Let the *eccentricity* of a vertex v in G be defined as $e_G(v) = \max_{u \in V} d_G(u, v)$. The *diameter* and the *radius* of a graph are defined as $diam(G) = \max_{u \in V} e_G(u)$ and $rad(G) = \min_{u \in V} e_G(u)$, respectively. Let the *center* of a graph G be defined as $C(G) = \{u \in V : e_G(u) = rad(G)\}$. Each vertex from $C(G)$ is called a *central* vertex. In this paper, we investigate the radius, diameter, and all eccentricities computation problems in α_i-metric graphs. Understanding the eccentricity function of a graph and being able to efficiently compute or estimate the diameter, the radius, and all vertex eccentricities is of great importance. For example, in the analysis of complex networks, the eccentricity of a vertex is used to measure its importance: the *eccentricity centrality index* of v [33] is defined as $\frac{1}{e(v)}$. Furthermore, the problem of finding a central vertex is one of the most famous facility location problems. In [39], the following nice relation between the diameter and the radius of an α_i-metric graph G was established: $diam(G) \geq 2rad(G) - i - 1$. Recall that for every graph G, $diam(G) \leq 2rad(G)$ holds. Authors of [39] also raised a question[1] whether the diameter of the center of an α_i-metric graph can

[1] It is conjectured in [39] that $diam(C(G)) \leq i + 2$ for every α_i-metric graph G.

be bounded by a linear function of i. It is known that the diameters of the centers of chordal graphs or of distance-hereditary graphs are at most 3 [11,39].

Related Work. There is a naive algorithm which runs a BFS from each vertex in order to compute all eccentricities. It has running time $O(nm)$ on an n-vertex m-edge graph. Interestingly, this is conditionally optimal for general graphs as well as for some restricted families of graphs [1,5,15,36] since, under plausible complexity assumptions, neither the diameter nor the radius can be computed in truly subquadratic time (i.e., in $O(n^a m^b)$ time, for some positive a, b such that $a + b < 2$). In a quest to break this quadratic barrier, there has been a long line of work presenting more efficient algorithms for computing the diameter and/or the radius, or even better all eccentricities, on some special graph classes. For example, linear-time algorithms are known for computing all eccentricities of interval graphs [26,34]. Extensions of these results to several superclasses of interval graphs are also known [6,7,16,17,21,28–31]. Chordal graphs are another well-known generalization of interval graphs, for which the diameter can unlikely be computed in subquadratic time [5]. For all that, there is an elegant linear-time algorithm for computing the radius and a central vertex of a chordal graph [12]. Until this work there has been little insight about how to extend this nice result to larger graph classes (a notable exception being the work in [13]). This intriguing question is partly addressed in our paper.

Since the existence of subquadratic time algorithm for exact diameter or radius computation is unlikely, even for simple families of graphs, a large volume of work was also devoted to approximation algorithms [1,2,9,36,38]. Authors of [9] additionally address a more challenging question of obtaining an additive c-approximation for the diameter, i.e., an estimate D such that $diam(G) - c \leq D \leq diam(G)$. A simple $\tilde{O}(mn^{1-\epsilon})$ time algorithm achieves an additive n^{ϵ}-approximation and, for any $\epsilon > 0$, getting an additive n^{ϵ}-approximation algorithm for the diameter running in $O(n^{2-\epsilon'})$ time for any $\epsilon' > 2\epsilon$ would falsify the Strong Exponential Time Hypothesis (SETH). However, much better additive approximations can be achieved for graphs with bounded (metric) parameters. For example, a vertex furthest from an arbitrary vertex has eccentricity at least $diam(G) - 2$ for chordal graphs [12] and at least $diam(G) - \lfloor k/2 \rfloor$ for k-chordal graphs [8]. Later, those results were generalized to all δ-hyperbolic graphs [14,15,23,24]. In [20], we also introduce a natural generalization of α_i-metric and hyperbolic graphs, which we call a *(λ, μ)-bow metric*: namely, if two shortest paths $P(u, w)$ and $P(v, x)$ share a common shortest subpath $P(v, w)$ of length more than λ (that is, they overlap by more than λ), then the distance between u and x is at least $d(u, v) + d(v, w) + d(w, x) - \mu$. δ-Hyperbolic graphs are $(\delta, 2\delta)$-bow metric and α_i-metric graphs are $(0, i)$-bow metric. $(\alpha_1, \varDelta)$-Metric graphs form an important subclass of α_1-metric graphs and contain all chordal graphs and all plane triangulations with inner vertices of degree at least 7. In [25], it was shown that every $(\alpha_1, \varDelta)$-metric graph admits an eccentricity 2-approximating spanning tree, i.e., a spanning tree T such that $e_T(v) - e_G(v) \leq 2$ for every vertex v. Finding similar results for general α_1-metric graphs was left as an open problem in [25].

Our Contribution. We prove several new results on intervals, eccentricity and centers in α_i-metric graphs, and their algorithmic applications, thus answering open questions in the literature [18,25,39]. To list our contributions, we need to introduce on our way some additional notations and terminology.

Section 2 is devoted to general α_i-metric graphs ($i \geq 0$). The set $S_k(u,v) = \{x \in I(u,v) : d(u,x) = k\}$ is called a *slice* of the interval $I(u,v)$ where $0 \leq k \leq d(u,v)$. An interval $I(u,v)$ is said to be λ-thin if $d(x,y) \leq \lambda$ for all $x,y \in S_k(u,v)$, $0 < k < d(u,v)$. The smallest integer λ for which all intervals of G are λ-thin is called the *interval thinness* of G. The disk of radius r and center v is defined as $\{u \in V : d(u,v) \leq r\}$, and denoted by $D(v,r)$. In particular, $N[v] = D(v,1)$ and $N(v) = N[v] \setminus \{v\}$ denote the closed and open neighbourhoods of a vertex v, respectively. More generally, for any vertex-subset S and a vertex u, we define $d(u,S) = \min_{v \in S} d(u,v)$, $D(S,r) = \bigcup_{v \in S} D(v,r)$, $N[S] = D(S,1)$ and $N(S) = N[S] \setminus S$. We say that a set of vertices $S \subseteq V$ of a graph $G = (V,E)$ is *d^k-convex* if for every two vertices $x,y \in S$ with $d(x,y) \geq k \geq 0$, the entire interval $I(x,y)$ is in S. For $k \leq 2$, this definition coincides with the usual definition of convex sets in graphs [4,10,37]. We show first that, in α_i-metric graphs G, the intervals are $(i+1)$-thin, and the disks (and, hence, the centers $C(G)$) are d^{2i-1}-convex. The main result of Sect. 2.1 states that the diameter of the center $C(G)$ of G is at most $3i+2$, thus answering a question raised in [39].

Let $F_G(v)$ be the set of all vertices of G that are most distant from v. A pair x,y is called a pair of mutually distant vertices if $x \in F_G(y), y \in F_G(x)$. In Sect. 2.2, we show that an additive $O(i)$-approximation of the radius and of the diameter of an α_i-metric graph G with m edges can be computed in $O(m)$ time. For that, we carefully analyze the eccentricities of most distant vertices from an arbitrary vertex and of mutually distant vertices. In Sect. 2.3, we present three approximation algorithms for all eccentricities, with various trade-offs between their running time and the quality of their approximation. Hence, an additive $O(i)$-approximation of all vertex eccentricities of an α_i-metric graph G with m edges can be computed in $O(m)$ time.

Section 3 is devoted to α_1-metric graphs. The eccentricity function $e(v)$ of a graph G is said to be *unimodal*, if for every non-central vertex v of G there is a neighbor $u \in N(v)$ such that $e(u) < e(v)$ (that is, every local minimum of the eccentricity function is a global minimum). We show in Sect. 3.1 that the eccentricity function on α_1-metric graphs is almost unimodal and we characterize non-central vertices that violate the unimodality (that is, do not have a neighbor with smaller eccentricity). Such behavior of the eccentricity function was observed earlier in chordal graphs [25], in distance-hereditary graphs [22] and in all (α_1, Δ)-metric graphs [25]. In Sect. 3.2, we show that the diameter of $C(G)$ is at most 3. This generalizes known results for chordal graphs [11] and for (α_1, Δ)-metric graphs [25]. Finally, based on these results we present in Sect. 3.3 a local-search algorithm for finding a central vertex of an arbitrary α_1-metric graph in subquadratic time. Our algorithm even achieves linear runtime on (α_1, Δ)-metric graphs, thus answering an open question from [25].

All omitted proofs can be found in our technical report [19].

2 General Case of α_i-Metric Graphs for Arbitrary $i \geq 0$

First we present two important lemmas for what follows.

Lemma 1. *Let G be an α_i-metric graph, and let u, v, x, y be vertices such that $x \in I(u,v)$, $d(u,x) = d(u,y)$, and $d(v,y) \leq d(v,x)+k$. Then, $d(x,y) \leq k+i+2$.*

Lemma 2. *If G is an α_i-metric graph, then its interval thinness is at most $i+1$.*

2.1 Centers of α_i-Metric Graphs

We provide an answer to a question raised in [39] whether the diameter of the center of an α_i-metric graph can be bounded by a linear function of i. For that, we show first that every disk must be d^{2i-1}-convex.

Lemma 3. *Every disk of an α_i-metric graph G is d^{2i-1}-convex. In particular, the center $C(G)$ of an α_i-metric graph G is d^{2i-1}-convex.*

Next auxiliary lemma is crucial in obtaining many results of this section.

Lemma 4. *Let G be an α_i-metric graph. For any $x, y, v \in V$ and any integer $k \in \{0, \ldots, d(x,y)\}$, there is a vertex $c \in S_k(x,y)$ such that $d(v,c) \leq \max\{d(v,x), d(v,y)\} - \min\{d(x,c), d(y,c)\} + i$ and $d(v,c) \leq \max\{d(v,x), d(v,y)\} + i/2$. For an arbitrary vertex $z \in I(x,y)$, we have $d(z,v) \leq \max\{d(x,v), d(y,v)\} - \min\{d(x,z), d(y,z)\} + 2i + 1$ and $d(z,v) \leq \max\{d(x,v), d(y,v)\} + 3i/2 + 1$. Furthermore, $e(z) \leq \max\{e(x), e(y)\} - \min\{d(x,z), d(y,z)\} + 2i + 1$ and $e(z) \leq \max\{e(x), e(y)\} + 3i/2 + 1$ when $v \in F(z)$.*

Using Lemma 4, one can easily prove that the diameter of the center $C(G)$ of an α_i-metric graph G is at most $4i+3$. Below we improve the bound.

Theorem 1. *If G is an α_i-metric graph, then $diam(C(G)) \leq 3i+2$.*

Proof. Let $r = rad(G)$. Suppose by contradiction $diam(C(G)) > 3i+2$. Since $C(G)$ is d^{2i-1}-convex, there exist $x, y \in C(G)$ such that $d(x,y) = 3i+3$ and $I(x,y) \subseteq C(G)$. Furthermore, for every $u \in V$ such that $\max\{d(u,x), d(u,y)\} < r$, $I(x,y) \subseteq D(u, r-1)$ because the latter disk is also d^{2i-1}-convex. Therefore, for every $z \in I(x,y)$, $F(z) \subseteq F(x) \cup F(y)$. Let ab be an edge on a shortest xy-path such that $d(a,x) < d(b,x)$. Assume $F(b) \not\subseteq F(a)$. Let $v \in F(b) \setminus F(a)$. Since G is α_i-metric, $d(v,y) \geq d(v,b) + d(b,y) - i = r + (d(b,y) - i)$. Therefore, $d(b,y) \leq i$. In the same way, if $F(a) \not\subseteq F(b)$, then $d(a,x) \leq i$. By induction, $F(z) \subseteq F(x)$ ($F(z) \subseteq F(y)$, respectively) for every $z \in I(x,y)$ such that $d(y,z) \geq i+1$ ($d(x,z) \geq i+1$, respectively). In particular, if $i+1 \leq t \leq d(x,y) - i - 1$, then $F(z) \subseteq F(x) \cap F(y)$ for every $z \in S_t(x,y)$. Note that the above properties are also true for every $x', y' \in C(G)$ with $d(x', y') \geq 2i-1$, as d^{2i-1}-convexity argument can still be used.

Let $c \in I(x,y)$ be such that $F(c) \subseteq F(x) \cap F(y)$ and $k := |F(c)|$ is minimized. We claim that $k < |F(x) \cap F(y)|$. Indeed, let $v \in F(x) \cap F(y)$ be arbitrary. By Lemma 4, some vertex $c_v \in S_{i+1}(x,y)$ satisfies $d(c_v, v) \leq r-1$. Then, $F(c_v) \subseteq (F(x) \cap F(y)) \setminus \{v\}$, and $k \leq |F(c_v)| \leq |F(x) \cap F(y)| - 1$ by minimality of c. Let $y_c \in I(x,y)$ be such that $F(y_c) \cap F(x) \cap F(y) \subseteq F(c)$ and $d(x, y_c)$ is maximized. We have $y_c \neq y$ because $F(x) \cap F(y) \not\subseteq F(c)$. Therefore, the maximality of $d(x,y_c)$ implies the existence of some $v \in (F(x) \cap F(y)) \setminus F(c)$ such that $d(v, y_c) = r-1$. Since G is α_i-metric, $d(v,y) \geq d(v,y_c) + d(y_c,y) - i = r + (d(y_c,y) - i - 1)$. As a result, $d(y_c,y) \leq i+1$. Then, for every $z \in S_{i+1}(x,y_c)$, since we have $d(z,y_c) = d(x,y) - i - 1 - d(y_c,y) \geq d(x,y) - 2i - 2 = i+1$, $F(z) \subseteq F(x) \cap F(y) \cap F(y_c) \subseteq F(c)$. By minimality of k, $F(z) = F(c)$. However, let $v \in F(c)$ be arbitrary. By Lemma 4, there exists some $c' \in S_{i+1}(x,y_c)$ such that $d(c',v) \leq r-1$, thus contradicting $F(c') = F(c)$. □

2.2 Approximating Radii and Diameters of α_i-Metric Graphs

In this subsection, we show that a vertex with eccentricity at most $rad(G) + O(i)$ and a vertex with eccentricity at least $diam(G) - O(i)$ of an α_i-metric graph G can be found in parameterized linear time. We summarize algorithmic results of this section in the following theorem.

Theorem 2. *There is a linear $(O(m))$ time algorithm which finds vertices v and c of an m-edge α_i-metric graph G such that $e(v) \geq diam(G) - 5i - 2$, $e(c) \leq rad(G) + 4i + (i+1)/2 + 2$ and $C(G) \subseteq D(c, 4i + (i+1)/2 + 2)$. Furthermore, there is an almost linear $(O(im))$ time algorithm which finds vertices v and c of G such that $e(v) \geq diam(G) - 3i - 2$, $e(c) \leq rad(G) + 2i + 1$ and $C(G) \subseteq D(c, 4i+3)$.*

Our algorithms are derived from the following new properties of an α_i-metric graph G, whose proofs are omitted due to lack of space.

- Let x, y be a pair of mutually distant vertices. Then, $d(x,y) \geq 2rad(G) - 4i - 3$ and $d(x,y) \geq diam(G) - 3i - 2$. Furthermore, any middle vertex z of a shortest path between x and y satisfies $e(z) \leq \lceil d(x,y)/2 \rceil + 2i + 1 \leq rad(G) + 2i + 1$, and $C(G) \subseteq D(z, 4i+3)$. There is also a vertex c in $S_{\lfloor d(x,y)/2 \rfloor}(x,y)$ with $e(c) \leq rad(G) + i$.
- Let $v \in V$, $x \in F(v)$ and $y \in F(x)$. Then, $e(x) = d(x,y) \geq 2rad(G) - 2i - diam(C(G)) \geq 2rad(G) - 5i - 2 \geq diam(G) - 5i - 2$. Furthermore, any middle vertex z of a shortest path between x and y satisfies $e(z) \leq \min\{rad(G) + 4i + (i+1)/2 + 2, \lceil d(x,y)/2 \rceil + 7i + 3\}$ and $C(G) \subseteq D(z, 4i + (i+1)/2 + 2)$.

In particular, using at most $O(i)$ *breadth-first-searches*, one can generate a sequence of vertices $v := v_0, x := v_1, y := v_2, v_3, \ldots v_k$ with $k \leq 5i + 4$ such that each v_i is most distant from v_{i-1} and v_k, v_{k-1} are mutually distant vertices (because the initial value $d(x,y) \geq diam(G) - 5i - 2$ can be improved at most $5i+2$ times). Therefore, a pair of mutually distant vertices of an α_i-metric graph can be computed in $O(im)$ total time, thus proving Theorem 2.

For every vertex $v \in V \setminus C(G)$ of a graph G we can define a parameter $loc(v) = \min\{d(v,x) : x \in V, e(x) < e(v)\}$ and call it the *locality* of v. It shows how far from v a vertex with a smaller eccentricity than that one of v exists. In α_i-metric graphs, the locality of each vertex is at most $i+1$.

Lemma 5. *Let G be an α_i-metric graph. Then, for every vertex v, $loc(v) \leq i+1$.*

In α_i-metric graphs, the difference between the eccentricity of a vertex v and the radius of G shows how far vertex v can be from the center $C(G)$ of G.

Lemma 6. *Let G be an α_i-metric graph and k be a positive integer. Then, for every vertex v of G with $e(v) \leq rad(G)+k$, $d(v,C(G)) \leq k+i$.*

Proof. Let x be a vertex from $C(G)$ closest to v. Consider a neighbor z of x on an arbitrary shortest path from x to v. Necessarily, $e(z) = e(x)+1 = rad(G)+1$. Consider a vertex $u \in F(z)$. We have $d(u,x) = rad(G)$ and $x \in I(z,u)$, $z \in I(x,v)$. By the α_i-metric property, $d(v,u) \geq d(v,x)+d(x,u)-i = d(v,x)-i+rad(G)$. As $e(v) \geq d(v,u)$ and $e(v) \leq rad(G)+k$, we get $rad(G)+k \geq e(v) \geq d(v,x)-i+rad(G)$, i.e., $d(v,x) \leq i+k$. □

As an immediate corollary of Lemma 6 we get.

Corollary 1. *Let G be an α_i-metric graph. Then, for every vertex v of G, $d(v,C(G))+rad(G) \geq e(v) \geq d(v,C(G))+rad(G)-i$.*

So, in α_i-metric graphs, to approximate the eccentricity of a vertex v up-to an additive one-sided error i, one needs to know only $rad(G)$ and the distance from v to the center $C(G)$ of G.

2.3 Approximating all Eccentricities in α_i-Metric Graphs

In this subsection, we show that the eccentricities of all vertices of an α_i-metric graph G can be approximated with an additive one-sided error at most $O(i)$ in (almost) linear total time. The following first result is derived from the interesting property that the distances from any vertex v to two mutually distant vertices give a very good estimation on the eccentricity of v.

Theorem 3. *Let G be an α_i-metric graph with m-edges. There is an algorithm which in total almost linear ($O(im)$) time outputs for every vertex $v \in V$ an estimate $\hat{e}(v)$ of its eccentricity $e(v)$ such that $e(v)-3i-2 \leq \hat{e}(v) \leq e(v)$.*

A spanning tree T of a graph G is called an *eccentricity k-approximating spanning tree* if for every vertex v of G $e_T(v) \leq e_G(v)+k$ holds [35]. All $(\alpha_1, \triangle)$-metric graphs (including chordal graphs and the underlying graphs of 7-systolic complexes) admit eccentricity 2-approximating spanning trees [25]. An eccentricity 2-approximating spanning tree of a chordal graph can be computed in linear time [18]. An eccentricity k-approximating spanning tree with minimum k can be found in $O(nm)$ time for any n-vertex, m-edge graph G [27]. It is also known [15,23] that if G is a δ-hyperbolic graph, then G admits an eccentricity $(4\delta+1)$-approximating spanning tree constructible in $O(\delta m)$ time and an eccentricity (6δ)-approximating spanning tree constructible in $O(m)$ time.

Lemma 7. *Let G be an α_i-metric graph with m edges. If c is a middle vertex of any shortest path between a pair x, y of mutually distant vertices of G and T is a $BFS(c)$-tree of G, then, for every vertex v of G, $e_G(v) \leq e_T(v) \leq e_G(v) + 4i + 2$. That is, G admits an eccentricity $(4i + 2)$-approximating spanning tree constructible in $O(im)$ time.*

Lemma 8. *Let G be an α_i-metric graph with m edges, and let $z \in V$, $x \in F(z)$ and $y \in F(x)$. If c is a middle vertex of any shortest path between x and y and T is a $BFS(c)$-tree of G, then, for every vertex v of G, $e_G(v) \leq e_T(v) \leq e_G(v) + 9i + 5$. That is, G admits an eccentricity $(9i+5)$-approximating spanning tree constructible in $O(m)$ time.*

It is a folklore by now that the eccentricities of all vertices in any tree $T = (V, U)$ can be computed in $O(|V|)$ total time. Consequently, by Lemma 7 and Lemma 8, we get the following additive approximations for the vertex eccentricities in α_i-metric graphs.

Theorem 4. *Let G be an α_i-metric graph with m edges. There is an algorithm which in total linear $(O(m))$ time outputs for every vertex $v \in V$ an estimate $\hat{e}(v)$ of its eccentricity $e(v)$ such that $e(v) \leq \hat{e}(v) \leq e(v) + 9i + 5$. Furthermore, there is an algorithm which in total almost linear $(O(im))$ time outputs for every vertex $v \in V$ an estimate $\hat{e}(v)$ of its eccentricity $e(v)$ such that $e(v) \leq \hat{e}(v) \leq e(v) + 4i + 2$.*

3 Graphs with α_1-Metric

Now we concentrate on α_1-metric graphs, which contain all chordal graphs and all plane triangulations with inner vertices of degree at least 7 (see [10,11,25,39]). For them we get much sharper bounds. First we recall some known results.

Theorem 5 ([39]). *G is an α_1-metric graph if and only if all disks $D(v, k)$ $(v \in V,\ k \geq 1)$ of G are convex and G does not contain the graph W_6^{++} from Fig. 1 as an isometric subgraph.*

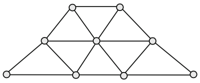


Fig. 1. Forbidden isometric subgraph W_6^{++}.

Lemma 9 ([37]). *All disks $D(v, k)$ $(v \in V,\ k \geq 1)$ of a graph G are convex if and only if for every vertices $x, y, z \in V$ and $v \in I(x, y)$, $d(v, z) \leq \max\{d(x, z), d(y, z)\}$.*

Letting z to be from $F(v)$, we get:

Corollary 2. *If all disks $D(v, k)$ $(v \in V,\ k \geq 1)$ of a graph G are convex then for every vertices $x, y \in V$ and $v \in I(x, y)$, $e(v) \leq \max\{e(x), e(y)\}$.*

Lemma 10 ([25]). *Let G be an α_1-metric graph and x be its arbitrary vertex with $e(x) \geq rad(G) + 1$. Then, for every vertex $z \in F(x)$ and every neighbor v of x in $I(x, z)$, $e(v) \leq e(x)$ holds.*

3.1 The Eccentricity Function on α_1-Metric Graphs is Almost Unimodal

We prove the following theorem.

Theorem 6. *Let G be an α_1-metric graph and v be an arbitrary vertex of G. If*

(i) $e(v) > rad(G) + 1$ or
(ii) $e(v) = rad(G) + 1$ and $diam(G) < 2\,rad(G) - 1$,

then there must exist a neighbor w of v with $e(w) < e(v)$.

Theorem 6 says that if a vertex v with $loc(v) > 1$ exists in an α_1-metric graph G then $diam(G) \geq 2rad(G) - 1$, $e(v) = rad(G) + 1$ and $d(v, C(G)) = 2$. Two α_1-metric graphs depicted in Fig. 2 show that this result is sharp.

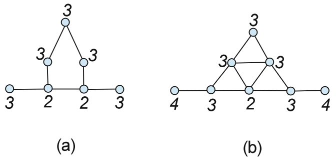


Fig. 2. Sharpness of the result of Theorem 6. (a) An α_1-metric graph G with $diam(G) = 2rad(G) - 1$ and a vertex (topmost) with locality 2. (b) A chordal graph (and hence an α_1-metric graph) G with $diam(G) = 2rad(G)$ and a vertex (topmost) with locality 2. The number next to each vertex indicates its eccentricity.

We formulate three interesting corollaries of Theorem 6.

Corollary 3. *Let G be an α_1-metric graph. Then,*

(i) if $diam(G) < 2rad(G) - 1$ (i.e., $diam(G) = 2rad(G) - 2$) then every local minimum of the eccentricity function on G is a global minimum.
(ii) if $diam(G) \geq 2rad(G) - 1$ then every local minimum of the eccentricity function on G is a global minimum or is at distance 2 from a global minimum.

Corollary 4. *For every α_1-metric graph G and any vertex v, the following formula is true: $d(v, C(G)) + rad(G) \geq e(v) \geq d(v, C(G)) + rad(G) - \epsilon$, where $\epsilon \leq 1$, if $diam(G) \geq 2rad(G) - 1$, and $\epsilon = 0$, otherwise.*

A path $(v = v_0, \ldots, v_k = x)$ of a graph G from a vertex v to a vertex x is called *strictly decreasing* (with respect to the eccentricity function) if for every i $(0 \leq i \leq k-1)$, $e(v_i) > e(v_{i+1})$. It is called *decreasing* if for every i $(0 \leq i \leq k-1)$, $e(v_i) \geq e(v_{i+1})$. An edge $ab \in E$ of a graph G is called *horizontal* (with respect to the eccentricity function) if $e(a) = e(b)$.

Corollary 5. *Let G be an α_1-metric graph and v be an arbitrary vertex. Then, there is a shortest path $P(v,x)$ from v to a closest vertex x in $C(G)$ such that:*

(i) if $diam(G) < 2rad(G) - 1$ (i.e., $diam(G) = 2rad(G) - 2$) then $P(v,x)$ is strictly decreasing;
(ii) if $diam(G) \geq 2rad(G) - 1$ then $P(v,x)$ is decreasing and can have only one horizontal edge, with an end-vertex adjacent to x.

3.2 Diameters of Centers of α_1-Metric Graphs

In this section, we provide sharp bounds on the diameter and the radius of the center of an α_1-metric graph. Previously, it was known that the diameter (the radius) of the center of a chordal graph is at most 3 (at most 2, respectively) [11]. To prove our result, we will need a few technical lemmas.

Lemma 11. *Let G be an α_1-metric graph. Then, for every shortest path $P = (x_1, x_2, x_3, x_4, x_5)$ and a vertex u of G with $d(u, x_i) = k$ for all $i \in \{1, \ldots, 5\}$, there exist vertices t, w, s such that $d(t, u) = d(s, u) = k - 1$, $k - 2 \leq d(w, u) \leq k - 1$, and t is adjacent to x_1, x_2, w and s is adjacent to x_4, x_5, w.*

Lemma 12. *Let G be an α_1-metric graph. Then, for every shortest path $P = (x_1, x_2, x_3, x_4, x_5)$ and a vertex u of G with $d(u, x_i) = k$ for all $i \in \{1, \ldots, 5\}$, there exists a shortest path $Q = (y_1, y_2, y_3)$ such that $d(u, y_i) = k - 1$, for each $i \in \{1, \ldots, 3\}$, and $N(y_1) \cap P = \{x_1, x_2\}$, $N(y_2) \cap P = \{x_2, x_3, x_4\}$ and $N(y_3) \cap P = \{x_4, x_5\}$.*

Theorem 7. *Let G be an α_1-metric graph. For every pair of vertices s, t of G with $d(s,t) \geq 4$ there exists a vertex $c \in I^o(s,t)$ such that $e(c) < \max\{e(s), e(t)\}$.*

Proof. It is sufficient to prove the statement for vertices s, t with $d(s,t) = 4$. We know, by Corollary 2, that $e(c) \leq \max\{e(s), e(t)\}$ for every $c \in I(s,t)$. Assume, by way of contradiction, that there is no vertex $c \in I^o(s,t)$ such that $e(c) < \max\{e(s), e(t)\}$. Let, without loss of generality, $e(s) \leq e(t)$. Then, for every $c \in I^o(s,t)$, $e(c) = e(t)$. Consider a vertex $c \in S_1(s,t)$. If $e(c) > e(s)$, then $e(c) = e(s) + 1$. Consider a vertex z from $F(c)$. Necessarily, $z \in F(s)$. Applying the α_1-metric property to $c \in I(s,t)$, $s \in I(c,z)$, we get $e(c) = e(t) \geq d(t,z) \geq d(c,t) + d(s,z) = 3 + e(s) = 2 + e(c)$, which is impossible. So, $e(s) = e(c) = e(t)$ for every $c \in I^o(s,t)$. Consider an arbitrary shortest path $P = (s = x_1, x_2, x_3, x_4, x_5 = t)$ connecting vertices s and t. We claim that for any vertex $u \in F(x_3)$ all vertices of P are at distance $k := d(u, x_3) = e(x_3)$ from u. As $e(x_i) = e(x_3)$, we know that $d(u, x_i) \leq k$ ($1 \leq i \leq 5$). Assume $d(u, x_i) = k - 1$, $d(u, x_{i+1}) = k$, and $i \leq 2$. Then, the α_1-metric property applied to $x_i \in I(u, x_{i+1})$ and $x_{i+1} \in I(x_i, x_{i+3})$ gives $d(x_{i+3}, u) \geq k - 1 + 2 = k + 1$, which is a contradiction with $d(u, x_{i+3}) \leq k$. So, $d(u, x_1) = d(u, x_2) = k$. By symmetry, also $d(u, x_4) = d(u, x_5) = k$. Hence, by Lemma 12, for the path $P = (x_1, x_2, x_3, x_4, x_5)$, there exists a shortest path $Q = (y_1, y_2, y_3)$ such that $d(u, y_i) = k - 1$, for each $i \in \{1, \ldots, 3\}$, and $N(y_1) \cap P = \{x_1, x_2\}$, $N(y_2) \cap P =$

$\{x_2, x_3, x_4\}$ and $N(y_3) \cap P = \{x_4, x_5\}$. As $y_i \in I^o(x_1, x_5) = I^o(s,t)$ for each $i \in \{1, \ldots, 3\}$, we have $e(y_i) = e(x_3) = k$.

All the above holds for every shortest path $P = (s = x_1, x_2, x_3, x_4, x_5 = t)$ connecting vertices s and t. Now, assume that P is chosen in such a way that, among all vertices in $S_2(s,t)$, the vertex x_3 has the minimum number of furthest vertices, i.e., $|F(x_3)|$ is as small as possible. As y_2 also belongs to $S_2(s,t)$ and has u at distance $k-1$, by the choice of x_3, there must exist a vertex $u' \in F(y_2)$ which is at distance $k-1$ from x_3. Applying the previous arguments to the path $P' := (s = x_1, x_2, y_2, x_4, x_5 = t)$, we will have $d(x_i, u') = d(y_2, u') = k$ for $i = 1, 2, 4, 5$ and, by Lemma 12, get two more vertices v and w at distance $k-1$ from u' such that $vx_1, vx_2, wx_4, wx_5 \in E$ and $vy_2, wy_2 \notin E$. By convexity of disk $D(u', k-1)$, also $vx_3, wx_3 \in E$. Now consider the disk $D(x_2, 2)$. Since y_3, w are in the disk and x_5 is not, vertices w and y_3 must be adjacent. But then vertices y_2, x_3, w, y_3 form an induced cycle C_4, which is forbidden because disks in G must be convex.

Thus, a vertex $c \in I^o(s,t)$ with $e(c) < \max\{e(s), e(t)\}$ must exist. □

Corollary 6. *Let G be an α_1-metric graph. Then, $diam(C(G)) \leq 3$ and $rad(C(G)) \leq 2$.*

Corollary 6 generalizes an old result on chordal graphs [11]. Finally, note that results of Theorem 7 and Corollary 6 are sharp.

3.3 Finding a Central Vertex of an α_1-Metric Graph

We present a local-search algorithm for computing a central vertex of an arbitrary α_1-metric graph in subquadratic time (Theorem 8). Our algorithm even achieves linear runtime on an important subclass of α_1-metric graphs, namely, (α_1, Δ)-metric graphs (Theorem 9), thus answering an open question from [25] where this subclass was introduced. The (α_1, Δ)-metric graphs are exactly the α_1-metric graphs that further satisfy the so-called triangle condition: for every vertices u, v, w such that u and v are adjacent, and $d(u,w) = d(v,w) = k$, there must exist some common neighbour $x \in N(u) \cap N(v)$ such that $d(x,w) = k-1$. Chordal graphs, and plane triangulations with inner vertices of degree at least 7, are (α_1, Δ)-metric graphs (see [10,11,25,39]).

We first introduce the required new notations and terminology for this part. In what follows, let $\texttt{proj}(v, A) = \{a \in A : d(v,a) = d(v,A)\}$ denote the metric projection of a vertex v to a vertex subset A. For every k such that $0 \leq k \leq d(v, A)$, we define $S_k(A, v) = \bigcup\{S_k(a, v) : a \in \texttt{proj}(v, A)\}$. A *distance-$k$ gate* of v with respect to A is a vertex v^* such that $v^* \in \bigcap\{I(a, v) : a \in \texttt{proj}(v, A)\}$ and $d(v^*, A) \leq k$. If $k = 1$, then following [12] we simply call it a gate. Note that every vertex v such that $d(v, A) \leq k$ is its own distance-k gate. A cornerstone of our main algorithms is that, in α_1-metric graphs, for every *closed neighbourhood* (for every *clique*, resp.), every vertex has a gate (a distance-two gate, resp.). Proofs are omitted due to lack of space.

The problem of computing gates has already attracted some attention, *e.g.*, see [12]. We use this routine in the design of our main algorithms.

Lemma 13 **([30]).** *Let A be an arbitrary subset of vertices in some graph G with m edges. In total $O(m)$ time, we can map every vertex $v \notin A$ to some vertex $v^* \in D(v, d(v, A) - 1) \cap N(A)$ such that $|N(v^*) \cap A|$ is maximized. Furthermore, if v has a gate with respect to A, then v^* is a gate of v.*

The efficient computation of distance-two gates is more challenging. We present a subquadratic-time procedure that only works in our special setting.

Lemma 14. *Let K be a clique in some α_1-metric graph G with m edges. In total $O(m^{1.41})$ time, we can map every vertex $v \notin K$ to some distance-two gate v^* with respect to K. Furthermore, in doing so we can also map v^* to some independent set $J_K(v^*) \subseteq D(v^*, 1)$ such that $\mathtt{proj}(v^*, K)$ is the disjoint union of neighbour-sets $N(w) \cap K$, for every $w \in J_K(v^*)$.*

Then, we turn our attention to the following subproblem: being given a vertex x in an α_1-metric graph G, either compute a neighbour y such that $e(y) < e(x)$, or assert that x is a local minimum for the eccentricity function (but not necessarily a central vertex). Our analysis of the next algorithms essentially follows from the results of Sect. 3.1. We first present the following special case, for which we obtain a better runtime than for the more general Lemma 16.

Lemma 15. *Let x be an arbitrary vertex in an α_1-metric graph G with m edges. If $e(x) \geq rad(G) + 2$, then $\bigcap\{N(x) \cap I(x, z) : z \in F(x)\} \neq \emptyset$, and every neighbour y in this subset satisfies $e(y) < e(x)$. In particular, there is an $O(m)$-time algorithm that either outputs a $y \in N(x)$ such that $e(y) < e(x)$, or asserts that $e(x) \leq rad(G) + 1$.*

Note that Lemma 15 relies on the existence of gates for every vertex with respect to $D(x, 1)$, and that it uses Lemma 13 as a subroutine. We can strengthen Lemma 15 as follows, at the expenses of a higher runtime.

Lemma 16. *Let x be an arbitrary vertex in an α_1-metric graph G with m edges. There is an $O(m^{1.41})$-time algorithm that either outputs a $y \in N(x)$ such that $e(y) < e(x)$, or asserts that x is a local minimum for the eccentricity function. If G is (α_1, Δ)-metric, then its runtime can be lowered down to $O(m)$.*

The improved runtime for (α_1, Δ)-metric graphs comes from the property that for every clique in such a graph, every vertex has a gate [25], and that gates are easier to compute than distance-two gates.

Theorem 8. *If G is an α_1-metric graph with m edges, then a vertex x_0 such that $e(x_0) \leq rad(G) + 1$ can be computed in $O(m)$ time. Furthermore, a central vertex can be computed in $O(m^{1.71})$ time.*

Let us sketch our algorithm for general α_1-metric graphs. By Theorem 2, we can compute in $O(m)$ time a vertex x_0 such that $e(x_0) \leq rad(G) + 3$. We repeatedly apply Lemma 15 until we can further assert that $e(x_0) \leq rad(G) + 1$ (and, hence, by Theorem 6, $d(x_0, C(G)) \leq 2$). Since there are at most two calls to this local-search procedure, the runtime is in $O(m)$. Now, if x_0 has small enough degree ($\leq m^{\cdot 29}$), then we can apply Lemma 16 to every vertex of $D(x_0, 1)$ in order to compute a minimum eccentricity vertex within $D(x_0, 2)$, which must be central. Otherwise, we further restrict our search for a central vertex to $X_1 := D(x_0, 5) \cap D(z_0, e(x_0) - 1)$, for some arbitrary $z_0 \in F(x_0)$, which must be a superset of $C(G)$ provided that x_0 is non-central. Starting from an arbitrary vertex of X_1, if we apply Lemma 15 at most five times, then we can either extract some $x_1 \in X_1$ such that $e(x_1) \leq e(x_0)$, or assert that x_0 is central. If $e(x_1) < e(x_0)$, then x_1 is central. Otherwise, we repeat our above procedure for x_1. Doing so, we compute a decreasing chain of subsets $X_0 = V \supset X_1 \supset \ldots \supset X_i \supset \ldots X_T$, and vertices $x_0, x_1, \ldots, x_i, \ldots, x_T$ such that $x_i \in X_i \setminus X_{i+1}$ for every i, $0 \leq i \leq T$. We continue until we compute a vertex of smaller eccentricity than x_0, or we reach $X_{T+1} = \emptyset$ (in which case, x_0 is central).

To lower the runtime to $O(m)$ for the (α_1, Δ)-metric graphs, we use a different approach that is based on additional properties of these graphs. Unfortunately, these properties crucially depend on the triangle condition.

Theorem 9. *If G is an (α_1, Δ)-metric graph with m edges, then a central vertex can be computed in $O(m)$ time.*

Roughly, the algorithm starts from a vertex x which is a local minimum for the eccentricity function of G. We run a core procedure which either outputs two adjacent vertices $u, v \in D(x, 1)$ such that $e(u) = e(v) = e(x)$ and $F(u), F(v)$ are not comparable by inclusion, or outputs a central vertex. In the former case, we can either assert that x is central, or extract a central vertex from $S_{e(x)-1}(y, z)$ for some arbitrary $y \in F(u) \setminus F(v)$, $z \in F(v) \setminus F(u)$. Indeed, we can apply Lemma 16 to the latter slice because $S_{e(x)-1}(y, z) \subseteq D(w, 1)$ for some w [25].

References

1. Abboud, A., Vassilevska Williams, V., Wang, J.: Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In: SODA, pp. 377–391. SIAM (2016)
2. Backurs, A., Roditty, L., Segal, G., Vassilevska Williams, V., Wein, N.: Towards tight approximation bounds for graph diameter and eccentricities. In: STOC 2018, pp. 267–280 (2018)
3. Bandelt, H.-J., Chepoi, V.: 1-hyperbolic graphs. SIAM J. Discr. Math. **16**, 323–334 (2003)
4. Boltyanskii, V.G., Soltan, P.S.: Combinatorial geometry of various classes of convex sets [in Russian]. Ştiinţa, Kishinev (1978)
5. Borassi, M., Crescenzi, P., Habib, M.: Into the square: on the complexity of some quadratic-time solvable problems. Electron. Notes TCS **322**, 51–67 (2016)
6. Brandstädt, A., Chepoi, V., Dragan, F.F.: The algorithmic use of hypertree structure and maximum neighbourhood orderings. DAM **82**, 43–77 (1998)
7. Corneil, D., Dragan, F.F., Habib, M., Paul, C.: Diameter determination on restricted graph families. DAM **113**, 143–166 (2001)

8. Corneil, D.G., Dragan, F.F., Köhler, E.: On the power of BFS to determine a graph's diameter. Networks **42**, 209–222 (2003)
9. Chechik, S., Larkin, D.H., Roditty, L., Schoenebeck, G., Tarjan, R.E., Vassilevska Williams, V.: Better approximation algorithms for the graph diameter. In: SODA 2014, pp. 1041–1052 (2014)
10. Chepoi, V.: Some d-convexity properties in triangulated graphs. In: Mathematical Research, vol. 87, pp. 164–177. Ştiinţa, Chişinău (1986). (Russian)
11. Chepoi, V.: Centers of triangulated graphs. Math. Notes **43**, 143–151 (1988)
12. Chepoi, V., Dragan, F.: A linear-time algorithm for finding a central vertex of a chordal graph. In: van Leeuwen, J. (ed.) ESA 1994. LNCS, vol. 855, pp. 159–170. Springer, Heidelberg (1994). https://doi.org/10.1007/BFb0049406
13. Chepoi, V., Dragan, F.F.: Finding a central vertex in an HHD-free graph. DAM **131**(1), 93–111 (2003)
14. Chepoi, V.D., Dragan, F.F., Estellon, B., Habib, M., Vaxès, Y.: Diameters, centers, and approximating trees of δ-hyperbolic geodesic spaces and graphs. In: Proceedings of the 24th Annual ACM Symposium on Computational Geometry (SoCG 2008), 9–11 June 2008, College Park, Maryland, USA, pp. 59–68 (2008)
15. Chepoi, V., Dragan, F.F., Habib, M., Vaxès, Y., Alrasheed, H.: Fast approximation of eccentricities and distances in hyperbolic graphs. J. Graph Algorithms Appl. **23**, 393–433 (2019)
16. Dragan, F.F.: Centers of graphs and the Helly property (in Russian). Ph.D. thesis, Moldava State University, Chişinău (1989)
17. Dragan, F.F.: HT-graphs: centers, connected R-domination and Steiner trees. Comput. Sci. J. Moldova (Kishinev) **1**, 64–83 (1993)
18. Dragan, F.F.: An eccentricity 2-approximating spanning tree of a chordal graph is computable in linear time. Inf. Process. Lett. **154**, 105873 (2020)
19. Dragan, F.F., Ducoffe, G.: α_i-metric graphs: radius, diameter and all eccentricities. CoRR, abs/2305.02545 (2023)
20. Dragan, F.F., Ducoffe, G.: α_i-metric graphs: hyperbolicity. In: Preparation (2022–2023)
21. Dragan, F.F., Ducoffe, G., Guarnera, H.M.: Fast deterministic algorithms for computing all eccentricities in (hyperbolic) Helly graphs. In: Lubiw, A., Salavatipour, M. (eds.) WADS 2021. LNCS, vol. 12808, pp. 300–314. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-83508-8_22
22. Dragan, F.F., Guarnera, H.M.: Eccentricity function in distance-hereditary graphs. Theor. Comput. Sci. **833**, 26–40 (2020)
23. Dragan, F.F., Guarnera, H.M.: Eccentricity terrain of δ-hyperbolic graphs. J. Comput. Syst. Sci. **112**, 50–65 (2020)
24. Dragan, F.F., Habib, M., Viennot, L.: Revisiting radius, diameter, and all eccentricity computation in graphs through certificates. CoRR, abs/1803.04660 (2018)
25. Dragan, F.F., Köhler, E., Alrasheed, H.: Eccentricity approximating trees. Discret. Appl. Math. **232**, 142–156 (2017)
26. Dragan, F.F., Nicolai, F., Brandstädt, A.: LexBFS-orderings and powers of graphs. In: d'Amore, F., Franciosa, P.G., Marchetti-Spaccamela, A. (eds.) WG 1996. LNCS, vol. 1197, pp. 166–180. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-62559-3_15
27. Ducoffe, G.: Easy computation of eccentricity approximating trees. DAM **260**, 267–271 (2019)
28. Ducoffe, G.: Around the diameter of AT-free graphs. JGT **99**(4), 594–614 (2022)

29. Ducoffe, G.: Beyond Helly graphs: the diameter problem on absolute retracts. In: Kowalik, Ł, Pilipczuk, M., Rzążewski, P. (eds.) WG 2021. LNCS, vol. 12911, pp. 321–335. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86838-3_25
30. Ducoffe, G.: Distance problems within Helly graphs and k-Helly graphs. Theor. Comput. Sci. **946**, 113690 (2023)
31. Ducoffe, G., Dragan, F.F.: A story of diameter, radius, and (almost) Helly property. Networks **77**, 435–453 (2021)
32. Howorka, E.: A characterization of distance-hereditary graphs. Quart. J. Math. Oxford Ser. **28**, 417–420 (1977)
33. Koschützki, D., Lehmann, K.A., Peeters, L., Richter, S., Tenfelde-Podehl, D., Zlotowski, O.: Centrality indices. In: Brandes, U., Erlebach, T. (eds.) Network Analysis. LNCS, vol. 3418, pp. 16–61. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31955-9_3
34. Olariu, S.: A simple linear-time algorithm for computing the center of an interval graph. Int. J. Comput. Math. **34**, 121–128 (1990)
35. Prisner, E.: Eccentricity-approximating trees in chordal graphs. Discret. Math. **220**, 263–269 (2000)
36. Roditty, L., Vassilevska Williams, V.: Fast approximation algorithms for the diameter and radius of sparse graphs. In: STOC, pp. 515–524. ACM (2013)
37. Soltan, V.P., Chepoi, V.D.: Conditions for invariance of set diameters under d-convexification in a graph. Cybernetics **19**, 750–756 (1983). (Russian, English transl.)
38. Weimann, O., Yuster, R.: Approximating the diameter of planar graphs in near linear time. ACM Trans. Algorithms **12**(1), 12:1–12:13 (2016)
39. Yushmanov, S.V., Chepoi, V.: A general method of investigation of metric graph properties related to the eccentricity. In: Mathematical Problems in Cybernetics, vol. 3, pp. 217–232. Nauka, Moscow (1991). (Russian)

Maximum Edge Colouring Problem On Graphs That Exclude a Fixed Minor

Zdeněk Dvořák[1] and Abhiruk Lahiri[2](✉)

[1] Charles University, 11800 Prague, Czech Republic
rakdver@iuuk.mff.cuni.cz

[2] Heinrich Heine University, 40225 Düsseldorf, Germany
abhiruk@hhu.de

Abstract. The maximum edge colouring problem considers the maximum colour assignment to edges of a graph under the condition that every vertex has at most a fixed number of distinct coloured edges incident on it. If that fixed number is q we call the colouring a maximum edge q-colouring. The problem models a non-overlapping frequency channel assignment question on wireless networks. The problem has also been studied from a purely combinatorial perspective in the graph theory literature.

We study the question when the input graph is sparse. We show the problem remains NP-hard on 1-apex graphs. We also show that there exists PTAS for the problem on minor-free graphs. The PTAS is based on a recently developed Baker game technique for proper minor-closed classes, thus avoiding the need to use any involved structural results. This further pushes the Baker game technique beyond the problems expressible in the first-order logic.

Keywords: Polynomial-time approximation scheme · Edge colouring · Minor-free graphs

1 Introduction

For a graph $G = (V, E)$, an *edge q-colouring* of G is a mapping $f\colon E(G) \to \mathbb{Z}^+$ such that the number of distinct colours incident on any vertex $v \in V(G)$ is bounded by q, and the *spread* of f is the total number of distinct colours it uses. The *maximum edge q-chromatic number* $\overline{\chi}'_q(G)$ of G is the maximum spread of an edge q-colouring of G.

A more general notion has been studied in the combinatorics and graph theory communities in the context of extremal problems, called *anti-Ramsey number*. For given graphs G and H, the *anti-Ramsey number* $\mathrm{ar}(G, H)$ denotes the maximum number of colours that can be assigned to edges of G so that there does not exist any subgraph isomorphic to H which is *rainbow*, i.e., all the edges

Supported by project 22-17398S (Flows and cycles in graphs on surfaces) of Czech Science Foundation

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 291–304, 2023.
https://doi.org/10.1007/978-3-031-43380-1_21

of the subgraph receive distinct colours under the colouring. The maximum edge q-chromatic number of G is clearly equal to $\mathrm{ar}(G, K_{1,q+1})$, where $K_{1,q+1}$ is a star with $q+1$ edges.

The notion of anti-Ramsey number was introduced by Erdős and Simonovits in 1973 [14]. The initial studies focused on determining tight bounds for $\mathrm{ar}(G, H)$. A lot of research has been done on the case when $G = K_n$, the complete graph, and H is a specific type of a graph (a path, a complete graph, ...) [26,29,31]. For a comprehensive overview of known results in this area, we refer interested readers to [17]. Bounds on $\mathrm{ar}(K_n, H)$ where H is a star graph are reported in [21,24]. Gorgol and Lazuka computed the exact value of $\mathrm{ar}(K_n, H)$ when H is $K_{1,4}$ with an edge added to it [18]. For general graph G, Montellano-Ballesteros studied $\mathrm{ar}(G, K_{1,q})$ and reported an upper bound [25].

The algorithmic aspects of this problem started gaining attention from researchers around fifteen years ago, due to its application to wireless networks [27]. At that time there was a great interest to increase the capacity of *wireless mesh networks* (which are commonly called *wireless broadband* nowadays). The solution that became the industry standard is to use multiple channels and transceivers with the ability to simultaneously communicate with many neighbours using multiple radios over the channels [27]. Wireless networks based on the IEEE 802.11a/b/g and 802.16 standards are examples of such systems. But, there is a physical bottleneck in deploying this solution. Enabling every wireless node to have multiple radios can possibly create an interface and thus reduce reliability. To circumvent that, there is a limit on the number of channels simultaneously used by any wireless node. In the IEEE 802.11 b/g standard and IEEE 802.11a standard, the numbers of permittable simultaneous channels are 3 and 12, respectively [34].

If we model a wireless network as a graph where each wireless node corresponds to a vertex of the graph, then the problem can be formulated as a maximum edge colouring problem. The nonoverlapping channels can be associated with distinct colours. On each vertex of the graph, the number of distinctly coloured edges allowed to be incident on it captures the limit on the number of channels that can be used simultaneously at each wireless node. The question of how many channels can be used simultaneously by a given network translates into the number of colours that can be used in a maximum edge colouring.

Devising an efficient algorithm for the maximum edge q-colouring problem is not an easy task. In [1], the problem is reported NP-hard for every $q \geq 2$. The authors further showed that the problem is hard to approximate within a factor of $(1 + \frac{1}{q})$ for every $q \geq 2$, assuming the *unique games conjecture* [2]. A simple 2-approximation algorithm for the maximum edge 2-colouring problem is reported in [15]. The same algorithm from [15] has an approximation ratio of 5/3 with the additional assumption that the graph has a perfect matching [2]. It is also known that the approximation ratio can be improved to 8/5 if the input graph is assumed to be triangle-free [7]. An almost tight analysis of the algorithm is known for the maximum edge q-colouring problem ($q \geq 3$) when the input graph satisfies certain degree constraints [6]. The $q = 2$ case is also known to be fixed-parameter tractable [19].

In spite of several negative theoretical results, the wireless network question continued drawing the attention of researchers due to its relevance in applications. There are several studies focusing on improving approximation under further assumptions on constraints that are meaningful in practical applications [23,30,33,34]. This motivates us to study the more general question on a graph class that captures the essence of wireless mesh networks. Typically, disk graphs and unit disk graphs are well-accepted abstract models for wireless networks. But they can capture more complex networks than what a real-life network looks like [30]. By definition, both unit disk graphs and disk graphs can have arbitrary size cliques. In a practical arrangement of a wireless mesh network, it is quite unlikely to place too many wireless routers in a small area. In other words, a real-life wireless mesh network can be expected to be fairly sparse and avoid large cliques. In this paper, we focus on a popular special case of sparse networks, those avoiding a fixed graph as a minor. In particular, this includes the graphs that can be made planar by deletion of a bounded number k of vertices (the *k-apex graphs*).

From a purely theoretical perspective, the graphs avoiding a fixed minor are interesting on their own merit. Famously, they admit the structural decomposition devised by Robertson and Seymour [28], but also have many interesting properties that can be shown directly, such as the existence of sublinear separators [3] and admitting layered decomposition into pieces of bounded weak diameter [22]. They have been also intensively studied from the algorithmic perspective, including the PTAS design. Several techniques for this purpose have been developed over the last few decades. The *bidimensionality technique* bounds the treewidth of the graph in terms of the size of the optimal solution and uses the balanced separators to obtain the approximation factor [10,16]. A completely different approach based on local search is known for unweighted problems [5,20]. Dvořák used thin systems of overlays [12] and a generalization of Baker's layering approach [4,13] to obtain PTASes for a wide class of optimization problems expressible in the first-order logic and its variations.

1.1 Our results

Our contribution is twofold. First, we show that the maximum edge q-colouring problem is NP-hard on 1-apex graphs. Our approach is similar in spirit to the approximation hardness reduction for the problem on general graphs [1].

Secondly, we show that there exists a PTAS for the maximum edge q-colouring problem for graphs avoiding a fixed minor. The result uses the *Baker game* approach devised in [13], avoiding the use of involved structural results. The technique was developed to strengthen and simplify the results of [9] giving PTASes for monotone optimization problems expressible in the first-order logic. Our work demonstrates the wider applicability of this technique to problems not falling into this framework.

2 Preliminaries

A graph H is a *minor* of a graph G if a graph isomorphic to H can be obtained from a subgraph of G by a series of edge contractions. We say that G is *H-minor-free* if G does not contain H as a minor. A graph is called *planar* if it can be drawn in the plane without crossings. A graph G is a *k-apex graph* if there exists a set $A \subseteq V(G)$ of size at most k such that $G - A$ is planar. The k-apex graphs are one of the standard examples of graphs avoiding a fixed minor; indeed, they are K_{k+5}-minor-free.

Given a function f assigning colours to edges of a graph G and a vertex $v \in V(G)$, we write $f(v)$ to denote the set $\{f(e) : e \text{ is adjacent to } v\}$, and $f(G) = \{f(e) : e \in E(G)\}$. Recall that f is an edge q-colouring of G if and only if $|f(v)| \leq q$ for every $v \in V(G)$, and the maximum edge q-chromatic number of G is

$$\overline{\chi}'_q(G) = \max\{|f(G)| : f \text{ is an edge } q\text{-colouring of } G\}.$$

A *matching* in a graph G is a set of edges of G where no two are incident with the same vertex. A matching M is *maximal* if it is not a proper subset of any other matching. Note that a maximal matching is not necessarily the largest possible. Let $|G|$ denote $|V(G)| + |E(G)|$. For all other definitions related to graphs not defined in this article, we refer readers to any standard graph theory textbook, such as [11].

3 PTAS for Minor-Free Graphs

Roughly speaking, we employ a divide-and-conquer approach to approximate $\overline{\chi}'_q(G)$, splitting G into vertex disjoint parts $G_1, \ldots, G_m$ in a suitable way, solving the problem for each part recursively, and combining the solutions. An issue that we need to overcome is that it may be impossible to compose the edge q-colourings, e.g., if an edge (v_1, v_2) joins distinct parts and disjoint sets of q colours are used on the neighbourhoods of v_1 and v_2 already. To overcome this issue, we reserve the colour 0 to be used to join the "boundary" vertices. This motivates the following definition.

For a set S of vertices of a graph G, an edge q-colouring f is *S-composable* if $|f(v) \setminus \{0\}| \leq q - 1$ for every $v \in S$. Let $\overline{\chi}'_q(G, S)$ denote the maximum number of non-zero colours that can be used by an S-composable edge q-colouring of G. Let us remark that G has an S-composable edge q-colouring using any non-negative number $k' \leq \overline{\chi}'_q(G, S)$ of non-zero colours, as all edges of any colour $c \neq 0$ can be recoloured to 0.

Observation 1 *For any graph G, we have $\overline{\chi}'_q(G) = \overline{\chi}'_q(G, \emptyset)$, and $\overline{\chi}'_q(G, S) \leq \overline{\chi}'_q(G)$ for any $S \subseteq V(G)$.*

We need the following approximation for $\overline{\chi}'_q(G, S)$ in terms of the size of a maximal matching, analogous to one for edge 2-colouring given in [15]. Let us remark that the S-composable edge q-colouring problem is easy to solve for $q = 1$, since we have to use colour 0 on all edges of each component intersecting S and we can use a distinct colour for all edges of any other component. Consequently, in all further claims, we assume $q \geq 2$.

Observation 2 *For any graph G, any $S \subseteq V(G)$, any maximal matching M in G, and any $q \geq 2$,*

$$|M| \leq \overline{\chi}'_q(G,S) \leq \overline{\chi}'_q(G) \leq 2q|M|.$$

Proof. We can assign to each edge of M a distinct positive colour and to all other edges (if any) the colour 0, obtaining an S-composable edge 2-colouring using $|M|$ non-zero colours. On the other hand, consider the set X of vertices incident with the edges of M. By the maximality of M, the set X is a vertex cover of G, i.e., each edge of G is incident with a vertex of X, and thus at most $q|X| = 2q|M|$ colours can be used by any edge q-colouring of G.

In particular, as we show next, the lower bound implies that the S-composable edge q-colouring problem is fixed-parameter tractable when parameterized by the value of the solution (a similar observation on the maximum edge 2-colouring is reported in [19]).

Observation 3 *There exists an algorithm that, given a graph G, a set $S \subseteq V(G)$, and integers $q \geq 2$ and s, in time $O_{q,s}(|G|)$ returns an S-composable edge q-colouring of G using at least $\min(\overline{\chi}'_q(G,S), s)$ colours.*

Proof. We can in linear time find a maximal matching M in G. If $|M| \geq s$, we return the colouring that gives each edge of M a distinct non-zero colour and all other edges colour 0. Otherwise, the set X of vertices incident with M is a vertex cover of G of size at most $2s-2$, and thus G has treewidth at most $2s-2$. Note also that for any s', there exists a formula $\varphi_{s',q}$ in monadic second-order logic such that $G, S, E_0, \ldots, E_{s'} \models \varphi_{s',q}$ if and only if $E_0, \ldots, E_{s'}$ is a partition of the edges of G with all parts except possibly for E_0 non-empty such that the function f defined by letting $f(e) = i$ for each $i \in \{0, \ldots, s'\}$ and $e \in E_i$ is an S-composable edge q-colouring of G. Therefore, we can find an S-composable edge q-colouring of G with the maximum number $s' \leq s$ of non-zero colours using Courcelle's theorem [8] in time $O_{q,s}(|G|)$.

A *layering* of a graph G is a function $\lambda \colon V(G) \to \mathbb{Z}^+$ such that $|\lambda(u) - \lambda(v)| \leq 1$ for every edge $(u,v) \in E(G)$. In other words, the graph is partitioned into layers $\lambda^{-1}(i)$ for $i \in \mathbb{Z}^+$ such that edges of G only appear within the layers and between the consecutive layers. Baker [4] gave a number of PTASes for planar graphs based on the fact that in a layering of a connected planar graph according to the distance from a fixed vertex, the union of a constant number of consecutive layers induces a subgraph of bounded treewidth. This is not the case for graphs avoiding a fixed minor in general, however, a weaker statement expressed in terms of Baker game holds. We are going to describe that result in more detail in the following subsection. Here, let us state the key observation that makes layering useful for approximating the edge q-chromatic number.

For integers $r \geq 2$ and m such that $0 \leq m \leq r-1$, the (λ, r, m)-stratification of a graph G is the pair (G', S') such that

- G' is obtained from G by deleting all edges uv such that $\lambda(u) \equiv m \pmod r$ and $\lambda(v) \equiv m+1 \pmod r$, and
- S' is the set of vertices of G incident with the edges of $E(G) \setminus E(G')$.

Lemma 1. *Let G be a graph, S a subset of its vertices, and $q, r \geq 2$ integers. Let λ be a layering of G. For $m \in \{0, \ldots, r-1\}$, let (G_m, S_m) be the (λ, r, m)-stratification of G.*

- *$\overline{\chi}'_q(G_m, S \cup S_m) \leq \overline{\chi}'_q(G, S)$ for every $m \in \{0, \ldots, r-1\}$.*
- *There exists $m \in \{0, \ldots, r-1\}$ such that $\overline{\chi}'_q(G_m, S \cup S_m) \geq \left(1 - \frac{6q}{r}\right)\overline{\chi}'_q(G, S)$.*

Proof. Given an $(S \cup S_m)$-composable edge q-colouring of G_m, we can assign the colour 0 to all edges of $E(G) \setminus E(G_m)$ and obtain an S-composable edge q-colouring of G using the same number of non-zero colours, which implies that $\overline{\chi}'_q(G_m, S \cup S_m) \leq \overline{\chi}'_q(G, S)$.

Conversely, consider an S-composable edge q-colouring f of G using $k = \overline{\chi}'_q(G, S)$ non-zero colours. For $m \in \{0, \ldots, r-1\}$, let B_m be the bipartite graph with vertex set S_m and edge set $E(G) \setminus E(G_m)$ and let M_m be a maximal matching in B_m. Let $\mathcal{P}$ be a partition of the set $\{0, \ldots, r-1\}$ into at most three disjoint parts such that none of the parts contains two integers that are consecutive modulo r. For each $P \in \mathcal{P}$, let $M_P = \bigcup_{m \in P} M_m$, and observe that M_P is a matching in G. By Observation 2, it follows that $k \geq |M_P|$, and thus

$$3k \geq |P|k \geq \sum_{P \in \mathcal{P}} |M_P| = \sum_{m=0}^{r-1} |M_m|.$$

Hence, we can fix $m \in \{0, \ldots, r-1\}$ such that $|M_m| \leq \frac{3}{r}k$. By Observation 2, any edge q-colouring of B_m, and in particular the restriction of f to the edges of B_m, uses at most $2q|M_m| \leq \frac{6q}{r}k$ distinct colours.

Let f' be the edge q-colouring of G obtained from f by recolouring all edges whose colour appears on the edges of B_m to colour 0. Clearly f' uses at least $\left(1 - \frac{6q}{r}\right)k$ non-zero colours. Moreover, each vertex $v \in S_m$ is now incident with an edge of colour 0, and thus $|f'(v) \setminus \{0\}| \leq q-1$. Therefore, the restriction of f' to $E(G_m)$ is an $(S \cup S_m)$-composable edge q-colouring, implying that

$$\overline{\chi}'_q(G_m, S \cup S_m) \geq \left(1 - \frac{6q}{r}\right)k = \left(1 - \frac{6q}{r}\right)\overline{\chi}'_q(G, S).$$

Hence, if $r \gg q$, then a good approximation of $\overline{\chi}'_q(G_m, S \cup S_m)$ for all $m \in \{0, \ldots, r-1\}$ gives a good approximation for $\overline{\chi}'_q(G, S)$. We will also need a similar observation for vertex deletion; here we only get an additive approximation in general, but as long as the edge q-chromatic number is large enough, this suffices (and if it is not, we can determine it exactly using Observation 3).

Lemma 2. *Let G be a graph, S a set of its vertices, and v a vertex of G. Let $S' = (S \setminus \{v\}) \cup N(v)$. For any integer $q \geq 2$, we have*

$$\overline{\chi}'_q(G, S) \geq \overline{\chi}'_q(G - v, S') \geq \overline{\chi}'_q(G, S) - q,$$

and in particular if $\varepsilon > 0$ and $\overline{\chi}'_q(G, S) \geq q/\varepsilon$, then

$$\overline{\chi}'_q(G - v, S') \geq (1 - \varepsilon)\overline{\chi}'_q(G, S).$$

Proof. Any S'-composable edge q-colouring of $G-v$ extends to an S-composable edge q-colouring of G by giving all edges incident on v colour 0, implying that $\overline{\chi}'_q(G, S) \geq \overline{\chi}'_q(G - v, S')$. Conversely, any S-composable edge q-colouring of G can be turned into an S'-composable edge q-colouring of $G - v$ by recolouring all edges whose colour appears on the neighbourhood of v to 0 and restricting it to the edges of $G - v$. This loses at most q non-zero colours (those appearing on the neighborhood of v), and thus $\overline{\chi}'_q(G - v, S') \geq \overline{\chi}'_q(G, S) - q$. □

3.1 Baker game

For an infinite sequence $\mathtt{r} = r_1, r_2, \ldots$ and an integer $s \geq 0$, let tail($\mathtt{r}$) denote the sequence $r_2, r_3, \ldots$ and let head($\mathtt{r}$) $= r_1$. *Baker game* is played by two players Destroyer and Preserver on a pair $(G, \mathtt{r})$, where G is a graph and $\mathtt{r}$ is a sequence of positive integers. The game stops when $V(G) = \emptyset$, and Destroyer's objective is to minimise the number of rounds required to make the graph empty. In each round of the game, either

- Destroyer chooses a vertex $v \in V(G)$, Preserver does nothing and the game moves to the state $(G \setminus \{v\}, \text{tail}(\mathtt{r}))$, or
- Destroyer selects a layering λ of G, Preserver selects an interval I of head($\mathtt{r}$) consecutive integers and the game moves to the state $(G[\lambda^{-1}(I)], \text{tail}(\mathtt{r}))$. In other words, Preserver selects head($\mathtt{r}$) consecutive layers and the rest of the graph is deleted.

Destroyer *wins* in k rounds on the state $(G, \mathtt{r})$ if regardless of Preserver's strategy, the game stops after at most k rounds. As we mentioned earlier Destroyer's objective is to minimise the number of rounds of this game and it is known that they will succeed if the game is played on a graph that forbids a fixed minor (the upper bound on the number of rounds depends only on the sequence $\mathtt{r}$ and the forbidden minor, not on G).

Theorem 1 (Dvořák [13]). *For every graph F and every sequence $\mathtt{r} = r_1, r_2, \ldots$ of positive integers, there exists a positive integer k such that for every graph G avoiding F as a minor,* Destroyer *wins Baker game from the state $(G, \mathtt{r})$ in at most k rounds. Moreover, letting $n = |V(G)|$, there exists an algorithm that preprocesses G in time $O_F(n^2)$ and then in each round determines a move for* Destroyer *(leading to winning in at most k rounds in total) in time $O_{F,\mathtt{r}}(n)$.*

Let us now give the algorithm for approximating the edge q-chromatic number on graphs for which we can quickly win Baker game.

Lemma 3. *There exists an algorithm that, given*

- *a graph G, a set $S \subseteq V(G)$, an integer $q \geq 2$, and*
- *a sequence $\mathtt{r} = r_1, r_2, \ldots$ of positive integers such that* Destroyer *wins Baker game from the state $(G, \mathtt{r})$ in at most k rounds, and in each state that arises in the game is able to determine the move that achieves this in time T,*

returns an S-composable edge q-colouring of G using at least $\Big(\prod_{i=1}^{k}\big(1-\frac{6q}{r_i}\big)\Big)\cdot \overline{\chi}'_q(G,S)$ non-zero colours, in time $O_{\mathtt{r},k,q}(|G|T)$.

Proof. First, we run the algorithm from Observation 3 with $s = \lceil r_1/3 \rceil$. If the obtained colouring uses less than s non-zero colours, it is optimal and we return it. Otherwise, we know that $\overline{\chi}'_q(G,S) \geq s$. In particular, $E(G) \neq \emptyset$, and thus Destroyer have not won the game yet.

Let $R = \Big(\prod_{i=2}^{k}\big(1-\frac{6q}{r_i}\big)\Big)$. Let us now consider two cases depending on Destroyer's move from the state $(G, \mathtt{r})$.

- Suppose that Destroyer decides to delete a vertex $v \in V(G)$. We apply the algorithm recursively for the graph $G - v$, set $S' = (S \setminus \{v\}) \cup N(v)$, and the sequence tail$(\mathtt{r})$, obtaining an S'-composable edge q-colouring f of $G - v$ using at least $R \cdot \overline{\chi}'_q(G - v, S')$ non-zero colours. By Lemma 2 with $\varepsilon = \frac{q}{s}$, we conclude that f uses at least
$$R \cdot \overline{\chi}'_q(G - v, S') \geq R(1-\varepsilon)\overline{\chi}'_q(G,S) \geq R\Big(1 - \frac{6q}{r_1}\Big)\overline{\chi}'_q(G,S)$$
non-zero colours. We turn f into an S-composable edge q-colouring of G by giving all edges incident on v colour 0 and return it.
- Suppose that Destroyer chooses a layering λ. We now recurse into several subgraphs, each corresponding to a valid move of Preserver. For each $m \in \{0, \ldots, r_1 - 1\}$, let (G_m, S_m) be the (λ, r_1, m)-stratification of G_m. Note that G_m is divided into parts $G_{m,1}, \ldots, G_{m,t_m}$, each contained in the union of r_1 consecutive layers of λ. For each $m \in \{0, \ldots, r_1 - 1\}$ and each $i \in \{1, \ldots, t_m\}$, we apply the algorithm recursively for the graph $G_{m,i}$, set $S_{m,i} = (S_m \cup S) \cap V(G_{m,i})$, and the sequence tail$(\mathtt{r})$, obtaining an $S_{m,i}$-composable edge q-colouring $f_{m,i}$ of $G_{m,i}$ using at least $R \cdot \overline{\chi}'_q(G_{m,i}, S_{m,i})$ non-zero colours. Let f_m be the union of the colourings $f_{m,i}$ for $i \in \{1, \ldots, t_m\}$ and observe that f_m is an $(S \cup S_m)$-composable edge q-colouring of G_m using at least $R \cdot \overline{\chi}'_q(G_m, S \cup S_m)$ non-zero colours. We choose $m \in \{0, \ldots, r_1 - 1\}$ such that f_m uses the largest number of non-zero colours, extend it to an S-composable edge q-colouring of G by giving all edges of $E(G) \setminus E(G_m)$ colour 0, and return it. By Lemma 1, the colouring uses at least
$$R \cdot \overline{\chi}'_q(G_m, S \cup S_m) \geq R\Big(1 - \frac{6q}{r_1}\Big)\overline{\chi}'_q(G,S)$$
non-zero colours, as required.

For the time complexity, note that each vertex and edge of G belongs to at most $\prod_{i=1}^{d} r_i$ subgraphs processed at depth d of the recursion, and since the depth of the recursion is bounded by k, the sum of the sizes of the processed subgraphs is $O_{\mathtt{r},k,q}(|G|)$. Excluding the recursion and time needed to select Destroyer's moves, the actions described above can be performed in linear time. Consequently, the total time complexity is $O_{\mathtt{r},k,q}(|G|T)$.

Our main result is then just a simple combination of this lemma with Theorem 1.

Theorem 2. *There exists an algorithm that given an F-minor-free graph G and integers $q, p \geq 2$, returns in time $O_{F,p,q}(|G|^2)$ an edge q-colouring of G using at least $(1 - 1/p)\overline{\chi}'_q(G)$ colours.*

Proof. Let $\mathtt{r}$ be the infinite sequence such that $r_i = 10pqi^2$ for each positive integer i, and let k be the number of rounds in which Destroyer wins Baker game from the state $(G', \mathtt{r})$ for any F-minor-free graph G', using the strategy given by Theorem 1. Note that

$$R = \prod_{i=1}^{k}\left(1 - \tfrac{6q}{r_i}\right) \geq 1 - \sum_{i=1}^{\infty} \frac{6q}{r_i}$$
$$= 1 - \frac{3}{5p}\sum_{i=1}^{\infty}\frac{1}{i^2} = 1 - \frac{3}{5p}\cdot\frac{\pi^2}{6} \geq 1 - \frac{1}{p}.$$

Let $n = |G|$. After running the preprocessing algorithm from Theorem 1, we apply the algorithm from Lemma 3 with $S = \emptyset$ and $T = O_{F,\mathtt{r}}(n) = O_{F,p,q}(n)$, obtaining an edge q-colouring of G using at least $R \cdot \overline{\chi}'_q(G, \emptyset) = R \cdot \overline{\chi}'_q(G) \geq (1 - 1/p)\overline{\chi}'_q(G)$ colours, in time $O_{F,p,q}(n^2)$. □

4 Hardness on 1-apex graphs

In this section, we study the complexity of the maximum edge 2-colouring problem on 1-apex graphs. We present a reduction from Planar ($\leq$3, 3)-SAT which is known to be NP-hard [32].

The *incidence graph* $G(\varphi)$ of a Boolean formula φ in conjunctive normal form is the bipartite graph whose vertices are the variables appearing in φ and the clauses of φ, and each variable is adjacent exactly to the clauses in which it appears. A Boolean formula φ in conjunctive normal form is called Planar ($\leq$3, 3)-SAT if

- each clause of φ contains at most three distinct literals,
- each variable of φ appears in exactly three clauses,
- the incidence graph $G(\varphi)$ is planar.

In Planar $(\leq 3, 3)$-SAT problem, we ask whether such a formula φ has a satisfying assignment.

We follow the strategy used in [1], using an intermediate *maximum edge* $1, 2$-*colouring* problem. The instance of this problem consists of a graph G, a function $g\colon V(G) \to \{1, 2\}$, and a number t. An *edge* g-*colouring* of G is an edge colouring f such that $|f(v)| \leq g(v)$ for each $v \in V(G)$. The objective is to decide whether there exists an edge g-colouring of G using at least t distinct colours. We show the maximum edge $\{1, 2\}$-colouring problem is NP-hard on 1-apex graphs by establishing a reduction from Planar $(\leq 3, 3)$-SAT problem. We then use this result to show that the maximum edge q-colouring problem on planar graphs is NP-hard when $q \geq 2$. Let us start by establishing the intermediate result.

Lemma 4. *The maximum edge* $\{1, 2\}$*-colouring problem is NP-hard even when restricted on the class of* 1*-apex graphs.*

Proof. Consider a given Planar $(\leq 3, 3)$-SAT formula φ with m clauses and n variables and a plane drawing of its incidence graph $G(\varphi)$. Let the clauses of φ be $c_1, \ldots, c_m$ and the variables $x_1, \ldots, x_n$; we use the same symbols for the corresponding vertices of $G(\varphi)$.

Let H be a graph obtained from $G(\varphi)$ as follows. For all $j \in \{1, 2, \ldots, n\}$, if the clauses in which x_j appears are $c_{\ell_{j,1}}$, $c_{\ell_{j,2}}$, and $c_{\ell_{j,3}}$, split x_j to three vertices $x_{j,1}$, $x_{j,2}$, and $x_{j,3}$, where $x_{j,a}$ is adjacent to $c_{\ell_{j,a}}$ for $a \in \{1, 2, 3\}$. For $1 \leq a < b \leq 3$, add a vertex $n_{j,a,b}$ and if x_j appears positively in $c_{\ell_{j,a}}$ and negatively in $c_{\ell_{j,b}}$ or vice versa, make it adjacent to $x_{j,a}$ and $x_{j,b}$ (otherwise leave it as an isolated vertex). Finally, we add a new vertex u adjacent to c_i for $i \in \{1, \ldots, m\}$ and to $n_{j,a,b}$ for $j \in \{1, \ldots, n\}$ and $1 \leq a < b \leq 3$. Clearly, H is a 1-apex graph, since $H - u$ is planar.

Let us define the function $g\colon V(H) \to \{1, 2\}$ as follows:

- $g(u) = 1$,
- $g(c_i) = 2$ for all $i \in \{1, 2, \ldots, m\}$,
- $g(x_{j,a}) = 1$ for all $j \in \{1, 2, \ldots, n\}$ and $a \in \{1, 2, 3\}$, and
- $g(n_{j,a,b}) = 2$ for all $j \in \{1, 2, \ldots, n\}$ and $1 \leq a < b \leq 3$.

First, we show if there exists a satisfying assignment for the formula φ, then H has an edge g-colouring using $n+1$ colours. For $i \in \{1, \ldots, n\}$, choose a vertex $x_{j,a}$ adjacent to c_i such that the (positive or negative) literal of c_i containing the variable x_j is true in the assignment, and give colour i to the edge $(c_i, x_{j,a})$ and all other edges incident on $x_{j,a}$ (if any). All other edges receive colour 0.

Clearly, u is only incident with edges of colour 0, for each $j \in \{1, \ldots, n\}$ and $a \in \{1, \ldots, 3\}$ all edges incident on $x_{j,a}$ have the same colour, and for each $i \in \{1, \ldots, m\}$, the edges incident on c_i have colours 0 and i. Finally, consider a vertex $n_{j,a,b}$ for some $j \in \{1, \ldots, n\}$ and $1 \leq a < b \leq 3$ adjacent to $x_{j,a}$ and $x_{j,b}$. By the construction of H, the variable x_j appears positively in $c_{\ell_{j,a}}$ and negatively in $c_{\ell_{j,b}}$ or vice versa, and thus at most one of the corresponding literals is true in the assignment. Hence, $n_{j,a,b}$ is incident with edges of colour 0 and of at most one of the colours $\ell_{j,a}$ and $\ell_{j,b}$.

Conversely, suppose that there exists an edge g-colouring f of H using at least $m+1$ distinct colours, and let us argue that there exists a satisfying assignment for φ. Since $g(u) = 1$, we can without loss of generality assume that each edge incident with u has colour 0. If a colour $c \neq 0$ is used to colour the edge $(n_{j,a,b}, x_{j,k})$ for some $j \in \{1, \ldots, n\}$, $1 \leq a < b \leq 3$, and $k \in \{a, b\}$, then since $g(x_{j,k}) = 1$, this colour is also used on the edge $(x_{j,k}, c_{\ell_{j,k}})$. Hence, every non-zero colour appears on an edge incident with a clause. Since each clause is also joined to u by an edge of colour 0, it can be only incident with edges of one other colour. Since f uses at least $m + 1$ colours, we can without loss of generality assume that for $i \in \{1, \ldots, m\}$, there exists an edge $(c_i, x_{j,a})$ for some $j \in \{1, \ldots, n\}$ and $a \in \{1, \ldots, 3\}$ of colour i. Assign to x_j the truth value that makes the literal of c_i in which it appears true.

We only need to argue that this rule does not cause us to assign to x_j both values true and false. If that were the case, then there would exist $1 \leq a < b \leq 3$ such that the variable x_j appears positively in clause $c_{\ell_{j,a}}$ and negatively in clause $c_{\ell_{j,b}}$ or vice versa, the edge corresponding to the variable $x_{j,a}$ has colour $\ell_{j,a}$ and the edge corresponding to the variable $x_{j,b}$ has colour $\ell_{j,b}$. However, since $g(x_{j,a}) = g(x_{j,b}) = 1$, this would imply that $n_{j,a,b}$ is incident with the edge $(n_{j,a,b}, x_{j,a})$ of colour $\ell_{j,a}$, the edge $(n_{j,a,b}, x_{j,b})$ of colour $\ell_{j,b}$, and the edge $(n_{j,a,b}, u)$ of colour 0, which is a contradiction.

Therefore, we described how to transform in polynomial time a Planar $(\leq 3, 3)$-SAT instance φ to an equivalent instance H, g, $t = m+1$ of the maximum edge $\{1, 2\}$-colouring problem. □

Now we are ready to prove the main theorem of this section. The proof strategy is similar to the APX-hardness proof in [1]. We include the details for completeness.

Theorem 3. *For an arbitrary integer $q \geq 2$ the maximum edge q-colouring problem is NP-hard even when the input instance is restricted to 1-apex graphs.*

Proof. We construct a reduction from the maximum edge $\{1, 2\}$-colouring problem on 1-apex graphs. Let G, g, t be an instance of this problem, and let $n = |V(G)|$ and $r = |\{v \in V(G) \colon g(v) = 1\}|$. We create a graph G' from G by adding for each vertex $v \in V(G)$ exactly $q - g(v)$ pendant vertices adjacent to v. Clearly, G' is an 1-apex graph. We show that G has an edge g-colouring using at least t distinct colours if and only if G' has an edge q-colouring using at least $t + r + (q - 2)n$ colours.

In one direction, given an edge g-colouring of G using at least t colours, we colour each of the added pendant edges using a new colour, obtaining an edge q-colouring of G using at least $t + r + (q - 2)n$ colours.

Conversely, let f be an edge q-colouring of G' using at least $t + r + (q - 2)n$ colours. Process the vertices $v \in V(G)$ one by one, performing for each of them the following operation: For each added pendant vertex u adjacent to v in order, let c' be the colour of the edge (u, v), delete u, and if v is incident with an edge e of colour $c \neq c'$, then recolour all remaining edges of colour c' to c. Note that the number of eliminated colours is bounded by the number $r + (q - 2)n$ of pendant vertices, and thus the resulting colouring still uses at least t colours. Moreover, at each vertex $v \in V(G)$, we either end up with all edges incident on v having the same colour or we eliminated one colour from the neighbourhood of v for each adjacent pendant vertex; in the latter case, since $|f(v)| \leq q$ and v is adjacent to $q - g(v)$ pendant vertices, at most $g(v)$ colours remain on the edges incident on v. Hence, we indeed obtain an edge g-colouring of G using at least t colours. □

5 Future directions

We conclude with some possible directions for future research. The maximum edge 2-colouring problem on 1-apex graphs is NP-hard. But the complexity of the problem is unknown when the input is restricted to planar graphs. We consider this an interesting question left unanswered. The best-known approximation ratio is known to be 2, without any restriction on the input instances. Whereas, a lower bound of $(\frac{1+q}{q})$, for $q \geq 2$ is known assuming unique games conjecture. There are not many new results reported in the last decade that bridge this gap. We think, even a $(2-\varepsilon)$ algorithm, for any $\varepsilon > 0$, will be a huge progress towards that direction. The Baker game technique can yield PTASes for monotone optimization problems beyond problems expressible in the first-order logic. Clearly, the technique can't be extended to the entire class of problems expressible in the monadic second-order logic. It will be interesting to characterise the problems expressible in the monadic second-order logic where the Baker game yield PTASes.

Acknowledgement. The second author likes to thank Benjamin Moore, Jatin Batra, Sandip Banerjee and Siddharth Gupta for helpful discussions on this project. He also likes to thank the organisers of Homonolo for providing a nice and stimulating research environment.

References

1. Adamaszek, A., Popa, A.: Approximation and hardness results for the maximum edge q-coloring problem. In: Cheong, O., Chwa, K.-Y., Park, K. (eds.) ISAAC 2010, Part II. LNCS, vol. 6507, pp. 132–143. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17514-5_12
2. Adamaszek, A., Popa, A.: Approximation and hardness results for the maximum edge Q-coloring problem. J. Discret. Algorithms **38-41**, 1–8 (2016). https://doi.org/10.1016/j.jda.2016.09.003
3. Alon, N., Seymour, P.D., Thomas, R.: A separator theorem for graphs with an excluded minor and its applications. In: Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, 13–17 May 1990, Baltimore, Maryland, USA, pp. 293–299. ACM (1990). https://doi.org/10.1145/100216.100254
4. Baker, B.S.: Approximation algorithms for NP-complete problems on planar graphs. J. ACM **41**(1), 153–180 (1994). https://doi.org/10.1145/174644.174650
5. Cabello, S., Gajser, D.: Simple PTAS's for families of graphs excluding a minor. Discret. Appl. Math. **189**, 41–48 (2015). https://doi.org/10.1016/j.dam.2015.03.004

6. Chandran, L.S., Hashim, T., Jacob, D., Mathew, R., Rajendraprasad, D., Singh, N.: New bounds on the anti-Ramsey numbers of star graphs. CoRR abs/1810.00624 (2018). https://arxiv.org/abs/1810.00624
7. Chandran, L.S., Lahiri, A., Singh, N.: Improved approximation for maximum edge colouring problem. Discrete Appl. Math. **319**, 42–52 (2022). https://doi.org/10.1016/j.dam.2021.05.017
8. Courcelle, B.: The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. Inf. Comput. **85**(1), 12–75 (1990). https://doi.org/10.1016/0890-5401(90)90043-H
9. Dawar, A., Grohe, M., Kreutzer, S., Schweikardt, N.: Approximation schemes for first-order definable optimisation problems. In: 21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12–15 August 2006, Seattle, WA, USA, Proceedings, pp. 411–420. IEEE Computer Society (2006). https://doi.org/10.1109/LICS.2006.13
10. Demaine, E.D., Hajiaghayi, M.T.: Bidimensionality: new connections between FPT algorithms and PTASs. In: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, 23–25 January 2005, pp. 590–601. SIAM (2005). https://dl.acm.org/citation.cfm?id=1070432.1070514
11. Diestel, R.: Graph Theory. Graduate Texts in Mathematics, vol. 173, 4th edn. Springer, Heidelberg (2012)
12. Dvorák, Z.: Thin graph classes and polynomial-time approximation schemes. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, 7–10 January 2018, pp. 1685–1701. SIAM (2018). https://doi.org/10.1137/1.9781611975031.110
13. Dvorák, Z.: Baker game and polynomial-time approximation schemes. In: Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, 5–8 January 2020, pp. 2227–2240. SIAM (2020). https://doi.org/10.1137/1.9781611975994.137
14. Erdös, P., Simonovits, M., Sós, V.T.: Anti-Ramsey theorems. Infinite Finite Sets (Colloquium, Keszthely, 1973; dedicated to P. Erdös on his 60th birthday) **10**(II), 633–643 (1975)
15. Feng, W., Zhang, L., Wang, H.: Approximation algorithm for maximum edge coloring. Theor. Comput. Sci. **410**(11), 1022–1029 (2009). https://doi.org/10.1016/j.tcs.2008.10.035
16. Fomin, F.V., Lokshtanov, D., Raman, V., Saurabh, S.: Bidimensionality and EPTAS. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, 23–25 January 2011, pp. 748–759. SIAM (2011). https://doi.org/10.1137/1.9781611973082.59
17. Fujita, S., Magnant, C., Ozeki, K.: Rainbow generalizations of Ramsey theory: a survey. Graphs Combin. **26**(1), 1–30 (2010). https://doi.org/10.1007/s00373-010-0891-3
18. Gorgol, I., Lazuka, E.: Rainbow numbers for small stars with one edge added. Discuss. Math. Graph Theory **30**(4), 555–562 (2010). https://doi.org/10.7151/dmgt.1513
19. Goyal, P., Kamat, V., Misra, N.: On the parameterized complexity of the maximum edge 2-coloring problem. In: Mathematical Foundations of Computer Science 2013–38th International Symposium, MFCS 2013, Klosterneuburg, Austria, 26–30 August 2013, pp. 492–503 (2013). https://doi.org/10.1007/978-3-642-40313-2_44

20. Har-Peled, S., Quanrud, K.: Approximation algorithms for polynomial-expansion and low-density graphs. SIAM J. Comput. **46**(6), 1712–1744 (2017). https://doi.org/10.1137/16M1079336
21. Jiang, T.: Edge-colorings with no large polychromatic stars. Graphs Combin. **18**(2), 303–308 (2002). https://doi.org/10.1007/s003730200022
22. Klein, P.N., Plotkin, S.A., Rao, S.: Excluded minors, network decomposition, and multicommodity flow. In: Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, 16–18 May 1993, San Diego, CA, USA, pp. 682–690. ACM (1993). https://doi.org/10.1145/167088.167261
23. Kodialam, M.S., Nandagopal, T.: Characterizing the capacity region in multi-radio multi-channel wireless mesh networks. In: Proceedings of the 11th Annual International Conference on Mobile Computing and Networking, MOBICOM 2005, Cologne, Germany, 28 August–2 September 2005, pp. 73–87. ACM (2005), https://doi.org/10.1145/1080829.1080837
24. Manoussakis, Y., Spyratos, M., Tuza, Z., Voigt, M.: Minimal colorings for properly colored subgraphs. Graphs Combin. **12**(1), 345–360 (1996). https://doi.org/10.1007/BF01858468
25. Montellano-Ballesteros, J.J.: On totally multicolored stars. J. Graph Theory **51**(3), 225–243 (2006). https://doi.org/10.1002/jgt.20140
26. Montellano-Ballesteros, J.J., Neumann-Lara, V.: An anti-Ramsey theorem. Combinatorica **22**(3), 445–449 (2002). https://doi.org/10.1007/s004930200023
27. Raniwala, A., Chiueh, T.: Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In: INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, 13–17 March 2005, Miami, FL, USA, pp. 2223–2234 (2005). https://doi.org/10.1109/INFCOM.2005.1498497
28. Robertson, N., Seymour, P.D.: Graph minors. XVI. Excluding a non-planar graph. J. Combin. Theory Ser. B **89**(1), 43–76 (2003). https://doi.org/10.1016/S0095-8956(03)00042-X
29. Schiermeyer, I.: Rainbow numbers for matchings and complete graphs. Discret. Math. **286**(1–2), 157–162 (2004). https://doi.org/10.1016/j.disc.2003.11.057
30. Sen, A., Murthy, S., Ganguly, S., Bhatnagar, S.: An interference-aware channel assignment scheme for wireless mesh networks. In: Proceedings of IEEE International Conference on Communications, ICC 2007, Glasgow, Scotland, UK, 24–28 June 2007, pp. 3471–3476. IEEE (2007). https://doi.org/10.1109/ICC.2007.574
31. Simonovits, M., Sós, V.: On restricted colourings of K_n. Combinatorica **4**(1), 101–110 (1984). https://doi.org/10.1007/BF02579162
32. Tovey, C.A.: A simplified NP-complete satisfiability problem. Discret. Appl. Math. **8**(1), 85–89 (1984). https://doi.org/10.1016/0166-218X(84)90081-7
33. Wan, P., Al-dhelaan, F., Jia, X., Wang, B., Xing, G.: Maximizing network capacity of MPR-capable wireless networks. In: 2015 IEEE Conference on Computer Communications, INFOCOM 2015, Kowloon, Hong Kong, 26 April–1 May 2015, pp. 1805–1813. IEEE (2015). https://doi.org/10.1109/INFOCOM.2015.7218562
34. Wan, P., Cheng, Y., Wang, Z., Yao, F.F.: Multiflows in multi-channel multi-radio multihop wireless networks. In: INFOCOM 2011. 30th IEEE International Conference on Computer Communications, 10–15 April 2011, Shanghai, China, pp. 846–854. IEEE (2011). https://doi.org/10.1109/INFCOM.2011.5935308

Bounds on Functionality and Symmetric Difference – Two Intriguing Graph Parameters

Pavel Dvořák[1,2](✉), Lukáš Folwarczný[1,3], Michal Opler[4], Pavel Pudlák[3], Robert Šámal[1], and Tung Anh Vu[1]

[1] Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic
{koblich,samal}@iuuk.mff.cuni.cz, tung@kam.mff.cuni.cz
[2] Tata Institute of Fundamental Research, Mumbai, India
[3] Institute of Mathematics, Czech Academy of Sciences, Prague, Czech Republic
{folwarczny,pudlak}@math.cas.cz
[4] Faculty of Information Technology, Czech Technical University, Prague, Czech Republic
michal.opler@fit.cvut.cz

Abstract. [Alecu et al.: Graph functionality, JCTB2021] define functionality, a graph parameter that generalizes graph degeneracy. They research the relation of functionality to many other graph parameters (tree-width, clique-width, VC-dimension, etc.). Extending their research, we prove a logarithmic lower bound for functionality of random graph $G(n, p)$ for large range of p. Previously known graphs have functionality logarithmic in number of vertices. We show that for every graph G on n vertices we have $\mathrm{fun}(G) \le O(\sqrt{n \log n})$ and we give a nearly matching $\Omega(\sqrt{n})$-lower bound provided by projective planes.

Further, we study a related graph parameter *symmetric difference*, the minimum of $|N(u)\Delta N(v)|$ over all pairs of vertices of the "worst possible" induced subgraph. It was observed by Alecu et al. that $\mathrm{fun}(G) \le \mathrm{sd}(G)+1$ for every graph G. We compare fun and sd for the class INT of interval graphs and CA of circular-arc graphs. We let INT_n denote the n-vertex interval graphs, similarly for CA_n.

Alecu et al. ask, whether fun(INT) is bounded. Dallard et al. answer this positively in a recent preprint. On the other hand, we show that $\Omega(\sqrt[4]{n}) \le \mathrm{sd}(\mathrm{INT}_n) \le O(\sqrt[3]{n})$. For the related class CA we show that $\mathrm{sd}(\mathrm{CA}_n) = \Theta(\sqrt{n})$.

We propose a follow-up question: is fun(CA) bounded?

P. Dvořák—Supported by Czech Science Foundation GAČR grant #22-14872O.
L. Folwarczný—Supported by Czech Science Foundation GAČR grant 19-27871X.
M. Opler—Supported by Czech Science Foundation GAČR grant 22-19557S.
P. Pudlák—Supported by Czech Science Foundation GAČR grant 19-27871X.
R. Šámal—Partially supported by grant 22-17398S of the Czech Science Foundation. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 810115).
T. A. Vu—Supported by Czech Science Foundation GAČR grant 22-22997S.

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 305–318, 2023.
https://doi.org/10.1007/978-3-031-43380-1_22

Keywords: Functionality · Finite Projective Plane · Symmetric Difference · Intersection Graphs

1 Introduction

Let $G = (V, E)$ be a graph and $v \in V$ be a vertex. The set of neighbors of v in G is denoted by $N_G(v)$, we omit the subscript if the graph is clear from the context. An adjacency matrix A_G of G is 0-1 matrix such that its rows and columns are indexed by vertices of G and $A[u, v] = 1$ if and only if u and v are connected by an edge. Now, we define the functionality and symmetric difference of a graph – two principle notion of this papers – as introduced by Alecu et al. [1], and implicitly also by Atminas et al. [3].

A vertex v of a graph $G = (V, E)$ is a *function* of vertices $u_1, \dots, u_k \in V$ (different from v) if there exists a boolean function f of k variables such that for any vertex $w \in V \setminus \{v, u_1, \dots, u_k\}$ it holds that $A[v, w] = f\big(A[v, u_1], \dots, A[v, u_k]\big)$. Informally, we can determine if v and w are connected from the adjacencies of v with the u_i's. The *functionality* $\mathrm{fun}_G(v)$ of a vertex v in G is the minimum k such that v is a function of k vertices of G. We drop the subscript and write just $\mathrm{fun}(v)$ if the graph G is clear from the context. Then, the *functionality* $\mathrm{fun}(G)$ of a graph G is defined as

$$\mathrm{fun}(G) = \max_{H \subseteq G} \min_{v \in V(H)} \mathrm{fun}_H(v),$$

where the maximum is taken over all induced subgraphs H of G.

It is observed in [3] that if $\mathrm{fun}(G) \leq k$ then we can encode G using $n(2^k + (k+1)\log n)$ bits, where n is the number of vertices of G. Thus, if every graph G in some graph class $\mathcal{G}$ has bounded functionality then $\mathcal{G}$ contains at most $2^{O(n \log n)}$ graphs on n vertices. Such classes are said to be of factorial growth [4] and include diverse classes of practical importance (interval graphs, line graphs, forests, planar graphs, more generally all proper minor-closed classes). Thus, Alecu et al. [1] introduce functionality as a tool to study graph classes of factorial growth, and the related Implicit graph conjecture (although this conjecture was recently disproved [6]). This was also our original motivation. Moreover, functionality is a natural generalization of the graph degeneracy, as the degree of a vertex v is a trivial upper bound for the functionality of v. Thus, it deserves a study for its own sake.

Alecu et al. [1] research the relation of functionality to many other graph parameters: in particular they provide a linear upper bound in terms of clique-width and a lower bound in terms of some function of VC-dimension. They also give a lower bound for the functionality of the hypercube that is linear in the dimension (i.e., logarithmic in the number of vertices).

Another parameter related to functionality is the so-called symmetric difference. Given two vertices u, v of G, let $\mathrm{sd}_G(u, v)$ (or just $\mathrm{sd}(u, v)$ when the graph is clear from the context) be the number of vertices different from u and v that are adjacent to exactly one of u and v. The *symmetric difference* $\mathrm{sd}(G)$ of a graph G is defined as

$$\max_{H\subseteq G}\min_{u,v\in V(H)}\mathrm{sd}_H(u,v),$$

where the maximum is again taken over all induced subgraphs. We may view $\mathrm{sd}(u,v)$ as the size of the set $(N(u)\Delta N(v))\setminus\{u,v\}$, which explains the term "symmetric difference". It is noted by Alecu et al. [1] that $\mathrm{fun}(G)\leq\mathrm{sd}(G)+1$. However, there is no lower bound in terms of sd as there are graphs of constant functionality and polynomial symmetric difference – for example the interval graphs. This was shown by Theorem 3.2 of Dallard et al. [5] and by our Theorem 7, while Corollary 5.3 of [5] shows that sd(INT) is unbounded, without providing explicit lower bounds.

The intersection graph of a family of sets F is a graph $G=(V,E)$ where $V=\{v_1,\ldots,v_n\}$ and two vertices v_i and v_j are connected if and only if the corresponding sets S_i and S_j of F intersect. An interval graph is an intersection graph of n intervals on a real line. A circular arc graph is an intersection graph of n arcs of a circle. We let INT_n denote the family of all intersection graphs with n vertices, INT the family of all interval graphs. In the same vein, we define CA_n and CA for circular arc graphs.

1.1 Our Results

In this paper, we show several lower and upper bounds for functionality and symmetric difference of various graph classes. As far as we know, there were only logarithmic lower bounds for functionality [1,3,5]. Thus, it would have been possible that the functionality is at most logarithmic (similarly to the VC-dimension [7]). However, we show that the functionality of the incidence graph of a finite projective plane of order k is exactly $k+1$, i.e., roughly $\sqrt{n}$. We complement this result with an almost matching upper bound that functionality of any graph is at most $O(\sqrt{n\log n})$. Further, we show for $\frac{3\log^2 n}{n}\leq p\leq 1-\frac{3\log^2 n}{n}$ that a random graph $G(n,p)$ has at least logarithmic functionality with probability $1-o(1)$. Note that if $p\leq o\left(\frac{\log n}{n}\right)$ then $\mathrm{fun}(G(n,p))\leq o(\log n)$ with high probability because the minimum degree of $G(n,p)$ for such p is $o(\log n)$ with high probability and $\mathrm{fun}(G)$ is bounded by the minimum degree of the graph G. Similarly for $p\geq 1-o\left(\frac{\log n}{n}\right)$, as the functionality of a graph G and its complement $\bar{G}$ is the same, and complement of $G(n,p)$ is $G(n,1-p)$. Thus, our range of p is almost optimal – up to a logarithmic factor. Further, we prove that any vertex of $G(n,\frac{1}{2})$ is a function of at most $O(\log n)$ vertices with high probability. Unfortunately, it does not imply that functionality of $G(n,\frac{1}{2})$ is logarithmic as it still can contain an induced subgraph of higher functionality. Overall, it suggests that graphs with polynomial functionality are quite rare and should be very structured (like the case of finite projective plane).

Further, we study symmetric difference of interval and circular arc graphs. We show that symmetric difference of circular arc graph is $\Theta(\sqrt{n})$, i.e., we prove that any circular arc graph has symmetric difference at most $O(\sqrt{n})$ and we present a circular arc graph of symmetric difference $\Omega(\sqrt{n})$. Recently, it was shown that interval graphs have bounded functionality and unbounded symmetric difference [5]. Even though it was not explicitly mentioned, the construction given by Dallard et al. [5] leads to the lower bound $\Omega(\sqrt[4]{n})$ for the symmetric difference of interval graphs. We independently came up with a different construction leading to the same lower bound, however analysis of our construction is simpler than the one from the previous work. For interval graphs, we also present the upper bound $O(\sqrt[3]{n})$ for symmetric difference. Thus, we are leaving a gap between the lower and upper bound. However, we show the symmetric difference of interval graphs is polynomial and strictly smaller than symmetric difference of circular arc graphs.

2 Functionality

2.1 Finite Projective Planes

Recall that a finite projective plane is a pair $(X, \mathcal{L})$, where X is a finite set and $\mathcal{L} \subseteq 2^X$, satisfying the following axioms [8]:

1. For every $p \neq q \in X$, there is exactly one subset of $\mathcal{L}$ containing p and q.
2. For every $L \neq M \in \mathcal{L}$, we have $|L \cap M| = 1$.
3. There exists a subset $Y \subseteq X$ of size 4 such that $|L \cap Y| \leq 2$ for every $L \in \mathcal{L}$.

Elements of X are called points and elements of $\mathcal{L}$ are called lines. We note that for every k which is a power of a prime, a finite projective plane with the following properties can be constructed. Each line contains exactly $k+1$ points. Each point is incident to exactly $k+1$ lines. The total number of points is k^2+k+1. The total number of lines is also k^2+k+1. The number k is called the order of the finite projective plane.

The incidence graph of a finite projective plane is a bipartite graph with one part X and the second part $\mathcal{L}$. In this graph, $x \in X$ is adjacent to $\ell \in \mathcal{L}$ iff x is incident to ℓ. The following theorem shows that the incidence graph of a finite projective plane has functionality approximately $\sqrt{n}$, where n is the number of vertices. Alecu et al. [1] have shown that there exists a function f such that for every graph G we have $\mathrm{vc}(G) \leq f(\mathrm{fun}(G))$. Our result complements this inequality by showing that the functionality of a graph cannot be upper bounded by its VC-dimension as it is known that the VC-dimension of a finite projective plane of any order is 2 [2].

Theorem 1. *Consider a finite projective plane of order k and its incidence graph G. Then,* $\mathrm{fun}(G) = k+1$. *Moreover, for any proper induced subgraph $G' \subset G$ we have* $\mathrm{fun}(G') \leq k$.

Proof. First, note that G is a $(k+1)$-regular graph, thus $\mathrm{fun}(G) \leq k+1$. Since G is connected, every proper subgraph $G' \subset G$ contains a vertex of degree at most k. Thus, $\mathrm{fun}(G') \leq k$.

It remains to prove that the functionality of every vertex $v \in V(G)$ is at least $k+1$. Let ℓ be a line of the projective plane. Because of the point-line duality of finite projective planes [8], it is enough to show that the functionality of ℓ is at least $k+1$. Let $S = \{p_1, \ldots, p_a, \ell_1, \ldots, \ell_b\}$ be a set of points p_i and lines ℓ_j (distinct from ℓ) such that $|S| = a + b \leq k$. We will prove that there exist vertices u and w satisfying:

1. The vertices u and w are not in S and they are not adjacent to any vertices in S. They are also different from ℓ.
2. The vertex u is adjacent to ℓ, i.e., u is a point incident to ℓ.
3. The vertex w is not adjacent to ℓ, i.e., w is a point not incident to ℓ or it is a line.

Once we prove the existence of these two vertices, we are done as their existence implies that ℓ is not a function of S.

There are $k+1$ points on each line and every two lines intersect in one point. It follows that there is a point q on ℓ which is distinct from each p_i and it is not incident with any ℓ_i. Thus, the vertex q is not adjacent to any vertex in S and it is adjacent to ℓ. We set $u = q$.

Let $L = \{\ell, \ell_1, \ldots, \ell_b\}$. We will prove the existence of the vertex w by considering two cases. The first case is $b = k$, i.e., the set S consists of k lines and no points. We may consider any line $\ell' \notin L$ as the vertex w. The vertex w is not adjacent to any vertex in S as it contains only lines and it is not adjacent to the line ℓ either. Thus, the vertex w satisfies the sought properties.

The second case is $b \leq k-1$. Let P be the set of points $p_1, \ldots, p_a$ together with all the points that are incident to some line from L. We will show that $|P| \leq k^2 + k$, which implies existence of a vertex $w \in V(G) \setminus P$ satisfying the sought properties. First, note that if $b = 0$ (i.e., S contains only points), then $|P| \leq k + 1 + a = 2k + 1$. Next, we suppose that $b \geq 1$ which means that L contains at least two lines. Recall that each pair of line intersects in a point. Thus, there are at most $(b+1)(k+1) - 1$ points incident to the lines in L (the -1 comes from the fact that we have at least two lines). Moreover, we have a points $p_1, \ldots, p_a$. Thus, $|P| \leq (b+1)(k+1) - 1 + a$. Since $a + b \leq k$, we have $a \leq k - b$. Therefore,

$$|P| \leq (b+1)(k+1) - 1 + k - b = bk + 2k \leq k^2 + k.$$

This concludes the second case and also the whole proof. □

2.2 Upper Bound for General Graphs

Theorem 2. *If G is a graph with n vertices, then* $\operatorname{fun}(G) \leq \sqrt{c \cdot n \ln n}$ *for any $c > 3$, if n is big enough.*

Proof. We show that there is $v \in V(G)$ such that $\operatorname{fun}_G(v) \leq d(n) := \sqrt{c \cdot n \ln n}$. As $d(n)$ is increasing, this suffices to show existence of such vertex also in a subgraph of G. We will write $d = d(n)$.

First, suppose that $|\operatorname{sd}(u, v)| < d$ for some u, v of G. Since $\operatorname{sd}(u, v) = (N(u) \Delta N(v)) \setminus \{u, v\}$, the vertex v is a function of the set $\operatorname{sd}(u, v) \cup \{u\}$. Next, suppose all sets $\mathrm{sd}(u,v)$ have at least d vertices. In this case we choose an arbitrary vertex $v \in V(G)$. We also choose a random set $S \subseteq V(G)$ by independently putting each vertex of G different from v to S with probability $p = d/n$. Suppose v is not a function of S. Then, there exists $u_1 \in N(v) \setminus S$ and $u_2 \notin N(v) \cup S \cup \{v\}$ such that neighbors of u_1 and u_2 in S are the same, that happens if and only if $\mathrm{sd}(u_1,u_2) \cap S = \emptyset$. We bound the probability of this event by the union bound

$$\begin{aligned}&\Pr\big[\exists u_1 \in N(v) \setminus S, u_2 \notin N(v) \cup S \cup \{v\} : S \cap \mathrm{sd}(u_1,u_2) = \emptyset\big]\\ &\qquad \leq \sum_{u_1,u_2} (1-p)^{|\mathrm{sd}(u_1,u_2)|-1} \leq n^2(1-p)^{d-1}.\end{aligned}$$

The -1 in the exponent is caused by $v \in \mathrm{sd}(u_1,u_2)$, but v cannot be chosen to belong to S. Thus, the probability that v is not a function of S is at most $n^2e^{-p(d-1)}$, which is strictly smaller than $\frac{1}{n}$ whenever $c > 3$ and n is big enough. Clearly, the expected size of S is $p(n-1) = \frac{n-1}{n}d$. Thus by Markov inequality, $\Pr[|S| > d] \leq \frac{n-1}{n} = 1-\frac{1}{n}$. This means that with the positive probability $|S| \leq d$ and v is function of S and we can conclude that $\mathrm{fun}_G(v) \leq d$. □

2.3 Random Graphs

In this section, we prove our results about functionality of random graphs.

Theorem 3. *The functionality of the random graph $G = G(n,p)$ is $\Omega(\log n)$ with probability at least $1 - \frac{1}{n^{\log n}}$ for any $\frac{3\log^2 n}{n} \leq p \leq 1 - \frac{3\log^2 n}{n}$.*

Proof. Note that $\mathrm{fun}(G) = \mathrm{fun}(\bar{G})$, where $\bar{G}$ is the complement of G. Since $G(n,1-p)$ is the complement of $G(n,p)$, we can suppose that $p \leq \frac{1}{2}$. We will prove that functionality of G is larger than $k = \frac{1}{2}\log n$ (w.h.p.). Let $v, u_1, \ldots, u_k$ be vertices of G. We will show that v is function of $u_1, \ldots, u_k$ only with small probability.

First suppose that $\frac{3\log^2 n}{\sqrt{n}} \leq p \leq \frac{1}{2}$. We divide the rest of vertices $V' = V(G) \setminus \{v, u_1, \ldots, u_k\}$ into buckets according to the adjacency to $u_1, \ldots, u_k$, i.e., for each subset $S \subseteq \{u_1, \ldots, u_k\}$ we create a bucket B_S consisting of vertices that are adjacent to all vertices in S and are not adjacent to any vertex in $\{u_1, \ldots, u_k\} \setminus S$. There are at most 2^k buckets and therefore, there is a bucket B containing at least $\frac{|V'|}{2^k} \geq \frac{\sqrt{n}}{2}$ vertices. However by definition of functionality, that means v is connected to all vertices in B or to none of them. This event occurs only with a probability

$$p^{|B|} + (1-p)^{|B|} \leq 2 \cdot (1-p)^{\frac{\sqrt{n}}{2}} \leq 2 \cdot e^{-p \cdot \frac{\sqrt{n}}{2}} \leq 2^{-\frac{3}{2} \cdot \log^2 n}.$$

We have $n \cdot \binom{n-1}{k} < n^{\frac{\log n}{2}}$ possibilities how to choose $v, u_1, \ldots, u_k$. Thus by the union bound, we have that $\mathrm{fun}(G) \leq k$ with probability at most $\frac{1}{n^{\log n}}$.

Now, suppose that $\frac{3\log^2 n}{n} \leq p \leq \frac{3\log^2 n}{\sqrt{n}}$. Let E' be edges between $u_1, \ldots, u_k$ and $V' = V \setminus \{v, u_1, \ldots, u_k\}$. We bound the expected size of E' as follows.

$$\mathbb{E}[|E'|] = p \cdot k \cdot |V'| \leq \frac{3}{2}\sqrt{n} \cdot \log^3 n$$

By the Chernoff bound, we have that probability that $|E'| \geq 3\sqrt{n} \cdot \log^3 n = \ell$ is at most $e^{-\sqrt{n}\log^3 n}$. Thus with high probability, there are at least $|V'| - \ell \geq \frac{2n}{3}$ vertices in V' such that there is no edge between them and $u_1, \ldots, u_k$. Denote such vertices as B_0. Now, we proceed similarly as in the previous case. Again, the vertex v has to be connected to all vertices in B_0 or none of them, which occurs only with the following probability.

$$p^{|B_0|} + (1-p)^{|B_0|} \leq 2 \cdot (1-p)^{\frac{2n}{3}} \leq 2 \cdot e^{-p \cdot \frac{2n}{3}} \leq 2^{-2 \cdot \log^2 n}.$$

Thus, the probability that v is function of $u_1, \ldots, u_k$ is at most

$$e^{-\sqrt{n}\log^3 n} + 2^{-2\cdot\log^2 n} \leq 2^{-\frac{3}{2}\cdot\log^2 n}.$$

By the same union bound as before, we have that $\mathrm{fun}(G) \leq k$ with probability at most $\frac{1}{n^{\log n}}$. □

Proposition 1. *Each vertex of the random graphs $G = G(n, \frac{1}{2})$ is a function of at most $3\log n$ vertices with probability at least $1 - \frac{1}{n}$.*

Proof. Let $k = 3\log n$ and V' be a set of k vertices. We will show that with high probability there are no two vertices $v_1, v_2 \notin V'$ such that $N(v_1) \cap V' = N(v_2) \cap V'$, i.e., they have the same neighborhood in V'. Let $v_1, v_2 \notin V'$, then

$$\Pr[v_1, v_2 : N(v_1) \cap V' = N(v_2) \cap V'] = 2^{-k}.$$

By the union bound, we have:

$$\Pr[\exists v_1, v_2 \notin V' : N(v_1) \cap V' = N(v_2) \cap V'] \leq \frac{n^2}{2^k} = \frac{1}{n}. \tag{1}$$

Thus, each vertex outside of V' has a unique neighborhood to V'. Therefore, V' determines adjacency for any vertex not in V'. □

3 Symmetric Difference

In this section, we will prove our lower and upper bounds for symmetric difference of interval and circular arc graphs.

3.1 Circular Arc Graphs

In this section, we prove that symmetric difference of circular arc graphs[1] is $\Theta(\sqrt{n})$. More formally, we prove the following two theorems.

Theorem 4. *Any circular arc graph $G \in \mathrm{CA}_n$ has symmetric difference at most $O(\sqrt{n})$.*

[1] Intersection graphs of arcs of a cycle, see Sect. 1 for the proper definition.

Theorem 5. *There is a circular arc graph $G \in \mathrm{CA}_n$ of symmetric difference at least $\Omega(\sqrt{n})$.*

First, we prove the upper bound, i.e., that every circular arc graph has symmetric difference at most $O(\sqrt{n})$. We use the following notations for arcs. Let a, b be two points of a circle. Then, an arc $r = [a, b]$ is an arc beginning in a and going in clockwise direction to b. We call a as the starting point of r and b as the ending point of a.

Proof (Proof of Theorem 4). Let circle arc graph $G = (V, E)$ be an intersection graph of a set of arcs $R = \{r_1, \dots, r_n\}$ of a circle C. Without loss of generality, we can suppose that the circumference of C is $2n$, endpoints of all arcs $r_i \in R$ are integer points and are different for all arcs. Thus, each integer point $\{0, \dots, 2n-1\}$ of C is an endpoint of exactly one arc in R.

Consider an arc $r = [a, b] \in R$. We again suppose that the points a and b are in clockwise order and the arc r starts in a and goes clockwise to b. We represent the arc r as a point (a, b) in the plane $\mathbb{R}^2$. Note that all these points are in the square S with corners in the points $(0, 0)$ and $(2n-1, 2n-1)$ (some points maybe on the border of S). We divide the square S into subsquares of size $k \times k$ for $k = \frac{2n-1}{\lfloor\sqrt{n}\rfloor - 1}$. Note that we have strictly less than n such subsquares and $k = \Theta(\sqrt{n})$. Thus, there is at least one subsquare that contains two points representing arcs, say $r = [a, b]$ and $r' = [a', b']$. It follows that $|a - a'|, |b - b'| \le k$. Suppose that $a' > a$ and $b' > b$, other cases are analogous. Then, each arc counted in $\mathrm{sd}(r, r')$ has to start or end in an integer point from the interval $[a, a'-1]$ or $[b, b'-1]$. Since there are at most $2k$ integer points in these two intervals and each integer point of C is an endpoint of exactly one arc of R, we conclude that $\mathrm{sd}(r, r') \le 2k \le O(\sqrt{n})$. □

Now, we give a construction of a circular arc graph of symmetric difference at least $\Omega(\sqrt{n})$. Let n be a square of an integer, i.e., $n = d^2$ for some $d \in \mathbb{N}$. We consider a circle C of circumference n and a set P of integer points of C, i.e., $P = \{0, \dots, n-1\}$ ordered in clockwise direction. The length $|r|$ of an arc $r = [a, b]$ is equal to $b - a \pmod{n}$. We say the arc $[a, b]$ is *integral* if both a and b are integers.

We will represent each point $p \in P$ as two integer indices $0 \le i, j < d = \sqrt{n}$ such that $p = i \cdot d + j$. Note that each point p has a unique such representation and we denote it as $(i, j)_d$. Let R be a set of arcs $\left[(i, j)_d, (j, i)_d\right]$ for all possible $i \ne j$ such that length of each arc in R is at most $\frac{n}{4}$. Note that we require that $i \ne j$, thus we do not consider zero-length arcs consisting only of a point of a form $(i, i)_d$. See Fig. 1 for an illustration. Let G be the intersection graph of arcs in R. We we will prove that $\mathrm{sd}(G) \ge \Omega(\sqrt{n})$. First, we prove two auxiliary lemmas, Lemma 1 and 2. Lemma 1 asserts that each sufficiently long arc of C contains a lot of starting and ending points of arcs in R. Lemma 2 states that for any two arcs r and r' we will find a long arc s that is a subarc of only one of the arcs r and r' (say r). Thus by Lemma 1, the arc r intersects many arcs that go "away" from the arc r' and that is enough to imply Theorem 5.

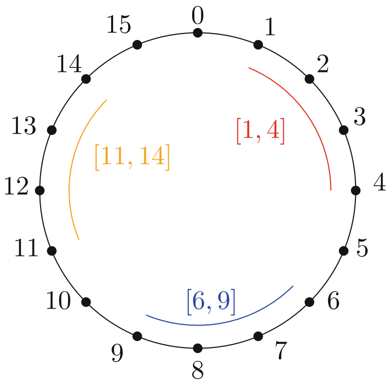


Fig. 1. An example of the circular arc graph lower bound construction for $n = 16$ and $d = 4$. This graph contains three arcs corresponding to points $1 = (0, 1)_d$, $6 = (1, 2)_d$, and $11 = (2, 3)_d$. For an example, the arc corresponding to point $3 = (0, 3)_d$ is omitted since $12 - 3 = 9 > 4 = \frac{n}{4}$ where $12 = (3, 0)_d$.

Lemma 1. *Let s be an integral arc of C of length at least $d-1$. Then, it contains at least $\frac{d}{5}$ integer points such that they are starting points of arcs in R. Similarly, it contains at least $\frac{d}{5}$ integer points such that they are ending points of arcs in R.*

Proof. Since s is integral and its length is at least $d - 1$, it contains at least d consecutive integer points. Thus, there is an integer j such that s contains all points of the set $\{(j, t)_d, \ldots, (j, d - 1)_d, (j + 1, 0)_d, \ldots, (j + 1, t - 1)_d\}$ for some $t \in [d]$. For any $\ell \in [d]$, we define $j_\ell = j$ if $\ell \geq t$ and $j_\ell = j + 1$ otherwise. It follows that s contains the point $(j_\ell, \ell)_d$ for every $\ell \in [d]$. Consider a subset of these points $S = \left\{\left(j_k, k\right)_d \mid 1 \leq (k - j \mod d) \leq \frac{d}{5}\right\}$. Observe that $|S| = \frac{d}{5}$. We claim that each point of S is a starting point of an arc in R. Let r be an arc $\left[(j_k, k)_d, (k, j_k)_d\right]$ where $1 \leq (k - j \mod d) \leq \frac{d}{5}$.

$$\begin{aligned} |r| &= k \cdot d + j_k - j_k \cdot d - k \mod n \\ &\leq (k - j_k) \cdot d + d \leq \frac{d}{5} \cdot d + d \leq \frac{n}{4} \qquad \text{for sufficiently large } n \end{aligned}$$

Thus, $r \in R$. Analogously, the set $E = \left\{\left(j_k, k\right)_d \mid 1 \leq (j + 1 - k \mod d) \leq \frac{d}{5}\right\}$ of size $\frac{d}{5}$ contains ending points of arcs in R. □

Lemma 2. *Let $r = [(i, j)_d, (j, i)_d]$ and $r' = [(i', j')_d, (j', i')_d]$ be two arcs in R. Then, at least one of the arcs r and r' contains an integral subarc s of length $d - 2$ that is disjoint from the other arc.*

Proof. If the arcs r and r' are disjoint then the existence of s is trivial as each arc in R has length at least $d - 1$ (the arcs in R of length exactly $d - 1$ are those of the form $[(k, k + 1)_d, (k + 1, k)_d]$).

Thus, suppose that r and r' intersect and without loss of generality suppose that the ending point of r lies inside r', i.e. we read in clockwise order the points

$(i',j')_d$, $(j,i)_d$ and $(j',i')_d$. Consider an arc $s' = [(j,i)_d,(j',i')_d]$. Note that arc s' is a subarc of r' and intersects r only in the point $(j,i)_d$. Thus, if $|s'| \geq d-1$, the arc r' would contain the sought integral subarc s. We have $|s'| = (j'-j)\cdot d + i' - i \pmod{n}$. Since $0 \leq i, i', j, j' \leq d-1$, it holds that $|(j'-j)\cdot d+i'-i| \leq d^2-1 < n$. Thus, if $|s'| \leq d-2$ then $j = j'$ or $j'-j = 1 \pmod{n}$ and $i - i' \geq 2$ (note that this holds even when $j = n-1$ and $j' = 0$).

First, suppose that $j'-j = 1 \pmod{n}$ (and $i-i' \geq 2$). Then, the start point of r also belongs to r' and the arc $s' = [(i',j')_d,(i,j)_d]$ has length at least $d-1$ and again it is a subarc of r' and intersects the arc r only in the single point $(i,j)_d$.

The last case is when $j' = j$. This implies that $i \neq i'$ as all endpoints are unique. Moreover, since the point $(j,i)_d$ precedes the point $(j',i')_d$ in clockwise order, we get that $i < i'$. Then, the arc $s' = [(i,j)_d,(i',j')_d] = [(i,j)_d,(i',j)_d]$ has length at least d and is a subarc of only the arc r with the exception of its ending point $(i',j')_d$. □

Now, we are ready to prove Theorem 5.

Proof (Proof of Theorem 5). Consider two arcs $r = [(i,j)_d,(j,i)_d]$ and $r' = [(i',j')_d,(j',i')_d]$ in R. Recall that all arcs in R have length at most $\frac{n}{4}$. Thus, we can suppose that the arc $s' = [(j',i')_d,(i,j)_d]$ has length at least $\frac{n}{4}$ and is disjoint from r and r', except for the endpoints $(j',i')_d$ and $(i,j)_d$. Note that, this implies that the endpoints $(j,i)_d$ and $(j',i')_d$ of r and r', respectively, are in clockwise order. By Lemma 2, we suppose that r contains a subarc s of length $d-2$ that is disjoint with r' (the other case when s is a subarc of r' is analogous). Let L be a set of points in s such that they are ending points of arcs in R (distinct from r). By Lemma 1, we have that $|L| \geq \frac{d}{5} - 1$.

Let $t = [a,b]$ be an arc with the ending point b in L. The arc t intersects the arc r as the ending point b is a point of $s \subseteq r$. Since the arc s' has length at least $\frac{n}{4}$ and $t \in R$, the starting point a has to be a point of s' and thus, the arc t is disjoint from r'. Therefore, $\mathrm{sd}(r,r') \geq |L| \geq \Omega(\sqrt{n})$. □

3.2 Interval Graphs

In this section, we will prove that symmetric difference of interval graphs[2] is still fixed power of n but strictly less than the symmetric difference of circular arc graphs. In particular, we will prove the following two theorems.

Theorem 6. *Any interval graph $G \in \mathrm{INT}_n$ has symmetric difference at most $O(\sqrt[3]{n})$.*

Theorem 7. *There is an interval graph $G \in \mathrm{INT}_n$ of symmetric difference at least $\Omega(\sqrt[4]{n})$.*

[2] Intersection graphs of intervals of the real line, see Sect. 1 for the proper definition.

Note that the existence of interval graphs of arbitrarily high symmetric difference is proved in Corollary 5.3 of Dallard et al. [5]. While they do not provide explicit bounds, their proof gives the same $\Omega(\sqrt[4]{n})$ bound as ours. However, our proof of the lower bound is self-contained and we believe it to be simpler. We start with a proof of the upper bound.

Proof (Proof of Theorem 6). Let $G = (V, E)$ be an intersection graph of intervals $R = \{r_1, \dots, r_n\}$ with $r_i = [a_i, b_i]$ and let $V = \{1, \dots, n\}$. Without loss of generality, we suppose for clarity that all a_i's and b_i's are different points. The intervals are numbered in the order given by their starting points, i.e., for every two indices $i, j \in [n]$ we have $i < j$ if and only if $a_i < a_j$. For an interval r_i, we define two sets:

1. $A_i = \{j \mid a_i < b_j\}$, i.e., it contains the indices of intervals in R that end after the interval r_i starts.
2. $B_i = \{j \mid a_j < b_i\}$, i.e., it contains the indices of intervals in R that start before the interval r_i ends.

Note that $N(i) = A_i \cap B_i$. Moreover, $\mathrm{sd}(i, j) \leq |N(i) \Delta N(j)| \leq |A_i \Delta A_j| + |B_i \Delta B_j|$. Note that for each pair i, j, it holds that $A_i \subseteq A_j$ or $A_j \subseteq A_i$, thus $A_i \Delta A_j$ is $A_i \setminus A_j$ or $A_j \setminus A_i$ and analogously with B_i and B_j. We will prove that there are two intervals r_i and r_j such that $|A_i \Delta A_j| + |B_i \Delta B_j| \leq O(\sqrt[3]{n})$.

Let $d \in \mathbb{N}$ be a parameter. We will find two vertices $i, j \in V$ such that $\mathrm{sd}(i, j) \leq O(\max\{d, d + \frac{n}{d^2}\})$. Thus, if we set $d = \sqrt[3]{n}$ we would get $\mathrm{sd}(i, j) \leq O(\sqrt[3]{n})$. Since any subgraph of an interval graph is again an interval graph, the upper bound for $\mathrm{sd}(G)$ will follow. Let $D_\ell = A_\ell \setminus A_{\ell+1}$. In other words, the set D_ℓ contains intervals of R that end after the interval r_ℓ starts but before the interval $r_{\ell+1}$ starts. Note that $B_i = \{1, \dots, \ell\}$ where ℓ is the unique index such that $i \in D_\ell$. Let k be the largest index such that $\sum_{\ell \leq k} |D_\ell| \leq d + 2$. Note that for any two indices $i, j \leq k$, it holds that $|A_i \Delta A_j| \leq \sum_{\ell \leq k} |D_\ell|$.

First, we prove that if $k \leq d^2$, then $\mathrm{sd}(i, j) \leq 2d + 2$. Suppose that actually $\sum_{\ell \leq k} |D_\ell| \leq d$. Then, $|D_{k+1}| \geq 3$. Let $i, j \in D_{k+1}$ such that $i, j \leq k$. Then, $B_i = B_j$ and $|A_i \Delta A_j| \leq d$, therefore $\mathrm{sd}(i, j) \leq d$.

From now, we suppose that $d \leq \sum_{\ell \leq k} |D_\ell| \leq d + 2$. Let $p \leq k$ be an index such that $|D_p| \geq 2$ (if such p exists) and $i, j \in D_p = A_p \setminus A_{p+1}$. Thus, $B_i = B_j$. Since $i, j \leq p \leq k$, then $|A_i \Delta A_j| \leq d + 2$ and $\mathrm{sd}(i, j) \leq d + 2$. Note that so far, we did not use the assumption that $k \leq d^2$.

Now, suppose that for all $\ell \leq k$ it holds that $|D_\ell| \leq 1$. Since $k \leq d^2$ there exists two indices $p < q \leq k$ such that $D_p, D_q \neq \emptyset$ and $q - p \leq d$ (there are at least d indices $\ell \leq k$ such that $|D_\ell| = 1$). Let $i \in D_p$ and $j \in D_q$. Since $B_i = \{1, \dots, p\}, B_j = \{1, \dots, q\}$, we have $|B_j \setminus B_i| \leq d$. Further, since $i, j \leq k$, we have $|A_i \Delta A_j| \leq d + 2$. Thus, $\mathrm{sd}(i, j) \leq 2d + 2$.

Now, we suppose that $k > d^2$. It follows there are two indices $i, j \leq k$ such that $i \in D_p$ and $j \in D_q$ such that $|p - q| \leq \frac{n}{d^2}$. Therefore, $|A_i \Delta A_j| \leq d + 2$ and $|B_i \Delta B_j| = |p - q| = \frac{n}{d^2}$ and $\mathrm{sd}(i, j) \leq d + 2 + \frac{n}{d^2}$. □

Now, we give a construction of an interval graph with symmetric difference $\Omega(\sqrt[4]{n})$. Let $d \in \mathbb{N}$ sufficiently large. We construct $\Theta(d^4)$ intervals on a line segment $[0,t]$ for $t = 20d^3$ such that the corresponding intersection graph G will have $\mathrm{sd}(G) \geq d$. There will be intervals of two types – short and long. See Fig. 2 for an illustration. Short intervals have length d and they start in each point $0, \ldots, t-d$, i.e.,

$$S = \{[i, i+d] \mid i \in \{0, \ldots, t-d\}\}.$$

Long intervals will have various lengths. For $i \geq 0$, let $\ell_i = 4d^2 \cdot (i+1)$. For $0 \leq i \leq 2d-1$, we define the i-th class of long intervals as

$$L_i = \{[a,b] \mid a, b \equiv i \pmod{2d}; \ell_i \leq b-a \leq \ell_i + 2d^2\},$$

i.e., the set L_i contains intervals such that they start and end in points congruent to i modulo $2d$ and their length is between ℓ_i and $\ell_i + 2d^2$. Let $I = S \cup \bigcup_{0 \leq i \leq 2d-1} L_i$ be the set of all constructed intervals. We start with two observations about I.

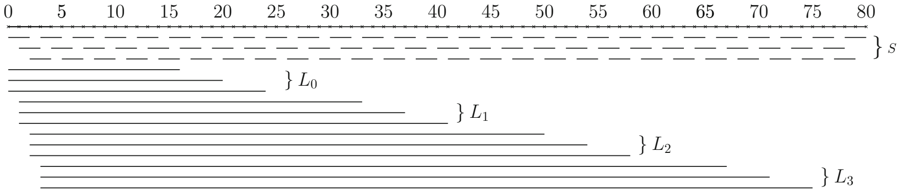


Fig. 2. An example of the interval graph lower bound construction for $d = 2$. For clarity, only the first three intervals are displayed from each set L_i.

Observation 8. *Any interval $[a,b]$ (for $a, b \in \mathbb{N}$) of the line segment $[0,t]$ of length $2d$ contains d short intervals.*

Observation 9. *There are at most $O(d^4)$ intervals in I.*

Proof. Clearly, there are $t - d + 1 = O(d^3)$ intervals in S. Note that for any $a \leq \frac{t}{2}, a \equiv i \pmod{2d}$, there are exactly $d+1$ intervals of a form $[a,b]$ in L_i because of the length constraints of the long interval. Analogously, for any $b \geq \frac{t}{2}, b \equiv i \pmod{2d}$, there are exactly $d+1$ intervals of a form $[a,b]$ in L_i. We remark this indeed holds even for L_{2d-1} as we set t to be large enough. There are no other intervals in L_i. Since there are $O(d^2)$ points $p \in [0,t]$ such that $p \equiv i \pmod{2d}$, it follows that $|L_i| \leq O(d^3)$. Therefore, there are at most $O(d^4)$ long intervals as there are $O(d)$ classes of long intervals. □

The graph G is an intersection graph of I. Now, we are ready to prove Theorem 7, i.e., $\mathrm{sd}(G) \geq \Omega(\sqrt[4]{n})$.

Proof (Proof of Theorem 7). Let $r = [a,b]$ and $r' = [a',b']$ be two intervals in I. Without loss of generality let $a \leq a'$. First, suppose that the $r, r' \in S$, i.e., both of them are short. In this case it holds that $a < a'$ and $b < b'$. First, if

$a' > a + 2d$, then all d short interval of form $[i, i + d]$ for $i \in r$ do not intersect r'. Thus, $|N(r)\Delta N(r')| \geq d$.

Now, suppose that $a' \leq a+2d$. Further, suppose that $b' \leq \frac{t}{2}$. Then as already observed, there are d intervals of the form $[b', c]$ in L_i for $b' \equiv i \pmod{2d}$. Since b' is not in r, we have that $|N(r)\Delta N(r')| \geq d$.

If $b' > \frac{t}{2}$, then $a > \frac{t}{2} - 3d$, as $b' - d = a' \leq a + 2d$. Analogously, it holds there are d intervals of the form $[c, a]$ in L_i for $a \equiv i \pmod{2d}$ as d and t is large enough.

Now, suppose that $r, r' \in L_i$ for some i. In this case $a, b, a', b' \equiv i \pmod{2d}$. Since $r \neq r'$, it follows that $|a - a'| \geq 2d$ or $|b - b'| \geq 2d$. Thus, at least one of the intervals r and r' has a private subinterval of length at least $2d$ and by Observation 8, we have that $|N(r)\Delta N(r')| \geq d$.

Let k be the difference of length of r and r'. For the remaining cases we will prove that $k \geq 4d$. Then, at least one of the intervals r and r' contains a private subinterval of length at least $2d$ and again by Observation 8, we have that $|N(r)\Delta N(r')| \geq d$. There are two remaining cases:

1. $r \in S, r' \in L_i$: Then, $|r| = d$ and $|r'| \geq 4d^2$.
2. $r \in L_i, r' \in L_j$ for $i < j$: Then, $|r| \leq 4d^2 \cdot (i+1) + 2d^2$ and $|r'| \geq 4d^2 \cdot (j + 1)$. It follows that $k = 4d^2 \cdot (j - i) - 2d^2 \geq 2d^2$.

Thus, in both cases we have that $k \geq 4d$ for $d \geq 2$. We have showed that $|N(r)\Delta N(r')| \geq d$ for all cases. Thus by Observation 9, we conclude that $\text{sd}(G) \geq \Omega(\sqrt[4]{n})$. □

Acknowledgements. The research presented in this paper has been started during the KAMAK workshop in 2021. We are grateful to the organizers of this wonderful event.

References

1. Alecu, B., Atminas, A., Lozin, V.V.: Graph functionality. J. Comb. Theory Ser. B **147**, 139–158 (2021). https://doi.org/10.1016/j.jctb.2020.11.002
2. Alon, N., Haussler, D., Welzl, E.: Partitioning and geometric embedding of range spaces of finite Vapnik-Chervonenkis dimension. In: Soule, D. (ed.) Proceedings of the Third Annual Symposium on Computational Geometry, Waterloo, Ontario, Canada, 8–10 June 1987, pp. 331–340. ACM (1987)
3. Atminas, A., Collins, A., Lozin, V., Zamaraev, V.: Implicit representations and factorial properties of graphs. Discret. Math. **338**(2), 164–179 (2015). https://doi.org/10.1016/j.disc.2014.09.008
4. Balogh, J., Bollobás, B., Weinreich, D.: The speed of hereditary properties of graphs. J. Comb. Theory Ser. B **79**(2), 131–156 (2000). https://doi.org/10.1006/jctb.2000.1952, https://www.sciencedirect.com/science/article/pii/S009589560091952X
5. Dallard, C., Lozin, V., Milanič, M., Štorgel, K., Zamaraev, V.: Functionality of box intersection graphs (2023). https://doi.org/10.48550/ARXIV.2301.09493
6. Hatami, H., Hatami, P.: The implicit graph conjecture is false. In: 63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31–3 November 2022, pp. 1134–1137. IEEE (2022)

7. Kranakis, E., Krizanc, D., Ruf, B., Urrutia, J., Woeginger, G.: The VC-dimension of set systems defined by graphs. Discret. Appl. Math. **77**(3), 237–257 (1997). https://doi.org/10.1016/S0166-218X(96)00137-0
8. Matoušek, J., Nešetřil, J.: Invitation to Discrete Mathematics, 2 ed. Oxford University Press, Oxford (2009)

Cops and Robbers on Multi-Layer Graphs

Jessica Enright[1], Kitty Meeks[1], William Pettersson[1](✉), and John Sylvester[1,2]

[1] School of Computing Science, University of Glasgow, Glasgow, UK
{jessica.enright,kitty.meeks,william.pettersson}@glasgow.ac.uk,
john.sylvester@liverpool.ac.uk

[2] Department of Computer Science, University of Liverpool, Liverpool, UK

Abstract. We generalise the popular *cops and robbers* game to multi-layer graphs, where each cop and the robber are restricted to a single layer (or set of edges). We show that initial intuition about the best way to allocate cops to layers is not always correct, and prove that the multi-layer cop number is neither bounded from above nor below by any function of the cop numbers of the individual layers. We determine that it is NP-hard to decide if k cops are sufficient to catch the robber, even if each layer is a tree plus some isolated vertices. However, we give a polynomial time algorithm to determine if k cops can win when the robber layer is a tree. Additionally, we investigate a question of worst-case division of a simple graph into layers: given a simple graph G, what is the maximum number of cops required to catch a robber over all multi-layer graphs where each edge of G is in at least one layer and all layers are connected? For cliques, suitably dense random graphs, and graphs of bounded treewidth, we determine this parameter up to multiplicative constants. Lastly we consider a multi-layer variant of Meyniel's conjecture, and show the existence of an infinite family of graphs whose multi-layer cop number is bounded from below by a constant times $n/\log n$, where n is the number of vertices in the graph.

Keywords: Cops and robbers · multi-layer graphs · pursuit-evasion games · Meyniel's conjecture

1 Introduction

We investigate the game of cops and robbers played on multi-layer graphs. Cops and robbers is a 2-player adversarial game played on a graph introduced independently by Nowakowski and Winkler [22], and Quilliot [25]. At the start of the game, the cop player chooses a starting vertex position for each of a specified number of cops, and the robber player then chooses a starting vertex position for the robber. Then in subsequent rounds, the cop player first chooses none, some, or all cops and moves them along exactly one edge to a new vertex. The robber player then either moves the robber along an edge, or leaves the robber on its current vertex. The cop player wins if after some finite number of rounds a cop

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 319–333, 2023.
https://doi.org/10.1007/978-3-031-43380-1_23

occupies the same vertex as the robber, and the robber wins otherwise. Both players have perfect information about the graph and the locations of cops and robbers. Initially, research focussed on games with only one cop and one robber, and graphs on which the cop could win were classed as *copwin* graphs. Aigner and Fromme [1] introduced the idea of playing with multiple cops, and defined the *cop number* of a graph as the minimum number of cops required for the cop player to win on that graph. Many variants of the game have been studied, and for an in-depth background on cops and robbers, we direct the reader to [5].

In this paper, we play cops and robbers on multi-layer graphs where each cop and the robber will be associated with exactly one layer, and during their respective turns, will move only over the edges in their own layer. While we define multi-layer graphs formally in upcoming sections, roughly speaking, here a multi-layer graph is a single set of vertices with each layer being a different (though possibly overlapping) set of edges on those vertices. The variants we study could intuitively be based on the premise that the cops are assigned different modes of transport. For instance, a cop in a car may be able to move quickly down streets, while a cop on foot may be slower down a street, but be able to quickly cut between streets by moving through buildings or down narrow alleys.

Extending cops and robbers to multi-layer graphs creates some new variants, and generalises some existing variants. Fitzpatrick [14] introduced the *precinct* variant, which assigns to each cop a subset of the vertices (called their *beat*). In the precinct variant, a cop can never leave their beat. This can be modelled as multi-layer cops and robbers by restricting each layer to a given beat. Fitzpatrick [14] mainly considers the case were a beat is an isometric path, we allow more arbitrary (though usually spanning and connected) beats/layers. Clarke [11] studies the problem of covering a graph with a number of cop-win subgraphs to upper bound the cop number of a graph — again such constructions can be modelled as multi-layer graphs with the edges of each layer forming a cop-win graph. Another commonly studied variant of cops and robbers defines a speed s (which may be infinite) such that the robber can move along a path of up to s edges on their turn [6, Section 3.2]. These can also be modelled as multi-layer graphs by adding edges between any pair of vertices of distance at most s that only belong to the layer the robber is occupying.

1.1 Further Related Work

Temporal graphs, in which edges are active only at certain time steps, are sometimes modelled as multi-layered graphs. There has been some work on cops and robbers on temporal graphs, though generally yielding quite a different game to the ones we consider here as a cop is not restricted to one layer. In particular, [3] considers cops and robbers on temporal graphs and when the full temporal graph is known they give a $O(n^3T)$ algorithm to determine the outcome of the game where T is the number of timesteps.

Variants of cops and robbers are also studied for their relationships to other parameters of graphs. For instance, the cop number of a graph G is at most one plus half the treewidth of G [17]. And if one considers the “helicopter” variant of cops and robbers, the treewidth of a graph is strictly less than the helicopter cop number of the graph [29]. Toruńczyk [32] generalises many graph parameters, including treewidth, clique-width, degeneracy, rank-width, and twin-width, through the use of variants of cops and robbers. We introduce our multi-layer variants of cops and robbers partially in the hopes of spurring research towards multi-layer graph parameters using similar techniques.

Recently Lehner, resolving a conjecture by Schröeder [27], showed the cop number of a toroidal graph is at most three [19]. There is also an interesting connection between cop number and the genus of the host graph [1,8,26,27]. It remains open whether any such connection can be made in the multi-layer setting.

1.2 Outline and Contributions

In Sect. 2 we define multi-layer graphs and multi-layer cops and robbers.

In Sect. 3 we develop several examples which highlight several counter-intuitive facts and properties of the multi-layer cops and robbers game. In particular, we show the multi-layer cop number is not bounded from above or below by a non-trivial function of the cop numbers of the individual cop layers.

In Sect. 4 we study the computational complexity of some multi-layer cops and robbers problems. We show that deciding if a given number of cops can catch a robber is NP-hard even if each layer is a tree plus some isolated vertices, but that if only the robber layer is required to be a tree the problem is FPT in the number of cops and the number of layers of the graph.

In Sect. 5 we consider an extremal version of multi-layer cop number over all divisions into layers of a single-layer graph. In particular, for a given single-layer graph G what is the maximum multi-layer cop number of any multi-layer graph $\mathcal{G}$ when all edges of G are present in at least one layer of $\mathcal{G}$.

In Sect. 6 we consider Meyniel's conjecture, which states that the single-layer cop number is $\mathcal{O}(\sqrt{|V|})$ and is a central open question in cops and robbers. We investigate whether a multi-layer analogue of Meyniel's conjecture can hold and, determine the worst case multi-layer cop number up to a multiplicative $\mathcal{O}(\log n)$ factor. This contrasts with the situation on simple graphs, where the worst-case is only known up to a multiplicative $n^{1/2-o(1)}$ factor.

Finally in Sect. 7 we reflect and conclude with some open problems.

Due to space limitations, most proofs have been omitted. For a complete version with all proofs, please refer to [13].

2 Definitions and Notation

We write $[n]$ to mean the set of integers $\{1, \dots, n\}$, and given a set V we write $\binom{V}{2}$ to mean all possible 2-element subsets (i.e., edges) of V. A simple graph is then defined as $G := (V, E)$ where $E \subseteq \binom{V}{2}$. For a vertex $v \in V$ we let $d_G(v) := |\{u : uv \in E\}|$ be the degree of vertex v in G, and $\delta(G) := \min_{v \in V(G)} d_G(v)$ denote the minimum degree in a graph G. If, for all $v \in V$, $d_G(v) = r$ for some integer r, we say that G is r-regular. If the exact value of r is not important, we may just say that G is regular. If, instead, for all $v \in V$, $d_G(v) \in \{r, r+1\}$ for some integer r, we say that G is almost-regular. The *distance* between two vertices u and v in a graph is the length of a shortest path between u and v.

A multi-layer graph $(V, \{E_1, \ldots, E_\tau\})$ consists of a vertex set V and a collection $\{E_1, \ldots, E_\tau\}$, for some integer $\tau \geqslant 1$, of edge sets (or *layers*), where for each i, $E_i \subseteq \binom{V}{2}$. We often slightly abuse terminology and refer to a layer E_i as a graph; when we do this, we specifically refer to the graph (V, E_i) (i.e., we always include every vertex in the original multi-layer graph, even if such a vertex is isolated in (V, E_i)). For instance, we often restrict ourselves to multi-layer graphs where, for each $i \in [\tau]$, the simple graph (V, E_i) is connected. We will say that each layer is *connected* to represent this notion. Given a multi-layer graph $(V, \{E_1, \ldots E_\tau\})$ let the *flattened* version of a multi-layer graph, written as $\mathsf{fl}(\mathcal{G})$, be the simple graph $G = (V, E_1 \cup \cdots \cup E_\tau)$.

Cops and robbers is typically played on a simple graph, with one player controlling some number of cops and the other player controlling the robber. On each turn, the cop player can move none, some, or all of the cops, however each cop can only move along a single edge incident to their current vertex. The robber player can then choose to move the robber along one edge, or have the robber stay still. The goal for the cop player is to end their turn with the robber on the same vertex as at least one cop, while the aim for the robber is to avoid capture indefinitely. If a cop player has a winning strategy on a graph G with k cops but not with $k-1$ cops, we say that the graph G has cop number k, denoted $\mathsf{c}(G) = k$, and that G is k-copwin. Given a multi-layer graph $(V, \{E_1, \ldots, E_\tau\})$, we will say the cop number of layer E_i to mean the cop number of the graph (V, E_i).

As this paper deals with both simple and multi-layer graphs, as well as cops and robbers variants played on these graphs, we will use *single-layer* as an adjective to denote when we are referring to either specifically a simple graph, or to cops and robbers played on a single-layer (i.e., simple) graph. This extends to parameters such as the cop number as well.

In this paper we consider the cops and robbers game on multi-layer graphs and so it will be convenient to define multi-layer graphs with a distinguished layer for the robber. More formally, for an integer $\tau \geqslant 1$, we use the notation $\mathcal{G} = (V, \{C_1, \ldots, C_\tau\}, R)$ to denote a multi-layer graph with vertex set V and collection $\{C_1, \ldots, C_\tau, R\}$ of layers, where $\{C_1, \ldots, C_\tau\}$ are the cop layers and R is the robber layer. In the cops and robber game on $\mathcal{G}$ each cop is allocated to a single-layer from $\{C_1, \ldots, C_\tau\}$, and the robber to R, and each cop (and the robber) will then only move along edges in their respective layer. We do not allow any cop or the robber to move between layers. We note that this is a slight abuse of notation, and that both $(V, \{C_1, \ldots, C_\tau, R\})$ and $(V, \{C_1, \ldots, C_\tau\}, R)$ both denote a multi-layer graphs with the the same collection $\{C_1, \ldots, C_\tau\} \cup \{R\}$ of edge sets, the latter has designated layers for the robber/cops whereas the former does not. We will use E_i to denote edge sets in multi-layer graphs that do not have a cop or robber labels.

A setting that appears often is $R = C_1 \cup \cdots \cup C_\tau$, where the robber can use any edge that exists in a cop layer. This setting is given by the multi-layer graph $\mathcal{G} := (V, \{C_1, \ldots, C_\tau\}, C_1 \cup \cdots \cup C_\tau)$, but for readability we will instead use $\mathcal{G} := (V, \{C_1, \ldots, C_\tau\}, *)$ to denote this.

We define several variants of cops and robbers on multi-layer graphs, however in each of them we have an *allocation* $\mathbf{k} := (k_1, \ldots, k_\tau)$ of cops to layers, such that there are k_i cops on layer C_i. We will often use $k := \sum_i k_i$ to refer to the total number of cops in a game.

We now define multi-layer cops and robbers: a two player game played with an allocation $\mathbf{k}$ on a multi-layer graph $\mathcal{G} = (V, \{C_1, \ldots, C_\tau\}, R)$. The two players are the cop player and the robber player. Each cop is assigned a layer such that there are exactly k_i cops in layer C_i. The game begins with the cop player assigning each cop to some vertex, and then the robber player assigns the robber to some vertex. The game then continues with each player taking turns in sequence, beginning with the cop player. On the cop player's turn, the cop player may move each cop along one edge in that cop's layer. The cop player is allowed to move none, some, or all of the cops. The robber player then takes their turn, either moving the robber along one edge in the robber layer or letting the robber stay on its current vertex. This game ends as a victory for the cop player if, at any point during the game, the robber is on a vertex that is also occupied by one or more cops. The robber wins if they can evade capture indefinitely.

We can now begin defining our problems, starting with ALLOCATED MULTI-LAYER COPS AND ROBBER.

> ALLOCATED MULTI-LAYER COPS AND ROBBER
> *Input:* A tuple $(\mathcal{G}, \mathbf{k})$ where $\mathcal{G} = (V, \{C_1, \ldots, C_\tau\}, R)$ is a multi-layer graph and $\mathbf{k}$ is an allocation of cops to layers.
> *Question:* Does the cop player have a winning strategy when playing multi-layer cops and robbers on $\mathcal{G}$ with allocation $\mathbf{k}$?

We also consider a variant in which the cop player has a given number k of cops, but gets to choose the layers to which the cops are allocated.

> MULTI-LAYER COPS AND ROBBER
> *Input:* A tuple $(\mathcal{G}, k)$ where $\mathcal{G} = (V, \{C_1, \ldots, C_\tau\}, R)$ is a multi-layer graph and $k \geqslant 1$ is an integer.
> *Question:* Is there an allocation $\mathbf{k}$ with $\sum_i k_i = k$ such that $(\mathcal{G}, \mathbf{k})$ is yes-instance for ALLOCATED MULTI-LAYER COPS AND ROBBER?

Lastly we consider MULTI-LAYER COPS AND ROBBER WITH FREE LAYER CHOICE, a variant of MULTI-LAYER COPS AND ROBBER in which, before the game is played, the layers in the multi-layer graph are not assigned to being either cop layers or robber layers. Instead the layers are simply labelled E_1 through E_τ, and in this variant the cop player first allocates each cop to one layer, and then the robber player is free to allocate the robber to any layer.

> Multi-layer cops and robber with free layer choice
> *Input:* A tuple $(\mathcal{G}, k)$ where $\mathcal{G} = (V, \{E_1, \ldots, E_\tau\})$ is a multi-layer graph and $k \geqslant 1$ is an integer.
> *Question:* Is there an allocation $\mathbf{k}$ with $\sum_i k_i = k$ such that for every j, $((V, \{E_1, \ldots, E_\tau\}, E_j), \mathbf{k})$ is a yes-instance for Multi-layer cops and robber?

We say that the multi-layer cop number of a multi-layer graph $\mathcal{G}$ is k if $(\mathcal{G}, k)$ is a yes-instance for Multi-layer cops and robber but $(\mathcal{G}, k-1)$ is a no-instance for Multi-layer cops and robber. We will denote this with $\mathsf{mc}(\mathcal{G})$. We round out this section with a number of basic observations.

Proposition 1. *Let $\mathcal{G} = (V, \{C_1, \ldots, C_\tau\}, R)$ and $\mathcal{G}' = (V, \{C_1, \ldots, C_\tau\}, R')$ be any two multi-layer graphs where $R \subseteq R' \subseteq \binom{V}{2}$. If $(\mathcal{G}, k)$ is a no-instance to* Multi-layer cops and robber, *then $(\mathcal{G}', k)$ is a no-instance to* Multi-layer cops and robber. *Consequently, $\mathsf{mc}(\mathcal{G}) \leqslant \mathsf{mc}(\mathcal{G}')$.*

Proof. To win, the robber on $\mathcal{G}'$ uses the strategy from $\mathcal{G}$. The robber can execute this strategy as any edge in R' is in R. Since the cop layers have no added edges, the strategy must be robber-win as else the cops would win on $\mathcal{G}$. □

Proposition 2. *Let $\mathcal{G} = (V, \{C_1, \ldots, C_\tau\}, R)$ and $\mathcal{G}' = (V, \{C'_1, \ldots, C'_\tau\}, R)$ be any two multi-layer graphs that satisfy $C_i \subseteq C'_i$ for every $i \in [\tau]$. If $(\mathcal{G}, k)$ is a yes-instance to* Multi-layer cops and robber, *then $(\mathcal{G}', k)$ is also a yes-instance to* Multi-layer cops and robber.

Proof. To win, the cops on $\mathcal{G}'$ use the strategy from $\mathcal{G}$. As no edge has been removed from $\mathcal{G}$ to create $\mathcal{G}'$, this must still result in the cops winning. □

Proposition 3. *Let $\mathcal{G} = (V, \{C_1, \ldots, C_\tau\}, *)$ be a multi-layer graph. If $(\mathcal{G}, k)$ is a yes-instance for* Multi-layer cops and robber, *then, letting $E_i = C_i$ for each $i \in [\tau]$, $((V, \{E_1, \ldots, E_\tau\}), k)$ is a yes-instance for* Multi-layer cops and robber with free layer choice.

Proof. Immediate from the problem definitions and Proposition 1. □

3 Counter Examples and Anti-Monotonicity Results

In this section we provide some concrete examples of cops and robbers on multilayer graphs illustrating some peculiarities of the game that may seem counterintuitive. We begin with the following that states that it is sometimes beneficial to put multiple cops on the same layer, and leave other layers empty.

Theorem 1. *For any $n \geqslant 4$ there exists a multi-layer graph $(V, \{C_H, C_V\}, *)$ on n vertices such that a cop player can win with two cops if both cops are on C_H, or if both cops are on C_V, but the robber player can win if one cop is on C_V and the other is on C_H.*

It is natural to ask if, given some multi-layer graph $\mathcal{G} = (V, \{C_1, \ldots, C_\tau\}, R)$, the multi-cop number of $\mathcal{G}$ is bounded from below by the minimum cop-number of a single cop layer; namely, does $\mathsf{mc}(\mathcal{G}) \geqslant \min_i \mathsf{c}((V, C_i))$ hold? Observe that, if $|V| = n$ and we let S_n denote the star graph on n vertices, any multi-layer graph $\mathcal{G} = (V, \{E(S_n), C_2, \ldots, C_\tau\}, R)$ has cop number 1, as the cop can start on the centre of the star and reach any other vertex in one move. This is not enough resolve the question directly, however in the next result we build on this idea to show a general bound of the form $\mathsf{mc}(\mathcal{G}) = \Omega(\min_i \mathsf{c}((V, C_i)))$ does not hold.

Proposition 4. *For any $c \geqslant 2$ there exist graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ such that $\mathsf{c}(G_1), \mathsf{c}(G_2) \geqslant c$ and $\mathsf{mc}((V, \{E_1, E_2\}, *)) = 2$.*

The idea of the proof is to take two n-vertex graphs with cop number c and add a $n-1$ pendent vertices from a single vertex in each graph (u_n and v_n respectively). The graphs are then identified as cop layers in such a way that a cop at u_n can police half the vertices, and a cop at v_n can cover the other half. See Fig. 1 for an illustration. In fact, in such a construction the two cops will catch the robber after at most one cop move. As a result, the edges present in the robber layer are irrelevant and we have the following corollary.

Corollary 1. *For any $c \geqslant 2$ there exist graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ such that $\mathsf{c}(G_1), \mathsf{c}(G_2) \geqslant c$, and for any set of edges $R \subseteq \binom{V}{2}$,*

$$\mathsf{mc}((V, \{E_1, E_2\}, R)) \leqslant 2.$$

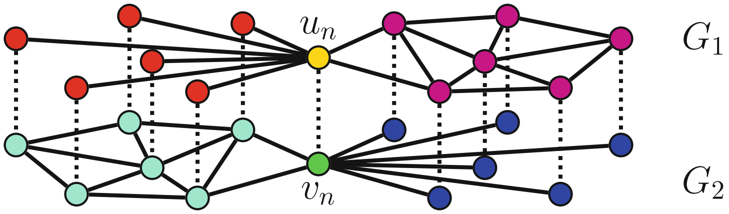


Fig. 1. Illustration of the construction in the proof of Proposition 4. The dotted edges signify an identification of the two end points of that edge.

We now consider the reverse inequality: is the multi-layer cop number bounded from above by a function of the cop numbers of the individual layers? If, in a multi-layer graph $\mathcal{G} = (V, \{C_1, \ldots, C_\tau\}, R)$, the robber layer is a subset of one of the cop layers, i.e. $R \subset C_i$ for some $i \in [\tau]$, then $\mathsf{mc}(\mathcal{G}) \leqslant \mathsf{c}((V, C_i))$ as the cop player can allocate $\mathsf{c}((V, C_i))$ cops to layer i, ignoring all other cop layers. The same reasoning gives an upper bound of $\sum_{i \in [\tau]} \mathsf{c}((V, C_i))$ on the cop number in the 'free choice layer' variant of the game. Thus, in this special case an upper bound that depends only on the cop numbers of individual layers does exist. However the next result shows that this is not the case in general.

Theorem 2. *For any positive integer k, there exists a multi-layer graph $\mathcal{G} = (V, \{C_1, C_2\}, R)$ on $O(k^3)$ vertices such that each of (V, R), (V, C_1), and (V, C_2) are connected,* $\mathsf{c}((V, R)) \leqslant 3$, $\mathsf{c}((V, C_i)) \leqslant 2$ *for* $i \in \{1, 2\}$, *and* $\mathsf{mc}(\mathcal{G}) \geqslant k$.

4 Complexity Results

In this section we will examine multi-layer cops and robbers from a computational complexity viewpoint. For a background on computational complexity, we point the reader to [30]. First note that as determining the cop-number of a simple graph is `EXPTIME`-complete [18], Multi-layer cops and robber with free layer choice is also `EXPTIME`-complete by the obvious reduction that creates a multi-layer graph with one layer from a simple graph. The same reduction, and the fact that, unless the strong exponential time hypothesis fails[1], determining if a graph is k-copwin requires $\Omega(n^{k-o(1)})$ time [9], we also get that Multi-layer cops and robber with free layer choice requires $\Omega(n^{k-o(1)})$ time.

An algorithm that determines whether a simple graph G is k-copwin in $O(kn^{k+2})$ time is given in [23]. Petr, Portier, and Versteegen show this by first constructing a *state graph* — a directed graph H wherein each vertex of H corresponds to a state of a game of cops and robbers played on the original graph G. They then give an $O(kn^{k+2})$ algorithm for finding all cop-win vertices of H, where a vertex is cop-win if the corresponding state either is a winning state for the cops, or can only lead to a winning state for the cops. We adapt their construction by only creating arcs of H where the move of a given cop or robber is allowed (i.e., the edge in the multi-layer graph exists in the same layer as the cop or robber that is moving). By doing this we obtain the following.

Theorem 3. Allocated multi-layer cops and robber *can be solved in* $O(k^2 n^{2k+2})$.

Note that τ, the number of layers, does not appear in the above as if $\tau \leqslant k$ then any dependence on τ is absorbed by the dependence on k, and if $k < \tau$ then at least $\tau - k$ layers must have zero cops allocated to them and can be ignored.

By taking an instance of dominating set $G = (V, E)$, and creating for each vertex $v \in V$ a layer E_v containing every edge incident to v, we create an instance of Multi-layer cops and robber with free layer choice that has a winning strategy for k cops if and only if G has a dominating set of size k, leading to the following.

Theorem 4. Multi-layer cops and robber with free layer choice *is* `NP`*-hard, even if each layer is a tree plus some isolated vertices.*

Note that the input size to Multi-layer cops and robber with free layer choice is the number of bits required to represent both the underlying graph and each of the layers.

[1] See [12, Chapter 14] for background on the strong exponential time hypothesis.

If it is only the robber that is limited to a tree, however, then determining if k cops can win is FPT in the number of cops and number of layers in the graph. In particular, this result applies even if the layers are not connected. We obtain this result by characterising whether the robber can win based on the existence of edges in the robber layer that cops can easily patrol.

Theorem 5. *Given a multi-layer graph $\mathcal{G} = (V, \{C_1, \ldots, C_\tau\}, R)$, if R is a tree, then* Multi-layer cops and robber *on $\mathcal{G}$ can be solved in time $O(f(k, \tau) \cdot \texttt{poly}(n))$, where k is the number of cops, τ is the number of layers of $\mathcal{G}$, f is a computable function independent of n, and $\texttt{poly}(n)$ is a fixed polynomial in n.*

The next result follows immediately from the proof technique used to prove Theorem 5.

Corollary 2. *Given a multi-layer graph $\mathcal{G} = (V, \{C_1, \ldots, C_\tau\}, R)$, if R is a tree, and each cop layer is connected, then $\mathsf{mc}(\mathcal{G}) \leqslant 2$.*

5 Extremal Multi-Layer Cop-Number

In this section we study, for a given simple connected graph $G = (V, E)$, the extremal multi-layer cop number of G. This is the multi-layer cop number maximised over the set of all multi-layer graphs with connected cop-layers, which when flattened give G. More formally, for given connected graph $G = (V, E)$, if we define the set

$$\mathcal{L}(G) = \{(V, \{C_1, \ldots, C_\tau\}, *) : E = C_1 \cup \cdots \cup C_\tau \text{ and for each } i \in [\tau], (V, C_i) \text{ is connected}\},$$

then the extremal multi-layer cop-number of G is given by

$$\mathsf{emc}_\tau(G) = \max_{\mathcal{G} \in \mathcal{L}} \mathsf{mc}(\mathcal{G})\,.$$

We generalise two tools for bounding the cop number of graphs to the setting of multi-layer graphs; $(1, k)$-existentially closed graphs [7] and bounds by domination number. See the arXiv version of this paper [13] for more details on the former method; here we will now outline our use of dominating sets.

Let $\mathcal{G} = (V, \{E_1, \ldots, E_\tau\})$ be a multi-layer graph (without designated layers). A *multi-layer dominating set* in $\mathcal{G}$ is a set $D \subseteq V \times \{1, \ldots, \tau\}$ of vertex-layer pairs such that for every $v \in V$, either $(v, i) \in D$ for some i, or there is a $(w, i) \in D$ such that $w \in V$ and $vw \in E_i$. We define the *domination number* $\gamma(\mathcal{G})$ of $\mathcal{G}$ to be the size of a smallest multi-layer dominating set in $\mathcal{G}$. Note that if $\mathcal{G}$ has a single-layer this definition aligns with the traditional notion of dominating set, which justifies the overloaded notation. It is a folklore result that the cop number is at most the size of any dominating set in the graph, this also holds in the multi-layer setting.

Theorem 6. *Let $\mathcal{G} := (V, \{E_1, \ldots, E_\tau\})$ be any multi-layer graph and $\mathcal{G}' := (V, \{E_1, \ldots, E_\tau\}, \binom{V}{2})$. Then, $\mathsf{mc}(\mathcal{G}') \leqslant \gamma(\mathcal{G})$.*

Proof. Let D be any multi-layer dominating set of size $|D| = \gamma(\mathcal{G})$ and for each $(v,i) \in D$ place one cop in layer i at the vertex v. The result now follows as if the robber is at an any vertex then they are adjacent to a cop in some layer and so the robber will be caught after the cops first move. □

We now introduce the parameter $\delta(\mathcal{G})$ which is an analogue of minimum degree for a multi-layer graph $\mathcal{G} = (V, \{E_1, \ldots, E_\tau\})$. This is given by

$$\delta(\mathcal{G}) := \min_{v \in V} \sum_{i \in [\tau]} d_{(V,E_i)}(v). \tag{1}$$

Using this notion we prove a bound on the domination number of a multi-layer graph, the proof is based on a classic application of the probabilistic method [2, Theorem 1.2.2].

Theorem 7. *Let $\mathcal{G} = (V, \{E_1, \ldots, E_\tau\})$ be any multi-layer graph. Then,*

$$\gamma(\mathcal{G}) \leqslant \frac{n\tau}{\tau + \delta(\mathcal{G})} \cdot \left(\ln\left(\frac{\tau + \delta(\mathcal{G})}{\tau} \right) + 1 \right).$$

Note that there are least two other sensible definitions of 'multi-layer minimum degree', namely the minimum degree of each layer $\min_{i \in [\tau]} \min_{v \in V} d_{(V,E_i)}(v)$, and minimum number of neighbours within any layer $\delta(\mathsf{fl}(\mathcal{G}))$. Our definition of $\delta(\mathcal{G})$ above in (1) can be thought of as the 'minimum number of edges incident in any layer', this is arguably a less natural notion than $\delta(\mathsf{fl}(\mathcal{G}))$ however it gives a better bound in our application (Theorem 7), in particular.

Proposition 5. *For any multi-layer graph $\mathcal{G}$ we have $\delta(\mathsf{fl}(\mathcal{G})) \leqslant \delta(\mathcal{G})$.*

Returning to extremal multi-layer cop numbers, for a complete graph we obtain Theorem 8. The upper bound we arrive at by placing all τ cops on a single vertex v; as each edge of K_n must be in some layer, there is no vertex that is not incident with v in some layer. The lower bound requires more work and relies on constructing cop layers with no overlap by combining colour classes of an edge colouring of the clique due to Sofier [31].

Theorem 8. *Let $n \geqslant 1$, $1 \leqslant \tau < \lfloor \frac{n}{2} \rfloor$ be integers. Then, $\lceil \frac{\tau}{10} \rceil \leqslant \mathsf{emc}_\tau(K_n) \leqslant \tau$.*

We now consider the extremal multi-layer cop number of the binomial random graph $G_{n,p}$. For any integer $n \geqslant 1$, this is the probability distribution over all n-vertex simple graphs generated by sampling each possible edge independently with probability $0 < p = p(n) < 1$, see [4] for more details. The following result shows that, for a suitably dense binomial random graph $G_{n,p}$, with probability tending to 1 as $n \to \infty$, $\mathsf{emc}_\tau(G_{n,p}) = \Theta(\tau \log(n)/p)$. The single-layer cop number of $G_{n,p}$ in the same range is known to be $\Theta(\log(n)/p)$ [7], so in some sense our result generalises this result.

Theorem 9. *For $\varepsilon > 0$, if $n^{1/2+\varepsilon} \leqslant np = o(n)$, and $1 \leqslant \tau \leqslant n^{\varepsilon}$ then,*

$$\mathbb{P}\left(\frac{\varepsilon}{10} \cdot \frac{\tau \cdot \ln n}{p} \leqslant \mathsf{emc}_{\tau}(G_{n,p}) \leqslant 10 \cdot \frac{\tau \cdot \ln n}{p}\right) \geqslant 1 - e^{-\Omega(\sqrt{n})}.$$

The upper bound in the proof of Theorem 9 follows from Theorems 6 and 7, whereas the lower bound follows by independently choosing cop layers which are each distributed as a random graph with edge probability $\Theta(p/\tau)$ and then applying a generalised form of the existential closure technique developed in [7]. See [24] for results on the cop number of $G_{n,p}$ for other ranges of p.

The extremal multi-layer cop number of a graph G is also bounded from above by the treewidth of G.

Theorem 10. *For any graph $G := (V, E)$, $\mathsf{emc}_{\tau}(G) \leqslant \mathsf{tw}(G)$. Furthermore, these cops can placed in any layers and still capture the robber.*

6 Multi-Layer Analogue of Meyniel's Conjecture

For the classical cop number, Meyniel's Conjecture [15] states that $\mathcal{O}(\sqrt{n})$ cops are sufficient to win cops and robbers on any graph G. After a sequence of results [10,15,16,20,28] the current best bound stands at $n \cdot 2^{-(1-o(1))\sqrt{\log_2 n}}$, see [5, Chapter 3] for a more detailed overview.

It is natural to explore analogues of Meyniel's Conjecture for the multi-layer cop number. Namely, what is the minimum number of cops needed to patrol any multi-layer graph with τ connected layers? Our results for the clique show that if τ is allowed to be arbitrary then no bound better than $\mathcal{O}(n)$ can hold. We conjecture that this is not the case when the number of layers is bounded.

Conjecture 1. For any fixed integer $\tau \geqslant 1$ and collection $(V, E_1), \ldots, (V, E_{\tau})$ of connected graphs where $|V| = n$, we have

$$\mathsf{mc}((V, \{E_1, \ldots, E_{\tau}\}, *)) = o(n).$$

Observe that the connected assumption is necessary in Conjecture 1 if we do not have divergent minimum degree, as shown by the example with two cop layers given by two edge disjoint matchings who's union forms an even cycle. This conjecture might seem very modest in comparison to Meyniel's conjecture, however the following result shows that it would be almost tight.

Theorem 11. *For any positive integer n there is a n-vertex multi-layer graph $\mathcal{G} = (V, \{C_1, C_2, C_3\}, *)$ such that $|V| = \Theta(n)$, each cop layer is connected and has cop-number 2, and*

$$\mathsf{mc}(\mathcal{G}) = \Omega\left(\frac{n}{\log n}\right).$$

The construction in Theorem 11 starts with a 3-edge coloured cubic expander graph X on N vertices, where each color class is a cop layer. The vertices of X are then connected to the leaves of a star that has been subdivided $\Theta(\log N)$

many times — these can be used by all cops. The idea is that k cops can police at most $2k$ vertices of X within $\Theta(\log N)$ steps as it takes each cop this long to change location in X (via the arms of the star). If $k = \Theta(N)$ is chosen to be a suitably small but constant fraction of N, then even with the vertices policed by the cops removed there is still an expander subgraph of X not adjacent to any cops. The robber can then use this expander subgraph to change position before any cop can threaten them.

Many of the current approaches to Meyniel's Conjecture use some variation the fact that a single cop can guard any shortest path between any two vertices. For example the first step of the approach in [28] is to iteratively remove long geodesics until the graph has small diameter (following this a more sophisticated argument matching randomly placed cops to possible robber trajectories is applied). What makes Conjecture 1 difficult to approach is that, even for two layers, a shortest path in the flattened graph $\mathsf{fl}(\mathcal{G})$ may not live within a single cop layer. We note that [16] use a different approach based on expansion, their approach is more versatile however the authors were unable to apply it in the multi-layer setting.

This suggests a new or more refined approach is needed. However, by a direct application of Theorems 6 and 7, using a simple dominating set approach we can prove Conjecture 1 for multi-layer graphs with diverging minimum degree.

Proposition 6. *For any n-vertex multi-layer graph $\mathcal{G} = (V, \{E_1, \dots, E_\tau\})$ satisfying $\delta(\mathcal{G})/\tau \to \infty$ as $n \to \infty$, we have $\mathsf{mc}\Big(\big(V, \{E_1, \dots, E_\tau\}, \binom{V}{2}\big)\Big) = o(n)$.*

7 Conclusion and Open Problems

We studied the game of cops and robbers on multi-layer graphs, via several different approaches, including concrete strategies for certain graphs, the construction of counter-intuitive examples, algorithmic and hardness results, and the use of probabilistic methods and expanders for extremal constructions. We find that the multi-layer cop number cannot be bound from above or below by (non-constant) functions of the cop numbers of the individual layers. We bound an extremal variant for cliques and dense binomial random graphs (extending some tools from the single layer case along the way). We also find that a naive transfer of Meyniel's conjecture to the multi-layer setting is not true: there are multi-layer graphs which have multi-layer cop number in $\Omega(n/\log n)$. Algorithmically, we find that even if each layer is a tree plus some isolated vertices, the free layer choice variant of the problem remains NP-hard. Positively, we find that the problem can be resolved by an algorithm that is FPT in the number of cops and layers if the layer the robber resides in is a tree.

We are hopeful that our contribution will spark future work in multi-layer variants of cops and robbers, and suggest a number of possible open questions:

We were not able to generalise some frequently used tools from single-layer cops and robbers: for example, we have no useful notion of a corner, or a retract, nor dismantleability - we are hopeful that such tools may exist.

We have made some progress on the parameterised complexity of our problems, but have only considered a limited set of parameters and have not considered any parameter that constraints the nature of interaction between the layers: if we, for example, require that the layers are very similar alongside other restrictions does that impact the computational complexity of our problems?

Single-layer cops and robbers has been very successful as a tool for defining useful graph parameters of simple graphs, and we ask whether multi-layer cops and robbers could be used to define algorithmically useful graph parameters.

Some of our bounds and extremal results are unlikely to be tight in number of layers or with respect to other graph characteristics: can they be improved?

Is the extremal multi-layer cop number of $G_{n,p}$ always $\Theta(\tau \cdot \mathsf{c}(G_{n,p}))$ w.h.p. for any p? Of particular is whether this holds even in the 'zig-zag' regime [21]?

While we showed that a naive adaptation of Meyniel's conjecture to our multi-layer setting fails, it is still possible that $o(|V|)$ cops are sufficient for a bounded number of connected layers. We have shown this for a special case related to degree: is it true in general?

Finally, while we introduced a particular notion of multi-layer dominating set for our use in proving other results (inspired by similar ideas in single-layer cops and robbers), we suggest that this multi-layer graph characteristic may also be interesting in its own right, in particular for algorithms for other problems on multi-layer graphs.

Acknowledgements. This work was supported by the Engineering and Physical Sciences Research Council [EP/T004878/1].

References

1. Aigner, M., Fromme, M.: A game of cops and robbers. Discret. Appl. Math. **8**(1), 1–12 (1984)
2. Alon, N., Spencer, J.H.: The Probabilistic Method, Third Edition. Wiley-Interscience series in discrete mathematics and optimization. Wiley, Hoboken (2008)
3. Balev, S., Laredo Jiménez, J.L., Lamprou, I., Pigné, Y., Sanlaville, E.: Cops and robbers on dynamic graphs: offline and online case. In: Richa, A.W., Scheideler, C. (eds.) SIROCCO 2020. LNCS, vol. 12156, pp. 203–219. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-54921-3_12
4. Bollobás, B.: Random graphs, volume 73 of Cambridge Studies in Advanced Mathematics, second edition. Cambridge University Press, Cambridge (2001)
5. Bonato, A., Nowakowski, R.J.: The Game of Cops and Robbers on Graphs. Student Mathematical Library. American Mathematical Society, New York (2011)
6. Bonato, A., Pralat, P.: Graph Searching Games and Probabilistic Methods. Discrete Mathematics and Its Applications. CRC Press, London, England, December 2017

7. Bonato, A., Pralat, P., Wang, C.: Pursuit-evasion in models of complex networks. Internet Math. **4**(4), 419–436 (2007)
8. Bowler, N.J., Erde, J., Lehner, F., Pitz, M.: Bounding the cop number of a graph by its genus. SIAM J. Discret. Math. **35**(4), 2459–2489 (2021)
9. Brandt, S., Pettie, S., Uitto, J.: Fine-grained lower bounds on cops and robbers. In: Azar, Y., Bast, H., Herman, G. (eds.), 26th Annual European Symposium on Algorithms (ESA 2018), volume 112 of Leibniz International Proceedings in Informatics (LIPIcs), pp. 9:1–9:12, Dagstuhl, Germany, 2018. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2018)
10. Chiniforooshan, E.: A better bound for the cop number of general graphs. J. Graph Theory **58**(1), 45–48 (2008)
11. Clarke, N.E.B.: Constrained cops and robber. ProQuest LLC, Ann Arbor, MI, 2002. Thesis (Ph.D.)-Dalhousie University (Canada) (2002)
12. Cygan, M., et al.: Parameterized Algorithms. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3
13. Enright, J., Meeks, K., Pettersson, W., Sylvester, J.: Cops and robbers on multi-layer graphs. arXiv:2303.03962 (2023)
14. Fitzpatrick, S.L.: Aspects of domination and dynamic domination. ProQuest LLC, Ann Arbor, MI, 1997. Thesis (Ph.D.)-Dalhousie University (Canada) (1997)
15. Frankl, P.: Cops and robbers in graphs with large girth and Cayley graphs. Discret. Appl. Math. **17**(3), 301–305 (1987)
16. Frieze, A.M., Krivelevich, M., Loh, P.-S.: Variations on cops and robbers. J. Graph Theory **69**(4), 383–402 (2012)
17. Joret, G., Kaminski, M., Theis, D.O.: The cops and robber game on graphs with forbidden (induced) subgraphs. Contrib. Discret. Math. **5**(2) (2010)
18. Kinnersley, W.B.: Cops and robbers is exptime-complete. J. Comb. Theory Ser. B **111**, 201–220 (2015)
19. Lehner, F.: On the cop number of toroidal graphs. J. Comb. Theory, Ser. B **151**, 250–262 (2021)
20. Linyuan, L., Peng, X.: On Meyniel's conjecture of the cop number. J. Graph Theory **71**(2), 192–205 (2012)
21. Luczak, T., Pralat, P.: Chasing robbers on random graphs: zigzag theorem. Random Struct. Algorithms **37**(4), 516–524 (2010)
22. Nowakowski, R.J., Winkler, P.: Vertex-to-vertex pursuit in a graph. Discret. Math. **43**(2–3), 235–239 (1983)
23. Petr, J., Portier, J., Versteegen, L.: A faster algorithm for cops and robbers. Discret. Appl. Math. **320**, 11–14 (2022)
24. Pralat, P., Wormald, N.C.: Meyniel's conjecture holds for random graphs. Random Struct. Algorithms **48**(2), 396–421 (2016)
25. Quilliot, A.: Jeux et pointes fixes sur les graphes. PhD thesis, Ph. D. Dissertation, Université de Paris VI (1978)
26. Quilliot, A.: A short note about pursuit games played on a graph with a given genus. J. Comb. Theory Ser. B **38**(1), 89–92 (1985)
27. Schroeder, B.S.W.: The copnumber of a graph is bounded by $\lfloor \frac{3}{2}$ genus $(G) \rfloor + 3$. In: Categorical perspectives (Kent, OH, 1998), Trends Math., pp. 243–263. Birkhäuser Boston, Boston, MA (2001)
28. Scott, A., Sudakov, B.: A bound for the cops and robbers problem. SIAM J. Discret. Math. **25**(3), 1438–1442 (2011)
29. Seymour, P.D., Thomas, R.: Graph searching and a min-max theorem for tree-width. J. Comb. Theory Ser. B **58**(1), 22–33 (1993)

30. Sipser, M.: Introduction to the Theory of Computation. Cengage Learning, Boston (2012)
31. Soifer, A.: The Mathematical Coloring Book. Springer, New York (2009). https://doi.org/10.1007/978-0-387-74642-5
32. Toruńczyk, S.: Flip-width: cops and robber on dense graphs. In: 2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS 2023), page to appear. IEEE (2023)

Parameterized Complexity of Broadcasting in Graphs

Fedor V. Fomin[1], Pierre Fraigniaud[2], and Petr A. Golovach[1](✉)

[1] Department of Informatics, University of Bergen, Bergen, Norway
{Fedor.Fomin,Petr.Golovach}@uib.no
[2] Institut de Recherche en Informatique Fondamentale,
Université Paris Cité and CNRS, Paris, France
pierre.fraigniaud@irif.fr

Abstract. The task of the broadcast problem is, given a graph G and a source vertex s, to compute the minimum number of rounds required to disseminate a piece of information from s to all vertices in the graph. It is assumed that, at each round, an informed vertex can transmit the information to at most one of its neighbors. The broadcast problem is known to be NP-hard. We show that the problem is FPT when parametrized by the size k of a feedback edge set, or by the size k of a vertex cover, or by $k = n - t$, where t is the input deadline for the broadcast protocol to complete.

Keywords: broadcasting · telephone model · parameterized complexity

1 Introduction

The aim of *broadcasting* in a network is to transmit a message from a given source node of the network to all the other nodes. Let $G = (V, E)$ be a connected simple graph modeling the network, and let $s \in V$ be the source of a message M. The standard *telephone model* [21] assumes that the communication proceeds in synchronous rounds. At any given round, any node $u \in V$ aware of M can forward M to at most one neighbor v of u. The minimum number of rounds for broadcasting a message from s in G to all other vertices is denoted by $b(G, s)$, and we let $b(G) = \max_{s \in V} b(G, s)$ be the broadcast time of graph G. As the number of informed nodes (i.e., nodes aware of the message) can at most double at each round, $b(G, s) \geq \lceil \log_2 n \rceil$ in n-node networks. On the other hand, since G is connected, at least one uninformed node receives the message at any given round, and thus $b(G) \leq n - 1$. Both bounds are tight, as witnessed by the complete graph K_n and the path P_n, respectively. The problem of computing

The research leading to these results has received funding from the Research Council of Norway via the project BWCA (grant no. 314528) and from the ANR project DUCAT. The full version of the paper is available in [12].

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 334–347, 2023.
https://doi.org/10.1007/978-3-031-43380-1_24

the broadcast time $b(G, s)$ for a given graph G and a given source $s \in V$ is NP-hard [30]. Also, the results of [27] imply that it is NP-complete to decide whether $b(G, s) \leq t$ for graphs with $n = 2^t$ vertices.

Three lines of research have emerged since the early days of studying broadcasting in the telephone model. One line is devoted to determining the broadcast time of specific classes of graphs judged important for their desirable properties as interconnection networks (e.g., hypercubes, de Bruijn graphs, Cube Connected Cycles, etc.). We refer to the surveys [15,22] for this matter. Another line of research takes inspiration from extremal graph theory. It aims at constructing n-node graphs G with optimal broadcast time $b(G) = \lceil \log_2 n \rceil$ and minimizing the number of edges sufficient to guarantee this property. Let $B(n)$ be the minimum number of edges of n-node graphs with broadcast time $\lceil \log_2 n \rceil$. It is known [18] that $B(n) = \Theta(n\, L(n))$ where $L(n)$ denotes the number of consecutive leading 1 s in the binary representation of $n - 1$. On the other hand, it is still not known whether $B(\cdot)$ is non-decreasing for $2^t \leq n < 2^{t+1}$, for every $t \in \mathbb{N}$. We are interested in a third, more recent line of research, namely the design of algorithms computing efficient broadcast protocols. Note that a protocol for broadcasting from a source s in a graph G can merely be represented as a spanning tree T rooted at s, with an ordering of all the children of each node in the tree.

Polynomial-time algorithms are known for trees [30] and some classes of tree-like graphs [4,17,20]. Several (polynomial-time) approximation algorithms have been designed for the broadcast problem. In particular, the algorithm in [25] computes, for every graph G and every source s, a broadcast protocol from s performing in $2\, b(G, s) + O(\sqrt{n})$ rounds, hence this algorithm has approximation ratio $2 + o(1)$ for graphs with broadcast time $\gg \sqrt{n}$, but $\tilde{\Theta}(\sqrt{n})$ in general. Later, a series of papers tighten the approximation ratio, from $O(\log^2 n/ \log\log n)$ [29], to $O(\log n)$ [1], and eventually $O(\log n/ \log\log n)$ [10], which is, up to our knowledge the current best approximation ratio for the broadcast problem. Better approximation ratios can be obtained for specific classes of graphs [2,3,19].

Despite all the achievements obtained on the broadcast problem, it has not yet been approached from the parameterized complexity viewpoint [8]. There might be a good reason for that. Since at most 2^t vertices can have received the message after t communication rounds, an instance of the broadcast problem with time-bound t in an n-vertex graph is a no-instance whenever $n > 2^t$. It follows that the broadcast problem has a trivial kernel when parameterized by the broadcast time. This makes the natural parameterization by the broadcast time not very significant. Nevertheless, as we shall show in this paper, there is a parameterization below the natural upper bound for the number of rounds that leads to interesting conclusions.

Our Results. Let Telephone Broadcast be the following problem: given a connected graph $G = (V, E)$, a source vertex $s \in V$, and a nonnegative integer t, decide whether there is a broadcast protocol from s in G that ensures that all the vertices of G get the message in at most t rounds. We first show that Telephone Broadcast can be solved in a single-exponential time by an exact algorithm. Our algorithm is based on the dynamic programming over subsets [13].

Theorem 1 **($\star$[1]).** Telephone Broadcast *can be solved in* $3^n \cdot n^{\mathcal{O}(1)}$ *time for* n*-vertex graphs.*

Motivated by the fact that the complexity of Telephone Broadcast remains open in pretty simple tree-like graphs (e.g., cactus graphs, and therefore outerplanar graphs), we first consider the *cyclomatic* number as a parameter, i.e., the minimum size of a feedback edge set, that is, the size of the smallest set of edges whose deletion results to an acyclic graph. We show that Telephone Broadcast is FPT when parameterized by this parameter.

Theorem 2. Telephone Broadcast *can be solved in* $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ *time for* n*-vertex graphs with cyclomatic number at most* k*.*

As far as we know, no NP-hardness result is known on graphs of treewidth at most $k \geq 2$. While we did not progress in that direction, we provide an interesting result for a stronger parameterization, namely the *vertex cover* number of a graph. (Note that, for all graphs, the treewidth never exceeds the vertex cover number.) As for the cyclomatic number, we do not only show that, for every fixed k, the broadcast time can be found in polynomial time on graphs with vertex cover at most k, but we prove a stronger result: the problem is FPT.

Theorem 3. Telephone Broadcast *can be solved in* $2^{\mathcal{O}(k2^k)} \cdot n^{\mathcal{O}(1)}$ *time for* n*-vertex graphs with a vertex cover of size at most* k*.*

Finally, we focus on graphs with very large broadcast time, for which the algorithm in [25] provides hope to derive very efficient broadcast protocols as this algorithm constructs a broadcast protocol performing in $2\,b(G) + O(\sqrt{n})$ rounds. While we were not able to address the problem over the whole range $\sqrt{n} \ll t \leq n-1$, we were able to provide answers for the range $n - O(1) \leq t \leq n-1$. More specifically, we consider the parameter $k = n - t$ and study the kernelization for the problem under such parametrization.

Theorem 4. Telephone Broadcast *admits a kernel with* $\mathcal{O}(k)$ *vertices in* n*-vertex graphs when parameterized by* $k = n - t$*.*

As a direct consequence of Theorem 4, Telephone Broadcast is FPT for the parameterization by $k = n - t$. Specifically the problem can be solved in $2^{O(k)} \cdot n^{O(1)}$ time.

Related Work. A classical generalization of the broadcast problem is the *multicast* problem, in which the message should only reach a given subset of target vertices in the input graph. Many of the previously mentioned approximation algorithms for the broadcast problem extend to the multicast problem, and, in particular, the algorithm in [10] is an $O(\log k / \log\log k)$-approximation algorithm for the multicast problem with k target nodes.

[1] The proofs of the statements labeled by ($\star$) are omitted and can be found in the full version of the paper [12].

Many variants of the telephone model have been considered in the literature, motivated by different network technologies. One typical example is the *line* model [11], in which a call between a vertex u and a vertex v is implemented by a path between u and v in the graph, with the constraint that all calls performed at the same round must be performed along edge-disjoint paths. (The intermediate nodes along the path do *not* receive the message, which "cut through" them.) Interestingly, the broadcast time of *every* n-node graph is exactly $\lceil \log_2 n \rceil$. The result extends to networks in which the paths are constructed by an underlying routing function [6]. The vertex-disjoint variant of the line model, i.e., the line model in which the calls performed at the same round must take place along vertex-disjoint paths, is significantly more complex. There is an $O(\log n / \log\log n)$-approximation algorithm for the vertex-disjoint line model [25], which naturally extend to an $O(\log n / \log OPT)$-approximation algorithm — see also [14] where an explicit $O(\log n / \log OPT)$-approximation algorithm is provided. It is also worth mentioning that the broadcast model has been also extensively studied in models aiming at capturing any type of node- or link-latencies, e.g., the message takes λ_e units of time to traverse edge e, and the algorithm in [1] also handles such constraints. Other variants take into account the size of the message, e.g., a message of L bits takes time $\alpha + \beta \cdot L$ to traverse an edge (see [23]). Under such a model, it might be efficient to split the original message into smaller packets and pipeline the broadcast of these packets through disjoint spanning trees [23,31].

2 Preliminaries

We refer to the book of Cygan et al. [8] for a detailed introduction to Parameterized Complexity. We consider only finite undirected graphs and refer to the textbook of Diestel [9] for basic notation. We always assume that the considered graphs are connected if it is not explicitly said to be otherwise. We use n and m to denote the number of vertices and edges if this does not create confusion. A set of edges S of a graph G is a *feedback edge set* if $G - S$ has no cycle. The *cyclomatic number* of a graph G is the minimum size of a feedback edge set. It is well-known, that for a connected graph G, the cyclomatic number is $m - n + 1$ and a feedback edge set can be found in linear time by constructing a spanning tree (see, e.g., [7,9]). A set of vertices S of a graph G is a *vertex cover* if each edge of G has at least one of its endpoints in G. The *vertex cover number* of G is the minimum size of a vertex cover. Note that for a vertex cover S, the set $I = V(G) \setminus S$ is an *independent set*, that is, any two distinct vertices of I are not adjacent.

Broadcasting. Let G be a graph and let $s \in V(G)$ be a source vertex from which a message is broadcasted. In general, a broadcasting protocol is a mapping that for each round $i \geq 1$, assigns to each vertex $v \in V(G)$ that is either a source or has received the message in rounds $1, \ldots, i-1$, a neighbor u to which v sends the message in the i-th round. However, it is convenient to note that it can be assumed that each vertex v that got the message, in the next $d \leq d_G(v)$ rounds, transmits the message to some neighbors in a certain order in such a way that each vertex receives the message only once. This allows us to formally define a *broadcasting protocol* as a pair $(T, \{C(v) \mid v \in V(T)\})$, where T is a spanning tree of G rooted in s and for each $v \in V(T)$, $C(v)$ is an ordered set of children of v in T. As soon as v gets the message, v starts to send it to the children in T in the order defined by $C(v)$. For G and $s \in V(G)$, we use $b(G, s)$ to denote the minimum integer $t \geq 0$ such that there is a broadcasting protocol such that every vertex of G gets the message after t rounds. We say that a broadcasting protocol ensuring that every vertex gets a message in $b(s, G)$ rounds is *optimal.*

As it was proved by Proskurowski [28] and Slater, Cockayne, and Hedetniemi [30], $b(G, s)$ can be computed in linear time for trees by dynamic programming.

Lemma 1 **([28,30]).** *For an n-vertex tree T and $s \in V(T)$, $b(T, s)$ can be computed in $\mathcal{O}(n)$ time.*

3 Telephone Broadcast Parameterized by the Cyclomatic Number

In this section, we sketch the proof of Theorem 2. We need some auxiliary results. Let T be a tree and let x and y be distinct leaves, that is, vertices of degree one in T. For an integer $h \geq 0$, we use $b_h(T, x, y)$ to denote the minimum number of rounds needed to broadcast the message from the source x to y in such a way that every vertex of T gets the message in at most h rounds. We assume that $b_h(T, x, y) = +\infty$ if $b(T, x) > h$. We prove that $b_h(T, x, y)$ can be computed in linear time similarly to $b(T, s)$ (see [28,30]). The difference is that it is more convenient to use recursion instead of dynamic programming.

Lemma 2 **($\star$).** *For an n-vertex tree T with given distinct leaves x and y of T and an integer $h \geq 0$, $b_h(T, x, y)$ can be computed in $\mathcal{O}(n)$ time.*

We also need a subroutine computing the minimum number of rounds for broadcasting from two sources with the additional constraint that the second source starts sending the message with a delay. Let T be a tree and let x and y be distinct leaves of T. Let also $h \geq 0$ be an integer. We use $d_h(T, x, y)$ to denote the minimum rounds needed to broadcast the message from x and y to every vertex of $T - y$ in such a way that y can send the message starting from the $(h + 1)$-th round (that is, we assume that y gets the message from outside in the h-th round).

Lemma 3 **($\star$).** *For an n-vertex tree T with given distinct leaves x and y of T and an integer $h \geq 0$, $d_h(T, x, y)$ can be computed in $\mathcal{O}(n^2)$ time.*

Sketch of the Proof of Theorem 2. Let (G, s, t) be an instance of Telephone Broadcast. If G is a tree, then we can compute $b(G, s)$ in linear time using

Lemma 1. Assume that this is not the case, and let $k = m - n + 1 \geq 1$ be the cyclomatic number of G. We find in linear time a feedback edge set S of size k by finding an arbitrary spanning tree F of G and setting $S = E(G) \setminus E(F)$.

We iteratively construct the set U as follows. Initially, we set $U := W = \{s\} \cup \{v \in V(G) \mid v \text{ is an endpoint of an edge of } S\}$. Then while $G - U$ has a vertex v such that G has three internally vertex-disjoint paths joining v and U, we set $U := U \cup \{v\}$. The properties of U are summarized in the following claims.

Claim 1. *We have that $|U| \leq 4k$, and for each connected component F of $G-U$, F is a tree such that each vertex $x \in U$ has at most one neighbor in F and*

(i) either U has a unique vertex x that has a neighbor in F,
(ii) or U contain exactly two vertices x and y having neighbors in F.

If F is a connected component of $G-U$ satisfying (i) of Claim 1, then we say that F is a x-tree and x is its *anchor*. For a connected component F of $G-U$ satisfying (ii), we say that F is an (x, y)*-tree* and call x and y *anchors* of F. We also say that F is *anchored* in x (x and y, respectively). Because $G - S$ is a tree and $|U| \leq 4k - 1$, we immediately obtain the next property.

Claim 2. *For every distinct $x, y \in U$, $G - U$ has at most one (x, y)-tree. Furthermore, the graph H with $V(H) = U$ such that $xy \in E(H)$ if and only if $G-U$ has an (x, y)-tree is a forest. In particular, the total number of (x, y)-trees is at most $4k - 1$.*

To prove the theorem, we have to verify the existence of a broadcasting protocol $P = (T, \{C(v) \mid v \in V(T)\})$ that ensures that every vertex receives the message after at most t rounds. To do it, we guess the *scheme* of P restricted to U. Namely, we consider the graph G' obtained from G by the deletion of the vertices of x-trees for all $x \in U$ and for each vertex $x \in U$, we guess how the message is broadcasted to x and from x to the neighbors of x in G'. Notice that $T' = T[V(G')]$ is a tree by the definition of G'. Observe also that for each x-tree F for $x \in U$, the message is broadcasted to the vertices of F from x, because $s \in U$. In particular, this means that the parents of the vertices of U in T are in G'. For each $v \in U$ distinct from s, we guess its parent $p(v) \in V(G')$ in T' and assume that $p(s) = s$. Then for each $v \in V(G)$, we guess the ordered subset $R(v)$ of vertices of $N_{G'}(v) \setminus \{p(v)\}$ such that $R(v) = C(v) \cap N_{G'}(v)$. We guess $p(v)$ and $R(v)$ for $v \in U$ by considering all possible choices. To guess $R(v)$ for each $v \in U$, we first guess the (unordered) set $S(v)$ and then consider all possible orderings of the elements of $S(v)$. The selection of $p(v)$ and $S(v)$ is done by brute force. However, we are only interested in choices, where the selection of the neighbors $p(v)$ and $S(v)$ of v for $v \in U$ can be extended to a spanning tree T' of G'.

Let T' be an arbitrary spanning tree of G' rooted in s. Let T'' be the tree obtained from T' by the iterative deletion of leaves not included in U. Observe that T'' is a tree such that $U \subseteq V(T'')$ and each leaf of T'' is a vertex of U. By Claim 1, each edge of T'' is either an edge of $G[U]$ or is an edge of an (x, y)-path Q for distinct $x, y \in U$ such that the internal vertices of Q are the vertices of the (x,y)-tree F; in the second case, each edge of Q is in T''. Notice also that $s \in V(T'')$ and for each $v \in U$ distinct from s, the parent of v in T is the parent of v in T'' with respect to the source vertex s. Hence, our first step in constructing $p(v)$ and $S(v)$, is to consider all possible choices of T''. Observe that $G[U]$ has at most $\binom{4k}{2}$ edges and the total number of (x,y)-trees is at most $4k-1$ by Claim 2. Because T'' is a tree, it contains $|U|-1$ edges of $G[U]$ and (x,y)-paths Q in total. We obtain that we have $k^{\mathcal{O}(k)}$ possibilities to choose T''. From now, we assume that T'' is fixed.

The choice of T'' defines $p(v)$ for $v \in U \setminus \{s\}$. For each $v \in U$, we initiate the construction of $S(v)$ by including in the set the neighbors of v in T'' distinct from $p(v)$. We proceed with guessing of $S(v)$ by considering (x,y)-trees F for $x,y \in U$ such that the (x,y)-path with the internal vertices in F is not included in T''. Clearly, the vertices of every F of such a type should receive the message either via x, or via y, or via both x and y. Let F be an (x,y)-tree of this type. Denote by x' and y' the neighbors of x and y, respectively. We have that either $x' \in S(x)$ and $y' \notin S(y)$, or $x \notin S(x)$ and $y \in S(Y)$, or $x' \in S(x)$, $y' \in S(y)$ and $x' \neq y'$. Thus, we have three choices for F. By Claim 2, the total number of choices is $2^{\mathcal{O}(k)}$. We go over all the choices and include the vertices to the sets $S(v)$ for $v \in U$ with respect to them. By Claim 1, this concludes the construction of the sets $S(v)$. From now, we assume that $S(v)$ for $v \in U$ are fixed.

We construct the ordered sets $R(v)$ by considering all possible orderings of the elements of $S(v)$. The number of these orderings is $\mathit{\Pi}_{v\in U}(|S(v)|!)$. Recall that the sets $S(v)$ are sets of neighbors of v in a spanning tree of G'. This and Claims 1 and 2 imply that $\sum_{v\in U}|S(v)| \leq 2(|U|-1)+2(4k-1) \leq 16k$. Therefore, the total number of orderings is $\mathit{\Pi}_{v\in U}(|S(v)|!) = k^{\mathcal{O}(k)}$. This completes the construction of $R(v)$. Now we can assume that $p(v)$ for each $v \in U \setminus \{s\}$ and $R(v)$ for each $v \in U$ are given.

The final part of our algorithm is checking whether the guessed scheme for a broadcasting protocol can be extended to the protocol itself. This is done in two stages.

In the first stage, we compute for each $v \in U$, the minimum number $r(v)$ of a round in which v can receive the message and the ordered set $C(v)$. Initially, we set $r(s) = 0$ and set $X := \{s\}$. Then we iteratively either compute $C(v)$ for $v \in X$ or extend X by including a new vertex $v \in U \setminus X$ and computing $r(v)$. We proceed until we get $X = U$ and compute $C(v)$ for every $v \in U$. We also stop and discard the current choice of the scheme if we conclude that the choice cannot be extended to a broadcasting protocol terminating in at most t steps.

Suppose that there is $v \in X$ such that $C(v)$ is not constructed yet. Notice that $r(v)$ is already computed. To construct $C(v)$, we observe that for each v-tree F anchored in v, the vertices of F should receive the message via v. Hence, to construct $C(v)$, we extend $R(v)$ by inserting the neighbors of v in the v-trees. If there is no v-tree anchored in v, then we simply set $C(v) = R(v)$. Assume that this is not the case and let $F_1,\ldots,F_k$ be the v-trees anchored in v. Denote by $u_1,\ldots,u_k$ the neighbors of v in $T_1,\ldots,T_k$, respectively. Because the message is broadcasted from u to each u_i, we can assume that to broadcast the message from u_i to the other vertices of T_i, an optimal protocol requiring $b(T_i, u_i)$ rounds is used. We compute the values $b(T_i, u_i)$ for all $i \in \{1, \dots, k\}$ and assume that $b(T_1, u_1) \geq \dots \geq b(T_k, u_k)$.

If $r(v) + |R(v)| + k > t$, we discard the current choice of the scheme, because we cannot transmit the message to the neighbors of v in t rounds. Notice also that $r(v) + \max\{b(T_i, u_i) + i \mid i \in \{1, \dots, k\}\}$ rounds are needed to transmit the message to the vertices of all v-trees. Hence, if $r(v) + \max\{b(T_i, u_i) + i \mid i \in \{1, \dots, k\}\} > t$, we discard the considered scheme. From now on, we assume that $r(v) + |R(v)| + k \leq t$ and $r(v) + \max\{b(T_i, u_i) + i \mid i\{1 \in, \dots, k\}\} \leq t$.

The main idea for constructing $C(v)$ is to ensure that the message is sent to the vertices of $R(v)$ as early as possible. To achieve this, we put $u_1, \dots, u_k$ in $C(v)$ in such a way, that the message is sent to each u_i as late as possible. Since $|C(v)| = |R(v)| + k$, we represent $C(v)$ as an $|R(v)| + k$-element array whose elements are indexed $1, 2, \dots, |R(v)| + k$. Because $b(T_1, u_1) \geq \dots \geq b(T_k, u_k)$, we can assume that the ordering of the vertices $u_1, \dots, u_k$ in $C(v)$ is $(u_1, \dots, u_k)$. Therefore, we insert u_i in $C(v)$ consecutively for $i = k, k-1, \dots, 1$.

Suppose that $i \in \{1, \dots, k\}$ and $u_{i+1}, \dots, u_k$ are in $C(v)$. Denote by h_{i+1} the index of u_{i+1} assuming that $h_{k+1} = |R(v)| + k + 1$. We find maximum positive integer $h < h_{i+1}$ such that $r(v) + h + b(T_i, u_i) \leq t$ and set the index $h_i = h$ for u_i. In words, we find the maximum index that is prior to the index of u_{i+1} such that if we transmit the message from v to u_i in the h-th round after v got aware of the message, then the vertices of T_i still may get the message in t rounds. After placing $u_1, \dots, u_k$ into the array, we place the vertices of $R(v)$ in the remaining $|R(v)|$ places following the order in $R(v)$. This completes the construction of $C(v)$.

Suppose that $U \setminus X \neq \emptyset$ and for each $v \in X$, $C(v)$ is given. We assume that for each $v \in X$, the elements of $C(v)$ are indexed $1, \dots, |C(v)|$ according to the order. By the constriction of the schemes, there is $y \in U \setminus X$ such that y receives the message from some vertex $x \in X$ either directly or via some (x, y)-tree F anchored in x and y. We find such a vertex y, compute $r(y)$, and include y in X.

If there is $y \in U \setminus X$ such that $p(y) = x \in X$, then we set $r(y) = r(x) + h$, where h is the index of y in $C(v)$ and set $X := X \cup \{y\}$. Since $r(x)$ is the minimum number of a round when x gets the message, $r(y)$ is the minimum number of a round in which y gets the message. Suppose that such a vertex y does not exist. Then by the construction of the considered scheme, there are $x \in X$ and $y \in U \setminus X$ such that the tree T'' which was used to construct the scheme contains an (x, y)-path whose internal vertices are in the (x, y)-tree F anchored in x and y. This means that the neighbor x' of x in F is included in $C(v)$. Let h be the index of x' in $C(v)$. We also have that $y' = p(y)$ is the unique neighbor of y in F. In other words, we have to transmit the message from x to y according to the scheme. To compute $r(y)$, we have to transmit the message as fast as possible. For this, we use Lemma 2. Notice that the vertices of F should receive the message in at most $t' = t - r(x) - h + 1$ rounds because x receives the message in the round $r(v)$ and $h - 1$ vertices of $C(v)$ get the message before x'. Let F' be the tree obtained from F by adding the vertices x, y and the edges xx', yy'. We compute $b_{t'}(F', x, y)$ using the algorithm from Lemma 2. If $b_{t'}(F', x, y) = +\infty$, we discard the considered scheme because y cannot receive the message in t rounds. Otherwise, we set $r(y) = r(x) + (h-1) + b_{t'}(F', x, y)$ and set $X := X \cup \{y\}$.

This completes the first stage where we compute $r(v)$ and $C(v)$ for $v \in U$. Observe that if we completed this stage without discarding the considered choice of the scheme, we already have a partially constructed broadcasting protocol that ensures that (i) the vertices of v-trees for $v \in U$ get the message in at most t rounds, (ii) the vertices of (x, y)-trees that are assigned by the scheme to transmit the message from x to y receive the message in at most t rounds, and (iii) for each $v \in U$, $r(v)$ is the minimum number of a round when v can receive the message according to the scheme. By Claim 1, it remains to check whether the vertices of (x, y)-trees F that are not assigned by the scheme to transmit the message from x to y or vice versa can receive the message in at most t rounds. We do it using Lemmas 1 and 3.

Suppose that F is a (x, y)-tree anchored in $x, y \in U$ such that $p(x), p(y) \notin V(F)$, that is, F is not assigned by the scheme to transmit the message from x to y or vice versa. Let x' and y' be the neighbors in F of x and y, respectively. By the construction of the schemes, we have three cases: (i) $x' \in C(x)$ and $y' \notin C(y)$, (ii) $x' \notin C(x)$, $y' \in C(y)$, and (iii) $x' \in C(x)$, $y' \in C(y)$, and $x' \neq y'$.

In case (i), the vertices of F should receive the message via x. Clearly, we can use an optimal protocol for F with the source x' to transmit the message from x'. Let h be the index of x' in $C(x)$. We use Lemma 1, to verify whether $t - r(x) - h \geq b(F, x)$. If the inequality holds, we conclude that the message can be transmitted to the vertices of F in at most t rounds. Otherwise, we conclude that this is impossible and discard the scheme. Case (ii) is symmetric and the arguments are the same.

Suppose that $x' \in C(x)$, $y' \in C(y)$, and $x' \neq y'$. Then the vertices of F are receiving the message from both x and y. Denote by i and j the indexes of x' and y' in $C(x)$ and $C(y)$, respectively. By symmetry, we assume without loss of generality that $r(x)+i \leq r(y)+j$ and let $h = (r(y)+j) - (r(x)+i)$. Notice that the vertices of F start to get the message after the round $r(v)+i-1$. Denote by F' the tree obtained from F by adding the vertices x, y and the edges xx', yy'. We use Lemma 3 and compute $d_h(F', x, y)$. If $d_h(F', x, y) \leq t - r(v) - i + 1$, then we obtain that the message can be transmitted to the vertices of F in at most t rounds. Otherwise, we cannot do it and discard the scheme.

This completes the description of the second stage of the verification of whether the considered scheme can be extended to a broadcasting protocol terminating in at most t rounds.

If we find a scheme that allows us to conclude that the message can be broadcasted in at most t rounds, we conclude that (G, s, t) is a yes-instance. Otherwise, if every scheme gets discarded, we return that (G, s, t) is a no-instance of Telephone Broadcast. This concludes the description of the algorithm. □

4 Telephone Broadcast Parameterized by the Vertex Cover Number

In this section, we briefly sketch the proof of Theorem 3. Recall that we aim to show that Telephone Broadcast is FPT on graphs with the vertex cover number at most k when the problem is parameterized by k. We start with some auxiliary claims about the broadcasting on a graph with a given vertex cover S.

Lemma 4 (⋆). *Let G be a graph with at least one edge and $s \in V(G)$. Let also S be a vertex cover of G. Then there is an optimal broadcasting protocol for G with the source s such that the vertices of S receive the message in at most $2|S|-1$ rounds.*

We also use the bound for the number of vertices of $I = V(G) \setminus S$ getting the message in the first p rounds.

Lemma 5 (⋆). *Let G be a graph with at least one edge and $s \in V(G)$. Let also S be a vertex cover of G and $p \geq 1$ be an integer. Then for any broadcasting protocol for G with the source s, at most $p|S|$ vertices of $I = V(G) \setminus S$ receive the message in the first p rounds.*

Sketch of the Proof of Theorem 3. Let (G, s, t) be an instance of Telephone Broadcast and let $k \geq 0$ be an integer. We use the algorithm of Chen, Kanj, and Xia [5] to find in $1.2738^k \cdot n^{\mathcal{O}(1)}$ time a vertex cover S of G of size at most k. If the algorithm fails to find such a set, then we stop and return the answer that G has no vertex cover of size at most k. From now on, we assume that S is given and $I = V(G) \setminus S$. We use the well-known fact that I has a partition $\{L_1, \ldots, L_p\}$ into classes of false twins with $p \leq 2^k$. By Lemma 4, we can assume that the vertices of S receive the message in the first $2k-1$ rounds. Then we apply Lemma 5 and guess the vertices of I which receive the message in the first $2k-1$ rounds using the fact that the vertices of each L_i are indistinguishable. Our task boils down to deciding whether the remaining vertices of I can receive the message in the following $t-2k+1$ rounds. The crucial observation is that these vertices can get the message only from S. This allows us to encode a broadcasting protocol as a system of linear inequalities over $\mathbb{Z}$. For every $v \in S$ and every $i \in \{1, \ldots, p\}$, we introduce an integer-valued variable x_{vi} meaning that exactly x_{vi} neighbors of v in $L_i \setminus Y_i$ receive the message from v in the last $t-2k+1$ rounds. Notice that the number of variables is upper bounded by $k2^k$. Thus, we obtain the system of integer linear inequalities with at most $k2^k$ variables, which can be solved in $2^{\mathcal{O}(k2^k)} \cdot n^{\mathcal{O}(1)}$ time by the results of Lenstra [26] and Kannan [24] (see also [16]). □

5 Kernelization for the Parameterization by $k = n - t$

In this section, we sketch the proof of Theorem 4. Recall that we parameterize Telephone Broadcast by $k = n - t$. Hence, it is convenient for us to denote the considered instances as triples (G, s, k) throughout the section instead of $(G, s, n - k)$. Let (G, s, k) be an instance of Telephone Broadcast. We exhaustively apply the following reduction rules in the order in which they are stated. The first rule is straightforward because $b(G, s) \leq n - 1$ and (G, s, k) is a yes-instance if $k \leq 1$. Also if $k > n$, then (G, s, k) is a no-instance.

Reduction Rule 1. *If $k \leq 1$, then return a trivial yes-instance, e.g., the instance with $G = (\{s\}, \emptyset)$ and $k = 0$, and stop. If $k > n$, then return a trivial no-instance, e.g., the instance with $G = (\{s, v\}, \{sv\})$ and $k = 1$, and stop.*

Observe that after applying Reduction Rule 1, $n \geq 2$, because if $n = 1$, then either $k \leq 1$ of $k > n$ and we would stop. Notice that if $d_G(s) = 1$, then the source s sends the message to its unique neighbor in the first round. This allows us to delete s and define a new source.

Reduction Rule 2. *If $d_G(s) = 1$, then let v be the neighbor of s, set $G := G - s$ and define $s := v$.*

Now we can assume that $d_G(s) \geq 2$. By the next rule, we delete certain pendent vertices.

Reduction Rule 3. *If there is a vertex $v \in V(G)$ such that for the set of vertices of degree one $W \subseteq N_G(v)$, it holds that $|W| \geq |V(G) \setminus W|$, then select an arbitrary $w \in W$ and set $G := G - w$.*

To state the following rule, we introduce an auxiliary notation. For a vertex v of a graph H, we define $\rho_H(v) = \max\{\mathrm{dist}_H(v, u) \mid u \in V(H)\}$.

Reduction Rule 4. *If G has a bridge $e = uv$ such that $G - e$ has two connected components G_1 and G_2, where $s, u \in V(G_1)$, $v \in V(G_2)$, $d_G(u) = 2$, and $|V(G_1)| < dist_{G_1}(s, u) + \rho_{G_2}(v)$, then set $G := G/e$.*

From now, we can assume that Reduction Rules 1–4 are not applicable. We run the standard breadth-first search (BFS) algorithm on G from s (see, e.g., [7] for the description). The algorithm produces a spanning tree B of G of shortest paths and the partition of $V(G)$ into *BFS-levels* $L_0, \ldots, L_r$, where L_i is the set of vertices at distance i from s for every $i \in \{1, \ldots, r\}$.

Using the observation that $b(B, s) \geq b(G, s)$ and Lemma 1, we apply the following rule.

Reduction Rule 5. *Compute $b(B, s)$ and if $b(B, s) \leq n - k$, then return a trivial yes-instance and stop.*

Then we apply the final rule.

Reduction Rule 6. *If there is $v \in L_i$ for some $i \in \{0, \ldots, r - 1\}$ such that for $X = N_G(v) \cap L_{i+1}$ and for the (s, v)-path P in B, it holds that (i) $|X| \geq 2k+1$ and (ii) the total number of vertices in nontrivial, i.e., having at least two vertices, connected components of $G - V(P)$ containing vertices of X is at least $4k - 2$, then return a trivial yes-instance and stop.*

The crucial property of the instance obtained by applying Reduction Rules 1–6 is given in the following lemma.

Lemma 6 (⋆). *Suppose that Reduction Rules 1–6 are not applicable to* (G, s, k)*. Then* $|V(G)| \leq 18k - 12$.

By Lemma 6, if we do not stop during the exhaustive applications of Reduction Rules 1–6, then for the obtained instance (G, s, k), $|V(G)| \leq 18k - 12$. Hence, to complete the kernelization algorithm, we return (G, s, k).

It is straightforward to see that Reduction Rules 1–6 can be applied in polynomial time. In particular, BFS and finding bridges can be done in linear time by classical graph algorithms (see, e.g., the textbook [7]). Thus, the total running time of the kernelization algorithm is polynomial. This completes the the sketch of the proof of Theorem 4.

6 Conclusion

In our paper, we initiated the study of Telephone Broadcast from the parameterized complexity viewpoint. In this section, we discuss further directions of research.

We observed that Telephone Broadcast is trivially FPT when parameterized by t and Theorem 1 implies that the problem can be solved in $3^{2^t} \cdot n^{\mathcal{O}(1)}$ time. Is it possible to get a better running time for the parameterization by t?

In Theorem 4, we obtained a polynomial kernel for the parameterization by $k = n - t$, that is, for the parameterization below the trivial upper bound for $b(G, s)$. This naturally leads to the question about parameterization below some other bounds for this parameter. We note that the parameterization of Telephone Broadcast above the natural lower bound $b(G, s) \geq \log n$ leads to a para-NP-complete problem. To see this, observe that for graphs with $n = 2^t$ vertices, $b(G, s) \leq t$ if and only if G has a binomial spanning tree rooted in s, and it is NP-complete to decide whether G contains such a spanning tree [27].

In Theorems 2 and 3, we considered structural parameterizations of Telephone Broadcast by the cyclomatic and vertex cover numbers, respectively. It is interesting to consider other structural parameterizations. In particular, is Telephone Broadcast FPT when parameterized by the *feedback vertex number* and *treewidth* (we refer to [8] for the definitions)? For the parameterization by treewidth, the complexity status of Telephone Broadcast is open even for the case when the treewidth of the input graphs is at most two, that is, for series-parallel graphs.

References

1. Bar-Noy, A., Guha, S., Naor, J., Schieber, B.: Message multicasting in heterogeneous networks. SIAM J. Comput. **30**(2), 347–358 (2000)

2. Bhabak, P., Harutyunyan, H.A.: Constant approximation for broadcasting in k-cycle graph. In: Ganguly, S., Krishnamurti, R. (eds.) CALDAM 2015. LNCS, vol. 8959, pp. 21–32. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-14974-5_3
3. Bhabak, P., Harutyunyan, H.A.: Approximation algorithm for the broadcast time in k-path graph. J. Interconnect. Netw. **19**(4), 1950006:1–1950006:22 (2019). https://doi.org/10.1142/S0219265919500063
4. Cevnik, M., Zerovnik, J.: Broadcasting on cactus graphs. J. Comb. Optim. **33**(1), 292–316 (2017)
5. Chen, J., Kanj, I.A., Xia, G.: Improved upper bounds for vertex cover. Theor. Comput. Sci. **411**(40–42), 3736–3756 (2010). https://doi.org/10.1016/j.tcs.2010.06.026
6. Cohen, J., Fraigniaud, P., König, J., Raspaud, A.: Optimized broadcasting and multicasting protocols in cut-through routed networks. IEEE Trans. Parallel Distrib. Syst. **9**(8), 788–802 (1998)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. 3rd edn. MIT Press, Cambridge (2009). https://mitpress.mit.edu/books/introduction-algorithms
8. Cygan, M., et al.: Parameterized Algorithms. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3
9. Diestel, R.: Graph Theory. GTM, vol. 173. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-53622-3
10. Elkin, M., Kortsarz, G.: Sublogarithmic approximation for telephone multicast. J. Comput. Syst. Sci. **72**(4), 648–659 (2006)
11. Farley, A.M.: Minimum-time line broadcast networks. Networks **10**(1), 59–70 (1980)
12. Fomin, F.V., Fraigniaud, P., Golovach, P.A.: Parameterized complexity of broadcasting in graphs. CoRR abs/2306.01536 (2023). https://doi.org/10.48550/arXiv.2306.01536
13. Fomin, F.V., Kratsch, D.: Exact Exponential Algorithms. TTCSAES, Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16533-7
14. Fraigniaud, P.: Approximation algorithms for minimum-time broadcast under the vertex-disjoint paths mode. In: auf der Heide, F.M. (ed.) ESA 2001. LNCS, vol. 2161, pp. 440–451. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44676-1_37
15. Fraigniaud, P., Lazard, E.: Methods and problems of communication in usual networks. Discret. Appl. Math. **53**(1–3), 79–133 (1994)
16. Frank, A., Tardos, É.: An application of simultaneous diophantine approximation in combinatorial optimization. Comb. **7**(1), 49–65 (1987). https://doi.org/10.1007/BF02579200
17. Gholami, M.S., Harutyunyan, H.A., Maraachlian, E.: Optimal broadcasting in fully connected trees. J. Interconnect. Netw. **23**(1), 2150037:1–2150037:20 (2023). https://doi.org/10.1142/S0219265921500377
18. Grigni, M., Peleg, D.: Tight bounds on minimum broadcast networks. SIAM J. Discret. Math. **4**(2), 207–222 (1991)
19. Harutyunyan, H.A., Hovhannisyan, N.: Broadcasting in split graphs. In: Mavronicolas, M. (ed.) CIAC 2023. LNCS, vol. 13898, pp. 278–292. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-30448-4_20
20. Harutyunyan, H.A., Maraachlian, E.: On broadcasting in unicyclic graphs. J. Comb. Optim. **16**(3), 307–322 (2008). https://doi.org/10.1007/s10878-008-9160-2

21. Hedetniemi, S.M., Hedetniemi, S.T., Liestman, A.L.: A survey of gossiping and broadcasting in communication networks. Networks **18**(4), 319–349 (1988). https://doi.org/10.1002/net.3230180406
22. Hromkovic, J., Klasing, R., Pelc, A., Ruzicka, P., Unger, W.: Dissemination of Information in Communication Networks - Broadcasting, Gossiping, Leader Election, and Fault-Tolerance. Texts in Theoretical Computer Science. An EATCS Series, Springer, Heidelberg (2005). https://doi.org/10.1007/b137871
23. Johnsson, S.L., Ho, C.: Optimum broadcasting and personalized communication in hypercubes. IEEE Trans. Comput. **38**(9), 1249–1268 (1989)
24. Kannan, R.: Minkowski's convex body theorem and integer programming. Math. Oper. Res. **12**(3), 415–440 (1987). https://doi.org/10.1287/moor.12.3.415
25. Kortsarz, G., Peleg, D.: Approximation algorithms for minimum-time broadcast. SIAM J. Discret. Math. **8**(3), 401–427 (1995)
26. Lenstra, H.W., Jr.: Integer programming with a fixed number of variables. Math. Oper. Res. **8**(4), 538–548 (1983). https://doi.org/10.1287/moor.8.4.538
27. Papadimitriou, C.H., Yannakakis, M.: The complexity of restricted spanning tree problems. J. ACM **29**(2), 285–309 (1982). https://doi.org/10.1145/322307.322309
28. Proskurowski, A.: Minimum broadcast trees. IEEE Trans. Comput. **30**(5), 363–366 (1981). https://doi.org/10.1109/TC.1981.1675796
29. Ravi, R.: Rapid rumor ramification: approximating the minimum broadcast time. In: 35th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 202–213 (1994)
30. Slater, P.J., Cockayne, E.J., Hedetniemi, S.T.: Information dissemination in trees. SIAM J. Comput. **10**(4), 692–701 (1981). https://doi.org/10.1137/0210052
31. Stout, Q.F., Wagar, B.: Intensive hypercube communication prearranged communication in link-bound machines. J. Parallel Distrib. Comput. **10**(2), 167–181 (1990)

Turán's Theorem Through Algorithmic Lens

Fedor V. Fomin[1], Petr A. Golovach[1], Danil Sagunov[2], and Kirill Simonov[3](✉)

[1] Department of Informatics, University of Bergen, Bergen, Norway
{fomin,petr.golovach}@ii.uib.no
[2] St. Petersburg Department of V.A. Steklov Institute of Mathematics,
St. Petersburg, Russia
[3] Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
kirillsimonov@gmail.com

Abstract. The fundamental theorem of Turán from Extremal Graph Theory determines the exact bound on the number of edges $t_r(n)$ in an n-vertex graph that does not contain a clique of size $r+1$. We establish an interesting link between Extremal Graph Theory and Algorithms by providing a simple compression algorithm that in linear time reduces the problem of finding a clique of size ℓ in an n-vertex graph G with $m \geq t_r(n) - k$ edges, where $\ell \leq r+1$, to the problem of finding a maximum clique in a graph on at most $5k$ vertices. This also gives us an algorithm deciding in time $2.49^k \cdot (n+m)$ whether G has a clique of size ℓ. As a byproduct of the new compression algorithm, we give an algorithm that in time $2^{\mathcal{O}(td^2)} \cdot n^2$ decides whether a graph contains an independent set of size at least $n/(d+1)+t$. Here d is the average vertex degree of the graph G. The multivariate complexity analysis based on ETH indicates that the asymptotical dependence on several parameters in the running times of our algorithms is tight.

Keywords: Parameterized algorithms · Extremal graph theory · Turan's theorem · Above guarantee · Kernelization · Exponential time hypothesis

1 Introduction

In 1941, Pál Turán published a theorem that became one of the central results in extremal graph theory. The theorem bounds the number of edges in an undirected graph that does not contain a complete subgraph of a given size. For positive integers $r \leq n$, the *Turán's graph* $T_r(n)$ is the unique complete r-partite n-vertex graph where each part consists of $\lfloor \frac{n}{r} \rfloor$ or $\lceil \frac{n}{r} \rceil$ vertices. In other words, $T_r(n)$ is isomorphic to $K_{a_1,a_2,\ldots,a_r}$, where $a_i = \lceil \frac{n}{r} \rceil$ if i is less than or equal to n modulo r and $a_i = \lfloor \frac{n}{r} \rfloor$ otherwise. We use $t_r(n)$ to denote the number of edges in $T_r(n)$.

The research leading to these results has received funding from the Research Council of Norway via the project BWCA (grant no. 314528) and DFG Research Group ADYN via grant DFG 411362735.

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 348–362, 2023.
https://doi.org/10.1007/978-3-031-43380-1_25

Theorem 1 (Turán's Theorem *[29]***).** *Let $r \leq n$. Then any K_{r+1}-free n-vertex graph has at most $t_r(n)$ edges. The only K_{r+1}-free n-vertex graph with exactly $t_r(n)$ edges is $T_r(n)$.*

The theorem yields a polynomial time algorithm that for a given n-vertex graph G with at least $t_r(n)$ edges decides whether G contains a clique K_{r+1}. Indeed, if a graph G is isomorphic to $T_r(n)$, which is easily checkable in polynomial time, then it has no clique of size $r + 1$. Otherwise, by Turán's theorem, G contains K_{r+1}. There are constructive proofs of Turán's theorem that also allow to find a clique of size $r + 1$ in a graph with at least $t_r(n)$ edges.

The fascinating question is whether Turán's theorem could help to find efficiently larger cliques in sparser graphs. There are two natural approaches to defining a "sparser" graph and a "larger" clique. These approaches bring us to the following questions; addressing these questions is the primary motivation of our work.

First, what happens when the input graph has a bit less edges than the Turán's graph? More precisely,

> Is there an efficient algorithm that for some $k \geq 1$, decides whether an n-vertex graph with at least $t_r(n) - k$ edges contains a clique of size $r+1$?

Second, could Turán's theorem be useful in finding a clique of size larger than $r + 1$ in an n-vertex graph with $t_r(n)$ edges? That is,

> Is there an efficient algorithm that for some $\ell > r$ decides whether an n-vertex graph with at least $t_r(n)$ edges contains a clique of size ℓ?

We provide answers to both questions, and more. We resolve the first question by showing a *simple* fixed-parameter tractable (FPT) algorithm where the parameter is k, the "distance" to the Turán's graph. Our algorithm builds on the cute ideas used by Erdős in his proof of Turán's theorem [11]. Viewing these ideas through algorithmic lens leads us to a simple preprocessing procedure, formally a linear-time polynomial compression. For the second question, unfortunately, the answer is negative.

Our Contribution. To explain our results, it is convenient to state the above questions in terms of the computational complexity of the following problem.

> TURÁN'S CLIQUE
> **Input:** An n-vertex graph G, positive integers $r, \ell \leq n$, and k such that $|E(G)| \geq t_r(n) - k$.
> **Question:** Is there a clique of size at least ℓ in G?

Our first result is the following theorem (Theorem 2). Let G be an n-vertex graph with $m \geq t_r(n) - k$ edges. Then there is an algorithm that for any $\ell \leq r+1$, in time $2.49^k \cdot (n+m)$ either finds a clique of size at least ℓ in G or correctly reports that G does not have a clique of size ℓ. Thus for $\ell \leq r + 1$, TURÁN'S CLIQUE is FPT parameterized by k. More generally, we prove that the problem admits a compression of size linear in k. That is, we provide a linear-time procedure that reduces an instance (G, r, ℓ, k) of Turán's Clique to an equivalent instance (G', p) of the Clique problem with at most $5k$ vertices. The difference between Clique and Turán's Clique is that we do not impose any bound on the number of edges in the input graph of Clique. This is why we use the term compression rather than kernelization,[1] and we argue that stating our reduction in terms of compression is far more natural and helpful. Indeed, after reducing the instance to the size linear in the parameter k, the difference between Clique and Turán's Clique vanes, as even the total number of edges in the instance is automatically bounded by a function of the parameter. On the other hand, Clique is a more general and well-studied problem than Turán's Clique.

Pipelined with the fastest known exact algorithm for Maximum Independent Set of running time $\mathcal{O}(1.1996^n)$ [30], our reduction provides the FPT algorithm for Turán's Clique parameterized by k. This algorithm is single-exponential in k and linear in $n+m$, and we also show that the existence of an algorithm subexponential in k would contradict Exponential Time Hypothesis (Corollary 5). Thus the running time of our algorithm is essentially tight, up to the constant in the base of the exponent.

The condition $\ell \leq r+1$ required by our algorithm is, unfortunately, unavoidable. We prove (Theorem 4) that for any fixed $p \geq 2$, the problem of deciding whether an n-vertex graph with at least $t_r(n)$ edges contains a clique of size $\ell = r + p$ is NP-complete. Thus for any $p \geq 2$, Turán's Clique parameterized by k is para-NP-hard. (We refer to the book of Cygan et al. [6] for an introduction to parameterized complexity.)

While our hardness result rules out finding cliques of size $\ell > r+1$ in graphs with $t_r(n)$ edges in FPT time, an interesting situation arises when the ratio $\xi := \lfloor \frac{n}{r} \rfloor$ is small. In the extreme case, when $n = r$, the n-vertex graph G with $t_r(n) = n(n-1)/2$ is a complete graph. In this case the problem becomes trivial.

To capture how far the desired clique is from the Turán's bound, we introduce the parameter

$$\tau = \begin{cases} 0, & \text{if } \ell \leq r, \\ \ell - r, & \text{otherwise.} \end{cases}$$

The above-mentioned compression algorithm into Clique with at most $5k$ vertices yields almost "for free" a compression of Turán's Clique into Clique with $\mathcal{O}(\tau\xi^2 + k)$ vertices. Hence for any ℓ, one can decide whether an n-vertex graph with $m \geq t_r(n) - k$ edges contains a clique of size ℓ in time $2^{\mathcal{O}(\tau\xi^2+k)} \cdot (n+m)$. Thus the problem is FPT parameterized by $\tau + \xi + k$. This result has an interesting interpretation when we look for a large independent set in the complement of a graph. Turán's theorem, when applied to the complement $\overline{G}$ of a graph G, yields a bound

$$\alpha(G) \geq \frac{n}{d+1},$$

[1] A *kernel* is by definition a reduction to an instance of the same problem. See the book [14] for an introduction to kernelization.

where $\alpha(G)$ is the size of the largest independent set in G (the independence number of G), and d is the average vertex degree of G. This motivates us to define the following problem.

Turán's Independent Set
Input: An n-vertex graph G with average degree d, a positive integer t.
Question: Is there an independent set of size at least $\frac{n}{d+1} + t$ in G?

By Theorem 3, we have a simple algorithm (Corollary 3) that compresses an instance of Turán's Independent Set into an instance of Independent Set with $\mathcal{O}(td^2)$ vertices. Pipelined with an exact algorithm computing a maximum independent set, the compression results in the algorithm solving Turán's Independent Set in time $2^{\mathcal{O}(td^2)} \cdot n^2$.

As we already mentioned, Turán's Clique is NP-complete for any fixed $\tau \geq 2$ and $k = 0$. We prove that the problem remains intractable being parameterized by any pair of the parameters from the triple $\{\tau, \xi, k\}$. More precisely, Turán's Clique is also NP-complete for any fixed $\xi \geq 1$ and $\tau = 0$, as well as for any fixed $\xi \geq 1$ and $k = 0$. These lower bounds are given in Theorem 4.

Given the algorithm of running time $2^{\mathcal{O}(\tau\xi^2+k)} \cdot (n+m)$ and the lower bounds for parameterization by any pair of the parameters from $\{\tau, \xi, k\}$, a natural question is, what is the optimal dependence of a Turán's Clique algorithm on $\{\tau, \xi, k\}$? We use the Exponential Time Hypothesis (ETH) of Impagliazzo, Paturi, and Zane [21] to address this question. Assuming ETH, we rule out the existence of algorithms solving Turán's Clique in time $f(\xi,\tau)^{o(k)} \cdot n^{f(\xi,\tau)}$, $f(\xi,k)^{o(\tau)} \cdot n^{f(\xi,k)}$, and $f(k,\tau)^{o(\sqrt{\xi})} \cdot n^{f(k,\tau)}$, for any function f of the respective parameters.

Related Work. Clique is a notoriously difficult computational problem. It is one of Karp's 21 NP-complete problems [23] and by the work of Håstad, it is hard to approximate Clique within a factor of $n^{1-\epsilon}$ [20]. Clique parameterized by the solution size is W[1]-complete [8]. The problem plays the fundamental role in the W-hierarchy of Downey and Fellows, and serves as the starting point in the majority of parameterized hardness reductions. From the viewpoint of structural parameterized kernelization, Clique does not admit a polynomial kernel when parameterized by the size of the vertex cover [3]. A notable portion of works in parameterized algorithms and kernelization is devoted to solving Independent Set (equivalent to Clique on the graph's complement) on specific graph classes like planar, H-minor-free graphs and nowhere-dense graphs [2,7,10,28].

Our algorithmic study of Turán's theorem fits into the paradigm of the "above guarantee" parameterization [26]. This approach was successfully applied to various problems, see e.g. [1,5,12,15–19,22,25,27].

Most relevant to our work is the work of Dvorak and Lidicky on independent set "above Brooks' theorem" [9]. By Brooks' theorem [4], every n-vertex graph of maximum degree at most $\varDelta \geq 3$ and clique number at most $\varDelta$ has an independent set of size at least $n/\varDelta$. Then the Independent Set over Brook's bound problem is to decide whether an input graph G has an independent set of size at least $\frac{n}{\varDelta} + p$. Dvorak and Lidicky [9, Corollary 3] proved that Indepen-

DENT SET OVER BROOK'S BOUND admits a kernel with at most $114p\Delta^3$ vertices. This kernel also implies an algorithm of running time $2^{\mathcal{O}(p\Delta^3)} \cdot n^{\mathcal{O}(1)}$. When average degree d is at most $\Delta - 1$, by Corollary 3, we have that INDEPENDENT SET OVER BROOK'S BOUND admits a compression into an instance of INDEPENDENT SET with $\mathcal{O}(p\Delta^2)$ vertices. Similarly, by Corollary 4, for $d \leq \Delta - 1$, INDEPENDENT SET OVER BROOK'S BOUND is solvable in time $2^{\mathcal{O}(p\Delta^2)} \cdot n^{\mathcal{O}(1)}$. When $d > \Delta - 1$, for example, on regular graphs, the result of Dvorak and Lidicky is non-comparable with our results.

2 Algorithms

While in the literature it is common to present Turán's theorem under the implicit assumption that n is divisible by r, here we make no such assumption. For that, it is useful to recall the precise value of $t_r(n)$ in the general setting, as observed by Turán [29].

Proposition 1 (Turán [29]). *For positive integers $r \leq n$,*

$$t_r(n) = \left(1 - \frac{1}{r}\right) \cdot \frac{n^2}{2} - \frac{s}{2} \cdot \left(1 - \frac{s}{r}\right)$$

where $s = n - r \cdot \lfloor \frac{n}{r} \rfloor$ is the remainder in the division of n by r.

Note that [29] uses the expression $t_r(n) = \frac{r-1}{2r} \cdot (n^2 - s^2) + \binom{s}{2}$, however it can be easily seen to be equivalent to the above.

We start with our main problem, where we look for a K_{r+1} in a graph that has slightly less than $t_r(n)$ edges. Later in this section, we show how to derive our other algorithmic results from the compression routine developed next.

2.1 Compression Algorithm for $\ell \leq r + 1$

First, we make a crucial observation on the structure of a TURÁN'S CLIQUE instance that will be the key part of our compression argument. Take a vertex v of maximum degree in G, partition $V(G)$ on $S = N_G(v)$ and $T = V(G) \setminus S$, and add all edges between S and T while removing all edges inside T. It can be argued that this operation does not decrease the number of edges in G while also preserving the property of being K_{r+1}-free. Performing this recursively yields that $T_r(n)$ has indeed the maximum number of edges for a K_{r+1}-free graph, and this is the gist of Erdős' proof of Turán's Theorem [11]. Now, we want to extend this argument to cover our above-guarantee case. Again, we start with the graph G and perform exactly the same recursive procedure to obtain the graph G'. While we cannot say that G' is equal to G, since the latter has slightly less than $t_r(n)$ edges, we can argue that every edge that gets changed from G to G' can be attributed to the "budget" k. Thus we arrive to the conclusion that G is different from G' at only $\mathcal{O}(k)$ places. The following lemma makes this intuition formal.

Lemma 1. *There is an $\mathcal{O}(m+k)$-time algorithm that for non-negative integers $k \geq 1$, $r \geq 2$ and an n-vertex graph G with $m \geq t_r(n) - k$ edges, finds a partition $V_1, V_2, \ldots, V_p$ of $V(G)$ with the following properties*

(i) $p \geq r - k$;

(ii) *For each $i \in \{1, \ldots, p\}$, there is a vertex $v_i \in V_i$ with $N_G(v_i) \supset V_{i+1} \cup V_{i+2} \cup \cdots \cup V_p$;*

(iii) *If $p \leq r$, then for the complete p-partite graph G' with parts $V_1, V_2, \ldots, V_p$, we have $|E(G')| \geq |E(G)|$ and $|E(G) \triangle E(G')| \leq 3k$. Moreover, all vertices covered by $E(G) \setminus E(G')$ are covered by $E(G') \setminus E(G)$ and $|E(G') \setminus E(G)| \leq 2k$.*

Let us clarify this technical definition. The lemma basically states that if a graph G has at least $t_r(n) - k$ edges, then it either has a clique of size $r+1$, or it has at most $3k$ edit distance to a complete multipartite graph G' consisting of $p \in [r-k, r]$ parts. Moreover, G has a clique of size p untouched by the edit, i.e. this clique is present in the complete p-partite graph G' as well.

We should also note that Lemma 1 is close to the concept of stability of Turán's theorem. This concept received much attention in extremal graph theory (see e.g. recent work of Korándi et al. [24]), and appeals the structural properties of graphs having number of edges close to the Turán's number $t_r(n)$. Lemma 1 can also be seen as a stability version of Turán's theorem, but from the algorithmic point of view. We move on to the proof of the lemma.

Proof (of Lemma 1). First, we state the algorithm, which follows from the Erdős' proof of Turán's Theorem from [11]. We start with an empty graph G' defined on the same vertex set as G, and set $G_1 = G$. Then we select the vertex $v_1 \in V(G_1)$ as an arbitrary maximum-degree vertex in G_1, i.e. $\deg_{G_1}(v_1) = \max_{u \in V(G_1)} \deg_{G_1}(u)$. We put $V_1 = V(G_1) \setminus N_{G_1}(v_1)$ and add to G' all edges between V_1 and $V(G_1) \setminus V_1$.

We then put $G_2 := G_1 - V_1$ and, unless G_2 is empty, apply the same process to G_2. That is, we select $v_2 \in V(G_2)$ with $\deg_{G_2}(v_2) = \max_{u \in V(G_2)} \deg_{G_2}(u)$ and put $V_2 = V(G_2) \setminus N_{G_2}(v_2)$ and add all edges between V_2 and $V(G_2) \setminus V_2$ to G'. We repeat this process with $G_{i+1} := G_i - V_i$ until G_{i+1} is empty. The process has to stop eventually as each V_i is not empty. In this way three sequences are produced: $G = G_1, G_2, \ldots, G_p, G_{p+1}$, where G_1 is G and G_{p+1} is the empty graph; $v_1, v_2, \ldots, v_p$, and $V_1, V_2, \ldots, V_p$. Note that the sequences $\{v_i\}$ and $\{V_i\}$ satisfy property *(ii)* by construction. Observe that this procedure can be clearly performed in time $\mathcal{O}(n^2)$, and for any $r \geq 2$, $m + k = t_r(n) = \Theta(n^2)$, thus the algorithm takes time $\mathcal{O}(m+k)$.

Clearly, G' is a complete p-partite graph with parts $V_1, V_2, \ldots, V_p$ as in G' we added all edges between V_i and $V(G_i) \setminus V_i = (V_{i+1} \cup V_{i+2} \cup \ldots \cup V_p)$ for each $i \in \{1, \ldots, p\}$ and never added an edge between two vertices in the same V_i. Since a p-partite graph is always K_{p+1}-free, by Theorem 1 $|E(G')| \leq t_p(n)$.

Claim. $|E(G')| - |E(G)| \geq \sum_{i=1}^{p} |E(G[V_i])|$ and for each $u \in V(G)$, $\deg_G(u) \leq \deg_{G'}(u)$.

Proof (of Claim). For each $i \in \{1, \ldots, p\}$, denote by E_i the edges of G' added in the i-th step of the construction. Formally, $E_i = V_i \times (V_{i+1} \cup V_{i+2} \cup \ldots \cup V_p)$ for $i < p$ and $E_p = \emptyset$. We aim to show that $|E_i| - |E(G_i) \setminus E(G_{i+1})| \geq |E(G[V_i])|$. The first part of the claim will follow as $|E(G')| = \sum_{i=1}^{p} |E_i|$ and $|E(G)| = \sum_{i=1}^{p} |E(G_i) \setminus E(G_{i+1})|$.

Denote by d_i the degree of v_i in G_i. Since $N_{G_i}(v_i) = (V_{i+1} \cup V_{i+2} \cup \ldots \cup V_p)$, $|E_i| = d_i|V_i|$. As v_i is a maximum-degree vertex in G_i, $d_i \geq \deg_{G_i}(u)$ for every $u \in V_i$, so $|E_i| \geq \sum_{u \in V_i} \deg_{G_i}(u)$. Recall that $G_{i+1} = G_i - V_i$. Then

$$\begin{aligned}|E(G_i) \setminus E(G_{i+1})| = \sum_{u \in V_i} \deg_{G_i}(u) - |E(G_i[V_i])| = \sum_{u \in V_i} \deg_{G_i}(u) - |E(G[V_i])| \\ \leq |E_i| - |E(G[V_i])|,\end{aligned}$$

and the first part of the claim follows.

To show the second part, note that for a vertex $u \in V_i$, $\deg_G(u) \leq \sum_{j=1}^{i-1} |V_j| + \deg_{G_i}(u)$. On the other hand, u is adjacent to every vertex in $V_1 \cup V_2 \cup \cdots \cup V_{i-1} \cup V_{i+1} \cup \cdots \cup V_p$ in G'. We have already seen that $|V_{i+1} \cup \cdots \cup V_p| \geq \deg_{G_i}(u)$. Thus, $\deg_G(u) \leq \deg_{G'}(u)$. Proof of the claim is complete. □

The claim yields that $|E(G)| \leq t_p(n)$, so $t_p(n) \geq t_r(n) - k$. By Theorem 1, we have that $t_i(n) > t_{i-1}(n)$, as $T_{i-1}(n)$ is distinct from $T_i(n)$, so $t_i(n) \geq t_{i-1}(n) + 1$ for every $i \in [n]$. Hence if $r \geq p$ then $k \geq t_r(n) - t_p(n) \geq r - p$. It concludes the proof of (i).

It is left to prove (iii), i.e. that $|E(G) \triangle E(G')| \leq 3k$ under assumption $p \leq r$. First note that $E(G) \setminus E(G') = \bigcup E(G[V_i])$. Second, since $|E(G')| \leq t_p(n) \leq t_r(n)$ and $|E(G)| \geq t_r(n) - k$, $|E(G')| - |E(G)| \leq k$. By Claim, we have that $|E(G')| - |E(G)| \geq \sum |E(G[V_i])|$. Finally

$$\begin{aligned}|E(G) \triangle E(G')| &= |E(G')| - |E(G)| + 2|E(G) \setminus E(G')| \\ &= |E(G')| - |E(G)| + 2\sum |E(G[V_i])| \leq 3k.\end{aligned}$$

By Claim, each vertex covered by $E(G) \setminus E(G')$ is covered by $E(G') \setminus E(G)$. The total size of these edge sets is at most $3k$, while $|E(G') \setminus E(G)| - |E(G) \setminus E(G')| = |E(G')| - |E(G)| \leq k$. Hence, the size of $|E(G) \setminus E(G')|$ is at most $2k$. This concludes the proof of (iii) and of the lemma. □

We are ready to prove our main algorithmic result. Let us recall that we seek a clique of size ℓ in an n-vertex graph with $t_r(n) - k$ edges, and that $\tau = \max\{\ell - r, 0\}$.

Theorem 2. TURÁN'S CLIQUE *with* $\tau \in \{0, 1\}$ *admits an* $\mathcal{O}(n + m)$*-time compression into* CLIQUE *on at most* $5k$ *vertices.*

Proof. Let (G, r, k, ℓ) be the input instance of TURÁN'S CLIQUE. If $r < 2$ or $n \leq 5k$, a trivial compression is returned. Apply the algorithm of Lemma 1 to (G, r, k, ℓ) and obtain the partition $V_1, V_2, \ldots, V_p$. Observe that this takes time $\mathcal{O}(m + k) = \mathcal{O}(n + m)$ since $n > 5k$. By the second property of Lemma 1,

$v_1, v_2, \ldots, v_p$ induce a clique in G, so if $p \geq \ell$ we conclude that (G, r, k, ℓ) is a yes-instance. Formally, the compression returns a trivial yes-instance of Clique in this case.

We now have that $r - k \leq p \leq r$. Then the edit distance between G and the complete p-partite graph G' with parts $V_1, V_2, \ldots, V_p$ is at most $3k$. Denote by X the set of vertices covered by $E(G) \triangle E(G')$. Denote $R = E(G') \setminus E(G)$ and $A = E(G) \setminus E(G')$. We know that $|R| + |A| \leq 3k$, $|R| \leq 2k$ and $|R| \geq |A|$. By Lemma 1, R covers all vertices in X, so $|X| \leq 2|R|$.

Clearly, (G, r, k, ℓ) as an instance of Turán's Clique is equivalent to an instance (G, ℓ) of Clique. We now apply the following two reduction rules exhaustively to (G, ℓ). Note that these rules are an adaption of the well-known two reduction rules for the general case of Clique (see, e.g., [30]). Here the adapted rules employ the partition $V_1, V_2, \ldots, V_p$ explicitly.

Reduction rule 1. *If there is $i \in [p]$ such that $V_i \not\subseteq X$ and V_i is independent in G, remove V_i from G and reduce ℓ by one.*

Reduction rule 2. *For each $i \in [p]$ with $|V_i \setminus X| > 1$, remove all but one vertices in $V_i \setminus X$ from G.*

Since the reduction rules are applied independently to parts V_1, V_2, …, V_p, and each rule is applied to each part at most once, clearly this can be performed in linear time. We now argue that these reduction rules always produce an equivalent instance of Clique.

Claim. Reduction rule 1 and Reduction rule 2 are safe.

Proof. For Reduction rule 1, note that there is a vertex $v \in V_i \setminus X$ such that $N_G(v) = N_G(V_i) = V(G) \setminus V_i$. Since V_i is independent, for any vertex set C that induces a clique in G, we have $|C \cap V_i| \leq 1$. On the other hand, if $C \cap V_i = \emptyset$, $C \cup \{v\}$ also induces a clique in G as $C \subseteq N_G(v)$. Hence, any maximal clique in G contains exactly one vertex from V_i, so Reduction rule 1 is safe.

To see that Reduction rule 2 is safe, observe that $N_G(u) = N_G(v)$ for any two vertices $u, v \in V_i \setminus X$. Then no clique contains both u and v, and if $C \ni v$ induces a clique in G, $C \setminus \{v\} \cup \{u\}$ also induces a clique in G of the same size. Hence, v can be safely removed from G so Reduction rule 2 is safe. □

It is left to upperbound the size of G after the exhaustive application of reduction rules. In this process, some parts among $V_1, V_2, \ldots, V_p$ are removed from G. W.l.o.g. assume that the remaining parts are $V_1, V_2, \ldots, V_t$ for some $t \leq p$. Note that parts that have no common vertex with X are eliminated by Reduction rule 1, so $t \leq |X|$. On the other hand, by Reduction rule 2, we have $|V_i \setminus X| \leq 1$ for each $i \in [t]$.

Consider $i \in [t]$ with $|V_i \setminus X| = 1$. By Reduction rule 1, $G[V_i]$ contains at least one edge. Since V_i is independent in G', $E(G[V_i]) \subseteq A$. Hence, the number of $i \in [t]$ with $|V_i \setminus X| = 1$ is at most $|A|$. We obtain

$$\begin{aligned} |V(G)| = &\sum_{i=1}^{t} |V_i| = \sum_{i=1}^{t} |V_i \cap X| + \sum_{i=1}^{t} |V_i \setminus X| \\ &\leq |X| + |A| \leq 2|R| + |A| \leq |R| + (|R| + |A|) \leq 5k. \end{aligned}$$

We obtained an instance of Clique that is equivalent to (G, r, k, ℓ) and contains at most $5k$ vertices. The proof is complete. □

Combining the polynomial compression of Theorem 2 with the algorithm of Xiao and Nagamochi [30] for Independent Set running in $\mathcal{O}(1.1996^n)$, we obtain the following.

Corollary 1. Turán's Clique *with* $\tau \leq 1$ *is solvable in time* $2.49^k \cdot (n+m)$.

Proof. Take a given instance of Turán's Clique and compress it into an equivalent instance (G, ℓ) of Clique with $|V(G)| \leq 5k$. Clearly, $(G, |V(G)| - \ell)$ is an instance of Independent Set equivalent to (G, ℓ). Use the algorithm from [30] to solve this instance in $\mathcal{O}(1.1996^{|V(G)|})$ running time. Since $1.1996^5 < 2.49$, the running time of the whole algorithm is bounded by $2.49^k \cdot n^{\mathcal{O}(1)}$. □

2.2 Looking for Larger Cliques

In this subsection we consider the situation when $\tau > 1$. As we will see in Theorem 4, an FPT algorithm is unlikely in this case, unless we take a stronger parameterization. Here we show that Turán's Clique is FPT parameterized by $\tau + \xi + k$. Recall that $\xi = \lfloor \frac{n}{r} \rfloor$. Theorem 4 argues that this particular choice of the parameter is necessary.

First, we show that the difference between $t_\ell(n)$ and $t_r(n)$ can be bounded in terms of τ and ξ. This will allow us to employ Theorem 2 for the new FPT algorithm by a simple change of the parameter. The proof of the next lemma is done via a careful counting argument.

Lemma 2. *Let* n, r, ℓ *be three positive integers with* $r < \ell \leq n$. *Let* $\xi = \lfloor \frac{n}{r} \rfloor$ *and* $\tau = \ell - r$. *Then for* $\tau = \mathcal{O}(r)$, $t_\ell(n) - t_r(n) = \Theta(\tau\xi^2)$.

Proof. Throughout the proof, we assume $\xi = \frac{n}{r}$ since this does not influence the desired Θ estimation. Let s_r be the remainder in the division of n by r and s_ℓ be the remainder in the division of n by ℓ. By Lemma 1,

$$t_\ell(n) - t_r(n) = \frac{\tau n^2}{2r\ell} + \left(\frac{s_r}{2} \cdot \left(1 - \frac{s_r}{r}\right) - \frac{s_\ell}{2} \cdot \left(1 - \frac{s_\ell}{\ell}\right)\right). \tag{1}$$

The first summand in (1) is $\Theta(\xi^2\tau)$. Indeed, since $\tau = \mathcal{O}(r)$ we have

$$\frac{\tau n^2}{2r\ell} = \frac{\tau}{2} \cdot \frac{n}{r} \cdot \frac{n}{r+\tau} = \frac{\xi^2\tau}{2} \cdot \frac{r}{r+\tau} = \Theta(\xi^2\tau). \tag{2}$$

For the second summand,

$$\begin{aligned}\frac{s_r}{2}\cdot\left(1-\frac{s_r}{r}\right)-\frac{s_\ell}{2}\cdot\left(1-\frac{s_\ell}{\ell}\right) &= \frac{\ell s_r(r-s_r)-rs_\ell(\ell-s_\ell)}{2r\ell} = \frac{(rs_\ell^2-\ell s_r^2)+r\ell(s_r-s_\ell)}{2r\ell}\\ &= \frac{(rs_\ell^2-rs_r^2-\tau s_r^2)+r\ell(s_r-s_\ell)}{2r\ell} \qquad (3)\\ &= \frac{r(s_\ell-s_r)(s_\ell+s_r)+r\ell(s_r-s_\ell)}{2r\ell}-\frac{\tau s_r^2}{2r\ell}\\ &= \frac{(s_r-s_\ell)(\ell-(s_\ell+s_r))}{2\ell}-\frac{\tau s_r^2}{2r\ell}. \qquad (4)\end{aligned}$$

Since $n=\lfloor\frac{n}{\ell}\rfloor\cdot\ell+s_\ell$, we have that

$$s_r \equiv \left\lfloor\frac{n}{\ell}\right\rfloor\cdot\ell+s_\ell \pmod r,$$

and

$$s_r \equiv \left\lfloor\frac{n}{\ell}\right\rfloor\cdot(r+\tau)+s_\ell \pmod r.$$

Hence,

$$s_r-s_\ell \equiv \left\lfloor\frac{n}{\ell}\right\rfloor\cdot\tau \pmod r.$$

By definition $s_r<r$, thus we get from the above that $s_r-s_\ell\leq\lfloor\frac{n}{\ell}\rfloor\cdot\tau\leq\xi\tau$.

Analogously,

$$s_\ell-s_r \equiv \left\lfloor\frac{n}{r}\right\rfloor\cdot(-\tau) \pmod \ell$$

Since $s_\ell-s_r>-r>-\ell$, we have that $s_\ell-s_r\geq\lfloor\frac{n}{r}\rfloor\cdot(-\tau)\geq-\xi\tau$. Therefore $|s_\ell-s_r|\leq\xi\tau$. It is easy to see that $|\ell-(s_\ell+s_r)|\leq\ell+(s_\ell+s_r)\leq3\ell$. Finally, $\frac{\tau s_r^2}{2r\ell}$ is non-negative and is upper bounded by $\frac{\tau r^2}{2r\ell}\leq\frac{\tau}{2}$. Thus, the absolute value of (4), is at most

$$\frac{\xi\tau\cdot3\ell}{2\ell}+\frac{\tau}{2}=\mathcal{O}(\xi\tau).$$

By putting together (2) and (4), we conclude that $t_\ell(n)-t_r(n)=\Theta(\xi^2\tau)+\mathcal{O}(\xi\tau)=\Theta(\xi^2\tau)$. □

The following compression algorithm is a corollary of Lemma 2 and Theorem 2. It provides a compression of size linear in k and τ.

Theorem 3. TURÁN'S CLIQUE *admits a compression into* CLIQUE *on* $\mathcal{O}(\tau\xi^2+k)$ *vertices.*

Proof. Let (G,k,r,ℓ) be the given instance of TURÁN'S CLIQUE. If $\ell\leq r+1$, then the proof follows from Theorem 2. Otherwise, reduce (G,k,r,ℓ) to an equivalent instance $(G,k+t_\ell(n)-t_r(n),\ell,\ell)$ of TURÁN'S CLIQUE just by modifying the parameters. This is a valid instance since $|E(G)|\geq t_r(n)-k\geq t_\ell(n)-(t_\ell(n)+t_r(n)+k)$. Denote $k'=k+(t_\ell(n)-t_r(n))$. By Lemma 2, $k'=k+\mathcal{O}(\tau\xi^2)$. Apply polynomial compression of Theorem 2 to (G,k',ℓ,ℓ) into CLIQUE with $\mathcal{O}(k')$, i.e. $\mathcal{O}(\tau\xi^2+k)$, vertices. □

Pipelined with a brute-force algorithm computing a maximum independent set in time $\mathcal{O}(2^n)$, Theorem 3 yields the following corollary.

Corollary 2. TURÁN'S CLIQUE *is solvable in time* $2^{\mathcal{O}(\tau\xi^2+k)}\cdot(n+m)$.

2.3 Independent Set above Turán's Bound

Another interesting application of Theorem 3 concerns computing Independent Set in graphs of small average degree. Recall that Turán's theorem, when applied to the complement $\overline{G}$ of a graph G, yields a bound

$$\alpha(G) \geq \frac{n}{d+1}.$$

Here $\alpha(G)$ is the size of the largest independent set in G (the independence number of G), and d is the average vertex degree of G. Then in Turán's Independent Set, the task is for an n-vertex graph G and positive integer t to decide whether there is an independent set of size at least $\frac{n}{d+1} + t$ in G.

Theorem 3 implies a compression of Turán's Independent Set into Independent Set. In other words, we give a polynomial time algorithm that for an instance (G, t) of Turán's Independent Set constructs an equivalent instance (G', p) of Independent Set with at most $\mathcal{O}(td^2)$ vertices. That is, the graph G has an independent set of size at least $\frac{n}{d+1} + t$ if and only if G' has an independent set of size p.

Corollary 3. Turán's Independent Set *admits a compression into* Independent Set *on $\mathcal{O}(td^2)$ vertices.*

Proof. For simplicity, let us assume that n is divisible by $d + 1$. (For arguments here this assumption does not make an essential difference.) We select $r = \frac{n}{d+1}$, $\tau = t$, and $k = 0$. Then $d = \frac{n}{r} - 1 = \xi - 1$. The graph $\overline{G}$ has at most $nd/2$ edges, hence G has at least $\frac{n(n-1)}{2} - nd/2 = \frac{n(n-1)}{2} - n(\xi - 1)/2 \geq t_r(n)$ edges, see Lemma 1. An independent set of size $\frac{n}{d+1} + t$ in graph $\overline{G}$, corresponds in graph G to a clique of size $r + t$. Since Theorem 3 provides compression into a Clique with $\mathcal{O}(\tau\xi^2 + k) = \mathcal{O}(\tau\xi^2)$ vertices, for independent set and graph $\overline{G}$ this corresponds to a compression into an instance of Independent Set with $\mathcal{O}(td^2)$ vertices. □

By Corollary 3, we obtain the following corollary.

Corollary 4. Turán's Independent Set *is solvable in time $2^{\mathcal{O}(td^2)} \cdot n^2$.*

3 Lower Bounds

In this section, we investigate how the algorithms above are complemented by hardness results. First, observe that k has to be restricted, otherwise the Turán's Clique problem is not any different from Clique. In fact, reducing from Independent Set on sparse graphs, one can show that there is no $2^{o(k)}$-time algorithm for Turán's Clique even when $\tau \leq 1$. (The formal argument is presented in Theorem 5.) This implies that the $2^{\mathcal{O}(k)}$-time algorithm given by Corollary 1 is essentially tight.

Also, the difference between r and ℓ has to be restricted, as it can be easily seen that Turán's Clique admits no $n^{o(\ell)}$-time algorithm even when $k = 0$, assuming ETH. This is observed simply by considering the special case of Turán's Clique where $r = 1$, there the only restriction on G is that $|E(G)| \geq t_r(n) - k = 0$, meaning that the problem is as hard as Clique. However, Theorem 4 shows that even for any fixed $\tau \geq 2$ and $k = 0$ Turán's Clique is NP-complete. This motivates Theorem 3, where the exponential part of the running time has shape $2^{\mathcal{O}(\tau\xi^2 k)}$. In the rest of this section, we further motivate the running time of Theorem 3. First, in Theorem 4 we show that not only setting τ and k to constants is not sufficient to overcome NP-hardness, but also that the same holds for any choice of two parameters out of $\{\tau, \xi, k\}$.

Theorem 4. Turán's Clique *is NP-complete. Moreover, it remains NP-complete in each of the following cases*

(i) for any fixed $\xi \geq 1$ and $\tau = 0$;
(ii) for any fixed $\xi \geq 1$ and $k = 0$;
(iii) for any fixed $\tau \geq 2$ and $k = 0$.

Proof. Towards proving (i) and (ii), we provide a reduction from Clique. Let $\xi \geq 1$ be a fixed constant. Let (G, ℓ) be a given instance of Clique and let $n = |V(G)|$. We assume that $\ell \geq \xi$, otherwise we can solve (G, ℓ) in polynomial time. Construct a graph G' from G as follows. Start from $G' = G$ and $\ell' = \ell$. Then add $\max\{\xi\ell - n, 0\}$ isolated vertices to G'. Note that (G, k) and (G', k') are equivalent and $|V(G')| \geq \xi\ell'$. If we have $\xi\ell' \leq |V(G')| < (\xi + 1)\ell'$, we are done with the construction of G'. Otherwise, repeatedly add a universal vertex to G', increasing ℓ' by one, so $|V(G')| - (\xi + 1)\ell'$ decreases by ξ each time. We repeat this until $|V(G')|$ becomes less than $(\xi + 1)\ell'$. Since the gap between $\xi\ell'$ and $(\xi+1)\ell'$ is at least ξ at any moment, we derive that $\xi\ell' \leq |V(G')| < (\xi+1)\ell'$. The construction of G' is complete. Note that(G', ℓ') is an instance of Clique equivalent to (G, ℓ). We added at most $\max\{n, \xi\ell\}$ vertices to G', hence this is a polynomial-time reduction.

By the above, $\lfloor V(G')/\ell' \rfloor = \xi$, so we can reduce (G', ℓ') to an equivalent instance $(G', \ell', \binom{|V(G')|}{2}, \ell')$ of Turán's Clique. Clearly, this instance has the required fixed value of ξ and $\tau = 0$. This proves (i). For (ii), we use the fact that $t_1(n) = 0$ for every $n > 0$ and reduce (G', ℓ') to $(G', 1, 0, \ell')$.

To show (iii), we need another reduction from Clique. Let $\tau \geq 2$ be a fixed integer constant. Take an instance (G, ℓ) of Clique with $\ell \geq 2\tau$. We denote $n = |V(G)|$. To construct G' from G, we start from a large complete $(\ell-1)$-partite graph with equal-sized parts. The size of each part equals x, so $|V(G')| = (\ell-1)x$. We denote $N = |V(G')|$ and choose the value of x later, for now we only need that $N \geq n$. Clearly, $|E(G')| = t_{\ell-1}(N)$ at this point. To embed G into G', we select arbitrary n vertices in G' and make them isolated. This removes at most $n(\ell - 2)x$ edges from G'. Then we identify these n isolated vertices with $V(G)$ and add edges of G between these vertices in G' correspondingly. This operation does not decrease $|E(G')|$. This completes the construction of G'. Since G' is isomorphic to a complete $(\ell-1)$-partite graph united disjointly with G, we have that (G, ℓ) and (G', ℓ) are equivalent instances of Clique.

We now want to reduce (G', ℓ) to an instance $(G', \ell-\tau, 0, \ell)$ of Turán's Clique. To do so, we need $|E(G')| \geq t_{\ell-\tau}(N)$. By Lemma 2, $t_{\ell-1}(N) - t_{\ell-\tau}(N) \geq C \cdot (\tau-1) \cdot \left(\frac{N}{\ell-\tau}\right)^2$ for some constant $C > 0$. Since $|E(G')| \geq t_{\ell-1}(N) - n(\ell-2)x$, we want to choose x such that

$$n(\ell-2)x \leq C \cdot (\tau-1) \cdot \left(\frac{N}{\ell-\tau}\right)^2 .$$

By substituting $N = (\ell-1)x$, we derive that x should satisfy

$$\frac{n}{C} \cdot \frac{(\ell-2)(\ell-\tau)}{(\ell-1)^2} \cdot \frac{\ell-\tau}{\tau-1} \leq x.$$

Now simply pick as x the smallest integer that satisfies the above. Then $(G', \ell-\tau, 0, \ell)$ is an instance of Turán's Clique that is equivalent to the instance (G, k) of Clique and is constructed in polynomial time. □

Now, recall that Theorem 3 gives an FPT-algorithm for Turán's Clique that is single-exponential in $\tau\xi^2 + k$. The previous theorem argues that all three of τ, ξ, k have to be in the exponential part of the running time. However, that result does not say anything about what can be the best possible dependency on these parameters. The next Theorem 5 aims to give more precise lower bounds based on ETH, in particular it turns out that the dependency on τ and k cannot be subexponential unless ETH fails. First, we need to show the relation between the parameter ξ and the average degree of $\overline{G}$. The proof of the following proposition is available in the full version of the paper [13].

Proposition 2. *Let G be an n-vertex graph, $r \leq n$ be an integer, and denote $\xi = \lfloor \frac{n}{r} \rfloor$. Let $\overline{G}$ denote the complement of G and $\overline{d}$ denote the average degree of $\overline{G}$. Then $\overline{d} \leq \xi$ if $|E(G)| \geq t_r(n)$ and $|E(G)| \geq t_r(n)$ if $\overline{d} \leq \xi - 1$.*

We are ready to give lower bounds for algorithms solving Turán's Clique in terms of the parameters τ, ξ, and k.

Theorem 5. *Unless the Exponential Time Hypothesis fails, for any function f there is no $f(\xi,\tau)^{o(k)} \cdot n^{f(\xi,\tau)}$, $f(\xi,k)^{o(\tau)} \cdot n^{f(\xi,k)}$, or $f(k,\tau)^{o(\sqrt{\xi})} \cdot n^{f(k,\tau)}$ algorithm for* Turán's Clique.

The proof of this result is available in the full version of the paper [13]. It is based on the proof of Theorem 4, but is much more careful to details and contains some new ideas. Moreover, the proof of the first point of the theorem lets us observe that our $2.49^k \cdot (n+m)$-time algorithm for Turán's Clique with $\tau \leq 1$ is essentially tight.

Corollary 5. *Assuming ETH, there is no $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ algorithm for* Turán's Clique *with $\ell \leq r+1$.*

4 Conclusion

We conclude by summarizing natural questions left open by our work. Theorem 5 rules out (unless ETH fails) algorithms with running times subexponential in τ and k. However, when it comes to ξ, the dependency in the upper bound of Corollary 2 is $2^{\mathcal{O}(\tau\xi^2+k)} \cdot n^{\mathcal{O}(1)}$, while Theorem 5 only rules out the running time of $f(k,\tau)^{o(\sqrt{\xi})} \cdot n^{f(k,\tau)}$ under ETH. Thus, whether the correct dependence in ξ is single-exponential or subexponential, is left open. Similarly, the question whether TURÁN'S CLIQUE admits a compression into CLIQUE whose size is linear in ξ, τ, and k, is open. A weaker variant of this question (for the case $k = 0$) for TURÁN'S INDEPENDENT SET, whether it admits a compression or kernel linear in d and in t, is also open.

References

1. Alon, N., Gutin, G., Kim, E.J., Szeider, S., Yeo, A.: Solving MAX-r-SAT above a tight lower bound. Algorithmica **61**(3), 638–655 (2011)
2. Bodlaender, H.L., Fomin, F.V., Lokshtanov, D., Penninkx, E., Saurabh, S., Thilikos, D.M.: (Meta) kernelization. J. ACM **63**(5), 44:1–44:69 (2016). https://doi.org/10.1145/2973749
3. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Kernelization lower bounds by cross-composition. SIAM J. Discret. Math. **28**(1), 277–305 (2014). https://doi.org/10.1137/120880240
4. Brooks, L.R.: On colouring the nodes of a network. Proc. Camb. Philos. Soc. **37**, 194–197 (1941)
5. Crowston, R., Jones, M., Muciaccia, G., Philip, G., Rai, A., Saurabh, S.: Polynomial kernels for lambda-extendible properties parameterized above the Poljak-Turzik bound. In: IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS). Leibniz International Proceedings in Informatics (LIPIcs), vol. 24, pp. 43–54. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2013)
6. Cygan, M., et al.: Parameterized Algorithms. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3
7. Demaine, E.D., Fomin, F.V., Hajiaghayi, M., Thilikos, D.M.: Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs. J. ACM **52**(6), 866–893 (2005)
8. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer-Verlag, New York (1999)
9. Dvorák, Z., Lidický, B.: Independent sets near the lower bound in bounded degree graphs. In: Proceedings of the 34th International Symposium on Theoretical Aspects of Computer Science (STACS). Leibniz International Proceedings in Informatics (LIPIcs), vol. 66, pp. 28:1–28:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017). https://doi.org/10.4230/LIPIcs.STACS.2017.28
10. Dvorák, Z., Mnich, M.: Large independent sets in triangle-free planar graphs. SIAM J. Discret. Math. **31**(2), 1355–1373 (2017). https://doi.org/10.1137/16M1061862
11. Erdős, P.: On the graph theorem of Turán. Mat. Lapok **21**, 249–251 (1970)
12. Fomin, F.V., Golovach, P.A., Lokshtanov, D., Panolan, F., Saurabh, S., Zehavi, M.: Going far from degeneracy. SIAM J. Discret. Math. **34**(3), 1587–1601 (2020). https://doi.org/10.1137/19M1290577

13. Fomin, F.V., Golovach, P.A., Sagunov, D., Simonov, K.: Turán's theorem through algorithmic lens (2023)
14. Fomin, F.V., Lokshtanov, D., Saurabh, S., Zehavi, M.: Kernelization: Theory of Parameterized Preprocessing. Cambridge University Press, Cambridge (2019)
15. Garg, S., Philip, G.: Raising the bar for vertex cover: fixed-parameter tractability above a higher guarantee. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1152–1166. SIAM (2016). https://doi.org/10.1137/1.9781611974331.ch80
16. Gutin, G., van Iersel, L., Mnich, M., Yeo, A.: Every ternary permutation constraint satisfaction problem parameterized above average has a kernel with a quadratic number of variables. J. Comput. Syst. Sci. **78**(1), 151–163 (2012)
17. Gutin, G., Kim, E.J., Lampis, M., Mitsou, V.: Vertex cover problem parameterized above and below tight bounds. Theory Comput. Syst. **48**(2), 402–410 (2011)
18. Gutin, G.Z., Patel, V.: Parameterized traveling salesman problem: beating the average. SIAM J. Discret. Math. **30**(1), 220–238 (2016)
19. Gutin, G.Z., Rafiey, A., Szeider, S., Yeo, A.: The linear arrangement problem parameterized above guaranteed value. Theory Comput. Syst. **41**(3), 521–538 (2007). https://doi.org/10.1007/s00224-007-1330-6
20. Håstad, J.: Clique is hard to approximate within $n^{1-\epsilon}$. Acta Math. **182**(1), 105–142 (1999). https://doi.org/10.1007/BF02392825
21. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity. J. Comput. Syst. Sci. **63**(4), 512–530 (2001)
22. Jansen, B.M.P., Kozma, L., Nederlof, J.: Hamiltonicity below Dirac's condition. In: Sau, I., Thilikos, D.M. (eds.) WG 2019. LNCS, vol. 11789, pp. 27–39. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30786-8_3
23. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of Computer Computations, pp. 85–103. Plenum Press, New York (1972)
24. Korándi, D., Roberts, A., Scott, A.: Exact stability for turán's theorem. Adv. Comb. 31079 (2021)
25. Lokshtanov, D., Narayanaswamy, N.S., Raman, V., Ramanujan, M.S., Saurabh, S.: Faster parameterized algorithms using linear programming. ACM Trans. Algorithms **11**(2), 15:1–15:31 (2014). https://doi.org/10.1145/2566616
26. Mahajan, M., Raman, V.: Parameterizing above guaranteed values: MaxSat and MaxCut. J. Algorithms **31**(2), 335–354 (1999)
27. Mahajan, M., Raman, V., Sikdar, S.: Parameterizing above or below guaranteed values. J. Comput. Syst. Sci. **75**(2), 137–153 (2009)
28. Pilipczuk, M., Siebertz, S.: Kernelization and approximation of distance-r independent sets on nowhere dense graphs. Eur. J. Comb. **94**, 103309 (2021). https://doi.org/10.1016/j.ejc.2021.103309
29. Turán, P.: Eine Extremalaufgabe aus der Graphentheorie. Mat. Fiz. Lapok **48**, 436–452 (1941)
30. Xiao, M., Nagamochi, H.: Exact algorithms for maximum independent set. Inf. Comput. **255**, 126–146 (2017). https://doi.org/10.1016/j.ic.2017.06.001

On the Frank Number and Nowhere-Zero Flows on Graphs

Jan Goedgebeur[1,2], Edita Máčajová[3], and Jarne Renders[1(✉)]

[1] Department of Computer Science, KU Leuven Kulak, 8500 Kortrijk, Belgium
{jan.goedgebeur,jarne.renders}@kuleuven.be
[2] Department of Applied Mathematics, Computer Science and Statistics,
Ghent University, 9000 Ghent, Belgium
[3] Comenius University, Mlynská dolina, 842 48 Bratislava, Slovakia
macajova@dcs.fmph.uniba.sk

Abstract. An edge e of a graph G is called *deletable* for some orientation o if the restriction of o to $G - e$ is a strong orientation. In 2021, Hörsch and Szigeti proposed a new parameter for 3-edge-connected graphs, called the *Frank number*, which refines k-edge-connectivity. The Frank number is defined as the minimum number of orientations of G for which every edge of G is deletable in at least one of them. They showed that every 3-edge-connected graph has Frank number at most 7 and that in case these graphs are also 3-edge-colourable graphs the parameter is at most 3. Here we strengthen the latter result by showing that such graphs have Frank number 2, which also confirms a conjecture by Barát and Blázsik. Furthermore, we prove two sufficient conditions for cubic graphs to have Frank number 2 and use them in an algorithm to computationally show that the Petersen graph is the only cyclically 4-edge-connected cubic graph up to 36 vertices having Frank number greater than 2.

Keywords: Frank number · Connectivity · Orientation · Snark · Nowhere-zero flows

1 Introduction

An *orientation* o of a graph G is a directed graph with vertices $V(G)$ such that each edge $uv \in E(G)$ is replaced by exactly one of the arcs $u \rightarrow v$ or $v \rightarrow u$. An orientation is called *strong* if for every two distinct vertices u and v there exists an oriented uv-path, i.e. an oriented path with endpoints u and v. An edge e is *deletable* in a strong orientation o of G if the restriction of o to $E(G) - \{e\}$ is a strong orientation of $G - e$. The *cyclic edge connectivity* of a graph is the smallest number of edges k whose removal separates the graph into two components, each of which contains a cycle. Such a graph is called *cyclically k-edge-connected*.

In 2021, Hörsch and Szigeti [9] proposed a new parameter for 3-edge-connected graphs called the Frank number, which can be seen as a generalisation

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 363–375, 2023.
https://doi.org/10.1007/978-3-031-43380-1_26

of a theorem by Nash-Williams [11] stating that a graph has a k-arc-connected orientation if it is $2k$-edge-connected. For a 3-edge-connected graph G, the *Frank number* – denoted by $fn(G)$ – is defined to be the minimum number k for which G admits k orientations such that every edge $e \in E(G)$ is deletable in at least one of them.

Hörsch and Szigeti proved in [9] that every 3-edge-connected graph G has $fn(G) \leq 7$ and that the Berge-Fulkerson conjecture [12] implies that $fn(G) \leq 5$. They conjectured that every 3-edge-connected graph G has $fn(G) \leq 3$ and showed that the Petersen graph has Frank number equal to 3. In this paper we will mainly investigate the following stronger problem: Is it true that the Petersen graph is the only cyclically 4-edge-connected cubic graph with Frank number greater than 2? Let G_1 and G_2 be cubic graphs. Create a new graph G by removing a vertex from each of G_1 and G_2 and adding three edges between the resulting 2-valent vertices in such a way that the edges have one vertex in G_1 and one vertex in G_2 and the result is cubic. We call G the *3-join* of G_1 and G_2. We can equivalently formulate the problem as follows.

Problem 1. Can every 3-edge-connected cubic graph G with $fn(G) > 2$ be created by a sequence of 3-joins from the Petersen graph?

Note that a cyclically 3-edge-connected graph uniquely decomposes into cyclically 4-edge connected graphs and that a 3-join of a graph with Frank number 3 with any other graph has Frank number at least 3.

Barát and Blázsik showed in [1] that for any 3-edge-connected graph G, there exists a 3-edge-connected cubic graph H with $fn(H) \geq fn(G)$. Hence, it is sufficient to study this problem in the cubic case.

Hörsch and Szigeti proved in [9] that every 3-edge-connected 3-edge-colourable graph has Frank number at most 3. In Sect. 2 we strengthen this result by showing that these graphs have Frank number equal to 2. Note that such graphs are always cubic.

Barát and Blázsik also verified that several well-known infinite families of 3-edge-connected graphs have Frank number 2. This includes wheel graphs, Möbius ladders, prisms, flower snarks and an infinite subset of the generalised Petersen graphs. Note that except for the wheel graphs and flower snarks, these families all consist of 3-edge-colourable graphs. They also conjectured that every 3-edge-connected hamiltonian cubic graph has Frank number 2. Since every hamiltonian cubic graph is 3-edge-colourable, our result that every 3-edge-connected 3-edge-colourable graph has Frank number 2 also proves this conjecture.

Our proof of this result uses nowhere-zero integer flows. We give a sufficient condition for an edge to be deletable in an orientation which is the underlying orientation of some all-positive nowhere-zero k-flow and construct two specific nowhere-zero 4-flows that show that the Frank number is 2.

Moreover, in Sect. 2 we give two sufficient conditions for cyclically 4-edge-connected cubic graphs to have Frank number 2. In Sect. 3 we propose a heuristic algorithm and an exact algorithm for determining whether the Frank number of a 3-edge-connected cubic graph is 2. The heuristic algorithm makes use of the sufficient conditions for cyclically 4-edge-connected graphs shown in the previous section. Using an implementation of these algorithms we show that the Petersen graph is the only cyclically 4-edge-connected *snark*, i.e. cubic graph which does not admit a 3-edge-colouring, up to and including 36 vertices with Frank number greater than 2.

Due to space constraints we had to omit several proofs. A full-length version of this article containing all proofs can be found on arXiv [6].

1.1 Preliminaries

For some integer k, a *k-flow* (o, f) on a graph consists of an orientation o of the edges of G and a valuation $f : E(G) \to \{0, \pm 1, \pm 2 \ldots, \pm(k-1)\}$ such that at every vertex the sum of the values on incoming edges equals the sum on the outgoing edges. A k-flow (o, f) is said to be *nowhere-zero* if the value of f is not 0 for any edge of $E(G)$. A nowhere-zero k-flow on G is said to be *all-positive* if the value $f(e)$ is positive for every edge e of G. Every nowhere-zero k-flow can be transformed to an all-positive nowhere-zero k-flow by changing the orientation of the edges with negative $f(e)$ and changing negative values of $f(e)$ to $-f(e)$.

Let (G, o) be a graph with orientation o. Let H be a subgraph of G. If the context is clear we write (H, o) to be the graph H with the orientation of o restricted to H. We define the set $D(G, o) \subset E(G)$ to be the set of all edges of G which are deletable in o. Let $u, v \in V(G)$, if the edge uv is oriented from u to v, we write $u \to v$.

2 Theoretical Results

Let (o, f) be an all-positive nowhere-zero k-flow on a cubic graph G. An edge e with $f(e) = 2$ is called a *strong* 2*-edge* if there is no 3-edge-cut containing the edge e (cycle-separating or not) such that the remaining edges of the cut have value 1 in f.

Lemma 1. *Let G be a* 3*-edge-connected graph and let (o, f) be an all-positive nowhere-zero k-flow on G. Then all edges of G which receive value 1 and all strong* 2*-edges in (o, f) are deletable in o.*

For the proof we make use of the fact that if (o, f) is a k-flow on G, then in every edge-cut the sum of the flow values on the edges oriented in one direction equals the sum of the flow values on the edges oriented in the other direction. Due to page limits we omit the full proof.

The following theorem can be shown using Lemma 1 and careful application of the fact that every nowhere-zero 4-flow can be expressed as a combination of two 2-flows.

Theorem 1. *If G is a graph admitting a nowhere-zero* 4*-flow, then $fn(G) = 2$.*

Due to page limits we omit the proof.

It is known that a cubic graph is 3-edge-colourable if and only if it admits a nowhere-zero 4-flow. Therefore we have the following corollary.

Corollary 1. *If G is a 3-edge-connected 3-edge-colourable graph, then $fn(G) = 2$.*

Since every hamiltonian cubic graph is 3-edge-colourable, we have also shown the following conjecture by Barát and Blázsik [1].

Corollary 2. *If G is a 3-edge-connected cubic graph admitting a hamiltonian cycle, then $fn(G) = 2$.*

The following lemmas and theorems give two sufficient conditions for a cyclically 4-edge-connected cubic graph to have Frank number 2. These will be used in the algorithm in Sect. 3.

Lemma 2. *Let o be a strong orientation of a cubic graph G. Let $e_1 = u_1v_1$ and $e_2 = u_2v_2$ be two nonadjacent edges in G such that o contains $u_1 \to v_1$ and $u_2 \to v_2$. Assume that both e_1 and e_2 are deletable in o. Create a cubic graph G' from G by subdividing the edges e_1 and e_2 with vertices x_1 and x_2, respectively, and adding a new edge between x_1 and x_2. Let o' be the orientation of G' containing $u_1 \to x_1$, $x_1 \to v_1$, $x_1 \to x_2$, $u_2 \to x_2$, $x_2 \to v_2$ and such that $o'(e) = o(e)$ for all the remaining edges of G'. Then*

$$D(G', o') \supseteq (D(G, o) - \{e_1, e_2\}) \cup \{x_1v_1, x_1x_2, u_2x_2\}.$$

Due to page limits we omit the proof.

Let C be a 2-factor of G with exactly two odd circuits, say N_1 and N_2, (and possibly some even circuits). Let x_1x_2 be an edge of G such that $x_i \in N_i$ for $i \in \{1, 2\}$. Let $F = G - C$. Let M be a maximum matching in $C - \{x_1, x_2\}$. For $i \in \{1, 2\}$ denote by u_i and v_i the vertices of N_i which are adjacent to x_i. Denote by z_i the edge of N_i incident with u_i and not incident with x_i, denote by y_i the edge of N_i incident with v_i and not incident with x_i. The vertices of the graph $F - \{x_1x_2\} \cup M$ have degree 2, so the components of this graph are circuits. An orientation of these circuits is *consistent on* N_i if the edges z_i and y_i are oriented in the same direction with regards to N_i, see Fig. 1.

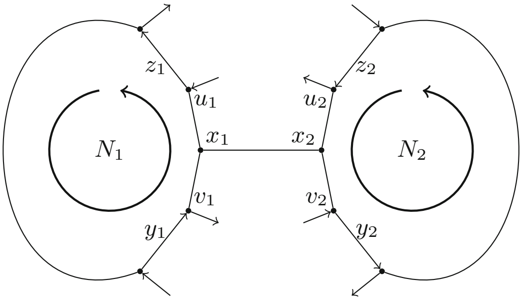


Fig. 1. Consistent orientation of the circuits from $F - \{x_1x_2\} \cup M$.

Theorem 2. *Let G be a cyclically 4-edge-connected cubic graph. Let C be a 2-factor of G with exactly two odd circuits, say N_1 and N_2, (and possibly some even circuits). Let $e = x_1x_2$ be an edge of G such that $x_1 \in N_1$ and $x_2 \in N_2$. For $i \in \{1,2\}$ denote by u_i and v_i the vertices of N_i which are adjacent to x_i. Let $F = G - C$. Let M be a maximum matching in $C - \{x_1, x_2\}$. If there exists an orientation of the circuits in $F - \{x_1x_2\} \cup M$ such that the edges of $N_i \cap M$ incident with u_i and v_i are consistent on N_i for $i \in \{1,2\}$, then $fn(G) = 2$.*

Due to space constains we omit the full proof.

The proof of the following Lemma is similar to that of Lemma 2, but is also omitted due to the page limit.

Lemma 3. *Let o be a strong orientation of a cubic graph G. Let $e_1 = u_1v_1$, $e_2 = u_2v_2$, and $f = w_2w_1$ be pairwise nonadjacent edges in G such that o contains $u_1 \to v_1$, $u_2 \to v_2$, and $w_2 \to w_1$. Let a cubic graph G' be created from G by performing the following steps:*

- *subdivide the edges e_1 and e_2 with the vertices x_1 and x_2, respectively,*
- *subdivide the edge w_1w_2 with the vertices y_1 and y_2 (in this order), and*
- *add the edges x_1y_1 and x_2y_2.*

Let o' be the orientation of G' containing $u_1 \to x_1$, $x_1 \to v_1$, $y_1 \to w_1$, $y_2 \to y_1$, $w_2 \to y_2$, $u_2 \to x_2$, $x_2 \to v_2$ and such that $o'(e) = o(e)$ for all the remaining edges of G' except for x_1y_1 and x_2y_2. Then

(a) if o' contains $y_1 \to x_1$ and $x_2 \to y_2$, o' will be a strong orientation of G' and $D(G', o') \supseteq D(G, o) - \{e_1, e_2, f\} \cup \{u_1x_1, x_1y_1, y_1w_1, y_2w_2, x_2y_2, x_2v_2\}$ (Fig. 2(left));

(b) if o' contains $x_1 \to y_1$ and $y_2 \to x_2$, o' will be a strong orientation of G' and $D(G', o') \supseteq D(G, o) - \{e_1, e_2, f\} \cup \{x_1v_1, y_1y_2, u_2x_2\}$ (Fig. 2(right)).

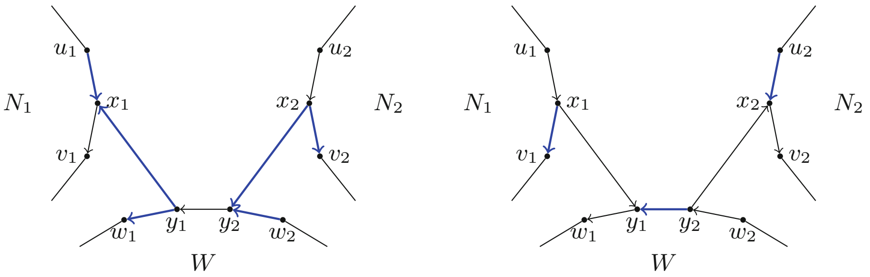


Fig. 2. A part of G' and orientation o' as defined in Lemma 3. The left-hand-side corresponds with the orientation of (a) and the right-hand-side corresponds with the orientation of (b). If the conditions of Lemma 3 are met the thick, blue edges will be deletable. (Color figure online)

Let C be a 2-factor of G with exactly two odd circuits, say N_1 and N_2 and at least one even circuit W. Let x_1y_1, y_1y_2 and y_2x_2 be edges of G such that

$x_i \in N_i$ and $y_i \in W$ for $i \in \{1, 2\}$. Let $F = G - C$ and let M be a maximum matching in $C - \{x_1, x_2, y_1, y_2\}$. For $i \in \{1, 2\}$ denote by u_i and v_i the vertices of N_i incident with x_i and by w_i the vertex of $W - \{y_1, y_2\}$ adjacent to y_i. Denote by z_i the edge of $N_i \cap M$ incident with u_i, by z'_i the edge of $N_i \cap M$ incident with v_i, by z the edge of $W \cap M$ incident with y_1 and by z' the edge of $W \cap M$ incident with y_2. The vertices of the graph $F - \{x_1y_1, y_2x_2\} \cup M$ have degree 2, so the components are circuits. An orientation of these circuits is *consistent on N_i and W* if the edges z_i and z'_i are oriented in the same direction with regards to N_i and the edges z and z' are oriented in the same direction with regards to W, see Fig. 3.

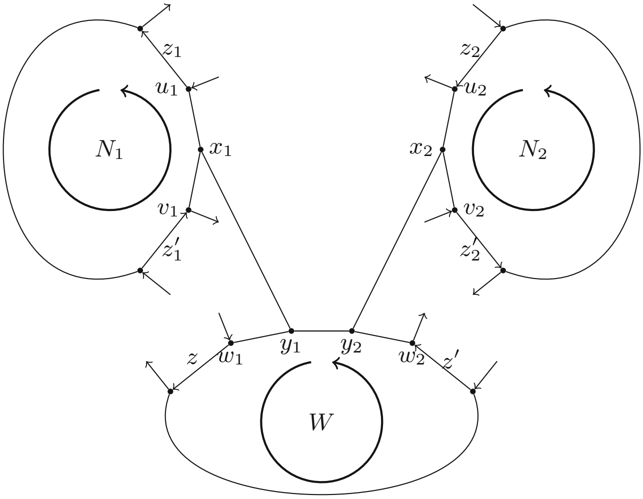


Fig. 3. Consistent orientation of the circuits from $F - \{x_1y_1, y_2x_2\} \cup M$.

Theorem 3. *Let G be a cyclically 4-edge-connected cubic graph with a 2-factor C containing precisely two odd circuits N_1 and N_2 and at least one even circuit W. Let x_1y_1, y_1y_2 and y_2x_2 be edges of G such that $x_1 \in V(N_1)$, $x_2 \in V(N_2)$ and $y_1, y_2 \in V(W)$. For $i \in \{1, 2\}$ denote by u_i and v_i the vertices of N_i which are adjacent to x_i and by w_i the neighbour of y_i in $W - \{y_1, y_2\}$. Let $F = G - C$. Let M be a maximum matching in $C - \{x_1, y_1, y_2, x_2\}$. If there exists an orientation of the circuits in $F - \{x_1y_1, x_2y_2\} \cup M$ such that the edges of $N_i \cap M$ incident with u_i and v_i are consistent on N_i for $i \in \{1, 2\}$ and the edges of $W \cap M$ incident with y_1 and y_2 are consistent on W and $G \sim x_1y_1 \sim x_2y_2$ has no cycle-separating set of three edges $\{e_1, e_2, e_3\}$ with $e_1 \in \{u_1v_1, u_2v_2, w_1w_2\}$ and $e_2, e_3 \in E(F - \{x_1y_1, x_2y_2\} \cup M)$, then $fn(G) \leq 2$.*

Due to page limits we omit the full proof.

3 Algorithm

We propose two algorithms for computationally verifying whether or not a given 3-edge-connected cubic graph has Frank number 2, i.e. a heuristic and an exact algorithm. Note that the Frank number for 3-edge-connected cubic graphs is always at least 2. Our algorithms are intended for graphs which are not 3-edge-colourable, since 3-edge-colourable graphs have Frank number 2 (cf. Corollary 1).

The first algorithm is a heuristic algorithm, which makes use of Theorem 2 and Theorem 3. Hence, it can only be used for cyclically 4-edge-connected cubic graphs. For every 2-factor in the input graph G, we verify if one of the configurations of these theorems is present. If that is the case, the graph has Frank number 2. The pseudocode of this algorithm can be found in Algorithm 1. In this algorithm we look at every 2-factor of G by generating every perfect matching and looking at its complement. We then count how many odd cycles there are in the 2-factor under investigation. If there are precisely two odd cycles, then we check for every edge connecting the two odd cycles whether or not the conditions of Theorem 2 hold. If they hold for one of these edges, we stop the algorithm and return that the graph has Frank number 2. If these conditions do not hold for any of these edges or if there are none, we check for all triples of edges x_1y_1, y_1y_2, y_2x_2, where x_1 and x_2 lie on the two odd cycles and y_1 and y_2 lie on some even cycle, whether the conditions of Theorem 3 hold. If they do, then G has Frank number 2 and we stop the algorithm.

The second algorithm is an exact algorithm for determining whether or not a 3-edge-connected cubic graph has Frank number 2. The pseudocode of this algorithm can be found in Algorithm 2. Due to space constraints, we omitted some technical subroutines from the pseudocode and focused on the main routines. These subroutines can be found in [6]. For a graph G, we start by looking at each of its strong orientations o and try to find a complementary orientation o' such that every edge is deletable in either o or o'. If there is a vertex in G for which none of its adjacent edges are deletable in o, then there exists no complementary orientation as no orientation of a cubic graph has three deletable edges incident to the same vertex. If o is suitable, we look for a complementary orientation using some tricks to reduce the search space. More precisely, we first we start with an empty *partial orientation*, i.e. a directed spanning subgraph of some orientation of G, and orient some edge. Note that we do not need to consider the opposite orientation of this edge, since an orientation of a graph in which all arcs are reversed has the same deletable edges as the original orientation. We then recursively orient edges of G that have not yet been oriented. After orienting an edge, the rules of Lemma 4 may enforce the orientation of edges which are not yet oriented. We orient them in this way before proceeding with the next edge. This heavily restricts the number edges which need to be added. As soon as a complementary orientation is found, we can stop the algorithm and return that the graph G has Frank number 2. If for all strong orientations of G no such complementary orientation is found, then the Frank number of G is higher than 2.

Since the heuristic algorithm is much faster than the exact algorithm, we will first apply the heuristic algorithm. After this we will apply the exact algorithm for those graphs for which the heuristic algorithm was unable to decide whether or not the Frank number is 2. In Sect. 3.1 we give more details on how many graphs pass this heuristic algorithm.

An implementation of these algorithms can be found on GitHub [5]. Our implementation uses bitvectors to store adjacency lists and lists of edges and bitoperations to efficiently manipulate these lists.

Theorem 4. *Let G be a cyclically 4-edge-connected cubic graph. If Algorithm 1 is applied to G and returns True, G has Frank number 2.*

Proof. Suppose the algorithm returns True for G. This happens in a specific iteration of the outer for loop corresponding to a perfect matching F. The complement of F is a 2-factor, say C, and since the algorithm returns True, C has precisely two odd cycles, say N_1 and N_2, and possibly some even cycles.

Suppose first that the algorithm returns True on Line 16. Then there is an edge x_1x_2 in G with $x_1 \in V(N_1)$ and $x_2 \in V(N_2)$, a maximal matching M of $C - \{x_1, x_2\}$ and an orientation o of the cycles in $F - \{x_1x_2\} \cup M$ such that o is consistent on N_1 and N_2. Now by Theorem 2 it follows that G has Frank number 2.

Now suppose that the algorithm returns True on Line 35. Then there are edges x_1y_1, y_1y_2 and y_2x_2 such that $x_1 \in V(N_1)$, $x_2 \in V(N_2)$ and $y_1, y_2 \in V(W)$ where W is some even cycle in C. Since the algorithm returns True, there is a maximal matching M of $C - \{x_1, y_1, y_2, x_2\}$ and an orientation o of the cycles in $F - \{x_1y_1, x_2y_2\} \cup M$ such that o is consistent on N_1, N_2 and W. Denote the neighbours of x_1 and x_2 in C by u_1, v_1 and u_2, v_2, respectively and denote the neighbour of y_1 in $C - y_2$ by w_1 and the neighbour of y_2 in $C - y_1$ by w_2. Since no triple e, e_1, e_2, where $e \in \{u_1x_1, w_1y_1, u_2x_2\}$, $e_1, e_2 \in E(F - \{x_1y_1, x_2y_2\} \cup M)$, is a cycle-separating edge-set of $G - \{x_1y_1, x_2y_2\}$, $G \sim x_1y_1 \sim x_2y_2$ has no cycle-separating edge-set $\{e, e_1, e_2\}$, where $e \in \{u_1v_1, u_2v_2, w_1w_2\}$ and $e_1, e_2 \in E(F - \{x_1y_1, x_2y_2\} \cup M)$. Now by Theorem 3 it follows that G has Frank number 2. □

We will use the following Lemma for the proof of the exact algorithm's correctness.

Lemma 4. *Let G be a cubic graph with $fn(G) = 2$ and let o and o' be two orientations of G such that every edge $e \in E(G)$ is deletable in either o or o'. Then the following hold for o':*

1. *every vertex has at least one incoming and one outgoing edge in o',*
2. *let $uv \notin D(G, o)$, then the remaining edges incident to u are one incoming and one outgoing in o',*
3. *let $uv, vw \notin D(G, o)$, then both are incoming to v or both are outgoing from v in o'.*

Due to page limits we omit the full proof.

Theorem 5. *Let G be a cubic graph. Algorithm 2 applied to G returns True if and only if G has Frank number 2.*

Algorithm 1. heuristicForFrankNumber2(Graph G)

```
for each perfect matching $F$ do
    Store odd cycles of $C := G - F$ in $\mathcal{O} = \{N_1, \ldots, N_k\}$
    if $|\mathcal{O}|$ is not 2 then
        Continue with the next perfect matching
    for all vertices $x_1$ in $N_1$ do
        if $x_1$ has a neighbour $x_2$ in $N_2$ then
            // Test if Theorem 2 can be applied
            Store a maximal matching of $C - \{x_1, x_2\}$ in $M$
            Denote the neighbours of $x_1$ and $x_2$ in $C$ by $u_1, v_1$ and $u_2, v_2$, respectively
            Create an empty partial orientation $o$ of the cycles in $F - \{x_1, x_2\} \cup M$
            for all $x \in \{u_1, v_1, u_2, v_2\}$ do
                if the cycle in $F - \{x_1, x_2\} \cup M$ containing $x$ is not yet oriented then
                    Orient the cycle in $F - \{x_1 x_2\} \cup M$ containing $x$
                    Store this in $o$
            if $o$ is consistent on $N_1$ and on $N_2$ then
                return True // Theorem 2 applies
        else if $x_1$ has a neighbour $y_1$ on an even cycle $W$ of $C$ then
            for each neighbour $y_2$ of $y_1$ in $C$ do
                if $y_2$ has a neighbour $x_2$ in $N_2$ then
                    // Test if Theorem 3 can be applied
                    Store a maximal matching of $C - \{x_1, y_1, y_2, x_2\}$ in $M$
                    Denote the neighbours of $x_1$ and $x_2$ in $C$ by $u_1, v_1$ and $u_2, v_2$
                    Denote the neighbour of $y_1$ in $C - y_2$ by $w_1$
                    Denote the neighbour of $y_2$ in $C - y_1$ by $w_2$
                    Create an empty partial orientation $o$ of the cycles in $F - \{x_1, y_1, y_2, x_2\} \cup M$
                    for all $x \in \{u_1, v_1, u_2, v_2, w_1, w_2\}$ do
                        if the cycle in $F - \{x_1, y_1, y_2, x_2\} \cup M$ containing $x$ is not oriented in $o$ then
                            Orient the cycle in $F - \{x_1, y_1, y_2, x_2\} \cup M$ containing $x$
                            Store this in $o$
                    if $o$ is consistent on $N_1$, $N_2$ and $W$ then
                        // Check cycle-separating edge-set condition
                        for all pairs of edges $e_1, e_2$ in $F - \{x_1, y_1, y_2, x_2\} \cup M$ do
                            for all $e \in \{u_1 x_1, w_1 y_1, u_2 x_2\}$ do
                                if $\{e, e_1, e_2\}$ is a cyclic edge-cut in $G - x_1 y_1 - x_2 y_2$ then
                                    return True // Theorem 3 applies
return False
```

Algorithm 2. frankNumberIs2(Graph G)

```
 1: for all orientations $o$ of $G$ do
 2:     if $o$ is not strong then
 3:         Continue with next orientation
 4:     Store deletable edges of $o$ in a set $D$
 5:     for all $v \in V(G)$ do
 6:         if no edge incident to $v$ is deletable then
 7:             Continue with next orientation
 8:     Create empty partial orientation $o'$ of $G$
 9:     Choose an edge $xy$ in $G$ and its orientation $x \rightarrow y$
10:     if not canAddArcsRecursively($G$, $D$, $o'$, $x \rightarrow y$) then // Algorithm 3
11:         Continue loop with next orientation
12:     while not all edges are oriented in $o'$ do
13:         Store a copy of $o'$ in $o''$
14:         Take an edge $uv$ of $G$ which is unoriented in $o'$
15:         if not canAddArcsRecursively($G$, $D$, $o'$, $u \rightarrow v$) then
16:             Reset $o'$ using $o''$
17:             if not canAddArcsRecursively($G$, $D$, $o'$, $v \rightarrow u$) then
18:                 Continue outer loop with next orientation
19:     if $D(G, o) \cup D(G, o') = E(G)$ then
20:         return True
21: return False
```

Proof. Suppose that frankNumberIs2(G) returns True. Then there exist two orientations o and o' for which $D(G, o) \cup D(G, o') = E(G)$. Hence, $fn(G) = 2$. Conversely, let $fn(G) = 2$. We will show that Algorithm 2 returns True. Let o_1 and o_2 be orientations of G such that every edge of G is deletable in either o_1 or o_2. If the algorithm returns True before we consider o_1 in the loop of Line 1, we are done. So, suppose we are in the iteration where o_1 is considered in the loop of Line 1. Without loss of generality assume that the orientation of xy we choose in Line 9 is in o_2. (If not, reverse all edges of o_2 to get an orientation with the same set of deletable edges.) Let o' be a partial orientation of G and assume that all oriented edges correspond to o_2. Let $u \rightarrow v$ be an arc of o_2. If $u \rightarrow v$ is present in o', then canAddArcsRecursively(G, $D(G, o)$, o', $u \rightarrow v$) (Algorithm 3) returns True and no extra edges become oriented in o'. If $u \rightarrow v$ is not present in o', it gets added on Line 8 of Algorithm 3, since the if-statement on Line 6 of Algorithm 3 will return True by Lemma 4. Note that this is the only place where an arc is added to o' in Algorithm 3. Hence, if we only call Algorithm 3 on arcs present in o_2, then all oriented edges of o' will always be oriented in the same way as in o_2. Now we will show that we only perform this call on arcs in o_2.

Again, suppose $u \rightarrow v$ is an arc in o_2, that it is not yet present in o' and that every oriented edge of o' has the same orientation as in o_2. Let u have two outgoing and no incoming arcs in o'. Let ux be the final unoriented edge incident to u. Then o_2 must have arc $x \rightarrow u$, otherwise it has three outgoing arcs from the same vertex. Let v have two incoming and no outgoing arcs in o'. Let vx be the final unoriented edge incident to v. Then o_2 must have arc $v \to x$, otherwise it has three incoming arcs to the same vertex.

Suppose uv is deletable in o_1. Let ux also be deletable in o_1. Denote the final edge incident to u by uy. Clearly, uy cannot be deletable in o_1, hence it is deletable in o_2. If o_2 contains $u \to x$, then uy is not deletable in o_2, hence, o_2 contains $x \to u$. Let vx be a deletable edge of o_1 and denote the final edge incident to v by vy. Since vy cannot be deletable in o_1, o_2 must contain arc $v \to x$. Suppose that the edges incident with u which are not uv are both not in $D(G, o_1)$. Then they must be oriented incoming to u in o_2. Similarly, if the edges incident with v which are not uv are both not in $D(G, o_1)$, they must both be outgoing from v in o_2.

Finally, suppose that uv is not a deletable edge in o_1. Suppose that o' still has one unoriented edge incident to u, say ux. If the other incident edges are one incoming and one outgoing from u, then o_2 contains the arc $u \to x$. Otherwise, uv cannot be deletable in o_2. Similarly, if o' still has one unoriented edge incident to v, say vx and the remaining incident edges are one incoming and one outgoing, then the arc $x \to v$ must be present in o_2. Otherwise, uv cannot be deletable in o_2. If ux is not deletable in o_1 $x \neq v$. Then o_2 contains the arc $u \to x$. Otherwise, not both of uv and ux can be deletable in o_2. Similarly, if vy is not deletable in o_1 and $y \neq u$, then o_2 must contain the arc $y \to v$. Otherwise, not both of uv and vy can be deletable in o_2.

It now inductively follows that during the execution of Algorithm 2 in the iteration of orientation o_1 on Line 19 that $o' = o_2$. Hence, the if-statement passes and the algorithm returns True. □

Algorithm 3. canAddArcsRecursively(Graph G, Set D, Partial Orientation o', Arc $u \to v$)

```
// Check if $u \to v$ can be added and recursively orient edges for which the orientation is enforced by the rules of Lemma 4
if $u \to v$ is present in $o'$ then
    return True
if $v \to u$ is present in $o'$ then
    return False
if adding $u \to v$ violates rules of Lemma 4 then // Algorithm 4 in Appendix of [6]
    return False
Add $u \to v$ to $o'$
if the orientation of edges enforced by Lemma 4 yields a contradiction then // Algorithm 5 in Appendix of [6]
    return False
return True
```

3.1 Results

Since by Corollary 1 all 3-edge-connected 3-edge-colourable (cubic) graphs have Frank number 2, we will focus in this section on cubic graphs which are *not* 3-edge-colourable.

In [3] Brinkmann et al. determined all cyclically 4-edge-connected snarks up to order 34 and of girth at least 5 up to order 36. This was later extended with all cyclically 4-edge-connected snarks on 36 vertices as well [7]. These lists of snarks can be obtained from the House of Graphs [4] at: https://houseofgraphs.org/meta-directory/snarks. Using our implementation of Algorithms 1 and 2, we tested for all cyclically 4-edge-connected snarks up to 36 vertices if they have Frank number 2 or not. This led to the following result.

Proposition 1. *The Petersen graph is the only cyclically* 4*-edge-connected snark up to and including order* 36 *which has Frank number not equal to* 2*.*

This was done by first running our heuristic Algorithm 1 on these graphs. It turns out that there are few snarks in which neither the configuration of Theorem 2 nor the configuration of Theorem 3 are present. For example: for more than 99.97% of the cyclically 4-edge-connected snarks of order 36, Algorithm 1 is sufficient to determine that their Frank number is 2. Thus we only had to run our exact Algorithm 2 (which is significantly slower than the heuristic) on the graphs for which our heuristic algorithm failed. In total about 214 CPU days of computation time was required to prove Proposition 1 using Algorithm 1 and 2.

In [10] Jaeger defines a snark G to be a *strong snark* if for every edge $e \in E(G)$, $G \sim e$, i.e. the unique cubic graph such that $G - e$ is a subdivision of $G \sim e$, is not 3-edge-colourable. Hence, a strong snark containing a 2-factor which has precisely two odd cycles, has no edge e connecting those two odd cycles, i.e. the configuration of Theorem 2 cannot be present. Therefore, they might be good candidates for having Frank number greater than 2.

In [3] it was determined that there are 7 strong snarks on 34 vertices having girth at least 5, 25 strong snarks on 36 vertices having girth at least 5 and no strong snarks of girth at least 5 of smaller order. By Proposition 1, their Frank number is 2. In [2] it was determined that there are at least 298 strong snarks on 38 vertices having girth at least 5 and the authors of [2] speculate that this is the complete set. We found the following.

Observation 6. *The* 298 *strong snarks of order* 38 *determined in [2] have Frank number* 2*.*

These snarks can be obtained from the House of Graphs [4] by searching for the keywords “strong snark”.

The configurations of Theorem 2 and Theorem 3 also cannot occur in snarks of *oddness* 4, i.e. the smallest number of odd cycles in a 2-factor of the graph is 4. Hence, these may also seem to be good candidates for having Frank number greater than 2. In [7,8] it was determined that the smallest snarks of girth at least 5 with oddness 4 and cyclic edge-connectivity 4 have order 44 and that there are precisely 31 such graphs of this order. We tested each of these and found the following.

Observation 7. *Let G be a snark of girth at least* 5*, oddness* 4*, cyclic edge-connectivity* 4 *and order* 44*. Then* $fn(G) = 2$*.*

These snarks of oddness 4 can be obtained from the House of Graphs [4] at https://houseofgraphs.org/meta-directory/snarks.

The correctness of our algorithm was shown in Theorem 4 and Theorem 5. We also performed several tests to verify that our implementations are correct. However, due to space constraints this had to be omitted. These tests can be found in [6].

References

1. Barát, J., Blázsik, Z.: Quest for graphs of Frank number 3 (2022). https://doi.org/10.48550/arXiv.2209.08804
2. Brinkmann, G., Goedgebeur, J.: Generation of cubic graphs and snarks with large girth. J. Graph Theory **86**(2), 255–272 (2017). https://doi.org/10.1002/jgt.22125
3. Brinkmann, G., Goedgebeur, J., Hägglund, J., Markström, K.: Generation and properties of snarks. J. Comb. Theory. Ser. B **103**(4), 468–488 (2013). https://doi.org/10.1016/j.jctb.2013.05.001
4. Coolsaet, K., D'hondt, S., Goedgebeur, J.: House of graphs 2.0: a database of interesting graphs and more. Discret. Appl. Math. **325**, 97–107 (2023). https://doi.org/10.1016/j.dam.2022.10.013
5. Goedgebeur, J., Máčajová, E., Renders, J.: Frank-Number (2023). https://github.com/JarneRenders/Frank-Number
6. Goedgebeur, J., Máčajová, E., Renders, J.: Frank number and nowhere-zero flows on graphs (2023). arXiv:2305.02133 [math.CO]
7. Goedgebeur, J., Máčajová, E., Škoviera, M.: Smallest snarks with oddness 4 and cyclic connectivity 4 have order 44. ARS Math. Contemp. **16**(2), 277–298 (2019). https://doi.org/10.26493/1855-3974.1601.e75
8. Goedgebeur, J., Máčajová, E., Škoviera, M.: The smallest nontrivial snarks of oddness 4. Discret. Appl. Math. **277**, 139–162 (2020). https://doi.org/10.1016/j.dam.2019.09.020
9. Hörsch, F., Szigeti, Z.: Connectivity of orientations of 3-edge-connected graphs. Eur. J. Comb. **94**, 103292 (2021). https://doi.org/10.1016/j.ejc.2020.103292
10. Jaeger, F.: A survey of the cycle double cover conjecture. In: Alspach, B.R., Godsil, C.D. (eds.) North-Holland mathematics studies, annals of discrete mathematics (27): cycles in graphs, vol. 115, pp. 1–12. North-Holland (1985). https://doi.org/10.1016/S0304-0208(08)72993-1
11. Nash-Williams, C.S.J.A.: On orientations, connectivity and odd-vertex-pairings in finite graphs. Canad. J. Math. **12**, 555–567 (1960). https://doi.org/10.4153/CJM-1960-049-6. publisher: Cambridge University Press
12. Seymour, P.D.: On multi-colourings of cubic graphs, and conjectures of Fulkerson and Tutte. Proc. London Math. Soc. **3**(3), 423–460 (1979). https://doi.org/10.1112/plms/s3-38.3.423

On the Minimum Number of Arcs in 4-Dicritical Oriented Graphs

Frédéric Havet[1], Lucas Picasarri-Arrieta[1(✉)], and Clément Rambaud[1,2]

[1] Université Côte d'Azur, CNRS, Inria, I3S, Sophia Antipolis, France
{frederic.havet,lucas.picasarri-arrieta}@inria.fr
[2] DIENS, École Normale Supérieure, CNRS, PSL University, Paris, France
clement.rambaud@ens.psl.eu

Abstract. We prove that every 4-dicritical oriented graph on n vertices has at least $(\frac{10}{3} + \frac{1}{51})n - 1$ arcs.

Keywords: dichromatic number · oriented graphs · directed graphs · dicritical · density

1 Introduction

Let G be a graph. We denote by $V(G)$ its vertex set and by $E(G)$ its edge set; we set $n(G) = |V(G)|$ and $m(G) = |E(G)|$. A **k-colouring** of G is a function $\varphi : V(G) \to [k]$. It is **proper** if for every edge $uv \in E(G)$, $\varphi(u) \neq \varphi(v)$. The smallest integer k such that G has a proper k-colouring is the **chromatic number**, and is denoted by $\chi(G)$. Since χ is non decreasing with respect to the subgraph relation, it is natural to consider the minimal graphs (for this relation) which are not $(k-1)$-colourable. Following this idea, Dirac defined **k-critical** graphs as the graphs G with $\chi(G) = k$ and $\chi(H) < k$ for every proper subgraph H of G. A first property of k-critical graph is that their minimum degree is at least $k-1$. Indeed, if a vertex v has degree at most $k-2$, then a $(k-1)$-colouring of $G - v$ can be easily extended to G, contradicting the fact that $\chi(G) = k$. As a consequence, the number of edges in a k-critical graph is at least $\frac{k-1}{2}n$. This bound is tight for complete graphs and odd cycles, but Dirac [3] proved an inequality of the form $m \geq \frac{k-1+\varepsilon_k}{2}n - c_k$ for every n-vertex k-critical graph with m edges, for some c_k and $\varepsilon_k > 0$. This shows that, for n sufficiently large, the average degree of a k-critical graph is at least $k - 1 + \varepsilon_k$. This initiated the quest after the best lower bound on the number of edges in n-vertex k-critical graphs. This problem was almost completely solved by Kostochka and Yancey in 2014 [11].

Theorem 1 (Kostochka and Yancey [11]). *Every k-critical graph on n vertices has at least $\frac{1}{2}(k - \frac{2}{k-1})n - \frac{k(k-3)}{2(k-1)}$ edges. For every k, this bound is tight for infinitely many values of n.*

Kostochka and Yancey [12] also characterised k-critical graphs for which this inequality is an equality, and all of them contain a copy of K_{k-2}, the complete graph on $k-2$ vertices. This motivated the following conjecture of Postle [13].

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 376–387, 2023.
https://doi.org/10.1007/978-3-031-43380-1_27

Conjecture 2 (Postle [13]*).* For every integer $k \geq 4$, there exists $\varepsilon_k > 0$ such that every k-critical K_{k-2}-free graph G on n vertices has at least $\frac{1}{2}\left(k - \frac{2}{k-1} + \varepsilon_k\right) n - \frac{k(k-3)}{2(k-1)}$ edges.

For $k = 4$, the conjecture trivially holds as there is no K_2-free 4-critical graph. Moreover, this conjecture has been confirmed for $k = 5$ by Postle [13], for $k = 6$ by Gao and Postle [5], and for $k \geq 33$ by Gould, Larsen, and Postle [6].

Let D be a digraph. We denote by $V(D)$ its vertex set and by $A(D)$ its arc set; we set $n(D) = |V(D)|$ and $m(D) = |E(D)|$. A k**-colouring** of D is a function $\varphi : V(D) \to [k]$. It is a k**-dicolouring** if every directed cycle C in D is not monochromatic for φ (that is $|\varphi(V(C))| > 1$). Equivalently, it is a k-dicolouring if every colour class induces an acyclic subdigraph. The smallest integer k such that D has a k-dicolouring is the **dichromatic number** of D and is denoted by $\vec{\chi}(D)$.

A **digon** in D is a pair of opposite arcs between two vertices. Such a pair of arcs $\{uv, vu\}$ is denoted by $[u, v]$. We say that D is a **bidirected graph** if every pair of adjacent vertices forms a digon. In this case, D can be viewed as obtained from an undirected graph G by replacing each edge $\{u, v\}$ of G by the digon $[u, v]$. We say that D is a bidirected G, and we denote it by $\overleftrightarrow{G}$. Observe that $\chi(G) = \vec{\chi}(\overleftrightarrow{G})$. Thus every statement on proper colouring of undirected graphs can be seen as a statement on dicolouring of bidirected graphs.

Exactly as in the undirected case, one can define k**-dicritical** digraphs to be digraphs D with $\vec{\chi}(D) = k$ and $\vec{\chi}(H) < k$ for every proper subdigraph H of D. It is easy to check that if G is a k-critical graph, then $\overleftrightarrow{G}$ is k-dicritical. Kostochka and Stiebitz [10] conjectured that the k-dicritical digraphs with the minimum number of arcs are bidirected graphs. Thus they conjectured the following generalisation of Theorem 1 to digraphs.

Conjecture 3 (Kostochka and Stiebitz [10]*).* Let $k \geq 2$. Every k-dicritical digraph on n vertices has at least $(k - \frac{2}{k-1})n - \frac{k(k-3)}{k-1}$ arcs. Moreover, equality holds only if D is bidirected.

In the case $k = 2$, this conjecture is easy and weak as it states that a 2-dicritical digraph on n vertices has at least two arcs, while, for all $n \geq 2$, the unique 2-dicritical digraph of order n is the directed n-cycle which has n arcs. The case $k = 3$ of the conjecture has been confirmed by Kostochka and Stiebitz [10]. Using a Brooks-type result for digraphs due to Harutyunyan and Mohar [7], they proved the following: if D is a 3-dicritical digraph of order $n \geq 3$, then $m(D) \geq 2n$ and equality holds if and only if n is odd and D is a bidirected odd cycle. The conjecture has also been proved for $k = 4$ by Kostochka and Stiebitz [10]. However, the conjecture is open for every $k \geq 5$. Recently, this problem has been investigated by Aboulker and Vermande [2] who proved the weaker bound $(k - \frac{1}{2} - \frac{2}{k-1})n - \frac{k(k-3)}{k-1}$ for the number of arcs in an n-vertex k-dicritical digraph.

For integers k and n, let $d_k(n)$ denote the minimum number of arcs in a k-dicritical digraph of order n. By the above observations, $d_2(n) = n$ for all $n \geq 2$, and $d_3(n) \geq 2n$ for all possible n, and equality holds if and only if n is odd and $n \geq 3$. Moreover, if n is even then $d_3(n) = 2n + 1$ (see [1]).

Kostochka and Stiebitz [9] showed that if a k-critical graph G is triangle-free (that is has no cycle of length 3), then $m(G)/n(G) \geq k - o(k)$ as $k \to +\infty$. Informally, this means that the minimum average degree of a k-critical triangle-free graph is (asymptotically) twice the minimum average degree of a k-critical graph. Similarly to this undirected case, it is expected that the minimum number of arcs in a k-dicritical digraph of order n is larger than $d_k(n)$ if we impose this digraph to have no short directed cycles, and in particular if the digraph is an **oriented graph**, that is a digraph with no digon. Let $o_k(n)$ denote the minimum number of arcs in a k-dicritical oriented graph of order n (with the convention $o_k(n) = +\infty$ if there is no k-dicritical oriented graph of order n). Clearly $o_k(n) \geq d_k(n)$.

Conjecture 4 (Kostochka and Stiebitz [10]*).* For any $k \geq 3$, there is a constant $\alpha_k > 0$ such that $o_k(n) > (1 + \alpha_k)d_k(n)$ for n sufficiently large.

For $k = 3$, this conjecture has been recently confirmed by Aboulker, Bellitto, Havet, and Rambaud [1] who proved that $o_3(n) \geq (2 + \frac{1}{3})n + \frac{2}{3}$.

In view of Conjecture 2, Conjecture 4 can be generalized to $\overleftrightarrow{K_{k-2}}$-free digraphs.

Conjecture 5. For any $k \geq 4$, there is a constant $\beta_k > 0$ such that every k-dicritical $\overleftrightarrow{K_{k-2}}$-free digraph D on n vertices has at least $(1 + \beta_k)d_k(n)$ arcs.

Together with Conjecture 3, this conjecture would imply the following generalisation of Conjecture 2.

Conjecture 6. For every integer $k \geq 4$, there exists $\varepsilon_k > 0$ such that every k-dicritical $\overleftrightarrow{K_{k-2}}$-free digraph D on n vertices has at least $(k - \frac{2}{k-1} + \varepsilon_k)n - \frac{k(k-3)}{k-1}$ arcs.

A $\overleftrightarrow{K_2}$-free digraph is an oriented graph, and there are infinitely many 4-dicritical oriented graphs. Thus, while Conjecture 2 holds vacuously for $k = 4$, this is not the case for Conjecture 6. In this paper, we prove that Conjectures 4, 5, and 6 hold for $k = 4$.

Theorem 7. *If $\vec{G}$ is a 4-dicritical oriented graph, then*

$$m(\vec{G}) \geq \left(\frac{10}{3} + \frac{1}{51}\right) n(\vec{G}) - 1.$$

To prove Theorem 7, we use an approach similar to the proof of the case $k = 5$ of Conjecture 2 by Postle [13]. This proof is based on the potential method, which was first popularised by Kostochka and Yancey [11] when they proved Theorem 1. The idea is to prove a more general result on every 4-dicritical digraphs that takes into account the digons.

With a slight abuse, we call **digon** a subdigraph isomorphic to $\overleftrightarrow{K_2}$, the bidirected complete graph on two vertices. We also call **bidirected triangle** a subdigraph isomorphic to $\overleftrightarrow{K_3}$, the bidirected complete graph on three vertices. A **packing** of digons and bidirected triangles is a set of vertex-disjoint digons and bidirected triangles. To take into account the digons, we define a parameter $T(D)$ as follows.

$$T(D) = \max\{d + 2t \mid \text{there exists a packing of } d \text{ digons and } t \text{ bidirected triangles}\}$$

Clearly, $T(D) = 0$ if and only if D is an oriented graph.

Let ε, δ be fixed non-negative real numbers. We define the **potential** (with respect to ε and δ) of a digraph D to be

$$\rho(D) = \left(\frac{10}{3} + \varepsilon\right) n(D) - m(D) - \delta T(D).$$

Thus Theorem 7 can be rephrased as follows.

Theorem 7. *Set $\varepsilon = \frac{1}{51}$ and $\delta = 6\varepsilon = \frac{2}{17}$. If $\vec{G}$ is a 4-dicritical oriented graph, then $\rho(\vec{G}) \leq 1$.*

In fact, we prove a more general statement which holds for every 4-dicritical digraph (with or without digons), except for some exceptions called the 4**-Ore digraphs**. Those digraphs, which are formally defined in Sect. 2, are the bidirected graphs whose underlying graph is one of the 4-critical graphs reaching equality in Theorem 1. In particular, every 4-Ore digraph D has $\frac{10}{3}n(D) - \frac{4}{3}$ arcs. Moreover, the statement holds for all non-negative constants ε and δ satisfying the following inequalities:

- $\delta \geq 6\varepsilon$;
- $3\delta - \varepsilon \leq \frac{1}{3}$;

Theorem 8. *Let $\varepsilon, \delta \geq 0$ be constants satisfying the aforementioned inequalities. If D is a 4-dicritical digraph with n vertices, then*

(i) $\rho(D) \leq \frac{4}{3} + \varepsilon n - \delta\frac{2(n-1)}{3}$ if D is 4-Ore, and
(ii) $\rho(D) \leq 1$ otherwise.

In order to provide some intuition to the reader, let us briefly describe the main ideas of our proof. We will consider a minimum counterexample D to Theorem 8, and show that every subdigraph of D must have large potential. To do so, we need to construct some smaller 4-dicritical digraphs to leverage the minimality of D. These smaller 4-dicritical digraphs will be constructed by identifying some vertices of D. This is why, in the definition of the potential, we consider $T(D)$ instead of the number of digons: when identifying a set of vertices, the number of digons may be arbitrary larger in the resulting digraph, but $T(D)$ increases at most by 1. Using the fact that every subdigraph of D has large potential, we will prove that some subdigraphs are forbidden in D. Using this, we get the final contradiction by a discharging argument.

In addition to Theorem 7, Theorem 8 has also the following consequence when we take $\varepsilon = \delta = 0$.

Corollary 9. *If D is a 4-dicritical digraph, then $m(D) \geq \frac{10}{3}n(D) - \frac{4}{3}$. Moreover, equality holds if and only if D is 4-Ore, otherwise $m(D) \geq \frac{10}{3}n(D) - 1$.*

This is a slight improvement on a result of Kostochka and Stiebitz [10] who proved the inequality $m(D) \geq \frac{10}{3}n(D) - \frac{4}{3}$ without characterising the equality case.

Another interesting consequence of our result is the following bound on the number of vertices in a 4-dicritical oriented graph embedded on a fixed surface. Since a graph on n vertices embedded on a surface of Euler characteristic c has at most $3n - 3c$ edges, we immediately deduce the following from Theorem 7.

Corollary 10. *If $\vec{G}$ is a 4-dicritical oriented graph embedded on a surface of Euler characteristic c, then $n(\vec{G}) \leq \frac{17}{6}(1 - 3c)$.*

The previous best upper bound was $n(\vec{G}) \leq 4 - 9c$ [10].

In Sect. 2 we prove some first preliminary results on 4-Ore digraphs, before proving Theorem 8 in Sect. 3. For the sake of brevity, we skip the proofs of lemmas and claims. All the detailed proofs can be found in [8].

2 The 4-Ore Digraphs and Their Properties

We start with a few notations. We denote by $[\![x_1, \dots, x_n]\!]$ the bidirected path with vertex set $\{x_1, \dots, x_n\}$ in this order. If $x_1 = x_n$, $[\![x_1, \dots, x_n]\!]$ denotes the bidirected cycle of order n with cyclic order $x_1, \dots, x_n$. If D is a digraph, for any $X \subseteq V(D)$, $D - X$ is the subdigraph induced by $V(D) \setminus X$. We abbreviate $D - \{x\}$ into $D - x$. Moreover, for any $F \subseteq V(D) \times V(D)$, $D \setminus F$ is the subdigraph $(V(D), A(D) \setminus F)$ and $D \cup F$ is the digraph $(V(D), A(D) \cup F)$.

Let D_1, D_2 be two bidirected graphs, $[x, y] \subseteq A(D_1)$, and $z \in V(D_2)$. An **Ore-composition** D of D_1 and D_2 with **replaced digon** $[x, y]$ and **split vertex** z is a digraph obtained by removing $[x, y]$ of D_1 and z of D_2, and adding the set of arcs $\{xz_1 \mid zz_1 \in A(D_2) \text{ and } z_1 \in Z_1\}$, $\{z_1x \mid z_1z \in A(D_2) \text{ and } z_1 \in Z_1\}$, $\{yz_2 \mid zz_2 \in A(D_2) \text{ and } z_2 \in Z_2\}$, $\{z_2y \mid z_2z \in A(D_2) \text{ and } z_2 \in Z_2\}$, where (Z_1, Z_2) is a partition of $N_{D_2}(z)$ into non-empty sets. We call D_1 the **digon side** and D_2 the **split side** of the Ore-composition. The class of the **4-Ore digraphs** is the smallest class containing $\overleftrightarrow{K_4}$ which is stable under Ore-composition. See Fig. 1 for an example of a 4-Ore digraph. Observe that all the 4-Ore-digraphs are bidirected.

Proposition 11 (Dirac [4]). *4-Ore digraphs are 4-dicritical.*

Proof. One can easily show that a bidirected digraph is 4-dicritical if and only if its undirected underlying graph is 4-critical. Then the result follows from Theorem 1 in [4]. □

Lemma 12. *Let D be a 4-dicritical bidirected digraph and $v \in V(D)$. Let (N_1^+, N_2^+) and (N_1^-, N_2^-) be two partitions of $N(v)$. Consider D' the digraph with vertex set $V(D) \setminus \{v\} \cup \{v_1, v_2\}$ with $N^+(v_i) = N_i^+, N^-(v_i) = N_i^-$ for $i = 1, 2$ and $D'\langle V(D) \setminus \{v\}\rangle = D - v$. Then D' has a 3-dicolouring with v_1 and v_2 coloured the same except if $N_1^+ = N_1^-$ (that is D' is bidirected).*

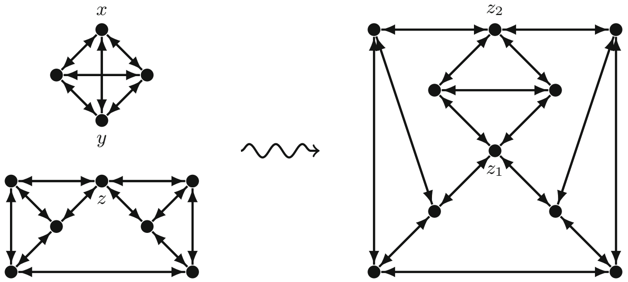


Fig. 1. An example of a 4-Ore digraph obtained by an Ore-composition of two smaller 4-Ore digraphs, with replaced digon $[x, y]$ and split vertex z.

Lemma 13. *Let D be a digraph. If v is a vertex of D, then $T(D-v) \geq T(D)-1$.*

Lemma 14. *If D_1, D_2 are two digraphs, and D is an Ore-composition of D_1 and D_2, then $T(D) \geq T(D_1) + T(D_2) - 2$. Moreover, if D_1 or D_2 is isomorphic to $\overleftrightarrow{K_4}$, then $T(D) \geq T(D_1) + T(D_2) - 1$.*

Lemma 15. *If D is 4-Ore, then $T(D) \geq \frac{2}{3}(n(D) - 1)$.*

Let D be a digraph. A **diamond** in D is a subdigraph isomorphic to $\overleftrightarrow{K_4}$ minus a digon $[u, v]$, with vertices different from u and v having degree 6 in D. An **emerald** in D is a subdigraph isomorphic to $\overleftrightarrow{K_3}$ whose vertices have degree 6 in D.

Let R be an induced subdigraph of D with $n(R) < n(D)$. The **boundary** of R in D, denoted by $\partial_D(R)$, or simply $\partial(R)$ when D is clear from the context, is the set of vertices of R having a neighbour in $V(D) \setminus R$. We say that R is **Ore-collapsible** if the boundary of R contains exactly two vertices u and v and $R \cup [u, v]$ is 4-Ore.

Lemma 16. *If D is 4-Ore and $v \in V(D)$, then there exists either an Ore-collapsible subdigraph of D disjoint from v or an emerald of D disjoint from v.*

Lemma 17. *If $D \neq \overleftrightarrow{K_4}$ is 4-Ore and T is a copy of $\overleftrightarrow{K_3}$ in D, then there exists either an Ore-collapsible subdigraph of D disjoint from T or an emerald of D disjoint from T.*

Lemma 18. *If R is an Ore-collapsible induced subdigraph of a 4-Ore digraph D, then there exists a diamond or an emerald of D whose vertices lie in $V(R)$.*

Lemma 19. *If D is a 4-Ore digraph and v is a vertex in D, then D contains a diamond or an emerald disjoint from v.*

Proof. Follows from Lemmas 16 and 18.

Lemma 20. *If D is a 4-Ore digraph and T is a bidirected triangle in D, then either $D = \overleftrightarrow{K_4}$ or D contains a diamond or an emerald disjoint from T.*

Proof. Follows from Lemmas 17 and 18.

The following theorem was formulated for undirected graphs, but by replacing every edge by a digon, it can be restated as follows:

Theorem 21 (Kostochka and Yancey [12]). *Let D be a 4-dicritical bidirected digraph.*

If $\frac{10}{3}n(D) - m(D) > 1$, then D is 4-Ore and $\frac{10}{3}n(D) - m(D) = \frac{4}{3}$.

Lemma 22. *If D is a 4-Ore digraph with n vertices, then $\rho(D) \leq \frac{4}{3} + \varepsilon n - \delta\frac{2(n-1)}{3}$.*

Proof. Follows from Theorem 21 and Lemma 15.

Lemma 23 (Kostochka and Yancey /[12], Claim 16). *Let D be a 4-Ore digraph. If $R \subseteq D$ and $5 \leq n(R) < n(D)$, then $\frac{10}{3}n(R) - m(R) \geq \frac{10}{3}$.*

Lemma 24. *Let D be a 4-Ore digraph obtained from a copy J of $\overleftrightarrow{K_4}$ by successive Ore-compositions with 4-Ore digraphs, vertices and digons in J being always on the digon side. Let $[u, v]$ be a digon in $D\langle V(J)\rangle$. For every 3-dicolouring φ of $D \setminus [u, v]$, vertices in $V(J)$ receives distinct colours except u and v.*

Lemma 25. *Let D be a 4-Ore digraph obtained from a copy J of $\overleftrightarrow{K_4}$ by successive Ore-compositions with 4-Ore digraphs, vertices and digons in J being always on the digon side. Let v be a vertex in $V(J)$. For every 3-dicolouring φ of $D - v$, vertices in J receives distinct colours.*

3 Proof of Theorem 8

Let D be a 4-dicritical digraph, R be an induced subdigraph of D with $4 \leq n(R) < n(D)$ and φ a 3-dicolouring of R. The **φ-identification** of R in D, denoted by $D_\varphi(R)$ is the digraph obtained from D by identifying for each $i \in [3]$ the vertices coloured i in $V(R)$ to a vertex x_i, adding the digons $[x_i, x_j]$ for all $i, j \in [3]$ and then deleting loops and parallel arcs. Observe that $D_\varphi(R)$ is not 3-dicolourable. Indeed, assume for a contradiction that $D_\varphi(R)$ has a 3-dicolouring φ'. Since $V(R)$ induces a $\overleftrightarrow{K_3}$, we may assume without loss of generality that $\varphi'(x_i) = i$ for $i \in [3]$. Consider the 3-colouring φ'' of D defined by $\varphi''(v) = \varphi'(v)$ if $v \notin R$ and $\varphi''(v) = \varphi(v)$ if $v \in R$. One easily checks that φ'' is a 3-dicolouring of D, a contradiction to the fact that $\vec{\chi}(D) \geq 4$.

Now let W be a 4-dicritical subdigraph of $D_\varphi(R)$ and $X = \{x_1, x_2, x_3\}$. Then we say that $R' = D\langle(V(W) \setminus X) \cup R\rangle$ is the **dicritical extension** of R with **extender** W. We call $X_W = X \cap V(W)$ the **core** of the extension. Note that X_W is not empty, because W is not a subdigraph of D. Thus $1 \leq |X_W| \leq 3$. See Fig. 2 for an example of a φ-identification and a dicritical extension.

Let D be a counterexample to Theorem 8 with minimum number of vertices. By Lemma 22, D is not 4-Ore. Thus $\rho(D) > 1$.

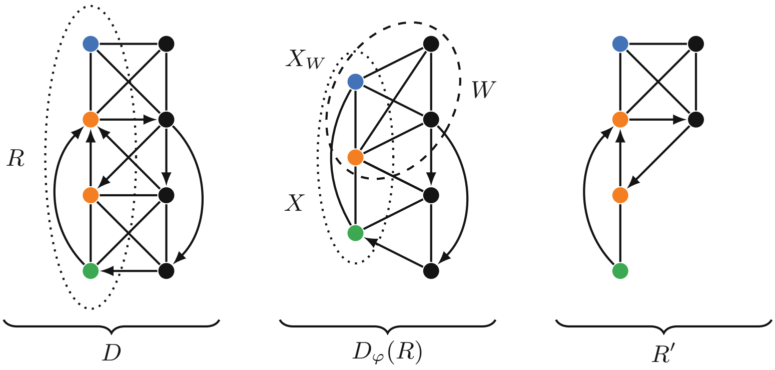


Fig. 2. A 4-dicritical digraph D together with an induced subdigraph R of D and φ a 3-dicolouring of R, the φ-identification $D_\varphi(R)$ of R in D and the dicritical extension R' of R with extender W and core X_W. For clarity, the digons are represented by undirected edges.

Claim 1. If $\tilde{D}$ is a 4-dicritical digraph with $n(\tilde{D}) < n(D)$, then $\rho(\tilde{D}) \leq \frac{4}{3} + 4\varepsilon - 2\delta$.

Claim 2. Let R be a subdigraph of D with $4 \leq n(R) < n(D)$. If R' is a dicritical extension of R with extender W and core X_W, then

$$\rho(R') \leq \rho(W) + \rho(R) - \left(\rho(\overleftrightarrow{K}_{|X_W|}) + \delta \cdot T(\overleftrightarrow{K}_{|X_W|})\right) + \delta \cdot (T(W) - T(W - X_W))$$

and in particular

$$\rho(R') \leq \rho(W) + \rho(R) - \frac{10}{3} - \varepsilon + \delta.$$

Claim 3. If R is a subdigraph of D with $4 \leq n(R) < n(D)$, then $\rho(R) \geq \rho(D) + 2 - 3\varepsilon + \delta > 3 - 3\varepsilon + \delta$.

As a consequence of Claim 3, any subdigraph (proper or not) of size at least 4 has potential at least $\rho(D)$.

We say that an induced subdigraph R of D is **collapsible** if, for every 3-dicolouring φ of R, its dicritical extension R' (with extender W and core X_W) is D, has core of size 1 (i.e. $|X_W| = 1$), and the border $\partial_D(R)$ of R is monochromatic in φ.

Claim 4. Let R be an induced subdigraph of D and φ a 3-dicolouring of R such that $\partial(R)$ is not monochromatic in φ. If D is a dicritical extension of R dicoloured by φ with extender W and core X_W with $|X_W| = 1$, then

$$\rho(R) \geq \rho(D) + 3 - 3\varepsilon + \delta.$$

Claim 5. If R is a subdigraph of D with $4 \leq n(R) < n(D)$ and R is not collapsible, then $\rho(R) \geq \rho(D) + \frac{8}{3} - \varepsilon - \delta > \frac{11}{3} - \varepsilon - \delta$.

Recall that a **k-cutset** in a graph G is a set S of k vertices such that $G - S$ is not connected. A graph is **k-connected** if it has more than k vertices and has no $(k-1)$-cutset. A **k-cutset** in a digraph is a k-cutset in its underlying graph, and a digraph is **k-connected** if its underlying graph is k-connected.

Claim 6. D is 2-connected.

Claim 7. D is 3-connected. In particular, D contains no diamond.

Claim 8. If R is a collapsible subdigraph of D, u, v are in the boundary of R and $D\langle R\rangle \cup [u, v]$ is 4-Ore, then there exists $R' \subseteq R$ such that

(i) either R' is an Ore-collapsible subdigraph of D, or
(ii) R' is an induced subdigraph of R, $n(R') < n(R)$, and there exist u', v' in $\partial_D(R')$ such that $R' \cup [u', v']$ is 4-Ore.

Claim 9. If R is a subdigraph of D with $n(R) < n(D)$ and $u, v \in V(R)$, then $R \cup [u, v]$ is 3-dicolourable. As a consequence, there is no collapsible subdigraph in D.

Claim 10. If R is a subdigraph of D with $n(R) < n(D)$ and $u, v, u', v' \in R$, then $R \cup \{uv, u'v'\}$ is 3-dicolourable. In particular, D contains no copy of $\overleftrightarrow{K_4}$ minus two arcs.

For any $v \in V(D)$, we denote by $n(v)$ its number of neighbours, that is $n(v) = |N^+(u) \cup N^-(v)|$, and by $d(v)$ its number of incident arcs, that is $d(v) = d^+(v) + d^-(v)$.

Claim 11. Vertices of degree 6 in D have either three or six neighbours.

Claim 12. There is no bidirected triangle containing two vertices of degree 6. In particular, D contains no emerald.

So now we know that D contains no emerald, and no diamond by Claim 7.

Claim 13. If R is an induced subdigraph of D with $4 \leq n(R) < n(D)$, then $\rho(R) \geq \rho(D) + 3 + 3\varepsilon - 3\delta$, except if $D - R$ contains a single vertex which has degree 6 in D.

In D, we say that a vertex v is a **simple in-neighbour** (resp. **simple out-neighbour**) if v is a in-neighbour (resp. out-neighbour) of u and $[u, v]$ is not a digon in D. If v is a simple in-neighbour or simple out-neighbour of u, we simply say that v is a **simple neighbour** of u.

Claim 14. Vertices of degree 7 have seven neighbours. In other words, every vertex of degree 7 has only simple neighbours.

The 8^+**-valency** of a vertex v, denoted by $\nu(v)$, is the number of arcs incident to v and a vertex of degree at least 8.

Let D_6 be the subdigraph of D induced by the vertices of degree 6 incident to digons. Let us describe the connected components of D_6 and their neighbourhoods. Remember that vertices of degree 7 are incident to no digon by Claim 14, and so they do not have neighbours in $V(D_6)$. If v is a vertex in D_6, we define its **neighbourhood valency** to be the sum of the 8^+-valency of its neighbours of degree at least 8. We denote the neighbourhood valency of v by $\nu_N(v)$.

Claim 15. If $[x, y]$ is a digon and both x and y have degree 6, then either

(i) the two neighbours of y distinct from x have degree at least 8, or
(ii) the two neighbours of x distinct from y have degree at least 8 and $\nu_N(x) \geq 4$.

Claim 16. Let C be a connected component of D_6. It is either

(i) a single vertex, or
(ii) a bidirected path on two vertices, or
(iii) a bidirected path on three vertices, whose extremities have neighbourhood valency at least 4, or
(iv) a star on four vertices, whose non-central vertices have neighbourhood valency at least 4.

An arc xy is said to be **out-chelou** if

(i) $yx \notin A(D)$, and
(ii) $d^+(x) = 3$, and
(iii) $d^-(y) = 3$, and
(iv) there exists $z \in N^-(y) \setminus N^+(y)$ distinct from x.

Symmetrically, we say that an arc xy is **in-chelou** if yx is out-chelou in the digraph obtained from D by reversing every arc. See Fig. 3 for an example of an out-chelou arc.

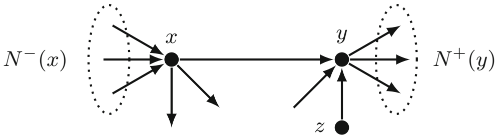


Fig. 3. An example of an out-chelou arc xy.

Claim 17. There is no out-chelou arc and no in-chelou arc in D.

We now use the discharging method. For every vertex v, let $\sigma(v) = \frac{\delta}{|C|}$ if v has degree 6 and is in a component C of D_6 of size at least 2, and $\sigma(v) = 0$ otherwise. Clearly $T(D)$ is at least the number of connected components of size at least 2 of D_6 so $\sum_{v \in V(D)} \sigma(v) \leq \delta T(D)$. We define the **initial charge** of v to be $w(v) = \frac{10}{3} + \varepsilon - \frac{d(v)}{2} - \sigma(v)$. We have

$$\rho(D) \leq \sum_{v \in V(D)} w(v).$$

We now redistribute this total charge according to the following rules:

(R1) A vertex of degree 6 incident to no digon sends $\frac{1}{12} - \frac{\varepsilon}{8}$ to each of its neighbours.

(R2) A vertex of degree 6 incident to digons sends $\frac{2}{d(v)-\nu(v)}(-\frac{10}{3} + \frac{d(v)}{2} - \varepsilon)$ to each neighbour v of degree at least 8 (so $\frac{1}{d(v)-\nu(v)}(-\frac{10}{3} + \frac{d(v)}{2} - \varepsilon)$ via each arc of the digon).

(R3) A vertex of degree 7 with $d^-(v) = 3$ (resp. $d^+(v) = 3$) sends $\frac{1}{12} - \frac{\varepsilon}{8}$ to each of its in-neighbours (resp. out-neighbours).

For every vertex v, let $w^*(v)$ be the final charge of v.

Claim 18. If v has degree at least 8, then $w^*(v) \leq 0$.

Claim 19. If v has degree 7, then $w^*(v) \leq 0$.

Claim 20. If v is a vertex of degree 6 incident to no digon, then $w^*(v) \leq 0$.

Claim 21. Let v be a vertex in D_6 having at least two neighbours of degree at least 8. Then $w^*(v) \leq 0$. Moreover, if v is not an isolated vertex in D_6 and $\nu_N(v) \geq 4$, then $w^*(v) \leq -\frac{1}{9} + \frac{5}{3}\varepsilon - \frac{\delta}{4}$.

Claim 22. If C is a connected component of D_6, then $\sum_{v \in V(C)} w^*(v) \leq 0$.

As a consequence of these last claims, we have $\rho(D) \leq \sum_{v \in V(D)} w(v) = \sum_{v \in V(D)} w^*(v) \leq 0 \leq 1$, a contradiction. This proves Theorem 8. □

References

1. Aboulker, P., Bellitto, T., Havet, F., Rambaud, C.: On the minimum number of arcs in k-dicritical oriented graphs. arXiv preprint arXiv:2207.01051 (2022)
2. Aboulker, P., Vermande, Q.: Various bounds on the minimum number of arcs in a k-dicritical digraph. arXiv preprint arXiv:2208.02112 (2022)
3. Dirac, G.A.: A theorem of R. L. Brooks and a conjecture of H. Hadwiger. Proc. London Math. Soc. **3**(1), 161–195 (1957)
4. Dirac, G.A.: On the structure of 5-and 6-chromatic abstract graphs. J. für die reine und angew. Math. (Crelles J.) **1964**(214–215), 43–52 (1964)
5. Gao, W., Postle, L.: On the minimal edge density of K_4-free 6-critical graphs. arXiv:1811.02940 [math] (2018)

6. Gould, R.J., Larsen, V., Postle, L.: Structure in sparse k-critical graphs. J. Comb. Theory Ser. B **156**, 194–222 (2022)
7. Harutyunyan, A., Mohar, B.: Gallai's theorem for list coloring of digraphs. SIAM J. Discret. Math. **25**(1), 170–180 (2011)
8. Havet, F., Picasarri-Arrieta, L., Rambaud, C.: On the minimum number of arcs in 4-dicritical oriented graphs. arXiv preprint arXiv:2306.10784 (2023)
9. Kostochka, A., Stiebitz, M.: On the number of edges in colour-critical graphs and hypergraphs. Combinatorica **20**(4), 521–530 (2000)
10. Kostochka, A., Stiebitz, M.: The minimum number of edges in 4-critical digraphs of given order. Graphs Comb. **36**(3), 703–718 (2020)
11. Kostochka, A., Yancey, M.: Ore's conjecture on color-critical graphs is almost true. J. Comb. Theory, Ser. B **109**, 73–101 (2014)
12. Kostochka, A., Yancey, M.: A Brooks-type result for sparse critical graphs. Combinatorica **38**(4), 887–934 (2018)
13. Postle, L.: On the minimum number of edges in triangle-free 5-critical graphs. Eur. J. Comb. **66**, 264–280 (2017). selected papers of EuroComb15

Tight Algorithms for Connectivity Problems Parameterized by Modular-Treewidth

Falko Hegerfeld and Stefan Kratsch(✉)

Institut für Informatik, Humboldt-Universität zu Berlin, Berlin, Germany
{hegerfeld,kratsch}@informatik.hu-berlin.de

Abstract. We study connectivity problems from a fine-grained parameterized perspective. Cygan et al. (TALG 2022) first obtained algorithms with single-exponential running time $\alpha^{\mathrm{tw}} n^{\mathcal{O}(1)}$ for connectivity problems parameterized by treewidth (tw) by introducing the cut-and-count-technique, which reduces the connectivity problems to locally checkable counting problems. In addition, the obtained bases α were proven to be optimal assuming the Strong Exponential-Time Hypothesis (SETH).

As only sparse graphs may admit small treewidth, these results are not applicable to graphs with dense structure. A well-known tool to capture dense structure is the *modular decomposition*, which recursively partitions the graph into *modules* whose members have the same neighborhood outside of the module. Contracting the modules, we obtain a *quotient graph* describing the adjacencies between modules. Measuring the treewidth of the quotient graph yields the parameter *modular-treewidth*, a natural intermediate step between treewidth and clique-width. While less general than clique-width, modular-treewidth has the advantage that it can be computed as easily as treewidth.

We obtain the first tight running times for connectivity problems parameterized by modular-treewidth. For some problems the obtained bounds are the same as relative to treewidth, showing that we can deal with a greater generality in input structure at no cost in complexity. We obtain the following randomized algorithms for graphs of modular-treewidth k, given an appropriate decomposition:

– Steiner Tree can be solved in time $3^k n^{\mathcal{O}(1)}$,
– Connected Dominating Set can be solved in time $4^k n^{\mathcal{O}(1)}$,
– Connected Vertex Cover can be solved in time $5^k n^{\mathcal{O}(1)}$,
– Feedback Vertex Set can be solved in time $5^k n^{\mathcal{O}(1)}$.

The first two algorithms are tight due to known results and the last two algorithms are complemented by new tight lower bounds under SETH.

Keywords: connectivity · modular-treewidth · tight algorithms

The first author was partially supported by DFG Emmy Noether-grant (KR 4286/1).

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 388–402, 2023.
https://doi.org/10.1007/978-3-031-43380-1_28

1 Introduction

Connectivity constraints are a very natural form of global constraints in the realm of graph problems. We study connectivity problems from a fine-grained parameterized perspective. The starting point is an influential paper of Cygan et al. [11] introducing the cut-and-count-technique which yields randomized algorithms with running time[1] $\mathcal{O}^*(\alpha^{\mathrm{tw}})$, for some constant *base* $\alpha > 1$, for connectivity problems parameterized by *treewidth* (tw). The obtained bases α were proven to be optimal assuming the Strong Exponential-Time Hypothesis[2] (SETH) [10].

Since dense graphs cannot have small treewidth, the results for treewidth do not help for graphs with dense structure. A well-known tool to capture dense structure is the *modular decomposition* of a graph, which recursively partitions the graph into *modules* whose members have the same neighborhood outside of the module. Contracting these modules, we obtain a *quotient graph* describing the adjacencies between the modules. Having isolated the dense part to the modules, measuring the complexity of the quotient graph by standard graph parameters such as treewidth yields e.g. the parameter *modular-treewidth* (mod-tw), a natural intermediate step between treewidth and clique-width. While modular-treewidth is not as general as clique-width, the algorithms for computing treewidth transfer to modular-treewidth, yielding e.g. reasonable constant-factor approximations for modular-treewidth in single-exponential time, whereas for clique-width we are currently only able to obtain approximations with exponential error.

We obtain the first tight running times for connectivity problems parameterized by modular-treewidth. To do so, we lift the algorithms using the cut-and-count-technique from treewidth to modular-treewidth. A crucial observation is that all vertices inside a module will be connected by choosing a single vertex from a neighboring module. In some cases, this observation is strong enough to lift the treewidth-based algorithms to modular-treewidth for free, i.e., the base α of the running time does not increase, showing that we can deal with a greater generality in input structure at no cost in complexity for these problems.

Theorem 1 (informal). *There are one-sided error Monte-Carlo algorithms that, given a decomposition witnessing modular-treewidth k, can solve*

- Steiner Tree *in time* $\mathcal{O}^*(3^k)$,
- Connected Dominating Set *in time* $\mathcal{O}^*(4^k)$.

These bases are optimal under SETH, by known results of Cygan et al. [10].

However, in other cases the interplay of the connectivity constraint and the remaining problem constraints does increase the complexity for modular-treewidth compared to treewidth. In these cases, we provide new algorithms adapting the cut-and-count-technique to this more intricate setting.

[1] The $\mathcal{O}^*$-notation hides polynomial factors in the input size.

[2] The hypothesis that for every $\delta < 1$, there is some q such that q-Satisfiability cannot be solved in time $\mathcal{O}(2^{\delta n})$, where n is the number of variables.

Theorem 2 (informal). *There are one-sided error Monte-Carlo algorithms that, given a decomposition witnessing modular-treewidth k, can solve* Connected Vertex Cover *and* Feedback Vertex Set *in time $\mathcal{O}^*(5^k)$.*

Both problems can be solved in time $\mathcal{O}^*(3^k)$ parameterized by treewidth [11]. In contrast, Vertex Cover (without the connectivity constraint) has complexity $\mathcal{O}^*(2^k)$ with respect to treewidth [22] and modular-treewidth simultaneously.

For these latter two problems, we provide new lower bounds to show that the bases are optimal under SETH. However, we do not need the full power of the modular decomposition to prove the lower bounds. The modular decomposition allows for *recursive* partitioning. When instead allowing for only a single level of partitioning and limited complexity inside the modules, we obtain parameters called *twinclass-pathwidth* (tc-pw) and *twinclass-treewidth*.

Theorem 3. *Unless SETH fails,* Connected Vertex Cover *and* Feedback Vertex Set *cannot be solved in time $\mathcal{O}^*((5-\varepsilon)^{\text{tc-pw}})$ for any $\varepsilon > 0$.*

As twinclass-pathwidth is a larger parameter than modular-treewidth, the lower bounds of Theorem 3 transfer to modular-treewidth.

The obtained results on connectivity problems parameterized by modular-treewidth are situated in the larger context of a research program aimed at determining the optimal running times for connectivity problems relative to width-parameters of differing generality, thus quantifying the price of generality in this setting. The known results are summarized in Table 1. Beyond the results for treewidth by Cygan et al. [10,11], Bojikian et al. [8] obtain tight results for the more restrictive *cutwidth* by either providing faster algorithms resulting from combining cut-and-count with the rank-based approach or by showing that the same lower bounds already hold for cutwidth. Hegerfeld and Kratsch [16] consider *clique-width* and obtain tight results for Connected Vertex Cover and Connected Dominating Set. Their algorithms combine cut-and-count with several nontrivial techniques to speed up dynamic programming on clique-expressions, where the interaction between cut-and-count and clique-width can yield more involved states compared to modular-treewidth, as clique-width is more general. These algorithms are complemented by new lower bound constructions following similar high-level principles as for modular-treewidth, but allow for more flexibility in the gadget design due to the mentioned generality. However, the techniques of Hegerfeld and Kratsch [16] for clique-width yield tight results for fewer problems compared to the present work; in particular, the optimal bases for Steiner Tree and Feedback Vertex Set parameterized by clique-width are currently not known.

Related Work. We survey some more of the literature on parameterized algorithms for connectivity problems relative to dense width-parameters. Bergougnoux [2] has applied cut-and-count to several width-parameters based on structured neighborhoods such as clique-width, rank-width, or mim-width. Building upon the rank-based approach of Bodlaender et al. [6], Bergougnoux and Kanté [4] obtain single-exponential running times $\mathcal{O}^*(\alpha^{\text{cw}})$ for a large

Table 1. Optimal running times of connectivity problems with respect to various width-parameters listed in increasing generality. The results in the penultimate column are obtained in this paper. The "?" denotes cases, where an algorithm with single-exponential running time is known by Bergougnoux and Kanté [4], but a gap between the lower bound and algorithm remains.

Parameters	cutwidth	treewidth	modular-tw	clique-width
Connected Vertex Cover	$\mathcal{O}^*(2^k)$	$\mathcal{O}^*(3^k)$	$\mathcal{O}^*(5^k)$	$\mathcal{O}^*(6^k)$
Connected Dominating Set	$\mathcal{O}^*(3^k)$	$\mathcal{O}^*(4^k)$	$\mathcal{O}^*(4^k)$	$\mathcal{O}^*(5^k)$
Steiner Tree	$\mathcal{O}^*(3^k)$	$\mathcal{O}^*(3^k)$	$\mathcal{O}^*(3^k)$	?
Feedback Vertex Set	$\mathcal{O}^*(2^k)$	$\mathcal{O}^*(3^k)$	$\mathcal{O}^*(5^k)$	?
References	[8]	[10,11]	here	[16]

class of connectivity problems parameterized by clique-width (cw). The same authors [5] also generalize this approach to other dense width-parameters via structured neighborhoods. All these works deal with general Connected (σ, ρ)-Dominating Set problems capturing a wide range of problems; this generality of problems (and parameters) comes at the cost of yielding running times that are far from optimal for specific problem-parameter combinations, e.g., the first article [2] is the most optimized for clique-width and obtains the running time $\mathcal{O}^*((2^{4+\omega})^{\text{cw}}) \geq \mathcal{O}^*(64^{\text{cw}})$, where ω is the matrix multiplication exponent [1], for Connected Dominating Set. Bergougnoux et al. [3] obtain XP algorithms parameterized by mim-width for problems expressible in a logic that can also capture connectivity constraints. Beyond dense width-parameters, cut-and-count has also been applied to the parameters branchwidth [28] and treedepth [14,26].

Our version of modular-treewidth was first used by Bodlaender and Jansen for Maximum Cut [7]. Several papers [21,24,27] also use the name modular-treewidth, but use it to refer to what we call *twinclass-treewidth.* In particular, Lampis [21] obtains tight results under SETH for q-Coloring with respect to twinclass-treewidth and clique-width. Hegerfeld and Kratsch [15] obtain tight results for Odd Cycle Transversal parameterized by twinclass-pathwidth and clique-width and for Dominating Set parameterized by twinclass-cutwidth. Kratsch and Nelles [20] combine modular decompositions with tree-depth in various ways and obtain parameterized algorithms for various efficiently solvable problems.

Organization. In Sect. 2 we discuss the general preliminaries and in Sect. 3 the cut-and-count-technique. We sketch Theorem 1 in Sect. 4 and Theorem 2 in Sect. 5. Everything marked with $\star$ has a more detailed version in the full version [17]; in particular, the lower bounds of Theorem 3 are completely contained in the full version due to space constraints.

2 Preliminaries

For two integers a, b we write $a \equiv_c b$ to indicate equality modulo $c \in \mathbb{N}$. We use Iverson's bracket notation: for a boolean predicate p, we have that $[p]$ is 1 if p

is true and 0 otherwise. For a function f we denote by $f[v \mapsto \alpha]$ the function $(f \setminus \{(v, f(v))\}) \cup \{(v, \alpha)\}$, viewing f as a set. By $\mathbb{F}_2$ we denote the field of two elements. For $n_1, n_2 \in \mathbb{Z}$, we write $[n_1, n_2] = \{x \in \mathbb{Z} : n_1 \leq x \leq n_2\}$ and $[n_2] = [1, n_2]$. For a function $f\colon V \to \mathbb{Z}$ and a subset $W \subseteq V$, we write $f(W) = \sum_{v \in W} f(v)$. Note that for functions $g\colon A \to B$, where $B \not\subseteq \mathbb{Z}$, and a subset $A' \subseteq A$, we still denote the *image of A' under g* by $g(A') = \{g(v) : v \in A'\}$. If $f\colon A \to B$ is a function and $A' \subseteq A$, then $f\big|_{A'}$ denotes the *restriction* of f to A' and for a subset $B' \subseteq B$, we denote the *preimage of B' under f* by $f^{-1}(B') = \{a \in A : f(a) \in B'\}$. The *power set* of a set A is denoted by $\mathcal{P}(A)$.

We use common graph-theoretic notation and the essentials of parameterized complexity. For two disjoint vertex subsets $A, B \subseteq V$, we define $E_G(A, B) = \{\{a, b\} \in E(G) : a \in A, b \in B\}$ and adding a *join* between A and B means adding an edge between every vertex in A and every vertex in B. We denote the *number of connected components* of G by $\texttt{cc}(G)$. A *cut* of G is a partition $V = V_L \cup V_R$, $V_L \cap V_R = \emptyset$, of its vertices into two parts.

Quotients and Twinclasses. Let Π be a partition of $V(G)$. The *quotient graph G/Π* is given by $V(G/\Pi) = \Pi$ and $E(G/\Pi) = \{\{B_1, B_2\} \subseteq \Pi : B_1 \neq B_2, \exists u \in B_1, v \in B_2\colon \{u, v\} \in E(G)\}$. We say that two vertices u, v are *twins* if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. The equivalence classes of this relation are called *twinclasses* and we let $\Pi_{tc}(G)$ denote the partition of $V(G)$ into twinclasses. A twinclass of size at least 2 either induces an independent set (false twins) or a clique (true twins). We define the *twinclass-treewidth* and *twinclass-pathwidth* of G by $\text{tc-tw}(G) = \text{tw}(G/\Pi_{tc}(G))$ and $\text{tc-pw}(G) = \text{pw}(G/\Pi_{tc}(G))$, respectively. The parameters twinclass-treewidth and twinclass-pathwidth were considered before under the name modular treewidth and modular pathwidth [21,24,27]. We use the prefix *twinclass* instead of *modular* to distinguish from the quotient graph arising from a *modular partition* of G.

Modular Decomposition. A vertex set $M \subseteq V(G)$ is a *module* of G if $N(v) \setminus M = N(w) \setminus M$ for every pair $v, w \in M$ of vertices in M. The modules $\emptyset$, $V(G)$, and all singletons are called *trivial*; A graph is *prime* if it only admits trivial modules. A module M is *proper* if $M \neq V(G)$. For two disjoint modules $M_1, M_2 \in \mathcal{M}(G)$, either $\{\{v, w\} : v \in M_1, w \in M_2\} \subseteq E(G)$ or $\{\{v, w\} : v \in M_1, w \in M_2\} \cap E(G) = \emptyset$; in the first case, M_1 and M_2 are *adjacent* and in the second case, they are *nonadjacent*. A module M is *strong* if for every module M' of G we have $M \cap M' = \emptyset$, $M \subseteq M'$, or $M' \subseteq M$. The family of nonempty strong modules is denoted $\mathcal{M}_{\text{tree}}(G)$, which can be arranged as the *modular decomposition tree* via the inclusion-relation. We freely switch between viewing $\mathcal{M}_{\text{tree}}(G)$ as a set family or as the modular decomposition tree of G; in the latter case, we refer also to the modules as *nodes*. Every graph G containing at least two vertices can be uniquely partitioned into a set of inclusion-maximal nonempty strong modules $\Pi_{mod}(G) = \{M_1, \ldots, M_\ell\}$, with $\ell \geq 2$, called *canonical modular partition*; $\Pi_{mod}(G)$ is undefined for $|V(G)| \leq 1$. For $M \in \mathcal{M}_{\text{tree}}(G)$ with $|M| \geq 2$, we write $\texttt{children}(M) = \Pi_{mod}(G[M])$ as the sets in $\Pi_{mod}(G[M])$ are precisely the children of M in the modular decomposition tree; if $|M| = 1$, then $\texttt{children}(M) = \emptyset$. Forming the *quotient graph* $G^q_M = G[M]/\Pi_{mod}(G[M])$ *at* M, there are three possible cases:

Theorem 4 ([12]). *If* $|M| \geq 2$, *then exactly one of the following holds:*

- ***Parallel node:*** $G[M]$ *is not connected and* G^q_M *is an independent set,*
- ***Series node:*** *the complement* $\overline{G[M]}$ *is not connected and* G^q_M *is a clique,*
- ***Prime node:*** $\Pi_{mod}(G[M])$ *consists of the inclusion-maximal proper modules of* $G[M]$ *and* G^q_M *is prime.*

We define the family $\mathcal{H}_p(G) = \{G^q_M : M \in \mathcal{M}_{\text{tree}}(G), |M| \geq 2, G^q_M \text{ is prime}\}$ and the *modular-pathwidth* by $\text{mod-pw}(G) = \max(2, \max_{H \in \mathcal{H}_p(G)} \text{pw}(H))$ and the *modular-treewidth* by $\text{mod-tw}(G) = \max(2, \max_{H \in \mathcal{H}_p(G)} \text{tw}(H))$. The modular decomposition tree can be computed in time $\mathcal{O}(n + m)$, see e.g. Tedder et al. [29] or the survey by Habib and Paul [13]. Running a treewidth-algorithm, such as the approximation algorithm of Korhonen [19], on every graph in $\mathcal{H}_p(G)$ and observing that[3] $|\mathcal{M}_{\text{tree}}(G)| \leq 2n$, which also bounds the total number of vertices appearing in quotient graphs, we obtain the following.

Theorem 5. *There is an algorithm, that given an* n*-vertex graph* G *and an integer* k, *in time* $2^{\mathcal{O}(k)}n$ *either outputs a tree decomposition of width at most* $2k+1$ *for every prime quotient graph* $G^q_M \in \mathcal{H}_p(G)$ *or determines that* $\text{mod-tw}(G) > k$.

Let $M \in \mathcal{M}_{\text{tree}}(G) \setminus \{V\}$ and $M^\uparrow \in \mathcal{M}_{\text{tree}}(G)$ be its *parent module*. We have $M \in \texttt{children}(M^\uparrow) = \Pi_{mod}(G[M^\uparrow])$, hence M appears as a vertex of the quotient graph $G^q_{M^\uparrow}$; we also denote this vertex by v^q_M. We define the *projection* at $M^\uparrow$ by $\pi_{M^\uparrow} : M^\uparrow \to V(G^q_{M^\uparrow})$ with $\pi_{M^\uparrow}(v) = v^q_M$ whenever $v \in M \in \Pi_{mod}(G[M^\uparrow])$.

Tree Decompositions ($\star$). The definition of (very nice) tree decompositions and treewidth is given in the full version of the paper. The very nice tree decompositions of Cygan et al. [11] augment the nice tree decompositions of Kloks [18] by empty root and leaf bags and every edge is introduced exactly once in an *introduce edge* bag.

Lemma 6 ([11]). *Any tree decomposition of* G *can be converted into a very nice tree decomposition of* G *with the same width in polynomial time.*

Given a very nice tree decomposition $(\mathcal{T}^q_{M^\uparrow}, (\mathbb{B}^q_t)_{t \in V(\mathcal{T}^q_{M^\uparrow})})$ of the quotient graph $G^q_{M^\uparrow}$, we associate to every node $t \in V(\mathcal{T}^q_{M^\uparrow})$ a subgraph $G^q_t = (V^q_t, E^q_t)$ of $G^q_{M^\uparrow}$ in the standard way. Based on the vertex subsets of the quotient graph $G^q_{M^\uparrow}$, we define vertex subsets of the original graph $G[M^\uparrow]$ as follows: $\mathbb{B}_t = \pi^{-1}_{M^\uparrow}(\mathbb{B}^q_t) =$

[3] The modular decomposition tree has n leaves and every internal node has at least two children, hence $|\mathcal{M}_{\text{tree}}(G)| \leq 2n$.

$\bigcup_{v^q_M \in \mathbb{B}^q_t} M$ and $V_t = \pi^{-1}_{M^\uparrow}(V^q_t) = \bigcup_{v^q_M \in V^q_t} M$. We also transfer the edge set as follows

$$E_t = \bigcup_{v^q_M \in V^q_t} E(G[M]) \cup \bigcup_{\{v^q_{M_1}, v^q_{M_2}\} \in E^q_t} \{\{u_1, u_2\} : u_1 \in M_1 \wedge u_2 \in M_2\},$$

allowing us to define the graph $G_t = (V_t, E_t)$ associated to any node $t \in V(\mathcal{T}^q_{M^\uparrow})$.

Parameter Relationships ($\star$). The standard definitions of clique-width, $\mathrm{cw}(G)$, and linear clique-width, $\text{lin-cw}(G)$, can be found in the full version. We have the following relationships between the considered parameters.

Lemma 7 ($\star$). *We have* $\mathrm{cw}(G) \leq \text{mod-pw}(G) + 2 \leq \max(2, \text{tc-pw}(G)) + 2$ *and* $\text{mod-tw}(G) \leq \max(2, \text{tc-tw}(G))$ *for every graph* G*.*

Note that Theorem 7 can only hold for modular-pathwidth and not modular-treewidth, as already for treewidth, Corneil and Rotics [9] show that for every k there exists a graph G_k with treewidth k and clique-width exponential in k.

Theorem 8 ([15]). *We have* $\mathrm{cw}(G) \leq \text{lin-cw}(G) \leq \text{tc-pw}(G) + 4 \leq \mathrm{pw}(G) + 4$*.*

Problem Definitions

Connected Vertex Cover

> **Input:** An undirected graph $G = (V, E)$, a cost function $\mathbf{c} \colon V \to \mathbb{N} \setminus \{0\}$ and an integer $\overline{b}$.
> **Question:** Is there a set $X \subseteq V$, $\mathbf{c}(X) \leq \overline{b}$, such that $G - X$ contains no edges and $G[X]$ is connected?

Connected Dominating Set

> **Input:** An undirected graph $G = (V, E)$, a cost function $\mathbf{c} \colon V \to \mathbb{N} \setminus \{0\}$ and an integer $\overline{b}$.
> **Question:** Is there a set $X \subseteq V$, $\mathbf{c}(X) \leq \overline{b}$, such that $N[X] = V$ and $G[X]$ is connected?

(Node) Steiner Tree

> **Input:** An undirected graph $G = (V, E)$, a set of terminals $K \subseteq V$, a cost function $\mathbf{c} \colon V \to \mathbb{N} \setminus \{0\}$ and an integer $\overline{b}$.
> **Question:** Is there a set $X \subseteq V$, $\mathbf{c}(X) \leq \overline{b}$, such that $K \subseteq X$ and $G[X]$ is connected?

Feedback Vertex Set

> **Input:** An undirected graph $G = (V, E)$, a cost function $\mathbf{c} \colon V \to \mathbb{N} \setminus \{0\}$ and an integer $\overline{b}$.
> **Question:** Is there a set $X \subseteq V$, $\mathbf{c}(X) \leq \overline{b}$, such that $G - X$ contains no cycles?

3 Cut and Count for Modular-Treewidth

Let $G = (V, E)$ denote a connected graph. For easy reference, we repeat the key definition and lemmas of the cut-and-count-technique [11] here. A cut (V_L, V_R) of an undirected graph $G = (V, E)$ is *consistent* if $u \in V_L$ and $v \in V_R$ implies $\{u, v\} \notin E$, i.e., $E_G(V_L, V_R) = \emptyset$. A *consistently cut subgraph* of G is a pair $(X, (X_L, X_R))$ such that $X \subseteq V$ and (X_L, X_R) is a consistent cut of $G[X]$. We denote the set of consistently cut subgraphs of G by $\mathcal{C}(G)$.

Lemma 9 ([11]). *Let $X \subseteq V$ be a subset of vertices. The number of consistently cut subgraphs $(X, (X_L, X_R))$ is equal to $2^{\mathtt{cc}(G[X])}$.*

Fix some $M^\uparrow \in \mathcal{M}_{\text{tree}}(G)$ with $|M^\uparrow| \geq 2$ and $X \subseteq M^\uparrow$ with $|\pi_{M^\uparrow}(X)| \geq 2$, i.e., X intersects at least two child modules of $M^\uparrow$, for this section. A simple exchange argument shows that the connectivity of $G[X]$ is not affected by the precise intersection $X \cap M$, $M \in \mathtt{children}(M^\uparrow)$, but only whether $X \cap M$ is empty or not. This observation allows us to reduce checking the connectivity of $G[X]$ to the quotient graph at $M^\uparrow$, as $G^q_{M^\uparrow}$ is isomorphic to the induced subgraph of G obtained by picking one vertex from each child module of $M^\uparrow$.

Lemma 10 ($\star$). *If $G[X]$ is connected, then for any $v^q_M \in \pi_{M^\uparrow}(X)$ and $\emptyset \neq Y \subseteq M$, the graph $G[(X \setminus M) \cup Y]$ is connected. Furthermore, $G[X]$ is connected if and only if $G^q_{M^\uparrow}[\pi_{M^\uparrow}(X)]$ is connected.*

Theorem 10 shows that we do not need to consider *heterogeneous* cuts, i.e., $(X, (X_L, X_R)) \in \mathcal{C}(G)$ with $X_L \cap M \neq \emptyset$ and $X_R \cap M \neq \emptyset$ for some module $M \in \Pi_{mod}(G)$, since we can assume that $|X \cap M| \leq 1$.

Definition 11. Let $M^\uparrow \in \mathcal{M}_{\text{tree}}(G)$. We say that a cut (X_L, X_R), with $X_L \cup X_R \subseteq M^\uparrow$, is *$M^\uparrow$-homogeneous* if $X_L \cap M = \emptyset$ or $X_R \cap M = \emptyset$ for every $M \in \mathtt{children}(M^\uparrow)$. We may just say that (X_L, X_R) is *homogeneous* when $M^\uparrow$ is clear from the context. We define for every subgraph G' of G the set $\mathcal{C}^{hom}_{M^\uparrow}(G') = \{(X, (X_L, X_R)) \in \mathcal{C}(G') : (X_L, X_R) \text{ is } M^\uparrow\text{-homogeneous}\}$.

Combining Theorem 9 with Theorem 10, the connectivity of $G[X]$ can be determined by counting $M^\uparrow$-homogeneous consistent cuts of $G[X]$ modulo 4.

Lemma 12 ($\star$). *We have*

$$|\{(X_L, X_R) : (X, (X_L, X_R)) \in \mathcal{C}^{hom}_{M^\uparrow}(G)\}| = 2^{\mathtt{cc}(G^q_{M^\uparrow}[\pi_{M^\uparrow}(X)])}.$$

Furthermore, $G[X]$ is connected if and only if $|\{(X_L, X_R) : (X, (X_L, X_R)) \in \mathcal{C}^{hom}_{M^\uparrow}(G)\}| \neq 0 \mod 4$.

With the isolation lemma we avoid unwanted cancellations in the cut-and-count-technique at the cost of introducing randomization.

Definition 13. A function $\mathbf{w} \colon U \to \mathbb{Z}$ *isolates* a set family $\mathcal{F} \subseteq \mathcal{P}(U)$ if there is a unique $S' \in \mathcal{F}$ with $\mathbf{w}(S') = \min_{S \in \mathcal{F}} \mathbf{w}(S)$, where for subsets X of U we define $\mathbf{w}(X) = \sum_{u \in X} \mathbf{w}(u)$.

Lemma 14 (Isolation Lemma, [25]). *Let $\emptyset \neq \mathcal{F} \subseteq \mathcal{P}(U)$ be a set family over a universe U. Let $N \in \mathbb{N}$ and for each $u \in U$ choose a weight $\mathbf{w}(u) \in [N]$ uniformly and independently at random. Then $\mathbb{P}[\mathbf{w}$ isolates $\mathcal{F}] \geq 1 - |U|/N$.*

4 Reduction to Treewidth ($\star$)

We sketch the ideas behind Theorem 1. For both problems, Steiner Tree and Connected Dominating Set, we use Theorem 10 to reduce the problems to a quotient graph and apply the treewidth-algorithms of Cygan et al. [11]. As Theorem 10 only applies to sets intersecting at least two modules, we separately search for solutions contained in a single module. We also handle the special cases of series and parallel nodes via special polynomial-time algorithms or recursing in the modular decomposition tree depending on the node type and problem.

Assuming that the topmost quotient graph $G^q_V = G/\Pi_{mod}(G)$ is prime and we are searching for solutions X intersecting at least two modules, i.e., $|\pi_V(X)| \geq 2$, we provide more details. First, consider such a Steiner Tree instance $(G, K, \mathbf{c}, \overline{b})$. Theorem 10 implies that the only sensible intersections are $X \cap M \in \{\emptyset, \{v_M\}, K \cap M\}$ for $M \in \Pi_{mod}(G)$, where v_M is a vertex of minimum cost inside M. In particular, we distinguish whether $K \cap M = \emptyset$ or $K \cap M \neq \emptyset$; in the former case, we can assume $X \cap M \in \{\emptyset, \{v_M\}\}$ and in the latter $X \cap M \in \{\emptyset, K \cap M\}$. This motivates the reduction to the quotient graph: we set $K^q = \pi_V(K)$ and $\mathbf{c}^q(v^q_M) = \mathbf{c}(K \cap M) = \sum_{v \in K \cap M} \mathbf{c}(v)$ if $K \cap M \neq \emptyset$ and $\mathbf{c}^q(v^q_M) = \mathbf{c}(v_M)$ otherwise, compressing the cost of $K \cap M$ into a single vertex or choosing a vertex of minimum cost respectively. Then, the instance $(G, K, \mathbf{c}, \overline{b})$ is equivalent to $(G^q_V, K^q, \mathbf{c}^q, \overline{b})$ and we can run a weighted variant of the Steiner Tree algorithm of Cygan et al. [11] on the latter instance.

The reduction for Connected Dominating Set uses a very similar principle by considering the cheapest vertex inside each module, which works as a module is completely dominated as soon as we take at least one vertex in an adjacent module. However, for Connected Dominating Set we might need to call the treewidth-algorithm due to more complicated recursions several times and not only once. This makes the algorithm more technical, as we have to be careful with the randomization to avoid increasing the error probability. By observing that an isolating weight function induces an isolating weight function for appropriate subinstances, we are able to maintain the error probability.

In the context of kernelization, Luo [23] uses similar reductions for Steiner Tree and Connected Dominating Set parameterized by modular-width, however, these reductions do not consider the weighted setting and do not have to contend with randomization.

5 Dynamic Programming Algorithms

In this section, we prove Theorem 2, by presenting novel algorithms using the cut-and-count-technique for Connected Vertex Cover and Feedback Vertex Set.

5.1 Connected Vertex Cover

We assume that $G = (V, E)$ is connected and contains at least two vertices, hence V cannot be a parallel node. We only consider cost functions $\mathbf{c}$ that are polynomially bounded in $|V|$. To solve Connected Vertex Cover, we begin by computing some optimum (possibly nonconnected) vertex cover Y_M with respect to $\mathbf{c}\big|_M$ for every module $M \in \Pi_{mod}(G)$ such that $G[M]$ contains an edge. If $G[M]$ contains no edges, then we set $Y_M = \{v^*_M\}$, where $v^*_M \in M$ is a vertex minimizing the cost inside M, i.e., $v^*_M := \arg\min_{v \in M} \mathbf{c}(v)$. The vertex covers can be computed in time $\mathcal{O}^*(2^{\text{mod-tw}(G)})$ by using a straightforward algorithm presented in the full version.

Definition 15. Let $X \subseteq V$ be a vertex subset. We say that X is *nice* if for every module $M \in \Pi_{mod}(G)$ it holds that $X \cap M \in \{\emptyset, Y_M, M\}$.

Via exchange arguments, in particular Theorem 10, we show that it is sufficient to only consider nice vertex covers. This shows that only a constant number of states per module in the dynamic programming algorithm are necessary.

Lemma 16 ($\star$). *If there exists a connected vertex cover X of G with $|\pi_V(X)| \geq 2$, then there exists a connected vertex cover X' of G that is nice with $|\pi_V(X')| \geq 2$ and $\mathbf{c}(X') \leq \mathbf{c}(X)$.*

Some simple observations allow us to handle the edge case of connected vertex covers contained in a single module $M \in \Pi_{mod}(G)$ and series nodes. We proceed by looking for connected vertex covers X with $|\pi_V(X)| \geq 2$ when $G^q := G^q_V = G/\Pi_{mod}(G)$ is prime. We are given a very nice tree decomposition $(\mathcal{T}^q, (\mathbb{B}^q_t)_{t \in V(\mathcal{T}^q)})$ of $G^q := G^q_V = G/\Pi_{mod}(G)$ of width k. Making use of Theorem 16 and Theorem 12, we can employ the cut-and-count-technique and perform dynamic programming along the tree decomposition $\mathcal{T}^q$ and extend our partial solutions module by module. The cut-and-count-formulation of the problem is as follows. For any subgraph G' of G, we define the *relaxed solutions* $\mathcal{R}(G') = \{X \subseteq V(G') : X \text{ is a nice vertex cover of } G'\}$ and *the cut solutions* $\mathcal{Q}(G') = \{(X, (X_L, X_R)) \in \mathcal{C}^{hom}_V(G') : X \in \mathcal{R}(G')\}$. For the isolation lemma, cf. Theorem 14, we sample a weight function $\mathbf{w} \colon V \to [2n]$ uniformly at random. We track the cost $\mathbf{c}(X)$, the weight $\mathbf{w}(X)$, and the number of intersected modules $|\pi_V(X)|$ of each partial solution $(X, (X_L, X_R))$. Accordingly, we define $\mathcal{R}^{\overline{c},\overline{w},\overline{m}}(G') = \{X \in \mathcal{R}(G') : \mathbf{c}(X) = \overline{c}, \mathbf{w}(X) = \overline{w}, |\pi_V(X)| = \overline{m}\}$ and $\mathcal{Q}^{\overline{c},\overline{w},\overline{m}}(G') = \{(X, (X_L, X_R)) \in \mathcal{Q}(G') : X \in \mathcal{R}^{\overline{c},\overline{w},\overline{m}}(G')\}$ for all $\overline{c} \in [0, \mathbf{c}(V)], \overline{w} \in [0, \mathbf{w}(V)], \overline{m} \in [0, |\Pi_{mod}(G)|]$.

As discussed, to every node $t \in V(\mathcal{T}^q)$ we associate a subgraph $G^q_t = (V^q_t, E^q_t)$ of G^q in the standard way, which in turn gives rise to a subgraph $G_t = (V_t, E_t)$ of G. The subgraphs G_t grow module by module and are considered by the dynamic program, hence we define $\mathcal{R}^{\overline{c},\overline{w},\overline{m}}_t = \mathcal{R}^{\overline{c},\overline{w},\overline{m}}(G_t)$ and $\mathcal{Q}^{\overline{c},\overline{w},\overline{m}}_t = \mathcal{Q}^{\overline{c},\overline{w},\overline{m}}(G_t)$ for all $\overline{c}$, $\overline{w}$, and $\overline{m}$. We will compute the sizes of the sets $\mathcal{Q}^{\overline{c},\overline{w},\overline{m}}_t$ by dynamic programming over the tree decomposition $\mathcal{T}^q$, but to do so we need to parameterize the partial solutions by their state on the current bag.

Disregarding the side of the cut, Theorem 16 tells us that each module $M \in \Pi_{mod}(G)$ has one of three states for some $X \in \mathcal{R}_t^{\overline{c},\overline{w},\overline{m}}$, namely $X \cap M \in \{\emptyset, Y_M, M\}$. Since we are considering homogeneous cuts there are two possibilities if $X \cap M \neq \emptyset$; $X \cap M$ is contained in the left side of the cut or in the right side. Thus, there are five total choices. We define **states** $= \{\mathbf{0}, \mathbf{1}_L, \mathbf{1}_R, \mathbf{A}_L, \mathbf{A}_R\}$ with $\mathbf{1}$ denoting that the partial solution contains at least one vertex, but not all, of the module and with $\mathbf{A}$ denoting that the partial solution contains all vertices of the module; the subscript denotes the side of the cut.

A function of the form $f \colon \mathbb{B}_t^q \to$ **states** is called *t-signature*. For every node $t \in V(\mathcal{T}^q)$, cost $\overline{c}$, weight $\overline{w}$, number of modules $\overline{m}$, and t-signature f, the family $\mathcal{A}_t^{\overline{c},\overline{w},\overline{m}}(f)$ consists of all $(X, (X_L, X_R)) \in \mathcal{Q}_t^{\overline{c},\overline{w},\overline{m}}$ that satisfy for all $v_M^q \in \mathbb{B}_t^q$:

$$f(v_M^q) = \mathbf{0} \leftrightarrow X \cap M = \emptyset,$$
$$f(v_M^q) = \mathbf{1}_L \leftrightarrow X_L \cap M = Y_M \neq M, \qquad f(v_M^q) = \mathbf{1}_R \leftrightarrow X_R \cap M = Y_M \neq M,$$
$$f(v_M^q) = \mathbf{A}_L \leftrightarrow X_L \cap M = M, \qquad f(v_M^q) = \mathbf{A}_R \leftrightarrow X_R \cap M = M.$$

Recall that by considering homogeneous cuts, we have that $X_L \cap M = \emptyset$ or $X_R \cap M = \emptyset$ for every module $M \in \Pi_{mod}(G)$. We use the condition $Y_M \neq M$ for the states $\mathbf{1}_L$ and $\mathbf{1}_R$ to ensure a well-defined state for modules of size 1. Note that the sets $\mathcal{A}_t^{\overline{c},\overline{w},\overline{m}}(f)$, ranging over f, partition $\mathcal{Q}_t^{\overline{c},\overline{w},\overline{m}}$ due to the consideration of nice vertex covers and homogeneous cuts.

Our goal is to compute the size of $\mathcal{A}_{\hat{r}}^{\overline{c},\overline{w},\overline{m}}(\emptyset) = \mathcal{Q}_{\hat{r}}^{\overline{c},\overline{w},\overline{m}} = \mathcal{Q}^{\overline{c},\overline{w},\overline{m}}(G)$, where $\hat{r}$ is the root vertex of the tree decomposition $\mathcal{T}^q$, modulo 4 for all $\overline{c}$, $\overline{w}$, $\overline{m}$. By Theorem 12, there is a connected vertex cover X of G with $\mathbf{c}(X) = \overline{c}$ and $\mathbf{w}(X) = \overline{w}$ if the result is nonzero. We present the recurrences for the various bag types to compute $A_t^{\overline{c},\overline{w},\overline{m}}(f) = |\mathcal{A}_t^{\overline{c},\overline{w},\overline{m}}(f)|$; if not stated otherwise, then $t \in V(\mathcal{T}^q)$, $\overline{c} \in [0, \mathbf{c}(V)]$, $\overline{w} \in [0, \mathbf{w}(V)]$, $\overline{m} \in [0, |\Pi_{mod}(G)|]$, and f is a t-signature. We set $A_t^{\overline{c},\overline{w},\overline{m}}(f) = 0$ whenever at least one of $\overline{c}$, $\overline{w}$, or $\overline{m}$ is negative.

Leaf Bag: We have that $\mathbb{B}_t^q = \mathbb{B}_t = \emptyset$ and t has no children. The only possible t-signature is $\emptyset$ and the only possible partial solution is $(\emptyset, (\emptyset, \emptyset))$. Hence, we only need to check the tracker values: $A_t^{\overline{c},\overline{w},\overline{m}}(\emptyset) = 1$ if $\overline{c} = \overline{w} = \overline{m} = 0$ and 0 otherwise.

Introduce Vertex Bag: We have that $\mathbb{B}_t^q = \mathbb{B}_s^q \cup \{v_M^q\}$, where $s \in V(\mathcal{T}^q)$ is the only child of t and $v_M^q \notin \mathbb{B}_s^q$. Hence, $\mathbb{B}_t = \mathbb{B}_s \cup M$. We have to consider all possible interactions of a partial solution with M, though since we are considering nice vertex covers these interactions are quite restricted. To formulate the recurrence, we let, as an exceptional case, f be an s-signature and not a t-signature. Since no edges of the quotient graph G^q incident to v_M^q are introduced yet, we only have to check some edge cases and update the trackers when introducing v_M^q:

$$A_t^{\overline{c},\overline{w},\overline{m}}(f[v_M^q \mapsto \mathbf{s}]) = \begin{cases} [G[M] \text{ is edgeless}] A_s^{\overline{c},\overline{w},\overline{m}}(f), & \mathbf{s} = \mathbf{0}, \\ [|M| > 1] A_s^{\overline{c}-\mathbf{c}(Y_M),\overline{w}-\mathbf{w}(Y_M),\overline{m}-1}(f), & \mathbf{s} \in \{\mathbf{1}_L, \mathbf{1}_R\}, \\ A_s^{\overline{c}-\mathbf{c}(M),\overline{w}-\mathbf{w}(M),\overline{m}-1}(f), & \mathbf{s} \in \{\mathbf{A}_L, \mathbf{A}_R\}. \end{cases}$$

Introduce Edge Bag: Let $\{v^q_{M_1}, v^q_{M_2}\}$ denote the introduced edge. We have that $\{v^q_{M_1}, v^q_{M_2}\} \subseteq \mathbb{B}^q_t = \mathbb{B}^q_s$. The edge $\{v^q_{M_1}, v^q_{M_2}\}$ corresponds to adding a join between the modules M_1 and M_2. We need to filter all solutions whose states at M_1 and M_2 are not consistent with M_1 and M_2 being adjacent. There are two possible reasons: either not all edges between M_1 and M_2 are covered, or the introduced edges go across the homogeneous cut. We implement this via the helper function $\mathrm{cons}\colon \mathbf{states} \times \mathbf{states} \to \{0, 1\}$ defined by $\mathrm{cons}(\mathbf{s}_1, \mathbf{s}_2) = [\{\mathbf{s}_1, \mathbf{s}_2\} \cap \{\mathbf{A}_L, \mathbf{A}_R\} \neq \emptyset][\mathbf{s}_1 \in \{\mathbf{1}_L, \mathbf{A}_L\} \to \mathbf{s}_2 \notin \{\mathbf{1}_R, \mathbf{A}_R\}][\mathbf{s}_1 \in \{\mathbf{1}_R, \mathbf{A}_R\} \to \mathbf{s}_2 \notin \{\mathbf{1}_L, \mathbf{A}_L\}]$. The recurrence is given by $A^{\overline{c},\overline{w},\overline{m}}_t(f) = \mathrm{cons}(f(v^q_{M_1}), f(v^q_{M_2}))A^{\overline{c},\overline{w},\overline{m}}_s(f)$.

Forget Vertex Bag: We have that $\mathbb{B}^q_t = \mathbb{B}^q_s \setminus \{v^q_M\}$, where $v^q_M \in \mathbb{B}^q_s$ and $s \in V(\mathcal{T}^q)$ is the only child of t. Here, we only need to forget the state at v^q_M and accumulate the contributions from the different states v^q_M could assume. As the states are disjoint no overcounting happens: $A^{\overline{c},\overline{w},\overline{m}}_t(f) = \sum_{\mathbf{s} \in \mathbf{states}} A^{\overline{c},\overline{w},\overline{m}}_s(f[v \mapsto \mathbf{s}])$.

Join Bag: We have $\mathbb{B}^q_t = \mathbb{B}^q_{s_1} = \mathbb{B}^q_{s_2}$, where $s_1, s_2 \in V(\mathcal{T}^q)$ are the children of t. Two partial solutions, one at s_1, and the other at s_2, can be combined when the states agree on all $v^q_M \in \mathbb{B}^q_t$. Since we update the trackers already at introduce vertex bags, we need to take care that the values of the modules in the bag are not counted twice. For this sake, define $S^f = \bigcup_{v^q_M \in f^{-1}(\{\mathbf{1}_L, \mathbf{1}_R\})} Y_M \cup \bigcup_{v^q_M \in f^{-1}(\{\mathbf{A}_L, \mathbf{A}_R\})} M$ for all t-signatures f. This definition satisfies $X \cap \mathbb{B}_t = S^f$ for all $(X, (X_L, X_R)) \in \mathcal{A}^{\overline{c},\overline{w},\overline{m}}(f)$. Then, the recurrence is given by

$$A^{\overline{c},\overline{w},\overline{m}}_t(f) = \sum_{\substack{\overline{c}_1 + \overline{c}_2 = \overline{c} + \mathbf{c}(S^f) \\ \overline{w}_1 + \overline{w}_2 = \overline{w} + \mathbf{w}(S^f)}} \; \sum_{\overline{m}_1 + \overline{m}_2 = \overline{m} + (|\mathbb{B}^q_t| - f^{-1}(\mathbf{0}))} A^{\overline{c}_1,\overline{w}_1,\overline{m}_1}_{s_1}(f) A^{\overline{c}_2,\overline{w}_2,\overline{m}_2}_{s_2}(f).$$

Theorem 17 ($\star$). *There exists a Monte-Carlo algorithm that given a tree decomposition of width at most k for every prime quotient graph $H \in \mathcal{H}_p(G)$, solves* Connected Vertex Cover *in time $\mathcal{O}^*(5^k)$. The algorithm cannot give false positives and may give false negatives with probability at most* $1/2$.

Proof (sketch). We compute the sets Y_M for all $M \in \Pi_{mod}(G)$ in time $\mathcal{O}^*(2^k)$ using a straightforward algorithm described in the full version [17]. For the remainder, we only need to consider the topmost quotient graph $G^q_V = G/\Pi_{mod}(G)$. In polynomial time, we can find an optimum connected vertex cover that is contained in a single module and also deal with the case where V is a parallel or series node. It remains to handle the case that G^q_V is prime. In that case, we run the presented dynamic programming algorithm along the very nice tree decomposition of G^q_V with the sampled weight function $\mathbf{w}\colon V \to [2n]$. The algorithm returns true if there are $\overline{c} \in [0, \overline{b}]$, $\overline{w} \in [0, \mathbf{w}(V)]$, $\overline{m} \in [2, |\Pi_{mod}(G)|]$ such that $A^{\overline{c},\overline{w},\overline{m}}_{\hat{r}}(\emptyset) \not\equiv_4 0$, where $\hat{r}$ is the root of the tree decomposition, otherwise the algorithm returns false.

We skip the correctness proofs of the recurrences here. Setting $\mathcal{S}^{\overline{c},\overline{w},\overline{m}} = \{X \in \mathcal{R}^{\overline{c},\overline{w},\overline{m}}(G) : G[X] \text{ is connected}\}$, we have that $A^{\overline{c},\overline{w},\overline{m}}_{\hat{r}}(\emptyset) = |\mathcal{Q}^{\overline{c},\overline{w},\overline{m}}(G)| = \sum_{X \in \mathcal{R}^{\overline{c},\overline{w},\overline{m}}(G)} 2^{\mathrm{cc}(G[X])} \equiv_4 2|\mathcal{S}^{\overline{c},\overline{w},\overline{m}}|$ by Theorem 12, so the algorithm cannot return false negatives. Common isolation lemma arguments and Theorem 16 show that we return correctly with probability at least 1/2.

By assumption the cost function **c** is polynomially bounded, hence there are $\mathcal{O}^*(5^k)$ table entries to compute. Furthermore, every recurrence can be computed in polynomial time, hence the running time of the algorithm follows. □

5.2 Feedback Vertex Set ($\star$)

The algorithm for Feedback Vertex Set is the most technical part of this work; we give a summary of the main ideas. The first step is to solve the complementary problem Induced Forest instead of Feedback Vertex Set as that matches the usage of the cut-and-count-technique better. By analyzing the structure of an induced forest X with respect to a module M, we see that there are four sensible possibilities for the intersection $X \cap M$: it is empty, a single vertex, an independent set, or an induced forest containing an edge.

In particular, the last possibility leads to many technical issues, as we must solve Induced Forest for every module $M \in \mathcal{M}_{\text{tree}}(G)$, whereas for Connected Vertex Cover only problems without connectivity constraints needed to be solved for all modules. Due to the randomization of cut-and-count, some subproblems might be solved incorrectly. We have to ensure that this does not cause an issue and that the error probability stays constant. In particular, we need to carefully define the subproblems as they rely on the output of the previous subproblems. We sample a global weight function once and, assuming that the weight function is isolating, we analyze where the restricted weight function remains isolating and hence which subproblems are solved correctly.

For the independent set state we distinguish whether a neighboring module is intersected (degree-1) or not (degree-0), as degree-2 or higher leads to a cycle. The degree-1 independent set state and the induced forest state behave the same with respect to any future neighboring module, as the intersection of X with this module has to be empty, otherwise X would contain a cycle. Hence, we would like to collapse these two states into a single one, this however causes issues in the join nodes. Instead we allow the induced forest state only for modules that are already forgotten; to be precise, when a degree-0 independent set is forgotten, we can safely exchange the independent set with an induced forest without introducing any cycles. Hence, the induced forest state will not affect the table sizes of the dynamic program.

Until now, we have not specified the cut sides of the modules. Since we can work with homogeneous cuts, any non-empty state naively turns into two states, one for the left side and one for the right side; this would yield 7 states in total. However, we can avoid specifying the cut side for the independent set states; in the degree-0 case, the cut side is independent of any other modules; in the degree-1 case, we can inherit the cut side from the unique non-empty neighboring module. Therefore, we obtain only the desired 5 states in total.

References

1. Alman, J., Williams, V.V.: A refined laser method and faster matrix multiplication. In: Marx, D. (ed.) Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, 10–13 January 2021, pp. 522–539. SIAM (2021). https://doi.org/10.1137/1.9781611976465.32
2. Bergougnoux, B.: Matrix decompositions and algorithmic applications to (hyper)graphs. Ph.D. thesis, University of Clermont Auvergne, Clermont-Ferrand, France (2019). https://tel.archives-ouvertes.fr/tel-02388683
3. Bergougnoux, B., Dreier, J., Jaffke, L.: A logic-based algorithmic meta-theorem for mim-width, pp. 3282–3304 (2023). https://doi.org/10.1137/1.9781611977554.ch125
4. Bergougnoux, B., Kanté, M.M.: Fast exact algorithms for some connectivity problems parameterized by clique-width. Theor. Comput. Sci. **782**, 30–53 (2019). https://doi.org/10.1016/j.tcs.2019.02.030
5. Bergougnoux, B., Kanté, M.M.: More applications of the d-neighbor equivalence: acyclicity and connectivity constraints. SIAM J. Discret. Math. **35**(3), 1881–1926 (2021). https://doi.org/10.1137/20M1350571
6. Bodlaender, H.L., Cygan, M., Kratsch, S., Nederlof, J.: Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. Inf. Comput. **243**, 86–111 (2015). https://doi.org/10.1016/j.ic.2014.12.008
7. Bodlaender, H.L., Jansen, K.: On the complexity of the maximum cut problem. Nord. J. Comput. **7**(1), 14–31 (2000)
8. Bojikian, N., Chekan, V., Hegerfeld, F., Kratsch, S.: Tight bounds for connectivity problems parameterized by cutwidth. In: Berenbrink, P., Bouyer, P., Dawar, A., Kanté, M.M. (eds.) 40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, 7–9 March 2023, Hamburg, Germany. LIPIcs, vol. 254, pp. 14:1–14:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2023). https://doi.org/10.4230/LIPIcs.STACS.2023.14
9. Corneil, D.G., Rotics, U.: On the relationship between clique-width and treewidth. SIAM J. Comput. **34**(4), 825–847 (2005). https://doi.org/10.1137/S0097539701385351
10. Cygan, M., Nederlof, J., Pilipczuk, M., Pilipczuk, M., van Rooij, J.M.M., Wojtaszczyk, J.O.: Solving connectivity problems parameterized by treewidth in single exponential time. CoRR abs/1103.0534 (2011)
11. Cygan, M., Nederlof, J., Pilipczuk, M., Pilipczuk, M., van Rooij, J.M.M., Wojtaszczyk, J.O.: Solving connectivity problems parameterized by treewidth in single exponential time. ACM Trans. Algorithms **18**(2), 17:1–17:31 (2022). https://doi.org/10.1145/3506707
12. Gallai, T.: Transitiv orientierbare graphen. Acta Math. Hungar. **18**(1–2), 25–66 (1967)
13. Habib, M., Paul, C.: A survey of the algorithmic aspects of modular decomposition. Comput. Sci. Rev. **4**(1), 41–59 (2010). https://doi.org/10.1016/j.cosrev.2010.01.001
14. Hegerfeld, F., Kratsch, S.: Solving connectivity problems parameterized by treedepth in single-exponential time and polynomial space. In: Paul, C., Bläser, M. (eds.) 37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, 10–13 March 2020, Montpellier, France. LIPIcs, vol. 154, pp. 29:1–29:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.STACS.2020.29

15. Hegerfeld, F., Kratsch, S.: Towards exact structural thresholds for parameterized complexity. In: Dell, H., Nederlof, J. (eds.) 17th International Symposium on Parameterized and Exact Computation, IPEC 2022, 7–9 September 2022, Potsdam, Germany. LIPIcs, vol. 249, pp. 17:1–17:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.IPEC.2022.17
16. Hegerfeld, F., Kratsch, S.: Tight algorithms for connectivity problems parameterized by clique-width. CoRR abs/2302.03627 (2023). https://doi.org/10.48550/arXiv.2302.03627, accepted at ESA 2023
17. Hegerfeld, F., Kratsch, S.: Tight algorithms for connectivity problems parameterized by modular-treewidth. CoRR abs/2302.14128 (2023). https://doi.org/10.48550/arXiv.2302.14128
18. Kloks, T. (ed.): Treewidth. LNCS, vol. 842. Springer, Heidelberg (1994). https://doi.org/10.1007/BFb0045375
19. Korhonen, T.: A single-exponential time 2-approximation algorithm for treewidth. In: 62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, 7–10 February 2022, pp. 184–192. IEEE (2021). https://doi.org/10.1109/FOCS52979.2021.00026
20. Kratsch, S., Nelles, F.: Efficient parameterized algorithms on graphs with heterogeneous structure: combining tree-depth and modular-width. CoRR abs/2209.14429 (2022). https://doi.org/10.48550/arXiv.2209.14429
21. Lampis, M.: Finer tight bounds for coloring on clique-width. SIAM J. Discret. Math. **34**(3), 1538–1558 (2020). https://doi.org/10.1137/19M1280326
22. Lokshtanov, D., Marx, D., Saurabh, S.: Known algorithms on graphs of bounded treewidth are probably optimal. ACM Trans. Algorithms **14**(2), 13:1–13:30 (2018). https://doi.org/10.1145/3170442
23. Luo, W.: Polynomial turing compressions for some graph problems parameterized by modular-width. CoRR abs/2201.04678 (2022)
24. Mengel, S.: Parameterized compilation lower bounds for restricted CNF-formulas. In: Creignou, N., Le Berre, D. (eds.) SAT 2016. LNCS, vol. 9710, pp. 3–12. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40970-2_1
25. Mulmuley, K., Vazirani, U.V., Vazirani, V.V.: Matching is as easy as matrix inversion. Combinatorica **7**(1), 105–113 (1987). https://doi.org/10.1007/BF02579206
26. Nederlof, J., Pilipczuk, M., Swennenhuis, C.M.F., Węgrzycki, K.: Hamiltonian cycle parameterized by Treedepth in single exponential time and polynomial space. In: Adler, I., Müller, H. (eds.) WG 2020. LNCS, vol. 12301, pp. 27–39. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60440-0_3
27. Paulusma, D., Slivovsky, F., Szeider, S.: Model counting for CNF formulas of bounded modular treewidth. Algorithmica **76**(1), 168–194 (2016). https://doi.org/10.1007/s00453-015-0030-x
28. Pino, W.J.A., Bodlaender, H.L., van Rooij, J.M.M.: Cut and count and representative sets on branch decompositions. In: Guo, J., Hermelin, D. (eds.) 11th International Symposium on Parameterized and Exact Computation, IPEC 2016, 24–26 August 2016, Aarhus, Denmark. LIPIcs, vol. 63, pp. 27:1–27:12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016). https://doi.org/10.4230/LIPIcs.IPEC.2016.27
29. Tedder, M., Corneil, D., Habib, M., Paul, C.: Simpler linear-time modular decomposition via recursive factorizing permutations. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008. LNCS, vol. 5125, pp. 634–645. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70575-8_52

Cops and Robber - When Capturing Is Not Surrounding

Paul Jungeblut, Samuel Schneider, and Torsten Ueckerdt(✉)

Karlsruhe Institute of Technology, Karlsruhe, Germany
{paul.jungeblut,torsten.ueckerdt}@kit.edu,
samuel.schneider@student.kit.edu

Abstract. We consider "surrounding" versions of the classic Cops and Robber game. The game is played on a connected graph in which two players, one controlling a number of cops and the other controlling a robber, take alternating turns. In a turn, each player may move each of their pieces. The robber always moves between adjacent vertices. Regarding the moves of the cops we distinguish four versions that differ in whether the cops are on the vertices or the edges of the graph and whether the robber may move on/through them. The goal of the cops is to surround the robber, i.e., occupying all neighbors (vertex version) or incident edges (edge version) of the robber's current vertex. In contrast, the robber tries to avoid being surrounded indefinitely. Given a graph, the so-called *cop number* denotes the minimum number of cops required to eventually surround the robber.

We relate the different cop numbers of these versions and prove that none of them is bounded by a function of the classical cop number and the maximum degree of the graph, thereby refuting a conjecture by Crytser, Komarov and Mackey [Graphs and Combinatorics, 2020].

1 Introduction

Cops and Robber is a well-known combinatorial game played by two players on a graph $G = (V, E)$. The robber player controls a single robber, which we shall denote by r, whereas the cop player controls k cops, denoted $c_1, \dots, c_k$, for some specified integer $k \geq 1$. The players take alternating turns, and in each turn may perform one move with each of their pieces (the single robber or the k cops). In the classical game (and also many of its variants) the vertices of G are the possible positions for the pieces, while the edges of G model the possible moves. Let us remark that no piece is forced to move, i.e., there is no *zugzwang*. On each vertex there can be any number of pieces.

The game begins with the cop player choosing vertices as the starting positions for the k cops $c_1, \dots, c_k$. Then (seeing the cops' positions) the robber player places r on a vertex of G as well. The cop player wins if the cops *capture* the robber, which in the classical version means that at least one cop stands on the same vertex as the robber. On the other hand, the robber player wins if the robber can avoid being captured indefinitely.

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 403–416, 2023.
https://doi.org/10.1007/978-3-031-43380-1_29

The *cop number* denoted by $c(G)$ of a given connected[1] graph $G = (V, E)$ is the smallest k for which k cops can capture the robber in a finite number of turns. Clearly, every graph satisfies $1 \leq c(G) \leq |V|$.

We consider several versions of the classical Cops and Robber game. In some of these the cops are placed on the edges of G and allowed to move to an *adjacent* edge (that is an edge sharing an endpoint) during their turn. In all our versions the robber acts as in the original game but loses the game if he is *surrounded*[2] by the cops, meaning that they have to occupy all adjacent vertices or incident edges. At all times, let us denote by v_r the vertex currently occupied by the robber. Specifically, we define the following versions of the game, each specifying the possible positions for the cops and the exact surrounding condition:

Vertex Version Cops are positioned on vertices of G (like the robber). They surround the robber if there is a cop on each neighbor of v_r. Let $c_V(G)$ denote the smallest number of cops needed to eventually surround the robber.

Edge Version Cops are positioned on edges of G. A cop on an edge e can move to any edge e' sharing an endpoint with e during its turn. The cops surround the robber if there is a cop on each edge incident to v_r. Let $c_E(G)$ denote the smallest number of cops needed to eventually surround the robber.

In both versions above, the robber sits on the vertices of G and moves along the edges of G. Due to the winning condition for the cops being a full surround, the robber may come very close to, say, a single cop without being threatened. As this can feel counterintuitive, let us additionally consider a restrictive version of each game where we constrain the possible moves for the robber when cops are close by. These *restrictive* versions are given by the following rules:

Restrictive Vertex Version After the robber's turn, there may not be any cop on v_r. In particular, the robber may not move onto a vertex occupied by a cop. Additionally, if a cop moves onto v_r, then in his next turn the robber must leave that vertex.

Restrictive Edge Version. The robber may not move along an edge that is currently occupied by a cop.

We denote the cop numbers of the restrictive versions by putting an additional "r" in the subscript, i.e., the smallest number of cops needed to eventually surround the robber in these versions is $c_{V,\mathrm{r}}(G)$ and $c_{E,\mathrm{r}}(G)$, respectively.

Clearly, the restrictive versions are favorable for cops as they only restrict the robber. Consequently, the corresponding cop numbers are always at most their non-restrictive counterparts. Thus, for every connected graph G we have

$$c_{V,\mathrm{r}}(G) \leq c_V(G) \quad \text{and} \quad c_{E,\mathrm{r}}(G) \leq c_E(G). \tag{1}$$

[1] Cops cannot move between different connected components, so the cop number of any graph is the sum over all components. We thus consider connected graphs only.

[2] To distinguish between the classical and our versions, we use the term *capture* to express that a cop occupies the same vertex as the robber. In contrast, a *surround* always means that all neighbors, respectively incident edges, are occupied.

A recent conjecture by Crytser, Komarov and Mackey [7] states that the cop number in the restrictive edge version can be bounded from above by the classical cop number and the maximum degree of the graph:

Conjecture 1 ([7]). For every connected graph G we have $c_{E,\mathrm{r}}(G) \leq c(G) \cdot \Delta(G)$.

Prałat [14] verified Conjecture 1 for the random graph $G(n, p)$, i.e., the graph on n vertices where each possible edge is chosen independently with probability p, for some ranges of p. Let us note that Conjecture 1, if true, would strengthen a theorem by Crytser, Komarov and Mackey [7] stating that $c_{E,\mathrm{r}}(G) \leq \gamma(G) \cdot \Delta(G)$, where $\gamma(G)$ denotes the size of a smallest dominating set in G.

1.1 Our Results

Our main contribution is to disprove Conjecture 1. In fact, we prove that there are graphs G for which none of the surrounding cop numbers can be bounded by any function of $c(G)$ and $\Delta(G)$. This proves that the classical game of Cops and Robber is sometimes fundamentally different from all its surrounding versions.

Theorem 2. *There is an infinite family of connected graphs G with classical cop number $c(G) = 2$ and $\Delta(G) = 3$ while neither $c_V(G)$, $c_{V,\mathrm{r}}(G)$, $c_E(G)$ nor $c_{E,\mathrm{r}}(G)$ can be bounded by any function of $c(G)$ and the maximum degree $\Delta(G)$.*

Additionally, we relate the different surrounding versions to each other. Equation (1) already gives an upper bound for the cop numbers in the restrictive versions in terms of their corresponding unrestrictive cop numbers. To complete the picture, our second contribution is to prove several lower and upper bounds for different combinations:

Theorem 3. *Each of the following holds (assuming G to be connected):*

1. $\forall G : c_V(G) \leq \Delta(G) \cdot c_{V,\mathrm{r}}(G)$ *and* $\exists G : c_V(G) \geq \Delta(G) \cdot c_{V,\mathrm{r}}(G)$
2. $\forall G : c_E(G) \leq \Delta(G) \cdot c_{E,\mathrm{r}}(G)$ *and* $\exists G : c_E(G) \geq \Delta(G)/4 \cdot c_{E,\mathrm{r}}(G)$
3. $\forall G : c_V(G) \leq 2 \cdot c_E(G)$ *and* $\exists G : c_V(G) \geq 2 \cdot (c_E(G) - 1)$
4. $\forall G : c_{V,\mathrm{r}}(G) \leq 2 \cdot c_{E,\mathrm{r}}(G)$ *and* $\exists G : c_{V,\mathrm{r}}(G) \geq c_{E,\mathrm{r}}(G)$
5. $\forall G : c_E(G) \leq \Delta(G) \cdot c_V(G)$ *and* $\exists G : c_E(G) \geq \Delta(G)/12 \cdot c_V(G)$
6. $\forall G : c_{E,\mathrm{r}}(G) \leq \Delta(G) \cdot c_{V,\mathrm{r}}(G)$ *and* $\exists G : c_{E,\mathrm{r}}(G) \geq \Delta(G)/48 \cdot c_{V,\mathrm{r}}(G)$

Note that all lower and upper bounds from Theorem 3 are tight up to a small additive or multiplicative constant. We prove the upper bounds in Sect. 2. The main idea is the same for all six inequalities: Given a winning strategy for a set of cops in one version we can simulate the strategy in any other version. Afterwards, in Sect. 3, we consider the lower bounds by constructing explicit families of graphs with the desired surrounding cop numbers. While some lower bounds already follow from standard graph families (like complete bipartite graphs), others need significantly more involved constructions. For example we construct a family of graphs from a set of $k - 1$ mutually orthogonal Latin squares (see Sect. 3.3 for a definition).

Trivial Bounds. Clearly, for the robber to be surrounded at a vertex v_r, at least $\deg(v_r)$ cops are required in all considered versions. This already gives that the minimum degree $\delta(G)$ of G is a lower bound on the cop numbers of G in these cases (stated in [6] for $c_{V,\mathrm{r}}(G)$). In fact, the robber could restrict himself to a subgraph H of G of highest minimum degree, which gives the lower bound of $d(G) := \max\{\delta(H) \mid H \subseteq G\}$, also called the *degeneracy* of G. Moreover, in those versions in which the robber could simply start at a vertex of *highest* degree and *never move*, that is in all but the restrictive vertex version, we get the *maximum degree* $\Delta(G)$ of G as a lower bound (stated in [7] for $c_{E,\mathrm{r}}(G)$):

Observation 4. *For every connected graph $G = (V, E)$ we have*

- $c_{V,\mathrm{r}}(G) \geq d(G)$ *as well as*
- $c_V(G) \geq \Delta(G)$, $c_E(G) \geq \Delta(G)$ *and* $c_{E,\mathrm{r}}(G) \geq \Delta(G)$.

1.2 Related Work

The game of Cops and Robber was introduced by Nowakowski and Winkler [13] as well as Quilliot [15] almost forty years ago. Both consider the case where a single cop chases the robber. The version with many cops and therefore also the notion of the cop number $c(G)$ was introduced shortly after by Aigner and Fromme [1], already proving that $c(G) \leq 3$ for all connected planar graphs G. Their version is nowadays considered the standard version of the game and we refer to it as the *classical version* throughout the paper. The most important open question regarding $c(G)$ is Meyniel's conjecture stating that a connected n-vertex graph G has $c(G) \in O(\sqrt{n})$ [2,8]. It is known to be EXPTIME-complete to decide whether $c(G) \leq k$ (for k being part of the input) [12].

By now, countless different versions of the game with their own cop numbers have been considered, see for example these books on the topic [3,4].

The restrictive vertex version was introduced by Burgess et al. [6]. They prove bounds for $c_{V,\mathrm{r}}(G)$ in terms of the clique number $\omega(G)$, the independence number $\alpha(G)$ and the treewidth $\mathrm{tw}(G)$, as well as considering several interesting graph families. They also show that deciding whether $c_{V,\mathrm{r}}(G) \leq k$ can be decided in polynomial time for every fixed value of k. The complexity is unknown for k being part of the input. Bradshaw and Hosseini [5] extend the study of $c_{V,\mathrm{r}}(G)$ to graphs of bounded genus, proving (among other results) that $c_{V,\mathrm{r}}(G) \leq 7$ for every connected planar graph G. See the bachelor's thesis of Schneider [16] for several further results on $c_{V,\mathrm{r}}(G)$ (including a version with zugzwang).

The restrictive edge version was introduced even more recently by Crytser, Komarov and Mackey [7] (under the name *containment variant*). Besides stating Conjecture 1, which is verified for some graphs by Prałat [14], they give several bounds on $c_{E,\mathrm{r}}(G)$ for different families of graphs.

To the best of our knowledge, $c_V(G)$ and $c_E(G)$ were not considered before.

In the light of the (restrictive) vertex and edge versions one might also define a *face version* for embedded planar graphs. Here the cops would occupy the faces and surround the robber if they occupy all faces incident to v_r. A restrictive face version could be that the robber must not move along an edge with either one or both incident faces being occupied by a cop. This version was introduced recently by Ha, Jungeblut and Ueckerdt [9]. Despite their similar motivation, the face versions seem to behave differently than the vertex or edge versions.

In each version, one might also add *zugzwang*, i.e., the obligation to actually move during one's turn. We are not aware of any results about this in the literature.

1.3 Outline of the Paper

Section 2 proves the upper bounds from Theorem 3. Then, in Sect. 3 we give constructions implying the corresponding lower bounds. Finally, in Sect. 4, we disprove Conjecture 1. Proofs of statements marked with ($\star$) are in the full version [10].

2 Relating the Different Versions

In this section we prove the upper bounds from Theorem 3. The main idea is always that a sufficiently large group of cops in one version can simulate a single cop in another version. We denote by $N_G(v)$ and $N_G[v]$ the open and closed neighborhood of vertex v in G, respectively.

Proof (only Item 1 of Theorem 3, others in the full version [10]). Let G be an arbitrary connected graph.

1. $c_V(G) \leq \Delta(G) \cdot c_{V,\mathrm{r}}(G)$: Let $\mathcal{S}_{V,\mathrm{r}}(G)$ be a winning strategy for $k \in \mathbb{N}$ restrictive vertex cops $c_1, \ldots, c_k$ in G. For $i \in \{1, \ldots, k\}$, replace c_i by a group of $\Delta(G)$ non-restrictive vertex cops $C_i := \{c_i^1, \ldots, c_i^{\Delta(G)}\}$. Initially all cops in C_i start at the same vertex as c_i and whenever c_i moves to an adjacent vertex, all cops in C_i copy its move.
   If the restrictive cops $c_1, \ldots, c_k$ arrive in a position where they surround the robber, then he is also surrounded by the groups of cops $C_1, \ldots, C_k$. It remains to consider the case that the robber ends their turn on vertex v currently occupied by some group C_i (a move that would be forbidden in the restrictive version). Then the cops in C_i can spread to the up to $\Delta(G)$ neighbors of v in G, thereby surrounding the robber. □

Corollary 5. *For every graph G the surrounding cop numbers $c_V(G)$, $c_E(G)$, $c_{V,\mathrm{r}}(G)$ and $c_{E,\mathrm{r}}(G)$ are always within a factor of $2\Delta(G)$ of each other.*

Proof. In each of the six upper bounds stated in Theorem 3 the number of cops increases by at most a factor of $\Delta(G)$. In all cases this is obtained by simulating a winning strategy of one surrounding variant by (groups of) cops in another variant. The only cases where two such simulations need to be combined is when changing both, the cop type (vertex-cops/edge-cops) and the restrictiveness. It is easy to check that in all but one combination the number of cops increases by at most a factor of $2\Delta(G)$. The only exception is when a winning strategy for restricted vertex-cops is simulated by unrestricted edge-cops, where the number of cops would increase by a factor of $\Delta(G)^2$. However, looking at the proof of Theorem 3, we can see that both simulations replace a single cop by a group of $\Delta(G)$ cops. In this particular case it suffices to do this replacement just once.

We remark that all upper bounds proven above in Theorem 3 result from simulating a winning strategy of another surrounding version. In the next section we show that, surprisingly, these are indeed (asymptotically) tight. After all, it would seem natural that every version comes with its own unique winning strategy (more involved than just simulating one from a different version).

3 Explicit Graphs and Constructions

In this section we shall mention or construct several families of graphs with some extremal behavior for their corresponding classical and surrounding cop numbers. Together, these graphs prove all lower bounds stated in Theorem 3.

3.1 Complete Bipartite Graphs

We start by considering complete bipartite graphs. They already serve to directly prove two of the lower bounds from Theorem 3 and also appear again in proofs in the subsequent subsections.

Proposition 6 ($\star$). *For any complete bipartite graph G it holds that $c(G) = \min\{2, \delta(G)\}$, $c_{V,\mathrm{r}}(G) = \delta(G)$ and $c_V(G) = c_{E,\mathrm{r}}(G) = c_E(G) = \Delta(G)$.*

Let us consider two special cases of Proposition 6 for all $\Delta \in \mathbb{N}$: First, the star $K_{\Delta,1}$ has $c_{V,\mathrm{r}}(K_{\Delta,1}) = 1$ while $c_V(K_{\Delta,1}) = \Delta$, thus proving the lower bound in Item 1 of Theorem 3. Second, the complete bipartite graph $K_{\Delta,\Delta}$ has $c_{V,\mathrm{r}}(K_{\Delta,\Delta}) = c_{E,\mathrm{r}}(K_{\Delta,\Delta}) = \Delta$, thus proving the lower bound in Item 4 of Theorem 3.

3.2 Regular Graphs with Leaves

Our first construction takes a connected k-regular graph H and attaches a set of ℓ new degree-1-vertices (*leaves*) to each vertex. Depending on H, k and ℓ we can give several bounds on the surrounding cop numbers of the resulting graph.

Lemma 7. *Let $H = (V_H, E_H)$ be a k-regular connected graph and let $G = (V_G, E_G)$ be the graph obtained from H by attaching to each vertex $v \in V_H$ a set of ℓ new leaves for some $\ell \geq 0$. Then each of the following holds:*

1. $c_V(G) \geq \begin{cases} k(k+\ell-1) & \textit{if } \mathrm{girth}(H) \geq 7 \\ (k+1)\ell & \textit{always} \end{cases}$
2. $c_{V,\mathrm{r}}(G) = \max\{c_{V,\mathrm{r}}(H), k+1\}$

3. $c_E(G) \geq \begin{cases} k(k+\ell-1) & \textit{if } \mathrm{girth}(H) \geq 6 \\ k\ell & \textit{if } \mathrm{girth}(H) \geq 4 \\ \frac{1}{2}(k+1)\ell & \textit{always} \end{cases}$

4. $c_{E,\mathrm{r}}(G) = \max\{c_{E,\mathrm{r}}(H), k+\ell\}$

Proof (only Item 1, others in the full version [10]*).* Note that most claimed inequalities hold trivially for the case that $\ell = 0$ (many lower bounds become 0 while others follow from $G = H$ in this case). Only the two cases requiring $\mathrm{girth}(H) \geq 6$, respectively $\mathrm{girth}(H) \geq 7$, are not directly clear. However, their proofs below hold for $\ell = 0$ as well. In all other cases, we implicitly assume $\ell \geq 1$ to avoid having to handle additional corner cases.

1. To prove the lower bounds on $c_V(G)$, consider any configuration of cops on the vertices of G. For a vertex $v \in V_H$ of G, let A_v be the set consisting of v and all leaves that are attached to it, i.e., $A_v = \{v\} \cup (N_G[v] \setminus V_H)$. We call a vertex $v \in V_H$ *safe* if there are fewer than ℓ cops on A_v in G. Note that if the robber ends his turn on a safe vertex, then the cops cannot surround him in their next turn. Let v_r be the current position of the robber. If the total number of cops is less than $(k+1)\ell$, then at least one of the $k+1$ vertices in the closed neighborhood $N_H[v_r]$ of v_r is safe, as no cop can be in A_v and A_w for $v \neq w$. Thus, the robber always has a safe vertex to move to (or to remain on), giving him a strategy to avoid being surrounded. It follows that $c_V(G) \geq (k+1)\ell$.
   Now, if $\mathrm{girth}(H) \geq 7$ and the robber is on $v_r \in V_H$, then we consider for each neighbor v of v_r in V_H additionally the set $B_v = N_G[N_G(v) \setminus \{v_r\}]$, i.e., all vertices w with $\mathrm{dist}(w, v) \leq 2$ except from $N_G[v_r] \setminus \{v\}$. Since $\mathrm{girth}(H) \geq 7$, we have that $B_v \cap B_w = \emptyset$ for distinct $v, w \in N_H(v_r)$. Similar to above, we call $v \in N_H(v_r)$ *safe* if B_v contains fewer than $k+\ell-1$ cops. Again, if the robber ends his turn on a safe vertex, the cops cannot surround him in their next turn. If the total number of cops is less than $k(k+\ell-1)$, then at least one of the k neighbors of v_r in H is safe. This would give the robber a strategy to avoid being surrounded. It follows that $c_V(G) \geq k(k+\ell-1)$ in the case that $\mathrm{girth}(H) \geq 7$. □

Applied to different host graphs H above Lemma 7 yields several interesting bounds. In particular, considering Theorem 3, Corollary 8 proves the lower bound in Item 2 for even Δ and Corollary 9 proves the lower bound in Item 3.

Corollary 8 (⋆). *For every $\Delta \geq 2$ there is a connected graph G with $\Delta(G) = \Delta$ such that $c_{V,\mathrm{r}}(G) = \lfloor \frac{\Delta}{2} \rfloor + 1$, $c_V(G) = (\lfloor \frac{\Delta}{2} \rfloor + 1)\lceil \frac{\Delta}{2} \rceil$, $c_{E,\mathrm{r}}(G) = \Delta(G)$ and $c_E(G) = \lfloor \frac{\Delta}{2} \rfloor \lceil \frac{\Delta}{2} \rceil$.*

The host graph H in Corollary 8 is the complete bipartite graph $K_{k,k}$ with $k = \lfloor \frac{\Delta}{2} \rfloor$ with $h = \lceil \frac{\Delta}{2} \rceil$ leaves attached to each vertex.

Corollary 9 (⋆). *For every $\Delta \geq 2$ there is a connected graph G with $\Delta(G) = \Delta$ such that $c_V(G) = 2(\Delta - 1)$ and $c_E(G) = \Delta$.*

The host graph H in Corollary 9 is a single edge, i.e., the graph K_2 (therefore $k = 1$) with $h = \Delta - 1$ leaves attached to both vertices.

1	4	2	3
4	1	3	2
2	3	1	4
3	2	4	1

1	4	2	3
3	2	4	1
4	1	3	2
2	3	1	4

$(1,1)$	$(4,4)$	$(2,2)$	$(3,3)$
$(4,3)$	$(1,2)$	$(3,4)$	$(2,1)$
$(2,4)$	$(3,1)$	$(1,3)$	$(4,2)$
$(3,2)$	$(2,3)$	$(4,1)$	$(1,4)$

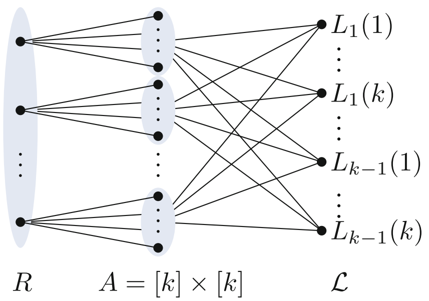


Fig. 1. Left: Two Latin squares and their juxtaposition, proving that they are orthogonal. Right: The graph G_k created from $k-1$ MOLS of order k. The vertices in R correspond to the rows of A, the middle vertices correspond to the cells of A (ordered row by row in the drawing) and the vertices in $\mathcal{L}$ correspond to the parts of the MOLS.

3.3 Graphs from Mutually Orthogonal Latin Squares

A *Latin square* of order $k \geq 1$ is a $k \times k$ array filled with numbers from $[k] = \{1, \ldots, k\}$ such that each row and each column contains each number from $[k]$ exactly once. Formally, a Latin square L is a partition $L(1) \cup \cdots \cup L(k)$ of $A = [k] \times [k]$ such that for the i-th row ($i \in [k]$) $A[i, \cdot] = \{(i, j) \in A \mid j \in [k]\}$ and every number $n \in [k]$ we have $|A[i, \cdot] \cap L(n)| = 1$, and symmetrically for the columns. See the left of Fig. 1 for two different Latin squares.

Let L_1 and L_2 be two Latin squares of order k. Their *juxtaposition* $L_1 \otimes L_2$ is the Latin square of order k that contains in each cell the ordered pair of the entries of L_1 and L_2 in that cell. We say that L_1 and L_2 are *orthogonal* if each ordered pair appears exactly once in $L_1 \otimes L_2$, i.e., if for every two distinct $n_1, n_2 \in [k]$ we have $|L_1(n_1) \cap L_2(n_2)| = 1$. It is well-known that $k-1$ *mutually orthogonal Latin squares (MOLS)* $L_1, \ldots, L_{k-1}$ (meaning that L_s and L_t are orthogonal whenever $s \neq t$) exist if and only if k is a prime power [11]. The two Latin squares in Fig. 1 (left) are indeed orthogonal, as can be seen by their juxtaposition below.

Construction of G_k. Let k be a prime power and $L_1, \ldots, L_{k-1}$ a set of $k-1$ mutually orthogonal Latin squares of order k. Let $A = [k] \times [k]$ denote the set of all positions, $R = \{A[i, \cdot] \mid i \in [k]\}$ denote the set of all rows in A, and $\mathcal{L} = \{L_s(n) \mid s \in [k-1] \wedge n \in [k]\}$ denote the set of all parts of the Latin squares $L_1, \ldots, L_{k-1}$. Let $G_k = (V, E)$ be the graph with

$$V = A \cup R \cup \mathcal{L} \quad \text{and} \quad E = \{pS \mid p \in A, S \in R \cup \mathcal{L}, p \in S\}.$$

We observe that G_k is a k-regular bipartite graph with $|A| + |R \cup \mathcal{L}| = k^2 + (k + k(k-1)) = 2k^2$ vertices with an edge between position $p \in A$ and a set $S \in R \cup \mathcal{L}$ if and only if p is in set S. See also the right of Fig. 1 for a schematic drawing.

Lemma 10 (⋆). *For a prime power k graph G_k has girth$(G_k) \geq 6$.*

Lemma 11 (⋆). *For a prime power k graph G_k has $c(G_k) = k$, $c_{V,\mathrm{r}}(G_k) \leq k+1$ and $c_E(G_k) \geq k(k-1)$.*

Remark 12. Burgess et al. [6] notice that for graphs G of many different families with a "large" value of $c_{V,\mathrm{r}}(G)$ the classical cop number $c(G)$ was "low" (often even constant). In fact, they only provide a single family of graphs (constructed from finite projective planes) where $c(G) \approx c_{V,\mathrm{r}}(G)$. They ask (Question 7 in [6]) whether graphs with large $c_{V,\mathrm{r}}(G)$ inherently possess some property that implies that $c(G)$ is low. Our graph G_k from $k-1$ MOLS satisfies $c(G_k), c_{V,\mathrm{r}}(G_k) \in \{k, k+1\}$. We interpret this as evidence that there is no such property.

3.4 Line Graphs of Complete Graphs

The *line graph* $L(G)$ of a given graph $G = (V, E)$ is the graph whose vertex set consists of the edges E of G and two vertices of $L(G)$ are connected if their corresponding edges in G share an endpoint. For $n \geq 3$, let K_n denote the complete graph on the set $[n] = \{1, \ldots, n\}$. For distinct $x, y \in [n]$, we denote by $\{x, y\}$ the vertex of $L(K_n)$ corresponding to the edge between x and y in K_n. Burgess et al. [6] showed that $c_{V,\mathrm{r}}(L(K_n)) = 2(n-2) = \delta(L(K_n))$. This is obtained by placing the cops on all vertices $\{1, x\}$ for $x \in \{2, \ldots n\}$ and $\{2, y\}$ for $y \in \{3, \ldots, n-1\}$. The cops can surround the robber in their first move.

Lemma 13 (⋆). *For every $n \geq 3$ we have $c_V(L(K_n)) = 2(n-2)$, $c_E(L(K_n)) \geq n(n-2)/3$, and $c_{E,\mathrm{r}}(L(K_n)) \geq n^2/12$.*

Stating the above bounds in terms of their maximum degree $\Delta := \Delta(L(K_n)) = 2(n-2)$, we obtain the claimed lower bounds in Items 5 and 6 of Theorem 3.

$$c_V(L(K_n)) = \Delta \qquad\qquad c_E(L(K_n)) \geq \frac{\Delta^2 + 4\Delta}{12}$$

$$c_{V,\mathrm{r}}(L(K_n)) = \Delta \qquad\qquad c_{E,\mathrm{r}}(L(K_n)) \geq \frac{\Delta^2 + 8\Delta + 16}{48}$$

4 When Capturing Is Not Surrounding

This section is devoted to the proof of Theorem 2, i.e., that none of the four surrounding cop numbers can be bounded by any function of the classical cop number and the maximum degree of the graph. In particular, we shall construct for infinitely many integers $k \geq 1$ a graph G_k with $c(G_k) = 2$ and $\Delta(G_k) = 3$, but $c_{V,\mathrm{r}}(G_k) \geq k$. Theorem 3 then implies that also $c_V(G_k)$, $c_E(G_k)$ and $c_{E,\mathrm{r}}(G_k)$ are unbounded for growing k.

The construction of G_k is quite intricate and we divide it into several steps.

The Graph H[s]. Let $s \geq 1$ be an integer and let $k = 2^{s-1}$. We start with a graph $H[s]$, which we call the *base graph*, with the following properties:

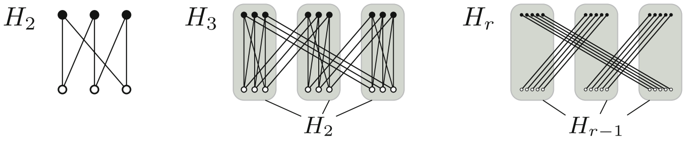


Fig. 2. Iterating $C_6 = H_2$ to obtain r-regular (bipartite) graphs H_r with $\text{girth}(H_r) \geq 5$.

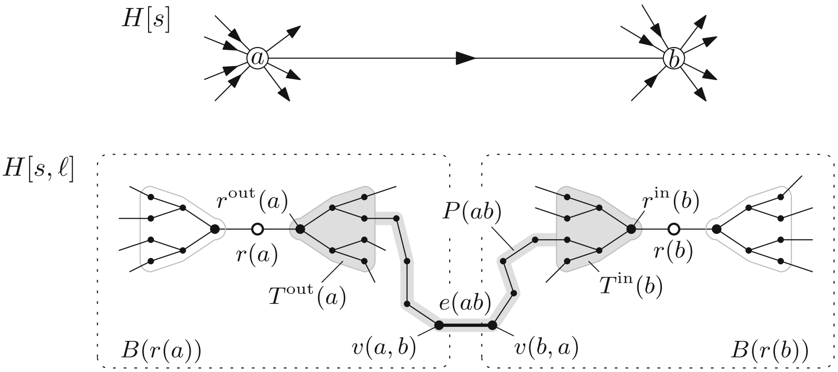


Fig. 3. Construction of $H[s,\ell]$ based on $H[s]$. A directed edge ab in $H[s]$ and the corresponding trees $T^{\text{out}}(a)$, $T^{\text{in}}(b)$, and path $P(ab)$ with middle edge $e(ab)$ in $H[s,\ell]$.

- $H[s]$ is $2k$-regular, i.e., every vertex of $H[s]$ has degree $2k$.
- $H[s]$ has girth at least 5.

There are many ways to construct such graphs $H[s]$, one being our graphs in Sect. 3.3 constructed from $2^s - 1$ mutually orthogonal Latin squares. An alternative construction (not relying on non-trivial tools) is an iteration of the 6-cycle as illustrated in Fig. 2. We additionally endow $H[s]$ with an orientation such that each vertex has exactly $k = 2^{s-1}$ outgoing and exactly $k = 2^{s-1}$ incoming edges. (For example, orient the edges according to an Eulerian tour in $H[s]$.)

The Graph *H[s,ℓ]*. Let $\ell \geq 1$ be another integer[3]. We define a graph $H[s,\ell]$ on the basis of $H[s]$ and its orientation. See Fig. 3 for an illustration with $s = 3$ and $\ell = 4$. For each vertex a in $H[s]$ take a complete balanced binary tree $T(a)$ of height $s = \log_2(k) + 1$ with root $r(a)$ and $2^s = 2k$ leaves. Let $r^{\text{in}}(a)$ and $r^{\text{out}}(a)$ denote the two children of $r(a)$ in $T(a)$, and let $T^{\text{in}}(a)$ and $T^{\text{out}}(a)$ denote the subtrees rooted at $r^{\text{in}}(a)$ and $r^{\text{out}}(a)$, respectively. We associate each of the $k = 2^{s-1}$ leaves in $T^{\text{in}}(a)$ with one of the k incoming edges at a in $H[s]$, and each of the k leaves in $T^{\text{out}}(a)$ with one of the k outgoing edges at a in $H[s]$. Finally, for each edge ab in $H[s]$ oriented from a to b, connect the associated leaf in $T^{\text{out}}(a)$ with the associated leaf in $T^{\text{in}}(b)$ by a path $P(ab)$ of length $2\ell + 1$, i.e., on 2ℓ new inner vertices. This completes the construction of $H[s,\ell]$.

[3] We shall choose $\ell \gg s$ later. So you may think of s as "short" and of ℓ as "long".

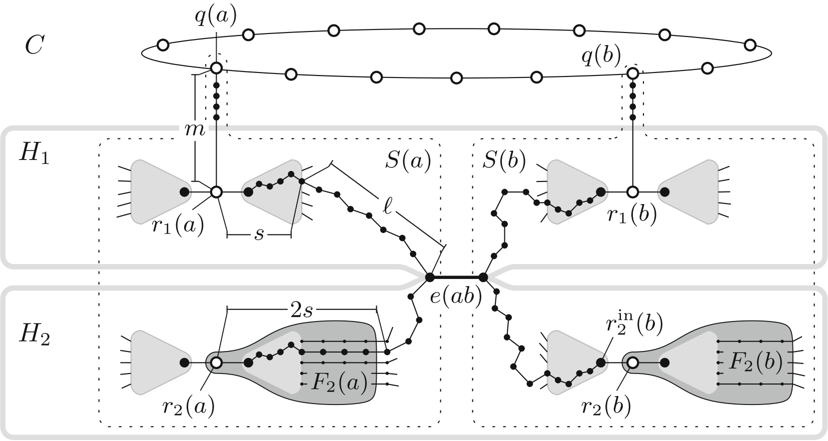


Fig. 4. Construction of $H[s,\ell,m]$ based on two copies of $H[s,\ell]$. A directed edge ab in $H[s]$ and the corresponding sets $S(a)$, $S(b)$, $F_2(a)$, etc. and edge $e(ab)$ in $H[s,\ell,m]$.

For each edge ab in $H[s]$, let $e(ab)$ denote the unique middle edge of $P(ab)$ in $H[s,\ell]$. I.e., $e(ab)$ connects a vertex in $B(r(a))$ which we denote by $v(a,b)$ with a vertex in $B(r(b))$ which we denote by $v(b,a)$; see Fig. 3.

The Graph *H[s,ℓ, m]*. Let $m \geq 1$ be yet another integer[4]. We start with two vertex-disjoint copies H_1 and H_2 of $H[s,\ell]$ and transfer our notation (such as R, $r(a)$, $v(a,b)$, etc.) for $H[s,\ell]$ to H_i for $i \in \{1,2\}$ by putting on the subscript i, e.g., R_1, $r_2(a)$, or $v_1(a,b)$. We connect H_1 and H_2 as follows: For each edge ab in $H[s]$ we identify the edge $e_1(ab)$ in H_1 with the edge $e_2(ab)$ in H_2 such that $v_1(a,b) = v_2(a,b)$ and $v_1(b,a) = v_2(b,a)$. Next, for each vertex a of $H[s]$ use the vertex $r_1(a)$ in H_1 as an endpoint for a new path $Q(a)$ of length m, and denote the other endpoint of $Q(a)$ by $q(a)$. Note that we do this only for the roots in H_1.

Finally, we connect the vertices $\{q(a) \mid a \in V(H[s])\}$ by a cycle C of length $|V(H[s])|$. This completes the construction of $H[s,\ell,m]$. See Fig. 4 for an illustration. Note that $H[s,\ell,m]$ has maximum degree 3 and degeneracy 2.

Lemma 14 (⋆). *For every $s \geq 1$, $m \geq 1$, and $\ell > |V(H[s])| + m + s$, it holds that $c(H[s,\ell,m]) \leq 2$.*

Let us give a vague idea of the winning strategy for two cops. Two cops could easily capture the robber on cycle C (including the attached paths $Q(a)$ for $a \in V(H[s])$). Thus they force him to “flee” to H_1 at some point. In a second phase they can even force him to H_2. Loosely speaking, cop c_1 stays on C to prevent the robber from getting back on C, while cop c_2 always goes towards the robber in $H[s,\ell,m] - E(C)$. Whenever the robber traverses one of the long paths $P_1(ab)$ for some $ab \in E(H[s])$, say from $T_1(a)$ towards $T_1(b)$, then c_1 can

[4] We shall choose $\ell \gg m \gg s$ later. So you may think of m as “medium”.

go in $|V(C)| + m + s < \ell$ steps along C to $q(b)$ and along $Q(b)$ and $T_1(b)$ to arrive at the other end of $P_1(ab)$ before the robber. However, to escape cop c_2 the robber must traverse a path $P_1(ab)$ eventually, with the only way to escape being to turn into H_2 at the middle edge $e(ab)$. This forces at some point the situation that the robber occupies some vertex v of H_2 and one of the cops, say c_1, occupies the corresponding copy of v in H_1. Now in a third phase the robber moves in H_2 and c_1 always copies these moves in H_1. This prohibits the robber to ever walk along a middle edge $e(ab)$ for some $ab \in E(H[s])$. But without these edges H_2 is a forest and thus cop c_2 can capture the robber in the tree component in which the robber is located.

Lemma 15 ($\star$). *For every $s \geq 1$, $m > 2s+1$, and $\ell > 3s+1$, it holds that $c_{V,\mathrm{r}}(H[s,\ell,m]) \geq k = 2^{s-1}$.*

Again, we give a vague idea for a strategy for the robber to avoid getting surrounded against $k-1$ vertex cops. To this end, the robber always stays in H_2 and moves from a root $r_2(a)$ to the next $r_2(b)$ for which the edge ab in $H[s]$ is directed from a to b. In $H[s]$, vertex a has k outgoing neighbors and we show that at least one such neighbor b is always safe for the robber to escape to, meaning that the region labeled $F_2(b)$ in Fig. 4 is free of cops when the robber reaches $r_2(b)$. However, it is quite tricky to identify this safe neighbor. Indeed, the robber has to start moving in the "right" direction down the tree $T_2(a)$ always observing the cops' response, before he can be absolutely sure which outgoing neighbor b of a is safe. With suitable choices of s, ℓ and m, the robber is fast enough at $r_2(b)$ to then choose his next destination from there. The crucial point is that the cops can "join" the robber when he traverses the middle edge $e(ab)$, but can never ensure being on any vertex of $P_2(ab)$ one step *before* the robber; and thus never surround him.

Finally, Lemmas 14 and 15 and Theorem 3 immediately give the following corollary, which proves Theorem 2.

Corollary 16. *For any $s \geq 1$, $k = 2^{s-1}$, $m \geq 2s+1$, and $\ell \geq |V(H[s])| + m + s$, the graph $G_k = H[s,\ell,m]$ has $\Delta(G_k) = 3$ and*

$$c(G_k) \leq 2, \quad c_V(G_k) \geq k, \quad c_{V,\mathrm{r}}(G_k) \geq k, \quad c_{E,\mathrm{r}}(G_k) \geq \frac{k}{2} \quad \textit{and} \quad c_E(G_k) \geq \frac{k}{2}.$$

5 Conclusion

We considered the cop numbers of four different surrounding versions of the well-known Cops and Robber game on a graph G, namely $c_V(G)$, $c_{V,\mathrm{r}}(G)$, $c_E(G)$ and $c_{E,\mathrm{r}}(G)$. Here index "V" denotes the vertex versions while index "E" denotes the edge versions, i.e., whether the cops occupy the vertices or the edges of the graph (recall that the robber always occupies a vertex). An additional index "r" stands for the corresponding restrictive versions, meaning that the robber must not end his turn on a cop or move along an edge occupied by a cop.

Only the two restrictive cop numbers have recently been considered in the literature, the vertex version $c_{V,\mathrm{r}}(G)$ in [5,6] (denoted by $\sigma(G)$ and $s(G)$) and the edge version $c_{E,\mathrm{r}}(G)$ in [7,14] (denoted by $\xi(G)$).

In this paper we related the four different versions to each other, showing that all of them lie (at most) within a factor of $2\Delta(G)$ of each other. For all combinations we presented explicit graph families showing that this is tight (up to small additive or multiplicative constants). It is an interesting open question to find out the exact constant factors for the lower and upper bounds in Theorem 3. We conjecture that all six presented upper bounds are tight (up to small *additive* constants). This would mean that optimal strategies for the cops in one surrounding version can indeed be obtained by simulating strategies from different surrounding versions.

As a second main result, we disproved a conjecture by Crytser, Komarov and Mackey [7] by constructing a family of graphs of maximum degree 3 and where the classical cop number is bounded whereas the cop number in all four surrounding versions is unbounded. It remains open to find other parameters that can be used to bound the surrounding cop numbers from above in combination with the classical cop number.

References

1. Aigner, M.S., Fromme, M.: A game of cops and robbers. Discret. Appl. Math. **8**(1), 1–12 (1984). https://doi.org/10.1016/0166-218X(84)90073-8
2. Baird, W., Bonato, A.: Meyniel's conjecture on the cop number: a survey (2013). https://arxiv.org/abs/1308.3385
3. Bonato, A.: An Invitation to Pursuit-Evasion Games and Graph Theory. American Mathematical Society, Providence (2022)
4. Bonato, A., Nowakowski, R.J.: The Game of Cops and Robbers on Graphs. American Mathematical Society, Providence (2011). https://doi.org/10.1090/stml/061
5. Bradshaw, P., Hosseini, S.A.: Surrounding cops and robbers on graphs of bounded genus (2019). https://arxiv.org/abs/1909.09916
6. Burgess, A.C., et al.: Cops that surround a robber. Discret. Appl. Math. **285**, 552–566 (2020). https://doi.org/10.1016/j.dam.2020.06.019
7. Crytser, D., Komarov, N., Mackey, J.: Containment: a variation of cops and robbers. Graphs Comb. **36**(3), 591–605 (2020). https://doi.org/10.1007/s00373-020-02140-5
8. Frankl, P.: Cops and robbers in graphs with large girth and Cayley graphs. Discret. Appl. Math. **17**(3), 301–305 (1987). https://doi.org/10.1016/0166-218X(87)90033-3
9. Ha, M.T., Jungeblut, P., Ueckerdt, T.: Primal-dual cops and robber. In: Seara, C., Huemer, C. (eds.) Proceedings of the 39th European Workshop on Computational Geometry (EuroCG) (2023). https://arxiv.org/abs/2301.05514
10. Jungeblut, P., Schneider, S., Ueckerdt, T.: Cops and Robber - When Capturing is not Surrounding (2023). https://doi.org/10.48550/arXiv.2302.10577
11. Keedwell, D.A., Dénes, J.: Latin Squares and their Applications. 2nd edn. Elsevier, Amsterdam (2015). https://doi.org/10.1016/C2014-0-03412-0
12. Kinnersley, W.B.: Cops and Robbers is EXPTIME-complete. J. Comb. Theory Ser. B **111**, 201–220 (2015). https://doi.org/10.1016/j.jctb.2014.11.002

13. Nowakowski, R.J., Winkler, P.: Vertex-to-vertex pursuit in a graph. Discret. Math. **43**(2–3), 235–239 (1983). https://doi.org/10.1016/0012-365X(83)90160-7
14. Prałat, P.: Containment game played on random graphs: another zig-zag theorem. Electron. J. Comb. **22**(2) (2015). https://doi.org/10.37236/4777
15. Quilliot, A.: Jeux et pointes fixes sur les graphes. Ph.D. thesis, Université de Paris VI (1978)
16. Schneider, S.: Surrounding cops and robbers. Bachelor's thesis, Karlsruhe Institute of Technology (2022). https://i11www.iti.kit.edu/_media/teaching/theses/ba_schneider22.pdf

Complexity Results for Matching Cut Problems in Graphs Without Long Induced Paths

Hoàng-Oanh Le[1] and Van Bang Le[2](✉)

[1] Independent Researcher, Berlin, Germany
HoangOanhLe@outlook.com
[2] Institut für Informatik, Universität Rostock, Rostock, Germany
van-bang.le@uni-rostock.de

Abstract. In a graph, a (perfect) matching cut is an edge cut that is a (perfect) matching. matching cut (mc), respectively, perfect matching cut (pmc), is the problem of deciding whether a given graph has a matching cut, respectively, a perfect matching cut. The disconnected perfect matching problem (dpm) is to decide if a graph has a perfect matching that contains a matching cut. Solving an open problem recently posed in [Lucke, Paulusma, Ries (ISAAC 2022) & Feghali, Lucke, Paulusma, Ries (arXiv:2212.12317)], we show that pmc is NP-complete in graphs without induced 14-vertex path P_{14}. Our reduction also works simultaneously for mc and dpm, improving the previous hardness results of mc on P_{19}-free graphs and of dpm on P_{23}-free graphs to P_{14}-free graphs for both problems.

Actually, we prove a slightly stronger result: within P_{14}-free graphs, it is hard to distinguish between

(i) those without matching cuts and those in which every matching cut is a perfect matching cut;
(ii) those without perfect matching cuts and those in which every matching cut is a perfect matching cut;
(iii) those without disconnected perfect matchings and those in which every matching cut is a perfect matching cut.

Moreover, assuming the Exponential Time Hypothesis, none of these problems can be solved in time $2^{o(n)}$ for n-vertex P_{14}-free input graphs. As a corollary from (i), computing a matching cut with a maximum number of edges is hard, even when restricted to P_{14}-free graphs. This answers a question asked in [Lucke, Paulusma & Ries (arXiv:2207.07095)]. We also consider the problems in graphs without long induced cycles. It is known that mc is polynomially solvable in graphs without induced cycles of length at least 5 [Moshi (JGT 1989)]. We point out that the same holds for dpm.

Keywords: Matching cut · Maximum matching cut · Perfect matching cut · Disconnected perfect matching · H-free graph · Computational complexity

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 417–431, 2023.
https://doi.org/10.1007/978-3-031-43380-1_30

1 Introduction and Results

In a graph $G = (V, E)$, a *cut* is a partition $V = X \cup Y$ of the vertex set into disjoint, non-empty sets X and Y. The set of all edges in G having an endvertex in X and the other endvertex in Y, written $E(X, Y)$, is called the *edge cut* of the cut (X, Y). A *matching cut* is an edge cut that is a (possibly empty) matching. Another way to define matching cuts is as follows; see [3,7]: a cut (X, Y) is a matching cut if and only if each vertex in X has at most one neighbor in Y and each vertex in Y has at most one neighbor in X. The classical NP-complete problem MATCHING CUT (MC) [3] asks if a given graph admits a matching cut.

An interesting special case, where the edge cut $E(X, Y)$ is a *perfect matching* of G, was considered in [8]. Such a matching cut is called a *perfect matching cut* and the PERFECT MATCHING CUT (PMC) problem asks whether a given graph admits a perfect matching cut. It was shown in [8] that this special case PMC of MC remains NP-complete.

A notion related to matching cut is *disconnected perfect matching* which has been considered recently in [1]: a disconnected perfect matching is a perfect matching that contains a matching cut. Observe that any perfect matching cut is a disconnected perfect matching but not the converse. Figure 1 provides some small examples for matching cuts, perfect matching cuts and disconnected perfect matchings.

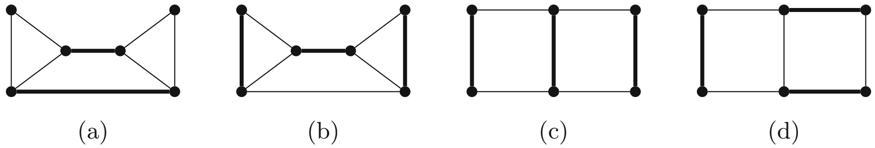


Fig. 1. Some example graphs; bold edges indicate a matching in question. (a): a matching cut. (b): a perfect matching that is neither a matching cut nor a disconnected perfect matching; this graph has no disconnected perfect matching, hence no perfect matching cut. (c): a perfect matching cut, hence a disconnected perfect matching. (d): a disconnected perfect matching that is not a perfect matching cut.

The related problem to MC and PMC, DISCONNECTED PERFECT MATCHING (DPM), asks if a given graph has a disconnected perfect matching; equivalently: if a given graph has a matching cut that is extendable to a perfect matching. It was shown in [1] that DPM is NP-complete. All these three problems have received much attention lately; see, e.g., [1,2,4,6,13–16] for recent results.

In this paper, we focus on the complexity of these three problems restricted to graphs without long induced paths and cycles. The current best known hardness results for MC and DPM in graphs without long induced paths are:

Theorem 1 ([14,15]). MC *remains* NP*-complete in* $\{4P_5, P_{19}\}$*-free graphs.*[1]

[1] Meanwhile the result has been improved to $\{3P_5, P_{15}\}$-free graphs [18].

Theorem 2 **([14,15]).** DPM *remains* NP*-complete in* $\{4P_7, P_{23}\}$*-free graphs.*[2]

Prior to the present paper, no similar hardness result for PMC was known. Indeed, it was asked in [4,14,15], whether there is an integer t such that PMC is NP-complete in P_t-free graphs. Polynomial-time algorithms exist for MC and PMC in P_6-free graphs [14,15] and for DPM in P_5-free graphs [1].

For graphs without long induced cycles (including chordal graphs and chordal bipartite graphs), the only result we are aware of is that MC is polynomially solvable:

Theorem 3 **([21]).** *There is a polynomial-time algorithm solving* MC *in graphs without induced cycles of length five and more.*

Previously, no similar polynomial-time results for DPM and PMC in long-hole-free graphs were known.

Our Contributions. We prove that PMC is NP-complete in graphs without induced path P_{14}, solving the open problem posed in [4,14,15]. For MC and DPM we improve the hardness results in Theorems 1 and 2 in graphs without induced path P_{19}, respectively, P_{23}, to graphs without induced path P_{14}. It is remarkable that all these hardness results for *three* problems will be obtained simultaneously by only *one* reduction, and can be stated in more details as follows.

Theorem 4. MC*,* PMC *and* DPM *are* NP*-complete in* $\{3P_6, 2P_7, P_{14}\}$*-free graphs. Moreover, under the ETH, no algorithm with runtime* $2^{o(n)}$ *can solve any of these problems for n-vertex* $\{3P_6, 2P_7, P_{14}\}$*-free input graphs.*

Actually, we prove the following slightly stronger result: within $\{3P_6, 2P_7, P_{14}\}$-free graphs, it is hard to distinguish between those without matching cuts (respectively perfect matching cuts, disconnected perfect matchings) and those in which every matching cut is a perfect matching cut. Moreover, under the ETH, this task cannot be solved in subexponential time in the vertex number of the input graph.

An interesting problem interposed between MC and PMC, called MAXIMUM MATCHING CUT (MAXMC), has recently been proposed in [15]. Here, given a graph G, we want to compute a matching cut of G (if any) with maximum number of edges. Formally, MAXMC in its decision version is as follows.

MAXIMUM MATCHING CUT (MAXMC)

Instance: A graph G and an integer k.

Question: Does G have a matching cut with k or more edges?

It has been asked in [15] what is the complexity of MAXMC on P_t-free graphs. Our next result answers this question.[3]

[2] Meanwhile the result has been improved to $\{3P_7, P_{19}\}$-free graphs [18].

[3] Meanwhile the complexity of MAXMC in H-free graphs has been completely determined [17].

Theorem 5. MAXMC *is* NP*-complete in* $\{3P_6, 2P_7, P_{14}\}$*-free graphs. Moreover, under the ETH, no algorithm with runtime* $2^{o(n)}$ *can solve* MAXMC *for* n*-vertex* $\{3P_6, 2P_7, P_{14}\}$*-free input graphs.*

On the positive side, we prove the following.

Theorem 6. *There is a polynomial-time algorithm solving* DPM *in graphs without induced cycle of length five and more.*

The paper is organized as follows. We recall some notion and notations in Sect. 2 which will be used. Then, we prove a slightly stronger result than Theorem 4 in Sect. 3 which then implies Theorem 5. The proof of Theorem 6 will be given in Sect. 4. Section 5 concludes the paper.

2 Preliminaries

For a set $\mathcal{H}$ of graphs, $\mathcal{H}$-free graphs are those in which no induced subgraph is isomorphic to a graph in $\mathcal{H}$. We denote by P_t the t-vertex path with $t-1$ edges and by C_t the t-vertex cycle with t edges. C_3 is also called a triangle, and a *hole* is a C_t for some $t \geq 4$; C_t with $t \geq 5$ are *long holes*. The union $G + H$ of two vertex-disjoint graphs G and H is the graph with vertex set $V(G) \cup V(H)$ and edge set $E(G) \cup E(H)$; we write pG for the union of p copies of G. For a subset $S \subseteq V(G)$, let $G[S]$ denote the subgraph of G induced by S; $G - S$ stands for $G[V(G) \setminus S]$. By 'G contains an H' we mean G contains H as an induced subgraph.

Given a matching cut $M = (X, Y)$ of a graph G, a vertex set $S \subseteq V(G)$ is *monochromatic* if S belongs to the same *part* of M, i.e., $S \subseteq X$ or else $S \subseteq Y$. Notice that every clique different from the P_2 is monochromatic with respect to any matching cut.

Algorithmic lower bounds in this paper are conditional, based on the Exponential Time Hypothesis (ETH) [9]. The ETH asserts that no algorithm can solve 3SAT in subexponential time $2^{o(n)}$ for n-variable 3-CNF formulas. As shown by the Sparsification Lemma in [10], the hard cases of 3SAT already consist of sparse formulas with $m = O(n)$ clauses. Hence, the ETH implies that 3SAT cannot be solved in time $2^{o(n+m)}$.

Recall that an instance for 1-IN-3SAT is a 3-CNF formula $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ over n variables, in which each clause C_j consists of three distinct literals. The problem asks whether there is a truth assignment of the variables such that every clause in ϕ has exactly one true literal. We call such an assignment an *1-in-3 assignment*. There is a polynomial reduction from 3SAT to 1-IN-3SAT ([20, Theorem 7.2]), which transforms an instance for 3SAT with n variables and m clauses to an equivalent instance for 1-IN-3SAT with $n + 4m$ variables and $3m$ clauses. Thus, assuming ETH, 1-IN-3SAT cannot be solved in time $2^{o(n+m)}$ on inputs with n variables and m clauses. We will need a restriction of 1-IN-3SAT, POSITIVE 1-IN-3SAT, in which each variable occurs positively. There is a well-known reduction from 1-IN-3SAT to POSITIVE 1-IN-3SAT, which transforms an instance for 1-IN-3SAT to an equivalent instance for POSITIVE 1-IN-3SAT, linear in the number of variables and clauses. Hence, we obtain: assuming ETH, POSITIVE 1-IN-3SAT cannot be solved in time $2^{o(n+m)}$ for inputs with n variables and m clauses.

3 Proof of Theorem 4 and Theorem 5

Recall that a perfect matching cut is in particular a matching cut, as well as a disconnected perfect matching. This observation leads to the following promise versions of MC, PMC and DPM. (We refer to [5] for background on promise problems.)

PROMISE-PMC MC
Instance: A graph G that either has no matching cut, or every matching cut is a perfect matching cut.
Question: Does G have a matching cut ?

PROMISE-PMC PMC
Instance: A graph G that either has no perfect matching cut, or every matching cut is a perfect matching cut.
Question: Does G have a perfect matching cut ?

PROMISE-PMC DPM
Instance: A graph G that either has no disconnected perfect matching, or every matching cut is a perfect matching cut.
Question: Does G have a disconnected perfect matching ?

In all the promise versions above, we are allowed not to consider certain input graphs. In PROMISE-PMC MC, PROMISE-PMC PMC and PROMISE-PMC DPM, we are allowed to ignore graphs having a matching cut that is not a perfect matching cut, for which MC must answer 'yes', and PMC and DPM may answer 'yes' or 'no'.

We slightly improve Theorem 4 by showing the following result.

Theorem 7. PROMISE-PMC MC, PROMISE-PMC PMC *and* PROMISE-PMC DPM *are* NP*-complete, even when restricted to* $\{3P_6, 2P_7, P_{14}\}$*-free graphs. Moreover, under the ETH, no algorithm with runtime* $2^{o(n)}$ *can solve any of these problems for* n*-vertex* $\{3P_6, 2P_7, P_{14}\}$*-free input graphs.*

Clearly, Theorem 7 implies Theorem 4. Theorem 7 shows in particular that distinguishing between graphs without matching cuts and graphs in which every matching cut is a perfect matching cut is hard, and not only between those without matching cuts and those with matching cuts which is implied by the NP-completeness of MC. Similar implications of Theorem 7 can be derived for PMC and DPM.

Also, Theorem 7 implies Theorem 5. Indeed, if MC or PMC is NP-hard in a graph class then MAXMC is NP-hard in the same class as well.

3.1 The Reduction

We give a polynomial-time reduction from POSITIVE 1-IN-3SAT to PROMISE-PMC PMC (and to PROMISE-PMC MC, PROMISE-PMC DPM at the same time).

Let ϕ be a 3-CNF formula with m clauses C_j, $1 \le j \le m$, and n variables x_i, $1 \le i \le n$, in which each clause C_j consists of three distinct variables. We will construct a $\{3P_6, 2P_7, P_{14}\}$-free graph G such that G has a perfect matching cut if and only if ϕ admits an 1-in-3 assignment. Moreover, every matching cut of G, if any, is a perfect matching cut.

For each clause C_j consisting of three variables c_{j1}, c_{j2} and c_{j3}, let $G(C_j)$ be the graph depicted in Fig. 2. We call c_j and c'_j the *clause vertices*, and c_{j1}, c_{j2} and c_{j3} the *variable vertices*. Then, the graph G is obtained from all $G(C_j)$ by adding

- all possible edges between variable vertices c_{jk} and $c_{j'k'}$ of the same variable. Thus, for each variable x,

$$Q(x) = \{c_{jk} \mid 1 \le j \le m, 1 \le k \le 3, x \text{ occurs in clause } C_j \text{ as } c_{jk}\}$$

  is a clique in G,
- all possible edges between the $2m$ clause vertices c_j and c'_j. Thus,

$$F = \{c_j \mid 1 \le j \le m\} \cup \{c'_j \mid 1 \le j \le m\}$$

  is a clique in G,
- all possible edges between the $3m$ vertices a_{jk}. Thus,

$$T = \{a_{jk} \mid 1 \le j \le m, 1 \le k \le 3\}$$

  is a clique in G.

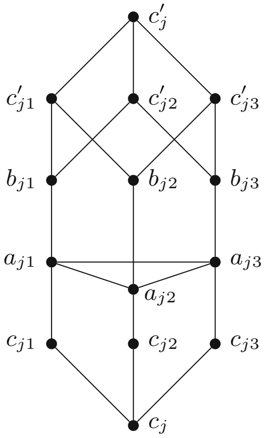


Fig. 2. The gadget $G(C_j)$.

The description of G is complete. As an example, the graph G from the formula ϕ with three clauses $C_1 = \{x, y, z\}, C_2 = \{u, z, y\}$ and $C_3 = \{z, v, w\}$ is depicted in Fig. 3.

Notice that no edge exists between the two cliques F and T. Notice also that $G - F - T$ has exactly $m + n$ components:

- For each $1 \le j \le m$, the 6-cycle D_j : $b_{j1}, c'_{j1}, b_{j2}, c'_{j3}, b_{j3}, c'_{j2}$ is a component of $G - F - T$, call it the clause 6-cycle (of clause C_j),
- For each variable x, the clique $Q(x)$ is a component of $G - F - T$, call it the variable clique (of variable x).

Lemma 1. *G is $\{3P_6, 2P_7, P_{14}\}$-free.*

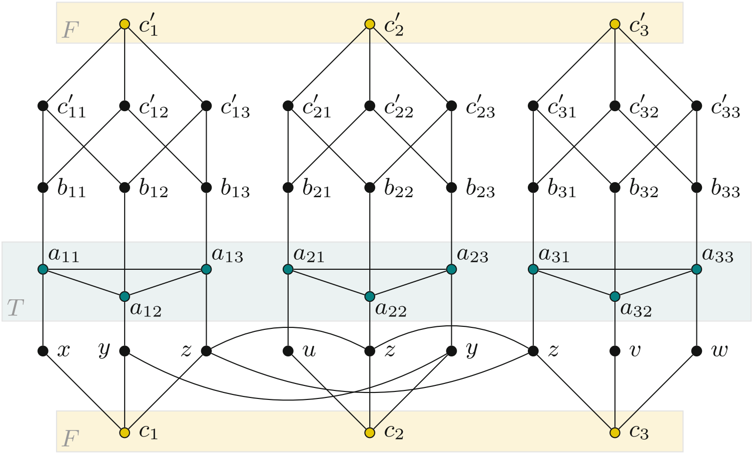


Fig. 3. The graph G from the formula ϕ with three clauses $C_1 = \{x, y, z\}, C_2 = \{u, z, y\}$ and $C_3 = \{z, v, w\}$. The 6 flax vertices $c_1, c_2, c_3, c'_1, c'_2, c'_3$ and the 9 teal vertices $a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}, a_{31}, a_{32}, a_{33}$ form the clique F and T, respectively.

Proof. First, observe that each component of $G - F - T$ is a clause 6-cycle D_j or a variable clique $Q(x)$. Hence,

$$G - F - T \text{ is } P_6\text{-free.} \tag{1}$$

Therefore, every induced P_6 in G must contain a vertex from the clique F or from the clique T. This shows that G is $3P_6$-free.

Observe next that, for each j, $c'_j \in F$ is the cut-vertex in $G-T$ separating the clause 6-cycle D_j and F, and $N(c'_j) \cap D_j = \{c'_{j1}, c'_{j2}, c'_{j3}\}$. Observe also that, for each x, $(G-T)[Q(x) \cup F]$ is a co-bipartite graph, the complement of a bipartite graph. Hence, it can be verified immediately that

$$G - T \text{ is } P_7\text{-free.} \tag{2}$$

Fact (2) implies that every induced P_7 in G must contain a vertex from the clique T. This shows that G is $2P_7$-free.

We now are ready to argue that G is P_{14}-free. Suppose not and let $P : v_1, v_2, \dots, v_{14}$ be an induced P_{14} in G, with edges v_iv_{i+1}, $1 \le i < 14$. For $i < j$, write $P[v_i, v_j]$ for the subpath of P between (and including) v_i and v_j. Then, by (2), each of $P[v_1, v_7]$ and $P[v_8, v_{14}]$ contains a vertex from the clique T. Since P has no chords, $P[v_1, v_7]$ has only the vertex v_7 in T and $P[v_8, v_{14}]$ has only the vertex v_8 in T. By (1), therefore, both $P[v_1, v_6]$ and $P[v_9, v_{14}]$ contain some vertex in the clique F, and thus P has a chord. This contradiction shows that G is P_{14}-free, as claimed. □

We remark that there are many induced P_{13} in G; we briefly discuss the limit of our construction in the appendix.

Lemma 2. *For any matching cut $M = (X, Y)$ of G,*

(i) *F and T are contained in different parts of M;*
(ii) *if $F \subseteq X$, then $|\{c_{j1}, c_{j2}, c_{j3}\} \cap Y| = 1$, and if $F \subseteq Y$, then $|\{c_{j1}, c_{j2}, c_{j3}\} \cap X| = 1$;*
(iii) *for any variable x, $Q(x)$ is monochromatic;*
(iv) *if $F \subseteq X$, then $|\{b_{j1}, b_{j2}, b_{j3}\} \cap Y| = 2$ and $|\{c'_{j1}, c'_{j2}, c'_{j3}\} \cap Y| = 1$, and if $F \subseteq Y$, then $|\{b_{j1}, b_{j2}, b_{j3}\} \cap X| = 2$ and $|\{c'_{j1}, c'_{j2}, c'_{j3}\} \cap X| = 1$.*

Proof. Notice that F and T are cliques with at least three vertices, hence F and T are monochromatic.

(i): Suppose not, and let F and T both be contained in X, say. Then all variable vertices c_{jk}, $1 \le j \le m, 1 \le k \le 3$, also belong to X because each of them has two neighbors in $F \cup T \subseteq X$. Now, if all b_{jk} are in X, then also all c'_{jk} are in X because in this case each of them has two neighbors in X, and thus $X = V(G)$. Thus some b_{jk} is in Y, and so are its two neighbors in $\{c'_{j1}, c'_{j2}, c'_{j3}\}$. But then c'_j, which is in X, has two neighbors in Y. This contradiction shows that F and T must belong to different parts of M, hence (i).

(ii): By (i), let $F \subseteq X$ and $T \subseteq Y$, say. (The case $F \subseteq Y$ is symmetric.) Then, for any j, at most one of c_{j1}, c_{j2} and c_{j3} can be outside X. Assume that, for some j, all c_{j1}, c_{j2}, c_{j3} are in X. The assumption implies that all b_{j1}, b_{j2}, b_{j3} belong to Y, and then all $c'_{j1}, c'_{j2}, c'_{j3}$ belong to Y, too. But then c'_j, which is in X, has three neighbors in Y. This contradiction shows (ii).

(iii): Suppose that two variable vertices c_{jk} and $c_{j'k'}$ in some clique $Q(x)$ are in different parts of M. Then, as c_{jk} and $c_{j'k'}$ have neighbor c_j and $c_{j'}$, respectively, in the monochromatic clique F, c_{jk} has two neighbors in the part of $c_{j'k'}$ or $c_{j'k'}$ has two neighbors in the part of c_{jk}. This contradiction shows (iii).

(iv): This fact can be derived from (i) and (ii). □

Lemma 3. *Every matching cut of G, if any, is a perfect matching cut.*

Proof. Let $M = (X, Y)$ be a matching cut of G. By Lemma 2 (i), let $F \subseteq X$ and $T \subseteq Y$, say. We argue that every vertex in X has a neighbor (hence exactly one) in Y. Indeed, for each j,

- $c_j \in F \subseteq X$ has a neighbor $c_{jk} \in Y$ (by Lemma 2 (ii)),
- $c'_j \in F \subseteq X$ has a neighbor $c'_{jk} \in Y$ (by Lemma 2 (iv)),
- each $c_{jk} \in X$ has a neighbor $a_{jk} \in T \subseteq Y$ (by construction of G),
- each $b_{jk} \in X$ has a neighbor $a_{jk} \in T \subseteq Y$ (by construction of G),
- each $c'_{jk} \in X$ has a neighbor in $\{b_{j1}, b_{j2}, b_{j3}\} \cap Y$ (by Lemma 2 (iv)).

Similarly, it can be seen that every vertex in Y has a neighbor in X. □

Lemma 4. *If ϕ has an 1-in-3 assignment, then G has a perfect matching cut.*

Proof. Partition $V(G)$ into disjoint subsets X and Y as follows. (Figure 4 shows the partition for the example graph in Fig. 3 given the assignment $y = v =$ True, $x = z = u = w =$ False.) First,

- put F into X, and for all variables x which are assigned with False, put $Q(x)$ into X;
- for each $1 \leq j \leq m$, let c_{jk} with $k = k(j) \in \{1, 2, 3\}$ be the variable vertex, for which the variable x of c_{jk} is assigned with True. Then put b_{jk} and its two neighbors in $\{c'_{j1}, c'_{j2}, c'_{j3}\}$ into X.

Let $Y = V(G) \setminus X$. Then, it is not difficult to verify that $M = (X, Y)$ is a perfect matching cut of G. □

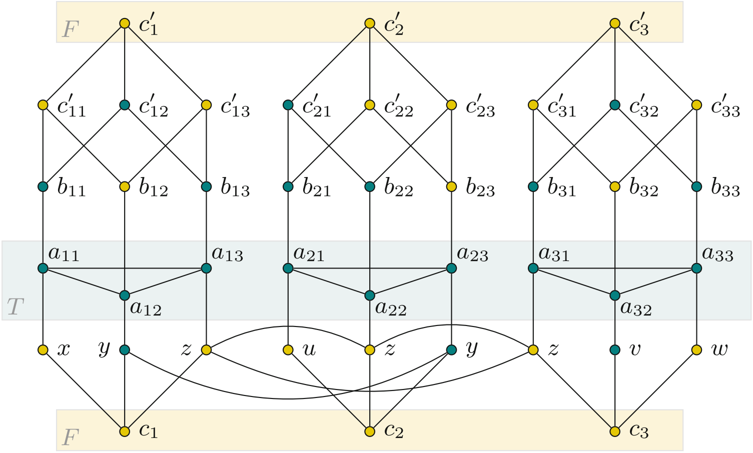


Fig. 4. The perfect matching cut (X, Y) of the example graph G in Fig. 3 given the assignment $y = v =$ True, $x = z = u = w =$ False. X and Y consist of the flax and teal vertices, respectively.

We now are ready to prove Theorem 7: First note that by Lemmas 1 and 3, G is $\{3P_6, 2P_7, P_{14}\}$-free and every matching cut of G (if any) is a perfect matching cut. In particular, every matching cut of G is extendable to a perfect matching.

Now, suppose ϕ has an 1-in-3 assignment. Then, by Lemma 4, G has a perfect matching cut. In particular, G has a disconnected perfect matching and, actually, a matching cut.

Conversely, let G have a matching cut $M = (X, Y)$, possibly a perfect matching cut or one that is contained in a perfect matching of G. Then, by Lemma 2 (i), we may assume that $F \subseteq X$, and set variable x to True if the corressponding variable clique $Q(x)$ is contained in Y and False if $Q(x)$ is contained in X. By Lemma 2 (iii), this assignment is well defined. Moreover, it is an 1-in-3 assignment for ϕ: consider a clause $C_j = \{x, y, z\}$ with $c_{j1} = x, c_{j2} = y$ and $c_{j3} = z$. By Lemma 2 (ii) and (iii), exactly one of $Q(x), Q(y)$ and $Q(z)$ is contained in Y, hence exactly one of x, y and z is assigned True.

Finally, note that G has $N = 14m$ vertices and recall that, assuming ETH, POSITIVE 1-IN-3SAT cannot be solved in $2^{o(m)}$ time. Thus, the ETH implies that no algorithm with runtime $2^{o(N)}$ exists for PROMISE-PMC MC, PROMISE-PMC PMC and PROMISE-PMC DPM, even when restricted to N-vertex $\{3P_6, 2P_7, P_{14}\}$-free graphs.

The proof of Theorem 7 is complete.

4 Proof of Theorem 6

Recall Theorem 3, MC is polynomially solvable for long-hole-free graphs (also called *quadrangulated graphs*). In this section, we point out that DPM is polynomially solvable for long-hole-free graphs, too, by following known approach [11,12,21].

Given a connected graph $G = (V, E)$ and two disjoint, non-empty vertex sets $A, B \subset V$ such that each vertex in A is adjacent to exactly one vertex of B and each vertex in B is adjacent to exactly one vertex of A. We say a matching cut of G is an *A, B-matching cut* (or a matching cut *separating A, B*) if A is contained in one side and B is contained in the other side of the matching cut. Observe that G has a matching cut if and only if G has an $\{a\}, \{b\}$-matching cut for some edge ab, and G has a disconnected perfect matching if and only if G has a perfect matching containing an $\{a\}, \{b\}$-matching cut for some edge ab.

For each edge ab of a long-hole-free graph G, we will be able to decide if G has a disconnected perfect matching containing a matching cut separating $A = \{a\}$ and $B = \{b\}$. This is done by applying known propagation rules ([11,12]), which are given below. Initially, set $X := A$, $Y := B$ and write $F = V(G) \setminus (X \cup Y)$ for the set of 'free' vertices. The sets A, B, X and Y will be extended, if possible, by adding vertices from F according to the following rules. The first three rules will detect certain vertices that ensure that G cannot have an A, B-matching cut.

(R1) Let $v \in F$ be adjacent to a vertex in A. If v is
- adjacent to a vertex in B, or
- adjacent to (at least) two vertices in $Y \setminus B$,

then G has no A, B-matching cut.

(R2) Let $v \in F$ be adjacent to a vertex in B. If v is
- adjacent to a vertex in A, or
- adjacent to (at least) two vertices in $X \setminus A$,

then G has no A, B-matching cut.

(R3) If $v \in F$ is adjacent to (at least) two vertices in $X \setminus A$ and to (at least) two vertices in $Y \setminus B$, then G has no A, B-matching cut.

The correctness of (R1), (R2) and (R3) is quite obvious. We assume that, before each application of the rules (R4) and (R5) below, none of (R1), (R2) and (R3) is applicable.

(R4) Let $v \in F$ be adjacent to a vertex in A or to (at least) two vertices in $X \setminus A$. Then $X := X \cup \{v\}$, $F := F \setminus \{v\}$. If, moreover, v has a unique neighbor $w \in Y \setminus B$ then $A := A \cup \{v\}$, $B := B \cup \{w\}$.

(R5) Let $v \in F$ be adjacent to a vertex in B or to (at least) two vertices in $Y \setminus B$. Then $Y := Y \cup \{v\}$, $F := F \setminus \{v\}$. If, moreover, v has a unique neighbor $w \in X \setminus A$ then $B := B \cup \{v\}$, $A := A \cup \{w\}$.

We refer to [12] for the correctness of rules (R4) and (R5), and for the following facts.

Fact 1. *The total runtime for applying (R1) – (R5) until none of the rules is applicable is bounded by $O(nm)$.*

Fact 2. *Suppose none of (R1) – (R5) is applicable. Then*

- *(X, Y) is an A, B-matching cut of $G[X \cup Y]$, and any A, B-matching cut of G must contain X in one side and Y in the other side;*
- *for any vertex $v \in F$,*

$$N(v) \cap A = \emptyset,\ N(v) \cap B = \emptyset \text{ and } |N(v) \cap X| \leq 1,\ |N(v) \cap Y| \leq 1.$$

We now are ready to prove Theorem 6: Let G be a connected, long-hole-free graph, and let ab be an edge of G. Set $A = \{a\}$ and $B = \{b\}$, and assume that none of (R1) − (R5) is applicable. Then, denoting $N(S)$ the set of vertices outside S adjacent to some vertex in S,

$$\text{for any connected component } S \text{ of } G[F], |N(S) \cap X| = 0 \text{ or } |N(S) \cap Y| = 0.$$

For, otherwise choose two vertices $s, s' \in S$ with a neighbor $x \in N(s) \cap X$ and a neighbor $y \in N(s') \cap Y$ such that the distance between s and s' in S is as small as possible. Then s, s', x and y and a shortest s, s'-path in S, a chordless x, y-path in $G[X \cup Y]$ together would induce a long hole in G. (Observe that, by the definition of X and Y, $G[X \cup Y]$ is connected.)

Partition F into disjoint subsets F_X and F_Y as follows:

$$F_X = \bigcup \{S : S \text{ is a connected component of } G[F] \text{with } N(S) \cap X \neq \emptyset\},$$

$$F_Y = \bigcup \{T : T \text{ is a connected component of } G[F] \text{ with } N(T) \cap Y \neq \emptyset\}.$$

Then, by the facts above and recall that G is connected,

$$F = F_X \cup F_Y \text{ and } F_X \cap F_Y = \emptyset.$$

Thus,

$$(X \cup F_X, Y \cup F_Y) \text{ is an } A, B\text{- matching cut of } G,$$

and it follows, that

$$G \text{ has a disconnected perfect matching containing an } A, B\text{-matching}$$
$$\text{cut if and only if } G - A - B \text{ has a perfect matching.}$$

Therefore, with Fact 1, in time $O(nm)$ we can decide whether G has a matching cut containing a given edge. Moreover, as a maximum matching can be computed in $O(\sqrt{n}m)$ time [19], we can decide in time $O(n\sqrt{n}m^2)$ whether G has a disconnected perfect matching containing an $\{a\}, \{b\}$-matching cut for a given edge ab. Since there are at most m edges to check, Theorem 6 follows.

5 Conclusion

We have shown that all three problems MC, PMC and DPM are NP-complete in P_{14}-free graphs. The hardness result for PMC solves an open problem posed in [4,14,15]. For MC and DPM, the hardness result improves the previously known one in P_{19}-free graphs, respectively, in P_{23}-free graphs, to P_{14}-free graphs. An obvious question is whether one of these problems remains NP-complete in P_t-free graphs for some $t < 14$.

We also pointed out that, like MC [21], DPM can be solved in polynomial time when restricted to long-hole-free graphs. We leave open the complexity of PMC restricted to long-hole-free graphs. More general, the chordality of a graph G is the length of a longest induced cycle in G. Chordal graphs and long-hole-free graphs (including weakly chordal and chordal bipartite graphs) have chordality 3 and 4, respectively. Notice that P_t-free graphs have chordality bounded by t, hence Theorem 4 implies that MC, PMC and DPM are NP-complete when restricted to graphs of chordality ≤ 14. We remark, however, that the graph constructed in the proof of Theorem 4 has chordality 8, and thus MC, PMC and DPM are NP-complete when restricted to graphs of chordality ≤ 8. Does there exist any class of graphs of chordality < 8 in which MC, PMC or DPM is NP-complete?

Acknowledgment. We thank the anonymous reviewers of WG 2023 for their very carefull reading. In particular, we thank all three reviewers for pointing out a small mistake in the earlier proof of Theorem 6.

A Limits of Our Reduction in the Proof of Theorem 4

As remarked, the graph G constructed from an instance of POSITIVE 1-IN-3SAT contains many induced paths P_{13}. For example, refer to Fig. 3; see also Fig. 5–7:

- $b_{11}, c'_{11}, b_{12}, c'_{13}, b_{13}, a_{13}, a_{22}, c_{22} = z, c_{31} = z, c_3, c_1, c_{12} = y, c_{23} = y$;
- $b_{11}, c'_{11}, b_{12}, c'_{13}, b_{13}, a_{13}, a_{21}, b_{21}, c'_{21}, c'_2, c'_3, c'_{33}, b_{33}$;
- $b_{11}, c'_{11}, b_{12}, c'_{13}, b_{13}, a_{13}, a_{21}, b_{21}, c'_{21}, c'_2, c_2, c_{23} = y, c_{12} = y$.

It can be seen that all P_{13} in G contain a P_5 from a 6-cycle D_j : $b_{j1}, c'_{j1}, b_{j2}, c'_{j3}, b_{j3}, c'_{j2}$. We now are going to describe how the gadget $G(C_j)$ used in the construction of G depicted in Fig. 2 was found. This could be useful when one is trying to improve the construction with shorter induced paths.

A general idea in constructing a graph without long induced paths from a given CNF-formula is to ensure that long induced paths must go through some, say at most three, cliques. Assuming we want to reduce POSITIVE 1-IN-3SAT (or NAE 3SAT) to PMC, the following observation gives a hint how to get such a clique: Let G be a graph, in which the seven vertices c, c_k, a_k, $1 \leq k \leq 3$, induce a tree with leaves a_1, a_2, a_3 and degree-2 vertices c_1, c_2, c_3 and the degree-3 vertex c. If G has a perfect matching cut, then a_1, a_2, a_3 must belong to the same part of the cut. Therefore, we can make $\{a_1, a_2, a_3\}$ adjacent to a clique and the resulting graph still has a perfect matching cut.

Now, a gadget $G(H;v)$ may be obtained from a suitable graph H with $v \in V(H)$ as follows. Let H be a graph having a vertex v of degree 3. Let b_1, b_2, b_3 be the neighbors of v in H. Let $G(H;v)$ be the graph obtained from $H - v$ by adding 7 new vertices a_1, a_2, a_3, c_1, c_2, c_3 and c, and edges cc_k, $c_k a_k$, $a_k b_k$, $1 \le k \le 3$, and a_1a_2, a_1a_3 and a_2a_3. (Thus, contracting the triangle $a_1a_2a_3$ from $G(H;v) \setminus \{c, c_1, c_2, c_3\}$ we obtain the graph H.)

Observation 1. *Assuming, for* any *neighbor w of v in H, H has a perfect matching cut (X, Y) such that $v \in X$ and $w \in Y$. Then, for* any *neighbor d of c in $G(H;v)$, the graph $G(H;v)$ has a perfect matching cut (X', Y') such that $c \in X'$ and $d \in Y'$.*

Examples of graphs H in Observation 1 include the cube, the Petersen graph and the 10-vertex Heggernes-Telle graph in [8, Fig. 3.1]. Our gadget $G(C_j)$ depicted in Fig. 2 is obtained by taking the cube. Take the Petersen graph or the Heggernes-Telle graph will produce induced P_t for some $t \ge 15$. If there exists another graph H 'better' than the cube, then our construction will yield a P_t-free graph for some $10 \le t \le 13$.

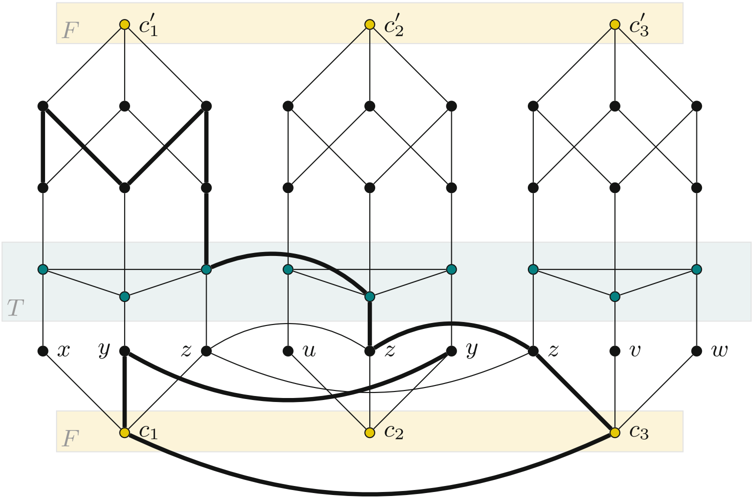


Fig. 5. The graph G from Fig. 3. The bold edges show the induced path P_{13}: $b_{11}, c'_{11}, b_{12}, c'_{13}, b_{13}, a_{13}, a_{22}, c_{22} = z, c_{31} = z, c_3, c_1, c_{12} = y, c_{23} = y$.

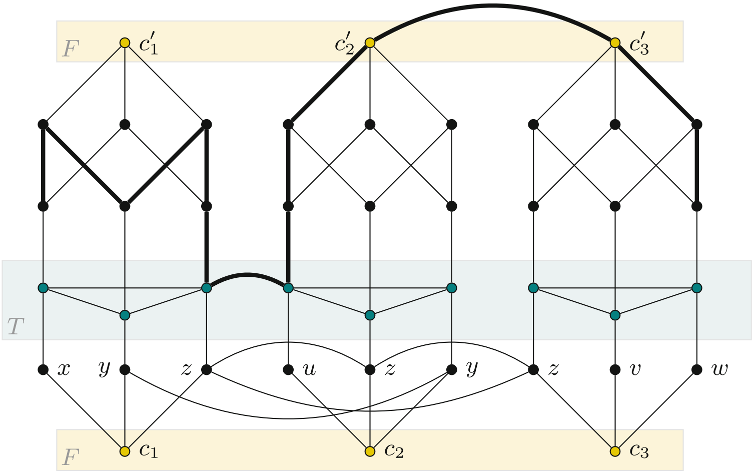


Fig. 6. The graph G from Fig. 3. The bold edges show the induced path P_{13}: $b_{11}, c'_{11}, b_{12}, c'_{13}, b_{13}, a_{13}, a_{21}, b_{21}, c'_{21}, c'_2, c'_3, c'_{33}, b_{33}$.

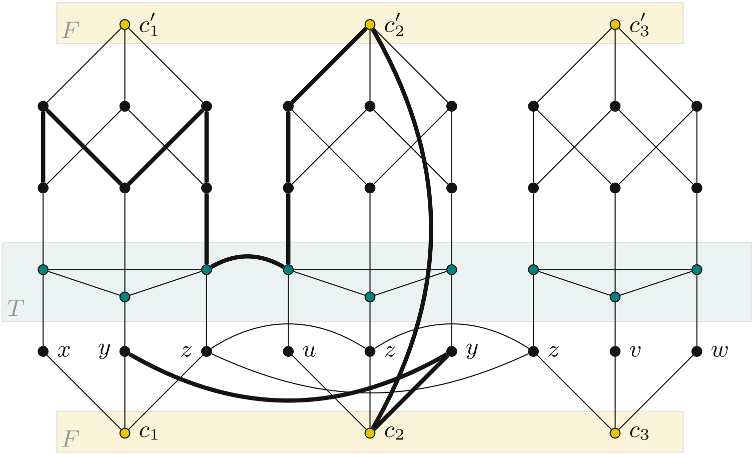


Fig. 7. The graph G from Fig. 3. The bold edges show the induced path P_{13}: $b_{11}, c'_{11}, b_{12}, c'_{13}, b_{13}, a_{13}, a_{21}, b_{21}, c'_{21}, c'_2, c_2, c_{23} = y, c_{12} = y$.

References

1. Bouquet, V., Picouleau, C.: The complexity of the perfect matching-cut problem. CoRR, abs/2011.03318v2 (2021). https://doi.org/10.48550/arXiv.2011.03318
2. Chen, C.-Y., Hsieh, S.-Y., Le, H.-O., Le, V.B., Peng, S.-L.: Matching cut in graphs with large minimum degree. Algorithmica **83**(5), 1238–1255 (2020). https://doi.org/10.1007/s00453-020-00782-8
3. Chvátal, V.: Recognizing decomposable graphs. J. Graph Theory **8**(1), 51–53 (1984). https://doi.org/10.1002/jgt.3190080106

4. Feghali, C., Lucke, F., Paulusma, D., Ries, B.: New hardness results for matching cut and disconnected perfect matching. CoRR, abs/2212.12317 (2022). https://doi.org/10.48550/arXiv.2212.12317
5. Goldreich, O.: On promise problems: a survey. In: Goldreich, O., Rosenberg, A.L., Selman, A.L. (eds.) Theoretical Computer Science. LNCS, vol. 3895, pp. 254–290. Springer, Heidelberg (2006). https://doi.org/10.1007/11685654_12
6. Golovach, P.A., Komusiewicz, C., Kratsch, D., Le, V.B.: Refined notions of parameterized enumeration kernels with applications to matching cut enumeration. STACS 2021, LIPIcs 187, 37:1–37:18 (2021). also. J. Comput. Syst. Sci., **123**, 76–102 (2022). https://doi.org/10.1016/j.jcss.2021.07.005
7. Graham, R.L.: On primitive graphs and optimal vertex assignments. Ann. N. Y. Acad. Sci. **175**(1), 170–186 (1970)
8. Pinar Heggernes and Jan Arne Telle: Partitioning graphs into generalized dominating sets. Nord. J. Comput. **5**(2), 128–142 (1998)
9. Impagliazzo, R., Paturi, R.: On the complexity of k-sat. J. Comput. Syst. Sci. **62**(2), 367–375 (2001). https://doi.org/10.1006/jcss.2000.1727
10. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. Comput. Syst. Sci. **63**(4), 512–530 (2001). https://doi.org/10.1006/jcss.2001.1774
11. Dieter Kratsch and Van Bang Le: Algorithms solving the matching cut problem. Theor. Comput. Sci. **609**, 328–335 (2016). https://doi.org/10.1016/j.tcs.2015.10.016
12. Le, H.O., Le, V.B.: A complexity dichotomy for matching cut in (bipartite) graphs of fixed diameter. Theor. Comput. Sci. **770**, 69–78 (2019). https://doi.org/10.1016/j.tcs.2018.10.029
13. Le , V.B., Telle, J.A.: The perfect matching cut problem revisited. In: Proceedings of the WG 2021, LNCS, vol. 12911, pp. 182–194 (2021). Also, Theor. Comput. Sci. **931**, 117–130 (2022). https://doi.org/10.1016/j.tcs.2022.07.035
14. Lucke, F., Paulusma, D., Ries, B.: Finding matching cuts in H-free graphs. In: Bae, S.W., Park, H. (eds.) 33rd International Symposium on Algorithms and Computation, ISAAC (2022), volume 248 of LIPIcs, pp. 22:1–22:16 (2022). https://doi.org/10.4230/LIPIcs.ISAAC.2022.22
15. Lucke, F., Paulusma, D., Ries, B.: Finding matching cuts in H-free graphs. CoRR, abs/2207.07095 (2022). https://doi.org/10.48550/arXiv.2207.07095
16. Lucke, F., Paulusma, D., Ries, B.: On the complexity of matching cut for graphs of bounded radius and H-free graphs. Theor. Comput. Sci. **936**, 33–42 (2022). https://doi.org/10.1016/j.tcs.2022.09.014
17. Lucke, F., Paulusma, D., Ries, B.: Dichotomies for maximum matching cut: H-freeness, bounded diameter, bounded radius. In: MFCS 2023. CoRR, abs/2304.01099 (2023). https://doi.org/10.48550/arXiv.2304.01099
18. Lucke, F., Paulusma, D., Ries, B.: Finding matching cuts in H-free graphs. Algorithmica (2023). https://doi.org/10.1007/s00453-023-01137-9
19. Micali, S., Vazirani, V.V.: An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In: 21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, USA, 13–15 October 1980, pp. 17–27. IEEE Computer Society (1980). https://doi.org/10.1109/SFCS.1980.12
20. Bernard, M., Moret, E.: Theory of Computation. Addison-Wesley-Longman, Boston (1998)
21. Moshi, A.M.: Matching cutsets in graphs. J. Graph Theory **13**(5), 527–536 (1989). https://doi.org/10.1002/jgt.3190130502

Upper Clique Transversals in Graphs

Martin Milanič[1](✉) and Yushi Uno[2]

[1] FAMNIT and IAM, University of Primorska, Koper, Slovenia
martin.milanic@upr.si
[2] Graduate School of Informatics, Osaka Metropolitan University, Sakai, Japan
yushi.uno@omu.ac.jp

Abstract. A *clique transversal* in a graph is a set of vertices intersecting all maximal cliques. The problem of determining the minimum size of a clique transversal has received considerable attention in the literature. In this paper, we initiate the study of the "upper" variant of this parameter, the *upper clique transversal number*, defined as the maximum size of a minimal clique transversal. We investigate this parameter from the algorithmic and complexity points of view, with a focus on various graph classes. We show that the corresponding decision problem is NP-complete in the classes of chordal graphs, chordal bipartite graphs, and line graphs of bipartite graphs, but solvable in linear time in the classes of split graphs and proper interval graphs.

Keywords: Clique transversal · Upper clique transversal number · Vertex cover

1 Introduction

A set of vertices of a graph G that meets all maximal cliques of G is called a *clique transversal* in G. Clique transversals in graphs have been studied by Payan in 1979 [36], by Andreae, Schughart, and Tuza in 1991 [4], by Erdős, Gallai, and Tuza in 1992 [20], and also extensively researched in the more recent literature (see, e.g., [3,5,10,13,15,19,23,28–31,37]). What most of these papers have in common is that they are interested in questions regarding the *clique transversal number* of a graph, that is, the minimum size of a clique transversal of the graph. For example, Chang, Farber, and Tuza showed in [13] that computing the clique transversal number for split graphs is NP-hard, and Guruswami and Pandu Rangan showed in [23] that the problem is NP-hard for cocomparability, planar, line, and total graphs, and solvable in polynomial time for Helly circular-arc graphs, strongly chordal graphs, chordal graphs of bounded clique size, and cographs.

In this paper, we initiate the study of the "upper" version of this graph invariant, the *upper clique transversal number*, denoted by $\tau_c^+(G)$ and defined as the maximum size of a minimal clique transversal, where a clique transversal in a graph G is said to be *minimal* if it does not contain any other clique transversal. The corresponding decision problem is defined as follows.

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 432–446, 2023.
https://doi.org/10.1007/978-3-031-43380-1_31

Upper Clique Transversal (UCT)

Input: A graph G and an integer k.

Question: Does G contain a minimal clique transversal S such that $|S| \geq k$?

Our study contributes to the literature on upper variants of graph minimization problems, which already includes the upper vertex cover (also known as maximum minimal vertex cover; see [11,16,41]), upper feedback vertex set (also known as maximum minimal feedback vertex set; see [18,27]), upper edge cover (see [26]), upper domination (see [2,6,25]), and upper edge domination (see [33]).

Our Results. We provide a first set of results on the algorithmic complexity of Upper Clique Transversal. Since clique transversals have been mostly studied in the class of chordal graphs and related classes, we also find it natural to first focus on this interesting graph class and its subclasses. In this respect, we provide an NP-completeness result as well as two very different linear-time algorithms. We show that UCT is NP-complete in the class of chordal graphs, but solvable in linear time in the classes of split graphs and proper interval graphs. Note that the result for split graphs is in contrast with the aforementioned NP-hardness result for computing the clique transversal number in the same class of graphs [13]. In addition, we provide NP-completeness proofs for two more subclasses of the class of perfect graphs, namely for chordal bipartite graphs, and for line graphs of bipartite graphs.

The diagram in Fig. 1 summarizes the relationships between various graph classes studied in this paper and indicates some boundaries of tractability of the UCT problem. We define those graph classes in the corresponding later sections in the paper. For further background and references on graph classes, we refer to [12].

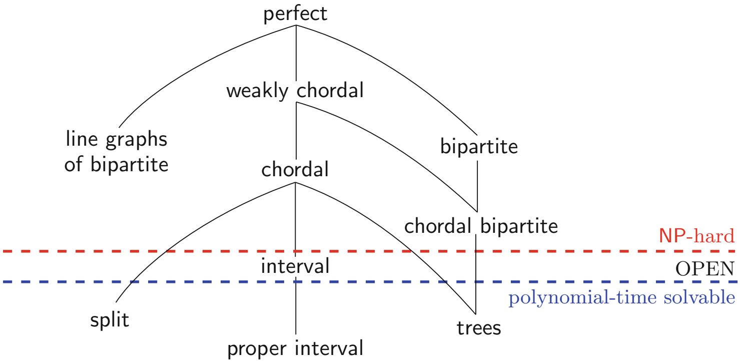


Fig. 1. The complexity of UCT in various graph classes studied in this paper.

Our Approach. Our approach is based on connections with a number of graph parameters. For example, the NP-completeness proofs for the classes of chordal bipartite graphs and of line graphs of bipartite graphs are based on the fact that for triangle-free graphs without isolated vertices, minimal clique transversals are exactly the minimal vertex covers, and they are closely related with minimal edge covers via the line graph operator. In particular, if G is a triangle-free graph without isolated vertices, then the upper clique transversal number of G equals the upper vertex cover number of G, that is, the maximum size of a minimal vertex cover. Since the upper vertex cover number of a graph G plus the independent domination number of G equals the order of G, there is also a connection with the independent dominating set problem. Let us note that, along with a linear-time algorithm for computing a minimum independent set in a tree [9], the above observations suffice to justify the polynomial-time solvability of the upper clique transversal problem on trees, as indicated in Fig. 1. The NP-completeness proof for the class of chordal graphs is based on a reduction from Spanning Star Forest, the problem of computing a spanning subgraph with as many edges as possible that consists of disjoint stars; this problem, in turn, is known to be closely related to the dominating set problem.

The linear-time algorithm for computing the upper clique transversal number of proper interval graphs relies on a linear-time algorithm for the maximum induced matching problem in bipartite permutation graphs due to Chang [14]. More precisely, we prove that the upper clique transversal number of a given graph cannot exceed the maximum size of an induced matching of a derived bipartite graph, the *vertex-clique incidence graph*, and show, using new insights on the properties of the matching computed by Chang's algorithm, that for proper interval graphs, the two quantities are the same.

The linear-time algorithm for computing the upper clique transversal number of a split graph is based on a characterization of minimal clique transversals of split graphs. A clique transversal that is an independent set is also called a *strong independent set* (or *strong stable set*; see [32] for a survey). It is not difficult to see that every strong independent set is a minimal clique transversal. We show that every split graph has a maximum minimal clique transversal that is independent (and hence, a strong independent set).

Structure of the Paper. In Sect. 2 we introduce the relevant graph theoretic background. Hardness results are presented in Sect. 3. Linear-time algorithms for UCT in the classes of split graphs and proper interval graphs are developed in Sects. 4 and 5, respectively. We conclude the paper in Sect. 6. Some proofs are omitted due to lack of space.

2 Preliminaries

Throughout the paper, graphs are assumed to be finite, simple, and undirected. We use standard graph theory terminology, following West [39]. A graph G with vertex set V and edge set E is often denoted by $G = (V, E)$; we write $V(G)$ and $E(G)$ for V and E, respectively. The set of vertices adjacent to a vertex $v \in V$ is the *neighborhood* of v, denoted $N(v)$; its cardinality is the *degree* of v. The *closed neighborhood* is the set $N[v]$, defined as $N(v) \cup \{v\}$. An *independent set* in a graph is a set of pairwise non-adjacent vertices; a *clique* is a set of pairwise adjacent vertices. An independent set (resp., clique) in a graph G is *maximal* if it is not contained in any other independent set (resp., clique). A *clique transversal* in a graph is a subset of vertices that intersects all the maximal cliques of the graph. A *dominating set* in a graph $G = (V, E)$ is a set S of vertices such that every vertex not in S has a neighbor in S. An *independent dominating set* is a dominating set that is also an independent set. The *(independent) domination number* of a graph G is the minimum size of an (independent) dominating set in G. Note that a set S of vertices in a graph G is an independent dominating set if and only if S is a maximal independent set. In particular, the independent domination number of a graph is a well-defined invariant leading to a decision problem called Independent Dominating Set.

The *clique number* of G is denoted by $\omega(G)$ and defined as the maximum size of a clique in G. An *upper clique transversal* of a graph G is a minimal clique transversal of maximum size. The *upper clique transversal number* of a graph G is denoted by $\tau_c^+(G)$ and defined as the maximum size of a minimal clique transversal in G. A *vertex cover* in G is a set $S \subseteq V(G)$ such that every edge $e \in E(G)$ has at least one endpoint in S. A vertex cover in G is *minimal* if it does not contain any other vertex cover. These notions are illustrated in Fig. 2. Note that if G is a triangle-free graph without isolated vertices, then the maximal cliques of G are exactly its edges, and hence the clique transversals of G coincide with its vertex covers.

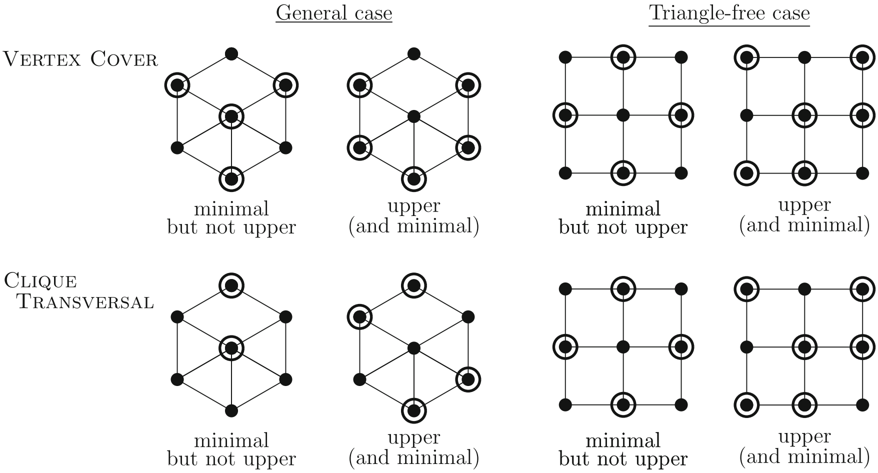


Fig. 2. Upper clique transversal and related notions.

3 Intractability of UCT for Some Graph Classes

In this section we prove that Upper Clique Transversal is NP-complete in the classes of chordal graphs, chordal bipartite graphs, and line graphs of bipartite graphs. First, let us note that for the class of all graphs, we do not know whether the problem is in NP. If S is a minimal clique transversal in G such that $|S| \geq k$, then a natural way to verify this fact would be to certify separately that S is a clique transversal and that it is a minimal one. Assuming that S is a clique transversal, one can certify minimality simply by exhibiting for each vertex $u \in S$ a maximal clique C in G such that $C \cap S = \{u\}$. However, unless $\mathsf{P} = \mathsf{NP}$, we cannot verify the fact that S is a clique transversal in polynomial time. This follows from a result of Zang [40], showing that it is co-NP-complete to check, given a weakly chordal graph G and an independent set S, whether S is a clique transversal in G. A graph G is *weakly chordal* if neither G nor its complement contain an induced cycle of length at least five.

We do not know whether Upper Clique Transversal is in NP when restricted to the class of weakly chordal graphs. However, for their subclasses chordal graphs and chordal bipartite graphs, membership of UCT in NP is a consequence of the following proposition.

Proposition 1. *Let $\mathcal{G}$ be a graph class such that every graph $G \in \mathcal{G}$ has at most polynomially many maximal cliques. Then,* Upper Clique Transversal *is in* NP *for graphs in $\mathcal{G}$.*

A *star* is a graph that has a vertex that is adjacent to all other vertices, and there are no other edges. A *spanning star forest* in a graph $G = (V, E)$ is a spanning subgraph (V, F) consisting of vertex-disjoint stars. Some of our hardness results will make use of a reduction from Spanning Star Forest, the problem that takes as input a graph G and an integer ℓ, and the task is to determine whether G contains a spanning star forest (V, F) such that $|F| \geq \ell$. This problem is NP-complete due to its close relationship with Dominating Set, the problem that takes as input a graph G and an integer k, and the task is to determine whether G contains a dominating set S such that $|S| \leq k$. The connection between the spanning star forests and dominating sets is as follows: a graph G has a spanning star forest with at least ℓ edges if and only if G has a dominating set with at most $|V| - \ell$ vertices (see [21,35]). Dominating Set is known to be NP-complete in the class of bipartite graphs (see, e.g., [8]) and even in the class of chordal bipartite graphs, as shown by Müller and Brandstädt [34]. The graphs constructed in the NP-hardness reduction from [34] do not contain any vertices of degree zero or one. Using the above-mentioned connection with Spanning Star Forest, we obtain the following.

Theorem 1. Spanning Star Forest *is* NP*-complete in the class of bipartite graphs with minimum degree at least* 2.

We present the hardness results in increasing order of difficulty of the proofs, starting with the class of chordal bipartite graphs. A *chordal bipartite* graph is a bipartite graph in which all induced cycles are of length four. Clearly, any chordal bipartite graph is triangle-free. Recall also that in any triangle-free graph G without isolated vertices, a set $S \subseteq V(G)$ is a minimal vertex cover if and only if it is a minimal clique transversal. Furthermore, in any graph G a set $S \subseteq V(G)$ is a minimal vertex cover if and only its complement $V(G) \setminus S$ is an independent dominating set. Using a reduction from the INDEPENDENT DOMINATING SET in chordal bipartite graphs (which is NP-complete [17]), we thus obtain the following.

Theorem 2. UPPER CLIQUE TRANSVERSAL *is* NP*-complete in the class of chordal bipartite graphs.*

We next consider the class of line graphs of bipartite graphs. The *line graph* of a graph G is the graph H with $V(H) = E(G)$ in which two distinct vertices are adjacent if and only if they share an endpoint as edges in G.

Lemma 1. *Let G be a triangle-free graph with minimum degree at least 2 and let H be the line graph of G. Then, the maximal cliques in H are exactly the sets E_v for $v \in V(G)$, where E_v is the set of edges in G that are incident with v.*

An *edge cover* of a graph G is a set F of edges such that every vertex of G is incident with some edge of F. Immediately from the definitions and Lemma 1 we obtain the following.

Lemma 2. *Let G be a triangle-free graph with minimum degree at least 2 and let H be the line graph of G. Then, a set $F \subseteq E(G)$ is a clique transversal in H if and only if F is an edge cover in G. Consequently, a set $F \subseteq E(G)$ is a minimal clique transversal in H if and only if F is a minimal edge cover in G.*

As shown by Hedetniemi [24], the maximum size of a minimal edge cover equals to the maximum number of edges in a spanning star forest, which is the number of vertices minus the domination number. Thus, using Proposition 1, Lemma 2, and a reduction from SPANNING STAR FOREST in the class of bipartite graphs with minimum degree at least 2 we obtain the following.

Theorem 3. UPPER CLIQUE TRANSVERSAL *is* NP*-complete in the class of line graphs of bipartite graphs.*

We now prove intractability of UCT in the class of chordal graphs. A graph is *chordal* if it does not contain any induced cycles on at least four vertices.

Theorem 4. UPPER CLIQUE TRANSVERSAL *is* NP*-complete in the class of chordal graphs.*

Proof (sketch). We reduce from SPANNING STAR FOREST. Let $G = (V, E)$ and ℓ be an input instance of SPANNING STAR FOREST. We may assume without loss of generality that G has an edge and that $\ell \geq 2$, since if any of these assumptions is violated, then it is trivial to verify if G has a spanning star forest with at least ℓ edges. We construct a chordal graph G' as follows. We start with a complete graph with vertex set V. For each edge $e = \{u, v\} \in E$, we introduce two new vertices x^e and y^e, and make x^e adjacent to u, to v, and to y^e. The obtained graph is G'. We thus have $V(G') = V \cup X \cup Y$, where $X = \{x^e : e \in E\}$ and $Y = \{y^e : e \in E\}$. See Fig. 3 for an example. Clearly, G' is chordal. Furthermore, let $k = \ell + |E|$.

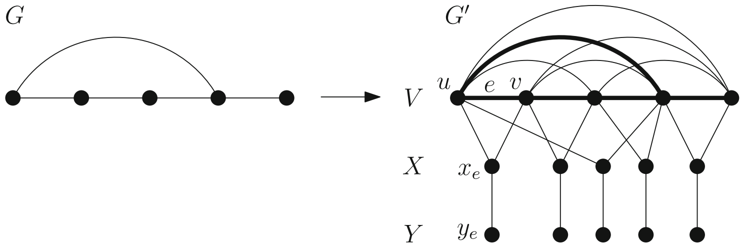


Fig. 3. Transforming G to G'.

To complete the proof, we show that G has a spanning star forest of size at least ℓ if and only if G' has a minimal clique transversal of size at least k.

First, assume that G has a spanning star forest (V, F) such that $|F| \geq \ell$. Since (V, F) is a spanning forest in which each component is a star, each edge of F is incident with a vertex of degree one in (V, F). Let S be a set obtained by selecting from each edge in F one vertex of degree one in (V, F). Then, every edge of F has one endpoint in S and the other one in $V \setminus S$. In particular, $|S| = |F| \geq \ell$. Let $S' = S \cup \{x^e : e \in E \setminus F\} \cup \{y^f : f \in F\}$ (see Fig. 4 for an example). The size of S' is at least $\ell + |E| = k$ and it can be shown that S' is a minimal clique transversal of G'.

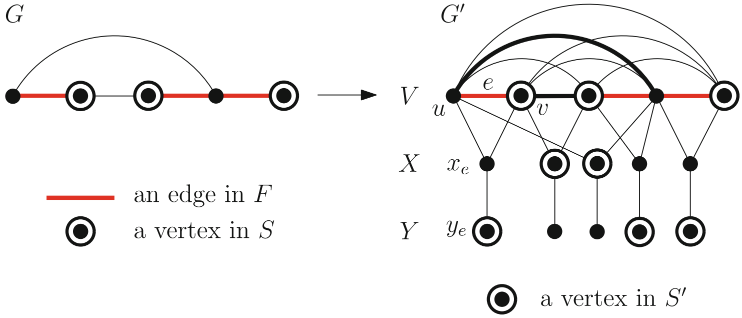


Fig. 4. Transforming a spanning star forest (V, F) in G into a minimal clique transversal S' in G'.

For the converse direction, let S' be a minimal clique transversal of G' such that $|S'| \geq k$. Let $S = S' \cap V$. It can be shown that we can associate to each vertex

$u \in S$ a vertex $v(u) \in V$ such that $e = \{u, v(u)\} \in E$ and $S' \cap \{u, v(u), x^e\} = \{u\}$. For each $u \in S$, let us denote by $e(u)$ the corresponding edge $\{u, v(u)\}$, and let $F = \{e(u) : u \in S\}$ (see Fig. 5). It can be shown that the set F satisfies $|F| = |S|$ and every vertex in S has degree one in (V, F). Therefore, the graph (V, F) is a spanning star forest of G.

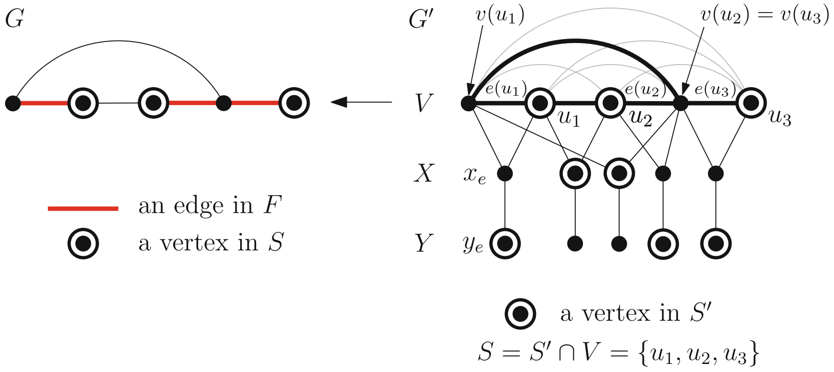


Fig. 5. Transforming a minimal clique transversal S' in G' into a spanning star forest (V, F) in G.

Since S' is a minimal clique transversal of G', for each edge $e \in E$ exactly one of x^e and y^e belongs to S'. Therefore, $|F| = |S| = |S'| - |E| \geq k - |E| = \ell$, and G has a spanning star forest of size at least ℓ. □

4 A Linear-Time Algorithm for UCT in Split Graphs

A *split graph* is a graph that has a *split partition*, that is, a partition of its vertex set into a clique and an independent set. We denote a split partition of a split graph G as (K, I) where K is a clique, I is an independent set, $K \cap I = \emptyset$, and $K \cup I = V(G)$. We may assume without loss of generality that I is a maximal independent set. In what follows, we repeatedly use the structure of maximal cliques of split graphs. If G is a split graph with a split partition (K, I), then the maximal cliques of G are as follows: the closed neighborhoods $N[v]$, for all $v \in I$, and the clique K, provided that it is a maximal clique, that is, every vertex in I has a non-neighbor in K.

Given a graph G and a set of vertices $S \subseteq V(G)$, we denote by $N(S)$ the set of all vertices in $V(G) \backslash S$ that have a neighbor in S. Moreover, given a vertex $v \in S$, an *S-private neighbor* of v is any vertex $w \in N(S)$ such that $N(w) \cap S = \{v\}$. The following proposition characterizes minimal clique transversals of split graphs.

Proposition 2. *Let G be a split graph with a split partition (K, I) such that I is a maximal independent set and let $S \subseteq V(G)$. Let $K' = K \cap S$ and $I' = I \cap S$. Then, S is a minimal clique transversal of G if and only if the following conditions hold:*

(i) $K' \neq \emptyset$ *if* K *is a maximal clique.*
(ii) $I' = I \setminus N(K')$.
(iii) Every vertex in K' *has a* K'*-private neighbor in* I.

Proposition 2 leads to the following result about maximum minimal clique transversals in split graphs. We denote by $\alpha(G)$ the *independence number* of a graph G, that is, the maximum size of an independent set in G.

Theorem 5. *Let* G *be a split graph with a split partition* (K, I) *such that* I *is a maximal independent set. Then:*

1. *If* K *is not a maximal clique in* G*, then* I *is a maximum minimal clique transversal in* G*; in particular, we have* $\tau_c^+(G) = \alpha(G)$ *in this case.*
2. *If* K *is a maximal clique in* G*, then for every vertex* $v \in K$ *with the smallest number of neighbors in* I*, the set* $\{v\} \cup (I \setminus N(v))$ *is a maximum minimal clique transversal in* G*; in particular, we have* $\tau_c^+(G) = \alpha(G) - \delta_G(I, K) + 1$ *in this case, where* $\delta_G(I, K) = \min\{|N(v) \cap I| : v \in K\}$.

Proof (sketch). Let S be a minimal clique transversal of G that is of maximum possible size and, subject to this condition, contains as few vertices from K as possible. Let $K' = K \cap S$ and $I' = I \cap S$. If $K' = \emptyset$, then K is not a maximal clique in G, and we have $S = I$, implying $\tau_c^+(G) = |S| = \alpha(G)$. Suppose now that $K' \neq \emptyset$. We first show that $|K'| = 1$. Suppose for a contradiction that $|K'| \geq 2$ and let $v \in K'$. Let I_v denote the set of K'-private neighbors of v in I and let $S' = (S \setminus \{v\}) \cup I_v$. Using Proposition 2, it can be verified that S' is a minimal clique transversal in G. Furthermore, since $v \in K'$, the set I_v is nonempty. This implies that $|S'| \geq |S|$; in particular, S' is a maximum minimal clique transversal in G. However, S' contains strictly fewer vertices from K than S, contradicting the choice of S. This shows that $|K'| = 1$, as claimed.

Let w be the unique vertex in K'. Since Condition (ii) from Proposition 2 holds for S, we have $I' = I \setminus N(w)$. Hence $S = \{w\} \cup (I \setminus N(w))$ and $|S| = 1 + |I| - |N(w) \cap I|$. Since $w \in K$, we have $|N(w) \cap I| \geq \delta_G(I, K)$ and hence $\tau_c^+(G) = |S| \leq \alpha(G) - \delta_G(I, K) + 1$. It can be verified that for every vertex $z \in K$ the set $X_z := \{z\} \cup (I \setminus N(z))$ satisfies Conditions (i)–(iii) from Proposition 2, and hence is a minimal clique transversal in G. Choosing z to be a vertex in K with the smallest number of neighbors in I, we obtain a set X_z of size $\alpha(G) - \delta_G(I, K) + 1$. Thus $\tau_c^+(G) \geq |X_z| = \alpha(G) - \delta_G(I, K) + 1$ and since we already proved that $\tau_c^+(G) \leq \alpha(G) - \delta_G(I, K) + 1$, any such X_z is optimal.

Since I is a maximal independent set and K is nonempty, we have $\delta_G(I, K) \geq 1$. Thus, $\tau_c^+(G) \leq \alpha(G)$. Suppose that K is not a maximal clique in G. Then I is a minimal clique transversal in G and therefore $\tau_c^+(G) \geq |I| = \alpha(G) \geq \tau_c^+(G)$. Hence equalities must hold throughout and I is a maximum minimal clique transversal. Finally, suppose that K is a maximal clique in G. Then every minimal clique transversal S in G satisfies $S \cap K \neq \emptyset$. In this case, the above analysis shows that for every vertex $v \in K$ with the smallest number of neighbors in I, the set $\{v\} \cup (I \setminus N(v))$ is a maximum minimal clique transversal in G. □

Corollary 1. Upper Clique Transversal *can be solved in linear time in the class of split graphs.*

5 A Linear-Time Algorithm for UCT in Proper Interval Graphs

A graph $G = (V, E)$ is an *interval graph* if it has an *interval representation*, that is, if its vertices can be put in a one-to-one correspondence with a family $(I_v : v \in V)$ of closed intervals on the real line such that two distinct vertices u and v are adjacent if and only if the corresponding intervals I_u and I_v intersect. If G has a *proper interval representation*, that is, an interval representation in which no interval contains another, then G is said to be a *proper interval graph.*

Our approach towards a linear-time algorithm for Upper Clique Transversal in the class of proper interval graphs is based on a relation between clique transversals in G and induced matchings in the so-called vertex-clique incidence graph of G. This relation is valid for arbitrary graphs.

UCT via Induced Matchings in the Vertex-Clique Incidence Graph

Given a graph $G = (V, E)$, we denote by B_G the *vertex-clique incidence graph* of G, a bipartite graph defined as follows. The vertex set of B_G consists of two disjoint sets X and Y such that $X = V$ and $Y = \mathcal{C}_G$, where $\mathcal{C}_G$ is the set of maximal cliques in G. The edge set of B_G consists of all pairs $x \in X$ and $C \in \mathcal{C}_G$ that satisfy $x \in C$. An *induced matching* in a graph G is a set M of pairwise disjoint edges such that the set of endpoints of edges in M induces no edges other than those in M. Given two disjoint sets of vertices A and B in a graph G, we say that A *dominates* B *in* G if every vertex in B has a neighbor in A. Given a matching M in a graph G and a vertex $v \in V(G)$, we say that v is *M-saturated* if it is an endpoint of an edge in M.

Clique transversals and minimal clique transversals of a graph G can be expressed in terms of the vertex-clique incidence graph as follows.

Lemma 3. *Let G be a graph, let $B_G = (X, Y; E)$ be its vertex-clique incidence graph, and let $S \subseteq V(G)$. Then:*

1. *S is a clique transversal in G if and only if S dominates Y in B_G.*
2. *S is a minimal clique transversal in G if and only if S dominates Y in B_G and there exists an induced matching M in B_G such that S is exactly the set of M-saturated vertices in X.*

The *induced matching number* of a graph G is the maximum size of an induced matching in G.

Corollary 2. *For every graph G, the upper clique transversal number of G is at most the induced matching number of B_G.*

As another consequence of Lemma 3, we obtain a sufficient condition for a set of vertices in a graph to be a minimal clique transversal of maximum size.

Corollary 3. *Let G be a graph, let $B_G = (X, Y; E)$ be its vertex-clique incidence graph, and let $S \subseteq V(G)$. Suppose that S dominates Y in B_G and there exists a maximum induced matching M in B_G such that S is exactly the set of M-saturated vertices in X. Then, S is a minimal clique transversal in G of maximum size.*

To apply Corollary 3 to proper interval graphs, we first state several characterizations of proper interval graphs in terms of their vertex-clique incidence graphs, establishing in particular a connection with bipartite permutation graphs.

Characterizing Proper Interval Graphs via Their Vertex-Clique Incidence Graphs

A bipartite graph $G = (X, Y; E)$ is said to be *biconvex* if there exists a *biconvex ordering* of (the vertex set of) G, that is, a pair $(<_X, <_Y)$ where $<_X$ is a linear ordering of X and $<_Y$ is a linear ordering of Y such that for every $x \in X$, the vertices in Y adjacent to x appear consecutively with respect to the ordering $<_Y$, and, similarly, for every $y \in Y$, the vertices in X adjacent to y appear consecutively with respect to the ordering $<_X$. Let $(<_X, <_Y)$ be a biconvex ordering of a biconvex graph $G = (X, Y; E)$. Two edges e and f of G are said to *cross* (each other) if there exist vertices $x_1, x_2 \in X$ and $y_1, y_2 \in Y$ such that $\{e, f\} = \{\{x_1, y_2\}, \{x_2, y_1\}\}$, $x_1 <_X x_2$, and $y_1 <_Y y_2$. A biconvex ordering $(<_X, <_Y)$ of a biconvex graph $G = (X, Y; E)$ is said to be *induced-crossing-free* if for any two crossing edges $e = \{x_1, y_2\}$ and $f = \{x_2, y_1\}$, either x_1 is adjacent to y_1 or x_2 is adjacent to y_2.

A *strongly induced-crossing-free ordering* (or simply a *strong ordering*) of G is a pair $(<_X, <_Y)$ of linear orderings of X and Y such that for any two crossing edges $e = \{x_1, y_2\}$ and $f = \{x_2, y_1\}$, vertex x_1 is adjacent to y_1 and vertex x_2 is adjacent to y_2. A *permutation graph* is a graph $G = (V, E)$ that admits a permutation model, that is, vertices of G can be ordered $v_1, \ldots, v_n$ such that there exists a permutation $(a_1, \ldots, a_n)$ of the set $\{1, \ldots, n\}$ such that for all $1 \leq i < j \leq n$, vertices v_i and v_j are adjacent in G if and only if $a_i > a_j$. A *bipartite permutation graph* is a graph that is both a bipartite graph and a permutation graph.

Theorem 6. *Let G be a graph. Then, the following statements are equivalent:*

1. *G is a proper interval graph.*
2. *B_G is a biconvex graph.*
3. *B_G is a bipartite permutation graph.*
4. *B_G has a strong ordering.*
5. *B_G has a strong biconvex ordering.*
6. *B_G has an induced-crossing-free biconvex ordering.*

The proof is based on showing that the vertex-clique incidence graph of every proper interval graph has a strong ordering, on characterizations of proper interval graphs and bipartite permutation graphs from [22] and [38], respectively, and on properties of biconvex graphs [1].

Maximum Induced Matchings in Bipartite Permutation Graphs, Revisited

Our goal is to show that the sufficient condition given by Corollary 3 is satisfied if G is a proper interval graph, namely, that there exists a maximum induced matching M in B_G such that the set S of M-saturated vertices in X dominates Y in B_G. We show the claimed property of B_G as follows. First, by applying Theorem 6, we infer that the graph B_G is a bipartite permutation graph. Second, by construction, no two distinct vertices in Y have comparable neighborhoods in X. It turns out that these two properties are already enough to guarantee the desired conclusion. We show this by a careful analysis of the linear-time algorithm due to Chang from [14] for computing a maximum induced matching in bipartite permutation graphs.

Theorem 7. *Given a bipartite permutation graph $G = (X, Y; E)$, there is a linear-time algorithm that computes a maximum induced matching M in G such that, if no two vertices in Y have comparable neighborhoods in G, then the set of M-saturated vertices in X dominates Y.*

Solving UCT in Proper Interval Graphs in Linear Time

We now have everything ready to prove the announced result.

Theorem 8. Upper Clique Transversal *can be solved in linear time in the class of proper interval graphs.*

Proof. The algorithm proceeds in three steps. In the first step, we compute from the input graph $G = (V, E)$ its vertex-clique incidence graph B_G, with parts $X = V$ and $Y = \mathcal{C}_G$. By Theorem 6, the graph B_G is a bipartite permutation graph. In the second step of the algorithm, we compute a maximum induced matching M of B_G using Theorem 7. Finally, the algorithm returns the set of M-saturated vertices in X.

By construction, the set M_X returned by the algorithm is a subset of X, and thus a set of vertices of G. Since the vertices of Y are precisely the maximal cliques of G, no two vertices in Y have comparable neighborhoods in B_G. Therefore, by Theorem 7, the set M_X dominates Y. By Corollary 3, M_X is a maximum minimal clique transversal in G. Computing the vertex-clique incidence graph B_G can be done in linear time (see [7]). Since B_G is a bipartite permutation graph, a maximum induced matching of B_G can be computed in linear time (see Theorem 7). The set of M-saturated vertices in X can also be computed in linear time. Thus, the overall time complexity of the algorithm is linear. □

The above proof also shows the following.

Theorem 9. *For every proper interval graph G, the upper clique transversal number of G is equal to the induced matching number of B_G.*

It can be shown that the result of Theorem 9 does not generalize to the class of interval graphs. In fact, there exist interval graphs with arbitrarily large difference between the induced matching number of their vertex-clique incidence graph and the upper clique transversal number of the graph (for example, the double stars).

6 Conclusion

We performed a systematic study of the complexity of Upper Clique Transversal in various graph classes, showing, on the one hand, NP-completeness of the problem in the classes of chordal graphs, chordal bipartite graphs, and line graphs of bipartite graphs, and, on the other hand, linear-time solvability in the classes of split graphs and proper interval graphs. Our work leaves open several questions:

- What is the complexity of computing a minimal clique transversal in a given graph?
- What is the complexity of Upper Clique Transversal in the class of interval graphs?
- For what graphs G does the upper clique transversal number equal to the induced matching number of the vertex-clique incidence graph? While not all interval graphs have the stated property, Theorem 9 shows that the property is satisfied by every proper interval graph. But there is more; for example, all cycles have the property.
- The upper clique transversal number is a trivial upper bound for the clique transversal number; however, the ratio between these two parameters can be arbitrarily large in general. For instance, in the complete bipartite graph $K_{1,q}$ the former one has value q while the latter one has value 1. For which graph classes is the ratio (or even the difference) between the clique transversal number and the upper clique transversal number bounded?

Acknowledgements. We are grateful to Nikolaos Melissinos, Haiko Müller, and the anonymous reviewers for their helpful comments. The work of the first named author is supported in part by the Slovenian Research Agency (I0-0035, research program P1-0285 and research projects N1-0102, N1-0160, J1-3001, J1-3002, J1-3003, J1-4008, and J1-4084). Part of the work was done while the author was visiting Osaka Prefecture University in Japan, under the operation Mobility of Slovene higher education teachers 2018–2021, co-financed by the Republic of Slovenia and the European Union under the European Social Fund. The second named author is partially supported by JSPS KAKENHI Grant Number JP17K00017, 20H05964, and 21K11757, Japan.

References

1. Abbas, N., Stewart, L.K.: Biconvex graphs: ordering and algorithms. Discrete Appl. Math. **103**(1–3), 1–19 (2000)

2. AbouEisha, H., Hussain, S., Lozin, V., Monnot, J., Ries, B., Zamaraev, V.: Upper domination: towards a dichotomy through boundary properties. Algorithmica **80**(10), 2799–2817 (2018)
3. Andreae, T., Flotow, C.: On covering all cliques of a chordal graph. Discrete Math. **149**(1–3), 299–302 (1996)
4. Andreae, T., Schughart, M., Tuza, Z.: Clique-transversal sets of line graphs and complements of line graphs. Discrete Math. **88**(1), 11–20 (1991)
5. Balachandran, V., Nagavamsi, P., Rangan, C.P.: Clique transversal and clique independence on comparability graphs. Inform. Process. Lett. **58**(4), 181–184 (1996)
6. Bazgan, C., et al.: The many facets of upper domination. Theoret. Comput. Sci. **717**, 2–25 (2018)
7. Berry, A., Pogorelcnik, R.: A simple algorithm to generate the minimal separators and the maximal cliques of a chordal graph. Inform. Process. Lett. **111**(11), 508–511 (2011)
8. Bertossi, A.A.: Dominating sets for split and bipartite graphs. Inform. Process. Lett. **19**(1), 37–40 (1984)
9. Beyer, T., Proskurowski, A., Hedetniemi, S., Mitchell, S.: Independent domination in trees. In: Proceedings of the Eighth Southeastern Conference on Combinatorics, Graph Theory and Computing (Louisiana State Univ., Baton Rouge, La., 1977), pp. 321–328. Congressus Numerantium, No. XIX (1977)
10. Bonomo, F., Durán, G., Safe, M.D., Wagler, A.K.: Clique-perfectness of complements of line graphs. Discrete Appl. Math. **186**, 19–44 (2015)
11. Boria, N., Della Croce, F., Paschos, V.T.: On the MAX MIN VERTEX COVER problem. Discrete Appl. Math. **196**, 62–71 (2015)
12. Brandstädt, A., Le, V.B., Spinrad, J.P.: Graph classes: a survey. In: Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, SIAM Monographs on Discrete Mathematics and Applications (1999)
13. Chang, G.J., Farber, M., Tuza, Z.: Algorithmic aspects of neighborhood numbers. SIAM J. Discrete Math. **6**(1), 24–29 (1993)
14. Chang, J.M.: Induced matchings in asteroidal triple-free graphs. Discrete Appl. Math. **132**(1–3), 67–78 (2003)
15. Cooper, J.W., Grzesik, A., Král, D.: Optimal-size clique transversals in chordal graphs. J. Graph Theory **89**(4), 479–493 (2018)
16. Damaschke, P.: Parameterized algorithms for double hypergraph dualization with rank limitation and maximum minimal vertex cover. Discrete Optim. **8**(1), 18–24 (2011)
17. Damaschke, P., Müller, H., Kratsch, D.: Domination in convex and chordal bipartite graphs. Inform. Process. Lett. **36**(5), 231–236 (1990)
18. Dublois, L., Hanaka, T., Khosravian Ghadikolaei, M., Lampis, M., Melissinos, N.: (In) approximability of maximum minimal FVS. J. Comput. System Sci. **124**, 26–40 (2022)
19. Eades, P., Keil, M., Manuel, P.D., Miller, M.: Two minimum dominating sets with minimum intersection in chordal graphs. Nordic J. Comput. **3**(3), 220–237 (1996)
20. Erdős, P., Gallai, T., Tuza, Z.: Covering the cliques of a graph with vertices. Discrete Math. **108**, 279–289 (1992)
21. Ferneyhough, S., Haas, R., Hanson, D., MacGillivray, G.: Star forests, dominating sets and Ramsey-type problems. Discrete Math. **245**(1–3), 255–262 (2002)
22. Gardi, F.: The Roberts characterization of proper and unit interval graphs. Discrete Math. **307**(22), 2906–2908 (2007)
23. Guruswami, V., Pandu Rangan, C.: Algorithmic aspects of clique-transversal and clique-independent sets. Discrete Appl. Math. **100**(3), 183–202 (2000)

24. Hedetniemi, S.T.: A max-min relationship between matchings and domination in graphs. Congr. Numer. **40**, 23–34 (1983)
25. Jacobson, M.S., Peters, K.: Chordal graphs and upper irredundance, upper domination and independence. Discrete Math. **86**(1–3), 59–69 (1990)
26. Khoshkhah, K., Ghadikolaei, M.K., Monnot, J., Sikora, F.: Weighted upper edge cover: complexity and approximability. J. Graph Algorithms Appl. **24**(2), 65–88 (2020)
27. Lampis, M., Melissinos, N., Vasilakis, M.: Parameterized max min feedback vertex set. In: Proceedings of 48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023) (2023). arXiv:2302.09604
28. Lee, C.M.: Algorithmic aspects of some variations of clique transversal and clique independent sets on graphs. Algorithms (Basel) **14**(1), 22 (2021)
29. Lee, C.M., Chang, M.S.: Distance-hereditary graphs are clique-perfect. Discrete Appl. Math. **154**(3), 525–536 (2006)
30. Lin, M.C., Vasiliev, S.: Approximation algorithms for clique transversals on some graph classes. Inform. Process. Lett. **115**(9), 667–670 (2015)
31. Liu, K., Lu, M.: Complete-subgraph-transversal-sets problem on bounded treewidth graphs. J. Comb. Optim. **41**(4), 923–933 (2021)
32. Milanič, M.: Strong cliques and stable sets. In: Topics in algorithmic graph theory, Encyclopedia of Mathematics and its Applications, vol. 178, pp. 207–227. Cambridge University Press, Cambridge (2021)
33. Monnot, J., Fernau, H., Manlove, D.: Algorithmic aspects of upper edge domination. Theoret. Comput. Sci. **877**, 46–57 (2021)
34. Müller, H., Brandstädt, A.: The NP-completeness of Steiner tree and dominating set for chordal bipartite graphs. Theoret. Comput. Sci. **53**(2–3), 257–265 (1987)
35. Nguyen, C.T., Shen, J., Hou, M., Sheng, L., Miller, W., Zhang, L.: Approximating the spanning star forest problem and its application to genomic sequence alignment. SIAM J. Comput. **38**(3), 946–962 (2008)
36. Payan, C.: Remarks on cliques and dominating sets in graphs. Ars Combin. **7**, 181–189 (1979)
37. Shan, E., Liang, Z., Kang, L.: Clique-transversal sets and clique-coloring in planar graphs. Eur. J. Combin. **36**, 367–376 (2014)
38. Spinrad, J., Brandstädt, A., Stewart, L.: Bipartite permutation graphs. Discrete Appl. Math. **18**(3), 279–292 (1987)
39. West, D.B.: Introduction to graph theory. Prentice Hall Inc, Upper Saddle River, NJ (1996)
40. Zang, W.: Generalizations of Grillet's theorem on maximal stable sets and maximal cliques in graphs. Discrete Math. **143**(1–3), 259–268 (1995)
41. Zehavi, M.: Maximum minimal vertex cover parameterized by vertex cover. SIAM J. Discrete Math. **31**(4), 2440–2456 (2017)

Critical Relaxed Stable Matchings with Two-Sided Ties

Meghana Nasre[1], Prajakta Nimbhorkar[2], and Keshav Ranjan[1](✉)

[1] IIT Madras, Chennai, India
meghana@cse.iitm.ac.in, ranjankeshav08@gmail.com
[2] Chennai Mathematical Institute and UMI ReLaX, Chennai, India
prajakta@cmi.ac.in

Abstract. We consider the stable marriage problem in the presence of ties in preferences and critical vertices. The input to our problem is a bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$ where $\mathcal{A}$ and $\mathcal{B}$ denote sets of vertices which need to be matched. Each vertex has a preference ordering over its neighbours possibly containing ties. In addition, a subset of vertices in $\mathcal{A} \cup \mathcal{B}$ are marked as critical and the goal is to output a matching that matches as many critical vertices as possible. Such matchings are called critical matchings in the literature and in our setting, we seek to compute a matching that is critical as well as optimal with respect to the preferences of the vertices.

Stability, which is a well-accepted notion of optimality in the presence of two-sided preferences, is generalized to weak-stability in the presence of ties. It is well known that in the presence of critical vertices, a matching that is critical as well as weakly stable may not exist. Popularity is another well-investigated notion of optimality for the two-sided preference list setting, however, in the presence of ties (even with no critical vertices), a popular matching need not exist. We, therefore, consider the notion of relaxed stability which was introduced and studied by Krishnaa et. al. (SAGT 2020). We show that in our setting a critical matching which is relaxed stable always exists although computing a maximum-sized relaxed stable matching turns out to be NP-hard. Our main contribution is a $\frac{3}{2}$-approximation to the maximum-sized critical relaxed stable matching for the stable marriage problem where ties as well as critical vertices are present on both the sides of the bipartition.

Keywords: Stable Matching · Ties in Preferences · Critical · Relaxed Stable · Approximation Algorithm

1 Introduction

We study the stable marriage problem in the presence of *ties* in preferences and *critical vertices*. Formally, the input to our problem is a bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$, where $\mathcal{A}$ and $\mathcal{B}$ are two sets of vertices and E denotes the set of all the acceptable vertex-pairs. Each vertex $u \in \mathcal{A} \cup \mathcal{B}$ ranks a subset of vertices in the other partition (its neighbours in G) in the order of its preference possibly

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 447–461, 2023.
https://doi.org/10.1007/978-3-031-43380-1_32

involving ties – this ordering is denoted as $\mathsf{Pref}(u)$. For a vertex u let v_1 and v_2 be two of its neighbours in G. The vertex u strictly prefers v_1 over v_2 (denoted as $v_1 \succ_u v_2$) if the rank of the edge (u, v_1) is smaller than the rank of the edge (u, v_2). The vertex u is tied between v_1 and v_2 (denoted as $v_1 =_u v_2$) if the ranks on the edges (u, v_1) and (u, v_2) are the same. We use $v_1 \succeq_u v_2$ to denote that the rank of v_1 is at least as good as the rank of v_2 in $\mathsf{Pref}(u)$. In addition, the input consists of a set $\mathcal{C} \subseteq (\mathcal{A} \cup \mathcal{B})$ of *critical vertices*. Our goal is to compute an assignment which minimizes the number of unassigned critical vertices.

Formally, an assignment or a *matching* $M \subseteq E$ in G is a set of edges that do not share an end-point. For each vertex $u \in \mathcal{A} \cup \mathcal{B}$, we denote by $M(u)$, the neighbour of u that is assigned to u in M. In the presence of critical vertices, we consider that the most important attribute of a matching is to match as many critical vertices as possible. A matching M is *critical* [11] if there is no matching that matches more critical vertices than M. In this work, we are interested in computing a critical matching that is *optimal* with respect to the preferences of the vertices in an instance of our setting.

Critical vertices or lower-quota positions naturally arise in applications like the Hospitals/Residents problem [7], where rural hospitals must be prioritized to ensure sufficient staffing. Another example is the problem of assigning sailors to billets [28] in the US Navy, where some critical billets cannot be left vacant [25,29]. Ties in preferences is yet another important practical consideration in matching problems and has been extensively investigated in the literature [2,8,9,13,18,19,24]. However, there is a limited investigation (see for example [5]) of matching problems with ties as well critical vertices and ours is the first work that allows ties as well as critical vertices on both sides of the bipartition.

Stability, which is the de-facto notion of optimality for two-sided preferences, is defined by the absence of a blocking pair. Informally, an assignment is stable if no unassigned pair wishes to deviate from it.

Definition 1 (Stable Matchings). *Given a matching M, a pair $(a, b) \in E \setminus M$ is called a blocking pair w.r.t. M if (i) either a is unmatched or $b \succ_a M(a)$ and (ii) either b is unmatched or $a \succ_b M(b)$. A matching M is stable if there is no blocking pair w.r.t. M.*

When all preferences are strict, that is, there are no ties, every instance of the stable marriage problem admits a stable matching, and it can be computed using the well-known Gale and Shapley algorithm [3]. In addition, it is also known [26,27] that all stable matchings have the same size.

Stable Matchings in the Presence of Ties: When preferences are allowed to have ties, the notion of stability defined above is called as *weak stability* (referred to as stability in the rest of the paper). We remark that, for a pair (a, b) to block a matching M, both a and b prefer each other *strictly* over their current partners in M. Every instance of the stable marriage problem with ties admits a stable matching, and it can be efficiently computed. However, unlike in the case of strict lists, all the stable matchings need not have the same size, and the problem of computing a maximum or minimum size stable matching is NP-hard [18] under severe restrictions – e.g. when ties occur at the end of preference lists and only on one side of the bipartition, there is at most one tie per list, and each tie is of length two.

Stable/Popular Matching in the Presence of Critical Vertices: When we have critical vertices as a part of the input, a stable matching which is also critical, may not exist – for example, consider an instance of the stable matching problem with strict lists obtained by arbitrarily breaking ties in the preference lists of all agents in the example shown in Fig. 1. Any critical matching in the instance must match b_2 with a_1, resulting in the blocking edge (a_1, b_1). Since stability and criticality are not simultaneously guaranteed, an alternate notion of optimality, namely *popularity* [4], is extensively investigated in the literature [11,20,22] for the case of strict lists. The goal is to compute a matching which is *popular* amongst the set of critical matchings. Informally, a matching M is *popular* in a set of matchings if no majority of vertices wish to deviate from M to any other matching in that set. It is known [11,22] that an instance with strict preference lists *always* admits a matching which is popular amongst critical matchings, and such a matching can be computed efficiently. Hence, it is natural to consider popularity in the presence of critical vertices and ties.

However, popular matchings are not guaranteed to exist even when ties are present in the preferences only on one side of the bipartition, without any critical vertices. Moreover, in the presence of ties, deciding whether a popular matching exists is NP-hard [1]. In light of this, we explore the notion of *relaxed stability.*

Relaxed Stability in the Presence of Ties and Critical Vertices: The notion of relaxed stability was introduced and studied by Krishnaa *et al.* [14] for the Hospitals/Residents problem with lower quotas (HRLQ). In their setting, preferences are assumed to be strict. The HRLQ setting is a many-to-one matching problem where a hospital h can accept at most $q^+(h)$ many residents and has $q^-(h) \leq q^+(h)$ many critical positions. To satisfy the critical positions at a hospital, certain residents may be *forced* to be matched to the hospital. The notion of relaxed stability allows only such residents to participate in blocking pairs. In addition, if a resident matched to h participates in a blocking pair then the hospital h should not be *surplus*, that is $|M(h)| \leq q^-(h)$.

In the HRLQ setting, preferences are strict, hospitals have capacities, and critical positions are allowed only for hospitals. In contrast, we allow ties in preferences as well as critical vertices to appear on *both sides* of the bipartition. However, our setting is one-to-one.

We now define the notion of relaxed stability (RSM) for our setting. Intuitively, a matching M is an RSM if every blocking pair (a, b) w.r.t. M is *justified* by either a or b or both. A vertex a justifies the blocking pair if $M(a)$ is a critical vertex. That is, $M(a)$ forces a to be matched to a lower-preferred vertex than b. Similarly, the vertex b can justify the blocking pair (a, b).

Definition 2 (Relaxed stability in our setting). *A matching M is* RSM *if for every blocking pair (a, b) w.r.t. M at least one of the following holds:*

1. a is matched and $b' = M(a)$ is critical, or
2. b is matched and $a' = M(b)$ is critical.

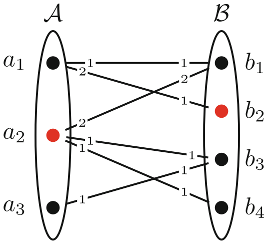


Fig. 1. Red vertices are critical, black vertices are non-critical. The numbers on the edges denote the ranks of the respective end-points. The instance does not admit any critical stable matching because b_2 remains unmatched in every stable matching. $M_1 = \{(a_1, b_2), (a_2, b_1), (a_3, b_3)\}$ is critical but not RSM because the blocking edge (a_2, b_4) is not justified. $M_2 = \{(a_1, b_2), (a_2, b_4), (a_3, b_3)\}$ is CRITICAL-RSM because the only blocking edge (a_1, b_1) is justified. (Color figure online)

A matching M is called *a critical relaxed stable matching (*CRITICAL-RSM*)* if it is *critical* as well as *relaxed stable*. In the instance shown in Fig. 1, the matching M_1 is critical but not RSM whereas M_2 is CRITICAL-RSM.

Our first contribution is to show that a CRITICAL-RSM always exists in our setting. We remark that when $\mathcal{C} = \emptyset$, an instance of our setting is the same as the stable marriage setting with ties but without critical vertices, and hence the set of CRITICAL-RSM is the same as the set of stable matchings. This immediately implies that computing a maximum size critical RSM is NP-hard [18] and hard to approximate within any factor smaller than $\frac{21}{19}$ [6]. For the problem of computing a maximum-sized stable matching when ties appear on both sides of the bipartition, the current best approximation factor [13,19,24] is $\frac{3}{2}$. The main result (Theorem 1) provides the same approximation size guarantee for a maximum sized CRITICAL-RSM in our setting.

Theorem 1. *Let $G = (\mathcal{A} \cup \mathcal{B}, E)$ be an instance of the stable marriage problem where ties and critical vertices can appear in both the bipartitions of G. Then G always admits a* CRITICAL-RSM *M such that $|M| \geq \frac{2}{3}|M'|$, where M' is a maximum size* CRITICAL-RSM *in G. Moreover, M can be computed in polynomial-time.*

Related Work: As mentioned earlier, the generalizations of the stable marriage problem to allow either ties in preferences or critical vertices/lower-quota positions has been extensively investigated. Very recently, Goko *et al.* [5] and Makino *et al.* [17] have considered the instances with both ties and critical vertices. They study the Hospitals/Residents problem with lower-quotas where ties appear on both sides. In their setting, only one side of the bipartition can have critical vertices. They define a matching with maximum satisfaction ratio, which for our one-to-one setting, coincides with critical matchings. However, their goal is to compute a matching that matches the maximum possible critical vertices amongst all stable matchings.

For strict preferences and lower-quotas/critical vertices, various notions like envyfreeness [15,30], popularity [11,20,22,23], and relaxed stability [14,15] have been studied. Relaxed stability and popularity do not define the same set of matchings even in the one-to-one strict-list setting and critical vertices restricted to one side only (see full version [21]) for the details. Hamada *et al.* [7] consider the problem of computing a matching with minimum number of blocking pairs or blocking residents, and give approximation algorithms for the same.

For the stable marriage problem with ties (without critical vertices) there is a long line of investigation [2,9,10,12,13,19,24] in order to improve the approximation ratio of the maximum size stable matching under various restricted settings. The best-known approximation algorithm for the case when ties are allowed only in one bipartition of the graph is by Lam and Plaxton [16] whereas the best-known for the case where ties are allowed on both sides is by [13,19,24]. We use Király's algorithm [13] in our work.

2 Preliminaries

Our algorithm described in the next section combines the ideas in (i) Király's algorithm [13] for computing a stable matching in instances where ties appear on both sides and (ii) Multi-level algorithm for computing popular critical matching [23] for strict preferences. We give an overview of the algorithms and also define terminology useful for our algorithm.

Overview of Király's Algorithm [13]. Király's algorithm [13] is a proposal-based algorithm where vertices in $\mathcal{A}$ propose and vertices in $\mathcal{B}$ accept or reject. We need the term *uncertain proposal* from [13] which is defined below.

Definition 3 (Uncertain Proposal). *Let b be some k^{th} rank neighbour of a in $\mathsf{Pref}(a)$. During the course of the algorithm, the proposal from a to b is uncertain if there exists another k^{th} rank neighbour b' of a which is unproposed by a and unmatched in the matching. Once a proposal (a, b) is uncertain, it remains uncertain until b rejects a.*

Each time an $a \in \mathcal{A}$ proposes to its *favourite* neighbour b (we define favourite neighbour formally in Definition 4), the vertex b accepts/rejects as follows:

1. If b is unmatched then b immediately accepts the proposal.
2. If b is matched, say to a', and (a', b) is an uncertain proposal, then b rejects a' and accepts the proposal from a, irrespective of the ranks of a and a' in $\mathsf{Pref}(b)$. In this case, b is *marked* by a'.
3. If b is matched, say to a', and (a', b) is not an uncertain proposal, then
   (i) if $a \succ_b a'$ then b rejects a' and accepts the proposal from a, or
   (ii) if $a' \succeq_b a$ then b rejects a.

The reason for a' marking the vertex b in (2) is as follows: In this case, b rejects the uncertain proposal from a' and accepts a *irrespective* of b's preference between a and a'. Later, when a' gets its chance to propose, and if none of the neighbours of a' at the rank of b accept the proposal from a', then a' will propose to the marked vertex b before proposing to the next lower-ranked neighbours. In contrast in (3)(i) above, when the proposal (a', b) is not uncertain and $a \succ_b a'$ then a' does not mark b. Note that a vertex $b \in \mathcal{B}$ can be part of an uncertain proposal at most once. Once a vertex receives its first proposal, it will remain matched and thereafter cannot be part of any uncertain proposal. Thus, any $b \in \mathcal{B}$ can be marked at most once during the course of the algorithm.

Now, we define the favourite neighbour of a vertex a, which is an adaptation of the definition in [13].

Definition 4 (Favourite neighbour of a). *Assume that k is the best rank at which some unproposed or marked neighbours of a exist in $\mathsf{Pref}(a)$. Then b is the favourite neighbour of a if one of the following conditions holds:*

(i) there exists at least one unmatched neighbour of a at the k^{th} rank and b has the lowest index among all such unmatched neighbours, or
(ii) all the k^{th} ranked neighbours of a are matched and b is the lowest index among all such neighbours which are unproposed by a, or
(iii) all the k^{th} ranked neighbours are already proposed by a and b has the lowest index among all the vertices which are marked by a.

Király's algorithm begins with every vertex $a \in \mathcal{A}$ being active. As long as there exists an active vertex which is unmatched and has not exhausted its preference list, the vertex proposes to its favourite neighbour. If $a \in \mathcal{A}$ remains unmatched after exhausting its preference list, it achieves a '$*$' status and starts proposing to vertices in $\mathsf{Pref}(a)$ with $*$ status. The $*$ status of a vertex a can be interpreted as improving the rank of a in $\mathsf{Pref}(b)$ by 0.5 for any neighbour b of a. Thus, the $*$ status vertex is used to decide between vertices in a tie, but does not affect strict preferences. It is shown in [13] that the resulting matching is a $\frac{3}{2}$-approximation of a maximum size stable matching.

Overview of the Popular Critical Matching Algorithm [23]. Now, we briefly describe the algorithm in [23] for computing the maximum size popular critical matching in the one-to-one strict list setting. Let s and t denote the number of critical vertices in $\mathcal{A}$ and $\mathcal{B}$, respectively. The algorithm in [23] is a multi-level algorithm which first matches as many critical vertices from $\mathcal{B}$ as possible. This is achieved by restricting unmatched vertices in $\mathcal{A}$ at levels $0, \ldots, t-1$ to propose only to critical vertices on the $\mathcal{B}$-side. At the level t, each vertex $a \in \mathcal{A}$ is allowed to propose *all* its neighbours. If a vertex $a \in \mathcal{A}$ remains unmatched even after exhausting its preference list at level t, a raises its level to $t+1$ and proposes to its neighbours until it is matched or it exhausts its preference list at the level $t+1$. If a critical vertex a remains unmatched then a raises its level above $t+1$ and continues proposing to all its neighbours until it is matched, or it exhausts its preference list at the highest level $s+t+1$. A vertex b which receives the proposal always prefers a higher level vertex a over any lower level vertex a' irrespective of the ranks of a and a' in $\mathsf{Pref}(b)$. It is shown in [23] that the resulting matching is a maximum size popular matching among all the critical matchings.

3 Algorithm for Computing CRITICAL-RSM

Our algorithm (see Algorithm 1) is a combination of Király's algorithm and the popular critical matching algorithm discussed in the previous section. In each level, vertices in $\mathcal{A}$ propose and vertices in $\mathcal{B}$ accept or reject. The set of vertices from $\mathcal{B}$ that $a \in \mathcal{A}$ proposes to depends on the level of a. Furthermore, depending on the level of a, the preference list at that level may be strict or may contain ties. Throughout Algorithm 1, b uses its original preference list $\mathsf{Pref}(b)$ which possibly contains ties. For a vertex $a \in \mathcal{A}$, let $\mathsf{PrefS}(a)$ denote a *strict* preference list obtained by breaking ties in $\mathsf{Pref}(a)$ in such a way that the vertices in ties are ordered by increasing order of their indices. Furthermore, let $\mathsf{PrefSC}(a)$ be the strict list obtained from $\mathsf{PrefS}(a)$ by omitting all the non-critical vertices from $\mathsf{PrefS}(a)$. For example, assume $\mathsf{Pref}(a) = (b_2, b_1), b_5, (b_3, b_4)$ where b_4 and b_5 are critical vertices. Here, a ranks b_1 and b_2 as rank-1, b_5 as rank-2 and b_3 and b_4 as rank-3. We have $\mathsf{PrefS}(a) = b_1, b_2, b_5, b_3, b_4$ and $\mathsf{PrefSC}(a) = b_5, b_4$ where comma separated vertices denote a strict ordering.

Initially, all the vertices in $\mathcal{A}$ have their levels set to 0. A vertex a at level ℓ is denoted as a^{ℓ}. At a level less than t, each $a \in \mathcal{A}$ proposes to vertices in $\mathsf{PrefSC}(a)$ (see Lines 4–8 of Algorithm 1). Each time it remains unmatched, it proposes to its *most preferred* neighbour b. The most preferred neighbour in $\mathsf{PrefSC}(a)$ or $\mathsf{PrefS}(a)$ is the best-ranked neighbour b to whom a has not yet proposed at the current level. If a remains unmatched after proposing to all its neighbours in $\mathsf{PrefSC}(a)$ at a level $\ell < t-1$, then a raises its level to $\ell+1$ and again proposes to vertices in $\mathsf{PrefSC}(a)$. In this part of the algorithm, we invoke `CriticalPropose()` which encodes the level-based accept/reject by b. A vertex $b \in \mathcal{B}$ prefers a_i^{ℓ} over $a_j^{\ell'}$ if :

(i) either $\ell > \ell'$ (ranks of a_i and a_j in $\mathsf{Pref}(b)$ do not matter) or
(ii) $\ell = \ell'$ and $a_i \succ_b a_j$.

If vertex a remains unmatched after exhausting $\mathsf{PrefSC}(a)$ at level $t-1$, a attains level t where it uses its original preference list $\mathsf{Pref}(a)$ which may contain ties. At level t our algorithm executes Király's algorithm [13]. This corresponds to Lines 10–13 of Algorithm 1. Király's algorithm is encoded in the procedure `TiesPropose()`. Since we have ties on both sides of the graph, at this level, we need the notion of a favourite neighbour and uncertain proposal defined in Sect. 2. If the vertex a remains unmatched after exhausting $\mathsf{Pref}(a)$ at level t, it attains the $*$ status, and for this, we have the sub-level t^*. The interpretation of the $*$ status is the same as discussed in Sect. 2.

If a *critical* vertex a remains unmatched after exhausting its preference list $\mathsf{Pref}(a)$ at level t^*, a raises its level to $t+1$, and starts proposing to vertices in $\mathsf{PrefS}(a)$ (see Lines 16–20 of Algorithm 1). It continues to do so until either it is matched or it has exhausted $\mathsf{PrefS}(a)$ at level $s+t$. In contrast, if a non-critical vertex a remains unmatched after exhausting its preference list $\mathsf{Pref}(a)$ at level t^*, a does not propose any further. Recall that $\mathsf{PrefS}(a)$ is a strict preference list containing all the neighbours (not restricted to critical vertices). Here, Algorithm 1, again invokes `CriticalPropose()` for the level-based accept/reject by b. The algorithm terminates when either (i) all the vertices in $\mathcal{A}$ are matched or

Algorithm 1: Critical relaxed stable matching in $G = (\mathcal{A} \cup \mathcal{B}, E)$

```
Set $M = \emptyset$, Initialize a queue $Q = \{a^0 \; : \; a \in \mathcal{A}\}$
while $Q$ is not empty do
    Let $a^\ell = dequeue(Q)$                              // $a$ is unmatched
    if $\ell < t$ then
        if $a^\ell$ has not exhausted $\mathsf{PrefSC}(a)$ then
            CriticalPropose($a^\ell, \mathsf{PrefSC}(a), M, Q$)
        else
            $\ell = \ell + 1$ and add $a^\ell$ to $Q$
    else if $\ell == t$ or $\ell == t^*$ then
        if $\exists\ b' \in \mathsf{Pref}(a)$ which is marked or unproposed by $a^\ell$ then
            TiesPropose($a^\ell, \mathsf{Pref}(a), M, Q$)
        else
            if $\ell == t$ then $\ell = t^*$ and add $a^\ell$ to $Q$
            if $\ell == t^*$ and $a$ is critical then $\ell = t + 1$ and add $a^\ell$ to $Q$
    else
        // $a$ is critical
        if $a^\ell$ has not exhausted $\mathsf{PrefS}(a)$ then
            CriticalPropose($a^\ell, \mathsf{PrefS}(a), M, Q$)
        else
            if $\ell < s + t$ and $a$ is critical then
                $\ell = \ell + 1$ and add $a^\ell$ to $Q$
return $M$
```

(ii) all unmatched critical $a \in \mathcal{A}$ have exhausted $\mathsf{PrefS}(a)$ at level $s + t$ and all unmatched non-critical $a \in \mathcal{A}$ have exhausted $\mathsf{Pref}(a)$ at level t^*. We note that $s + t = |\mathcal{C}| = O(n)$, where $n = |\mathcal{A} \cup \mathcal{B}|$ and each edge of G is explored at most $s + t + 3$ times (at most three times at level t, the Király's step, and at most once at every other level). Thus, the running time of our algorithm is $O(n \cdot |E|)$.

It is worth noting that in our algorithm, not all vertices in $\mathcal{A}$ propose at all levels. Similarly, not all vertices in $\mathcal{B}$ receive proposals from vertices at all levels. In other words, only critical vertices in $\mathcal{B}$ are allowed to receive proposals from vertices in $\mathcal{A}$ at levels at most $t - 1$, and only critical vertices in $\mathcal{A}$ are allowed to propose at levels above t. Also, note that when a vertex in $\mathcal{A}$ transitions to a higher level, it proposes to possibly a superset of vertices that it proposes to in the lower level (recall that $\mathsf{Pref}(a)$ and its strict counterpart $\mathsf{PrefS}(a)$ are both a superset of $\mathsf{PrefSC}(a)$). Therefore, we have the following useful observation.

Observation 1. *If a vertex $b \in \mathcal{B}$ receives a proposal from some $a' \in \mathcal{A}$ at a level z then b receives proposals from all its neighbours who exhausted their preference list at level z.*

4 Correctness of Our Algorithm

We prove that the matching M output by Algorithm 1 is

Procedure CriticalPropose(a^ℓ, $\mathsf{List}(a)$, M, Q)

```
Let $b$ be the most preferred unproposed vertex by $a^\ell$ in $\mathsf{List}(a)$
if $b$ is unmatched in $M$ then
    $M = M \cup \{(a^\ell, b)\}$
else
    Let $a_j^y = M(b)$
    if ($\ell > y$) or ($\ell == y$ and $a \succ_b a_j$) then
        $M = M \setminus \{(a_j^y, b)\} \cup \{(a^\ell, b)\}$ and add $a_j^y$ to $Q$
    else add $a^\ell$ to $Q$
```

Procedure TiesPropose(a^ℓ, $\mathsf{List}(a)$, M, Q)

```
Let $b$ be the favourite neighbour of $a^\ell$ in $\mathsf{List}(a)$ at rank $k$
if $b$ was marked by $a^\ell$ then $a^\ell$ unmarks $b$
if $b$ is unmatched then
    $M = M \cup \{(a^\ell, b)\}$
    if there exists an unmatched $b''$ at rank $k$ in $\mathsf{Pref}(a)$ then
        Set $(a^\ell, b)$ as uncertain proposal          // $\ell = t$ as $b''$ is unmatched
else if $b$ is part of an uncertain proposal $(a_j^y, b)$ then
    $M = (M \setminus \{(a_j^y, b)\}) \cup \{(a^\ell, b)\}$                // Here, $y = t$
    $a_j^y$ marks $b$ and add $a_j^y$ to $Q$
else if $b$ is not part of an uncertain proposal then
    Let $a_j^y = M(b)$
    if $\ell == t$ then
        if ($y < t$) or (($y == t$ or $y == t^*$) and $a \succ_b a_j$) then
            $M = M \setminus \{(a_j^y, b)\} \cup \{(a^\ell, b)\}$ and add $a_j^y$ to $Q$
        else add $a^\ell$ to $Q$
    if $\ell == t^*$ then
        if ($y < t$) or ($y == t$ and $a \succeq_b a_j$) or ($y == t^*$ and $a \succ_b a_j$) then
            $M = M \setminus \{(a_j^y, b)\} \cup \{(a^\ell, b)\}$ and add $a_j^y$ to $Q$
        else add $a^\ell$ to $Q$
```

(I) Critical as well as relaxed stable (RSM) and
(II) A $\frac{3}{2}$ approximation to the maximum size CRITICAL-RSM in G.

We define a partition of the vertices in $\mathcal{A} \cup \mathcal{B}$ based on the levels of vertices in $\mathcal{A}$ and the matching M. This partition is useful to establish the correctness of our algorithm.

Partition of Vertices: The vertex set $\mathcal{A}$ is partitioned into $\mathcal{A}_0 \cup \mathcal{A}_1 \cup \ldots \cup \mathcal{A}_t \cup \ldots \cup \mathcal{A}_{s+t}$, and the vertex set $\mathcal{B}$ is partitioned into $\mathcal{B}_0 \cup \mathcal{B}_1 \cup \ldots \cup \mathcal{B}_t \cup \ldots \cup \mathcal{B}_{s+t}$. For every matched vertex $a \in \mathcal{A}$ there exists $x \in \{0, \ldots, s+t\}$ such that $(a^x, b) \in M$. We use x to partition the vertex set. Note that if $(a^{t^*}, b) \in M$ then for the purpose of partitioning we consider $t^* = t$ as t^* is a sub-level of the level t.

- **Matched vertices in $\mathcal{A} \cup \mathcal{B}$**: Let $a \in \mathcal{A}, b \in \mathcal{B}$ and $(a^x, b) \in M$ for some $x \in \{0, \ldots, s+t\}$. We add a to $\mathcal{A}_x$ and b to $\mathcal{B}_x$.
- **Unmatched vertices in $\mathcal{A} \cup \mathcal{B}$**:
  - If a non-critical vertex $a \in \mathcal{A}$ is unmatched in M then we add a to $\mathcal{A}_t$.
  - If a critical vertex $a \in \mathcal{A}$ is unmatched in M then we add a to $\mathcal{A}_{s+t}$.
  - If a non-critical vertex $b \in \mathcal{B}$ is unmatched in M then we add b to $\mathcal{B}_t$.
  - If a critical vertex $b \in \mathcal{B}$ is unmatched in M then we add b to $\mathcal{B}_0$.

It is convenient to visualize the partitions as shown in Fig. 2. This particular drawing of the graph G is denoted by G_M throughout the rest of the section. It is useful to assume that the edges in G_M are implicitly directed from $\mathcal{A}$ to $\mathcal{B}$. By construction, the edges of M (shown in blue colour) are horizontal whereas the unmatched edges (shown as solid black edges) can be horizontal, upwards or downwards. We state the properties of the vertices and edges in G_M with respect to this partition in Property 1 (see the full version [21] for justification).

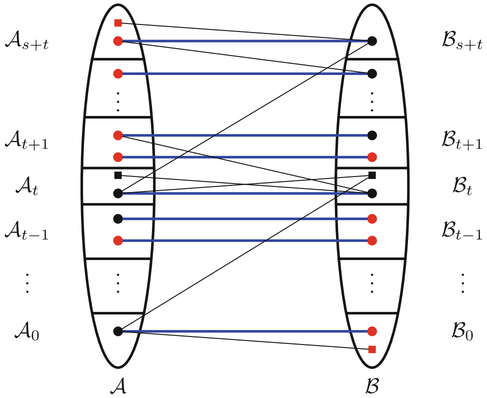


Fig. 2. The graph G_M. Red vertices are critical and black vertices are non-critical. Matched vertices are represented by circles, and unmatched vertices are represented by squares. The blue horizontal lines represent matched edges in M. Solid black lines represent edges which are not matched in M. Note that no edge in G_M is of the form $\mathcal{A}_x \times \mathcal{A}_y$ for $y \le x - 2$. (Color figure online)

Property 1. Let $a \in \mathcal{A}$ and $b \in \mathcal{B}$. Then the following hold in graph G_M.

1. If $a \in \bigcup_{x=t+1}^{s+t} \mathcal{A}_x$ then a is critical. Thus, $|\bigcup_{x=t+1}^{s+t} \mathcal{A}_x| \le s$.
2. If $b \in \bigcup_{x=0}^{t-1} \mathcal{B}_x$ then b is critical. Thus, $|\bigcup_{x=0}^{t-1} \mathcal{B}_x| \le t$.
3. If a is critical and is unmatched in M then $a \in \mathcal{A}_{s+t}$ and all the neighbours of a are matched and present in $\mathcal{B}_{s+t}$ only.
4. If a is not critical and is unmatched in M then $a \in \mathcal{A}_t$ and all the neighbours of a are matched and present in $\mathcal{B}_x$ for $x \ge t$.

5. If b is critical and is unmatched in M then $b \in \mathcal{B}_0$ and all the neighbours of b are present in $\mathcal{A}_0$ only.
6. If b is not critical and is unmatched in M then $b \in \mathcal{B}_t$ and all the neighbours of b are present in $\mathcal{A}_x$ for $x \leq t$.

Let $(a, b) \in E$ be an edge such that $a \in \mathcal{A}_x$ and $b \in \mathcal{B}_y$. We say that such an edge is of the form $\mathcal{A}_x \times \mathcal{B}_y$. Lemma 1 below gives an important property about the edges which cannot be present in G_M. An edge of the form $\mathcal{A}_x \times \mathcal{B}_y$ with $x > y + 1$ is referred to as a *steep downward* edge.

Lemma 1. *The graph G_M does not contain steep downward edges. That is, there is no edge in G_M of the form $\mathcal{A}_x \times \mathcal{B}_y$ such that $x > y + 1$.*

Proof. Let (a, b) be any edge in G_M such that $a \in \mathcal{A}_x$ and $b \in \mathcal{B}_y$. If b is unmatched, then irrespective of whether b is critical or not by Property 1(5) and Property 1(6), we have $x \leq y$. Now suppose that b is matched and $(a', b) \in M$. If $a = a'$ then by construction of G_M, $(a, b) \in \mathcal{A}_x \times \mathcal{B}_x$. If $a \neq a'$, then we use Claim 1, which is given below. It is immediate from this claim that b is in $\mathcal{B}_y$ for $y \geq x - 1$. □

Claim 1. *Let $(a, b) \in E \setminus M$ and b be matched in M to $\tilde{a}$ at level y, that is, $M(b) = \tilde{a}^y$. If the level x of a is at least 2 then $y \geq x - 1$.*

Proof. Suppose for contradiction that there exists $\tilde{a} \in \mathcal{A}$ such that $(\tilde{a}^y, b) \in M$ for $y < x - 1$. The fact that $(a, b) \in E$ and a achieves the level x implies that a remains unmatched after a^{x-1} exhausted its preference list $\mathsf{Pref}(a)$, $\mathsf{PrefS}(a)$ or $\mathsf{PrefSC}(a)$ as appropriate. Since b is matched to a vertex at level $y < x - 1$, and a^{x-1} exhausted its preference list, by Observation 1, b received a proposal from a^{x-1}. At this time, b must accept this proposal by rejecting $\tilde{a}^y$ because $y < x-1$. This implies $(\tilde{a}^y, b) \notin M$ which contradicts our assumption that $(\tilde{a}^y, b) \in M$ for $y < x - 1$. □

Lemma 2. *Let (a, b) be a blocking pair w.r.t. M. Then the corresponding edge in G_M is an upward edge.*

Proof. For the blocking pair (a, b) let a and b be at levels x and y, respectively. First, suppose that b is a critical vertex. Since (a, b) is a blocking pair, irrespective of whether a is matched or unmatched, a^x must have proposed to the critical vertex b. Thus, b cannot remain unmatched. This implies $M(b)$ exists. We consider the following two cases:

1. The proposal by a to b results in (a, b) to be uncertain: Note that a^x is rejected by b because b receives another proposal, and hence a^x marks b. Since (a, b) is a blocking pair, $M(a)$ is ranked lower than b. However, before proposing to any vertex ranked strictly lower than b, a^x must propose to the marked vertex b. At this point, either b is matched to a better preferred partner than a which contradicts that (a, b) blocks M, otherwise, b accepts the proposal from a^x. Thus, a^x is matched to either b or to some other vertex on the same rank as b. This implies (a, b) is not a blocking edge.

2. The proposal by a to b does not result in (a, b) to be uncertain: The fact that $a \succ_b M(b)$ implies $M(b)$ must be at a level y such that $y > x$. Thus, (a, b) edge is an upward edge in G_M.

Now, suppose that b is a non-critical vertex. Then by Property 1(2), $b \in \mathcal{B}_y$ for $y \geq t$. If $x < t$, then (a, b) is an upward edge, and we are done. Hence, assume that $x \geq t$. Since $x \geq t$, a^x is proposes to *all* of its neighbours. Again, since (a, b) is a blocking pair, irrespective of whether a is matched or unmatched, a^x must have proposed to b. Thus, b cannot be unmatched. Now, either b is matched to a better-preferred partner than a, which contradicts that (a, b) is a blocking pair or $M(b)$ is at a higher level than a and hence (a, b) is an upward edge. □

Lemma 3 below shows that the matching M output by Algorithm 1 is critical.

Lemma 3. *The output matching M is critical for G.*

Proof sketch: We prove the criticality of M by using the level structure of the graph G_M. The idea is to show that there is no alternating path ρ in G_M with respect to M such that the number of critical vertices matched in $M \oplus \rho$ is more than the number of critical vertices matched in M. We prove the criticality for the individual parts, that is, for $\mathcal{A}$-part and for $\mathcal{B}$-part. In other words, we show that M matches maximum possible critical nodes from $\mathcal{A}$-side, and maximum possible critical nodes from the $\mathcal{B}$-side. This immediately implies that M matches the maximum possible critical nodes that can be matched in *any* matching. Hence, M is critical. For the $\mathcal{A}$-part we show that the path $\rho = \langle u_0, v_1, u_1, \ldots \rangle$ begins at the highest level $s+t$ with an unmatched critical vertex $u_0 \in \mathcal{A}$. Using Property 1(5), we also show that at least the first two vertices on the $\mathcal{A}$-side (u_0 and u_1) on ρ are at the same level $s+t$. Then we argue that the other end of ρ must be at a level below $t+1$. Since there are no steep downward edges (Lemma 1), the path contains at least one vertex from each level $t+1, \ldots, s+t-1$. Thus, we have at least $s+1$ many vertices in $\mathcal{A}_{t+1} \cup \ldots \cup \mathcal{A}_{s+t}$. This contradicts Property 1(1). Proof for the $\mathcal{B}$-part is analogous. See full version [21] for the complete proof. □

Lemma 4. *The output matching M of Algorithm 1 is* RSM *for G.*

Proof. If there is no blocking pair w.r.t. M then we are done. Hence, assume that (a, b) is a blocking pair w.r.t. M. By Lemma 2, (a, b) is an upward edge. We consider two cases based on the level of b.

Case 1: $b \in \mathcal{B}_y$ **for** $y \leq t$. Clearly, $a \in \mathcal{A}_x$ for $x \leq t-1$. Thus, by the construction of G_M, a is matched, and hence $M(a)$ exists. Clearly, $M(a)$ is at level at most $t-1$. By Property 1(2), $M(a)$ is critical. Hence, the blocking pair (a, b) is justified by Condition 1 of Definition 2.

Case 2: $b \in \mathcal{B}_y$ **for** $y > t$. By construction of G_M, b is matched. Thus, $M(b)$ exists and $M(b) \in \mathcal{A}_x$ for $x \geq t+1$. By Property 1(1), $M(b)$ is critical. Hence, the blocking pair (a, b) is justified by Condition 2 of Definition 2. □

Lemma 5. *Let M' be any maximum size* CRITICAL-RSM *and M be the output of Algorithm 1 for an instance of our problem. Then $|M| \geq \frac{2}{3} \cdot |M'|$.*

Proof. We prove that $M \oplus M'$ does not admit any 1-length or 3-length augmenting path w.r.t. M. This immediately implies that $|M| \geq \frac{2}{3} \cdot |M'|$. If a is unmatched (critical or otherwise), we know from Property 1(3) and Property 1(4) that no neighbour b of a is unmatched in M. Thus, M is a maximal matching.

For contradiction assume that $M \oplus M'$ contains a 3-length augmenting path $\rho = \langle a_1, b, a, b_1 \rangle$ w.r.t. M. Here $(a, b) \in M$ and the other two edges are in M'. We show that (a, b) blocks M' and the blocking pair is not justified. This will contradict relaxed stability of M'. We first establish the levels of the vertices.

Levels of Vertices: The fact that a_1 remains unmatched in M implies that $a_1^{t^*}$ exhausted $\mathsf{Pref}(a_1)$. Thus, a_1 is at level at least t^*. Since b_1 remains unmatched in M, a did not exhaust $\mathsf{Pref}(a)$ at the level t. Thus, a is at level at most t. We claim that a_1 is not at level $t + 1$ or higher. If a_1 is at level $x \geq t + 1$ then a_1^x must have proposed to b as a_1 is unmatched in M. Since a is at level at most t, b must reject a and accept a_1 – a contradiction to $(a, b) \in M$. Thus, we conclude that a_1 is at level t^*. Now, if a is at level $y < t$ then b must reject a and accept a_1 as a_1 at level t^* proposed to it. Recall that t^* is a sub-level of t used in the algorithm, and t^* does not appear as a separate level in G_M. Thus, the vertices $a, a_1 \in \mathcal{A}_t$.

The Pair (a, b) **Blocks** M': If $a_1 \succ_b a$, then b would have accepted the proposal of a_1^t by rejecting a^t. Thus, $a \succeq_b a_1$. Since $a_1^{t^*}$ was rejected by b, it implies $M(b) = a$ and a_1 cannot be in tie for b, otherwise b would not have rejected a $*$ status vertex over a non $*$ status vertex. Thus, $a \succ_b a_1$. Now, we show that $b \succ_a b_1$. Suppose not. Then, if $b_1 \succ_a b$, then a^t must have proposed to b_1 before b and got matched to it – a contradiction that b_1 is unmatched. Hence, assume that $b =_a b_1$. In this case, when a^t proposes to b, the vertex b must also be unmatched; otherwise, b cannot be a favourite neighbour of a^t. This implies that a_1 proposes to b only *after* a proposes to b. Since b_1 was unmatched when a proposed to b, the proposal from a to b was uncertain. We claim that b must reject the proposal by a after the proposal (a, b) becomes uncertain due to a proposal by some vertex, possibly a_1^t. Such a vertex must exist because a_1^t proposed to b after (a, b) becomes uncertain. Since a has an unmatched neighbour b_1 at the same rank, a must have proposed b_1 before proposing to b again. This implies b_1 is matched, a contradiction. Thus, $b \succ_a b_1$; hence (a, b) blocks M'.

The Blocking Pair (a, b) **is not Justified:** In order to prove this, we show $b_1 = M'(a)$ and $a_1 = M'(b)$ are both non-critical. Note that b_1 is unmatched in M, hence if it is critical then $b_1 \in \mathcal{B}_0$ and the number of critical vertices on $\mathcal{B}$-side is at least 1 (that is $t \geq 1$). This implies that a cannot be at a level ≥ 1 since it has not yet proposed to at least one critical neighbour, namely b_1. Thus, b_1 is not critical. We finish the proof by showing that a_1 is also not critical. Note that a_1 is unmatched in M, hence, if it is critical then $a_1 \in \mathcal{A}_{s+t}$ and $s > 0$. This is a contradiction that $a_1 \in \mathcal{A}_t$. Thus, a_1 is not critical.

This finishes the proof that the claimed 3-length augmenting path w.r.t. M does not exist establishing the size guarantee. □

Using Lemma 3, Lemma 4 and Lemma 5, we establish Theorem 1.

5 Conclusion

In this work, we consider the problem of computing a matching in the stable marriage problem where ties and critical vertices can appear on both sides of the bipartition. We investigate a recently introduced notion of optimality called relaxed stability for our setting. We show that every instance of our problem admits a Relaxed Stable Matching (RSM) which is also critical. It follows from the known results [6,18] that computing a maximum size critical RSM is NP-hard and hard to approximate within any factor smaller than $\frac{21}{19}$. We present a polynomial-time algorithm to compute a $\frac{3}{2}$-approximation of the maximum size critical RSM.

References

1. Biró, P., Irving, R.W., Manlove, D.F.: Popular matchings in the marriage and roommates problems. In: Calamoneri, T., Diaz, J. (eds.) CIAC 2010. LNCS, vol. 6078, pp. 97–108. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13073-1_10
2. Dean, B., Jalasutram, R.: Factor revealing LPs and stable matching with ties and incomplete lists. In: Proceedings of the 3rd International Workshop on Matching Under Preferences, pp. 42–53 (2015)
3. Gale, D., Shapley, L.S.: College admissions and the stability of marriage. Am. Math. Mon. **69**(1), 9–15 (1962)
4. Gärdenfors, P.: Match making: assignments based on bilateral preferences. Behav. Sci. **20**(3), 166–173 (1975)
5. Goko, H., Makino, K., Miyazaki, S., Yokoi, Y.: Maximally satisfying lower quotas in the hospitals/residents problem with ties. In: 39th International Symposium on Theoretical Aspects of Computer Science (2022)
6. Halldórsson, M.M., Iwama, K., Miyazaki, S., Yanagisawa, H.: Improved approximation results for the stable marriage problem. ACM Trans. Algorithm (TALG) **3**(3), 30-es (2007)
7. Hamada, K., Iwama, K., Miyazaki, S.: The hospitals/residents problem with lower quotas. Algorithmica **74**(1), 440–465 (2016)
8. Hamada, K., Miyazaki, S., Yanagisawa, H.: Strategy-proof approximation algorithms for the stable marriage problem with ties and incomplete lists. In: International Symposium on Algorithms and Computation (2019)
9. Huang, C.C., Kavitha, T.: Improved approximation algorithms for two variants of the stable marriage problem with ties. Math. Program. **154**, 353–380 (2015)
10. Iwama, K., Miyazaki, S., Yanagisawa, H.: A 25/17-approximation algorithm for the stable marriage problem with one-sided ties. Algorithmica **68**(3), 758–775 (2014)
11. Kavitha, T.: Matchings, critical nodes, and popular solutions. In: 41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2021) (2021)
12. Király, Z.: Better and simpler approximation algorithms for the stable marriage problem. Algorithmica **60**(1), 3–20 (2011)
13. Király, Z.: Linear time local approximation algorithm for maximum stable marriage. Algorithms **6**(3), 471–484 (2013)

14. Krishnaa, P., Limaye, G., Nasre, M., Nimbhorkar, P.: Envy-freeness and relaxed stability: hardness and approximation algorithms. In: Harks, T., Klimm, M. (eds.) SAGT 2020. LNCS, vol. 12283, pp. 193–208. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-57980-7_13
15. Krishnaa, P., Limaye, G., Nasre, M., Nimbhorkar, P.: Envy-freeness and relaxed stability: hardness and approximation algorithms. J. Comb. Optim. **45**(1), 1–30 (2023)
16. Lam, C.K., Plaxton, C.G.: A (1+ 1/e)-approximation algorithm for maximum stable matching with one-sided ties and incomplete lists. In: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 2823–2840. SIAM (2019)
17. Makino, K., Miyazaki, S., Yokoi, Y.: Incomplete list setting of the hospitals/residents problem with maximally satisfying lower quotas. In: Kanellopoulos, P., Kyropoulou, M., Voudouris, A. (eds.) SAGT 2022. Lecture Notes in Computer Science, vol. 13584, pp. 544–561. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15714-1_31
18. Manlove, D.F., Irving, R.W., Iwama, K., Miyazaki, S., Morita, Y.: Hard variants of stable marriage. Theoret. Comput. Sci. **276**(1–2), 261–279 (2002)
19. McDermid, E.: A 3/2-approximation algorithm for general stable marriage. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 689–700. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02927-1_57
20. Nasre, M., Nimbhorkar, P.: Popular matchings with lower quotas. In: 37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2017), pp. 44:1–44:15 (2017)
21. Nasre, M., Nimbhorkar, P., Ranjan, K.: Critical relaxed stable matchings with two-sided ties. arXiv preprint arXiv:2303.12325 (2023)
22. Nasre, M., Nimbhorkar, P., Ranjan, K., Sarkar, A.: Popular matchings in the hospital-residents problem with two-sided lower quotas. In: 41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2021), vol. 213, pp. 30:1–30:21 (2021)
23. Nasre, M., Nimbhorkar, P., Ranjan, K., Sarkar, A.: Popular critical matchings in the many-to-many setting. arXiv:2206.12394 (2023)
24. Paluch, K.: Faster and simpler approximation of stable matchings. Algorithms **7**(2), 189–202 (2014)
25. Robards, P.A.: Applying two-sided matching processes to the united states navy enlisted assignment process. Technical report, NAVAL POSTGRADUATE SCHOOL MONTEREY CA (2001)
26. Roth, A.E.: Stability and polarization of interests in job matching. Econometrica: J. Econometric Soc. **52**, 47–57 (1984)
27. Roth, A.E.: On the allocation of residents to rural hospitals: a general property of two-sided matching markets. Econometrica: J. Econometric Soc. **54**, 425–427 (1986)
28. Tan, S.J., Yeong, C.M.: Designing economics experiments to demonstrate the advantages of an electronic employment market in a large military organization. Technical report, NAVAL POSTGRADUATE SCHOOL MONTEREY CA (2001)
29. Yang, W., Sycara, K.: Two-sided matching for the us navy detailing process with market complication. Technical report, Technical Report CMU-RI-TR-03-49, Robotics Institute, Carnegie-Mellon University (2003)
30. Yokoi, Y.: Envy-free matchings with lower quotas. Algorithmica **82**(2), 188–211 (2020)

Graph Search Trees and Their Leaves

Robert Scheffler(✉)

Institute of Mathematics, Brandenburg University of Technology, Cottbus, Germany
robert.scheffler@b-tu.de

Abstract. Graph searches and their respective search trees are widely used in algorithmic graph theory. The problem whether a given spanning tree can be a graph search tree has been considered for different searches, graph classes and search tree paradigms. Similarly, the question whether a particular vertex can be visited last by some search has been studied extensively in recent years. We combine these two problems by considering the question whether a vertex can be a leaf of a graph search tree. We show that for particular search trees, including DFS trees, this problem is easy if we allow the leaf to be the first vertex of the search ordering. We contrast this result by showing that the problem becomes hard for many searches, including DFS and BFS, if we forbid the leaf to be the first vertex. Additionally, we present several structural and algorithmic results for search tree leaves of chordal graphs.

Keywords: Graph search · Graph search trees · Leaves

1 Introduction

Graph searches are an extensively used concept in algorithmic graph theory. The searches BFS and DFS belong to the most basic algorithms and are used in a wide range of applications as subroutines. The same holds for more sophisticated searches as LBFS, LDFS, and MCS (see, e.g., [4,8,14]).

An important structure closely related to a graph search is the corresponding search tree. Such a tree contains all the vertices of the graph and for every vertex different from the start vertex exactly one edge to a vertex preceding it in the search ordering. Those trees can be of particular interest as for instance the tree obtained by a BFS contains the shortest paths from the root to all other vertices in the graph and DFS trees are used for fast planarity testing [19]. Furthermore, trees generated by LBFS were used to design a linear-time implementation of the search LDFS for chordal graphs [3].

The problem of deciding whether a given spanning tree of a graph can be obtained by a particular search was introduced by Hagerup [17] in 1985, who presented a linear-time algorithm that recognizes DFS trees. In the same year, Hagerup and Nowak [18] gave a similar result for the BFS tree recognition. In 2021, Beisegel et al. [2] presented a more general framework for the search tree recognition problem. They introduced the term $\mathcal{F}$-tree for search trees where a

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 462–476, 2023.
https://doi.org/10.1007/978-3-031-43380-1_33

vertex is connected to its first visited neighbor, i.e., BFS-like trees, and $\mathcal{L}$-trees for search trees where a vertex is connected to its most recently visited neighbor, i.e., DFS-like trees. They showed, among other things, that $\mathcal{F}$-tree recognition is NP-hard for LBFS, LDFS, and MCS on weakly chordal graphs, while the problem can be solved in polynomial time for all three searches on chordal graphs. These results are complemented in [29], where it is shown that the recognition of $\mathcal{F}$-trees of DFS and $\mathcal{L}$-trees of BFS is NP-hard, a strong contrast to the polynomial results for $\mathcal{F}$-trees of BFS and $\mathcal{L}$-trees of DFS.

Another feature of a graph search that was used several times within algorithms are its end-vertices, i.e., the vertices that can be visited last by the search. Some of these end-vertices have nice properties. One example are the end-vertices of LBFS on chordal graphs. These vertices are simplicial, a fact that was used by Rose et al. [27] to design a linear-time recognition algorithm for chordal graphs. Furthermore, the end-vertices of LBFS are strongly related to dominating pairs of AT-free graphs [13] and transitive orientations of comparability graphs [16]. Thus, it is well motivated to consider the end-vertex problem, i.e., the question whether a given vertex of a graph is an end-vertex of a particular search. Introduced in 2010 by Corneil et al. [11], the problem has gained much attention by several researchers, leading to a wide range of hardness results and algorithms for different searches on different graph classes (see, e.g., [1,6,15,22,25,33]).

If we compare the known complexity results for the end-vertex problem and the recognition problem of $\mathcal{F}$-trees, we notice strong similarities between these two problems. Motivated by that fact, a generalization of both problems, called *Partial Search Order Problem*, was introduced in [28]. This problem asks whether a given partial order on a graph's vertex set can be linearly extended by a search ordering. Another way to combine the end-vertex problem with the search tree recognition problems is motivated by the following observation: If a vertex is the end-vertex of some search ordering, then it is a leaf in the respective search tree, no matter whether we consider the $\mathcal{F}$-tree or the $\mathcal{L}$-tree. Therefore, we ask whether a given vertex can be a leaf of a search tree constructed by a particular search. Note that this problem was first suggested in 2020 by Michel Habib. Here, we study its complexity for $\mathcal{F}$-trees and $\mathcal{L}$-trees of several searches, including BFS, DFS, LBFS, LDFS, and MCS.

Our Contribution. We consider two different types of leaves of search trees. A leaf is a *root leaf* of a search tree if it is the start vertex of the respective search ordering. All other leaves of a search tree are called *branch leaves.* We show that it is easy for all the searches considered here to identify the possible root leaves both for $\mathcal{F}$-trees and for $\mathcal{L}$-trees. For some searches, including DFS, these results imply directly that the general problem of recognizing leaves of $\mathcal{L}$-trees is easy. This is contrasted by the result that, at least for DFS, the recognition of branch leaves of $\mathcal{L}$-trees is NP-hard. We show that the same holds for $\mathcal{F}$-tree branch leaves of several searches, including DFS and BFS. In contrast, the leaves of $\mathcal{L}$-trees of BFS can be recognized in polynomial time for bipartite graphs. This is quite surprising since the $\mathcal{L}$-tree recognition problem of BFS is NP-hard on bipartite graphs [29] while $\mathcal{F}$-trees of BFS can be recognized efficiently on general graphs [18]. In the final section we consider chordal graphs and show that on this graph class the branch leaves of almost all considered searches can be recognized in linear time.

Due to lack of space, the proofs of some results are omitted here. They can be found in the full version [30].

2 Preliminaries

General Notation. The graphs considered in this paper are finite, undirected, simple and connected. Given a graph G, we denote by $V(G)$ the *set of vertices* and by $E(G)$ the *set of edges.* The terms $n(G)$ and $m(G)$ describe the number of vertices and edges of G, respectively, i.e., $n(G) = |V(G)|$ and $m(G) = |E(G)|$. For a vertex $v \in V(G)$, we denote by $N_G(v)$ the *(open) neighborhood* of v in G, i.e., the set $N_G(v) = \{u \in V \mid uv \in E\}$ where uv denotes an edge between u and v. The *closed neighborhood* of a vertex v is the union of the open neighborhood of v with the set $\{v\}$ and is denoted by $N_G[v]$. Given a set $S \subseteq V(G)$, the term $G[S]$ describes the subgraph of G that is induced by S.

The *distance* $dist_G(v,w)$ of two vertices v and w in G is the length (i.e., the number of edges) of the shortest v-w-path in G. The *eccentricity* $ecc_G(v)$ of a vertex v in G is the largest distance of v to any other vertex in G. The *diameter* $diam(G)$ of G is the largest eccentricity of a vertex in G and the *radius* $rad(G)$ of G is the smallest eccentricity of a vertex in G. A vertex v with $ecc_G(v) = rad(G)$ is called *central vertex* of G. The set $N_G^{\ell}(v)$ contains all vertices whose distance to the vertex v in G is equal to ℓ.

A *vertex ordering* of G is a bijection $\sigma : \{1, 2, \ldots, |V(G)|\} \rightarrow V(G)$. We denote by $\sigma^{-1}(v)$ the position of vertex $v \in V(G)$. Given two vertices u and v in G we say that u is *to the left* (resp. *to the right*) of v if $\sigma^{-1}(u) < \sigma^{-1}(v)$ (resp. $\sigma^{-1}(u) > \sigma^{-1}(v)$) and we denote this by $u \prec_\sigma v$ (resp. $u \succ_\sigma v$).

A *clique* in a graph G is a set of pairwise adjacent vertices and an *independent set* in G is a set of pairwise nonadjacent vertices. A clique C is *dominating* if any vertex of G is either in C or has a neighbor in C. A vertex v is *simplicial* if its neighborhood induces a clique. A vertex v of a connected graph G is a *cut vertex* if $G - v$ is not connected. Two vertices u and w form a *two-pair* if any induced path between u and w has length two.

A graph is *bipartite* if its vertex set can be partitioned into two independent sets X and Y. A graph is *weakly chordal* if G contains neither an induced cycle of the length ≥ 5 nor the complement of such an induced cycle. A graph is *chordal* if it does not contain an induced cycle of length ≥ 4. A vertex ordering σ of a graph G is a *perfect elimination ordering* if any vertex v is simplicial in the graph $G[S(v)]$ with $S(v) := \{w \mid w \prec_\sigma v\}$. A graph G has a PEO if and only if G is chordal [26]. A *split graph* G is a graph whose vertex set can be partitioned into sets C and I, such that C is a clique in G and I is an independent set in G. It is easy to see that any split graph is chordal.

Algorithm 1: Label Search($\prec_{\mathcal{A}}$)

```
Input: A graph $G$
Output: A search ordering $\sigma$ of $G$
begin
    foreach $v \in V(G)$ do $label(v) \leftarrow \emptyset$ for $i \leftarrow 1$ to $n(G)$ do
        $Eligible \leftarrow \{x \in V(G) \mid x$ unnumbered and $\nexists$ unnumbered $y \in V(G)$
                                such that $label(x) \prec_{\mathcal{A}} label(y)\}$;
        let $v$ be an arbitrary vertex in $Eligible$;
        $\sigma(i) \leftarrow v$;                                  /* assigns to $v$ the number $i$ */
        foreach unnumbered vertex $w \in N(v)$ do $label(w) \leftarrow label(w) \cup \{i\}$
```

A *tree* is an acyclic connected graph. A *spanning tree* of a graph G is an acyclic connected subgraph of G which contains all vertices of G. A tree together with a distinguished *root vertex* r is said to be *rooted*. In such a rooted tree T, a vertex v is the *parent* of vertex w if v is an element of the unique path from w to the root r and the edge vw is contained in T. A vertex w is called the *child* of v if v is the parent of w.

Searches, Search Trees and Their Leaves. In the most general sense, a *graph search* $\mathcal{A}$ is a function that maps every graph G to a set $\mathcal{A}(G)$ of vertex orderings of G. The elements of the set $\mathcal{A}(G)$ are the $\mathcal{A}$-*orderings of* G. The graph searches considered in this paper can be formalized adapting a framework introduced by Corneil et al. [10] (a similar framework is given in [23]). This framework uses subsets of $\mathbb{N}^+$ as vertex labels. Whenever a vertex is numbered, its index in the search ordering is added to the labels of its unnumbered neighbors. The search $\mathcal{A}$ is defined via a strict partial order $\prec_{\mathcal{A}}$ on the elements of $\mathcal{P}(\mathbb{N}^+)$ (see Algorithm 1). The respective $\mathcal{A}$-orderings are exactly those vertex orderings that can be found by this framework using the partial label order $\prec_{\mathcal{A}}$.

In the following, we define the searches considered in this paper by presenting suitable partial orders $\prec_{\mathcal{A}}$ (see [10]). The *Generic Search* (GS) is equal to the Label Search($\prec_{GS}$) where $A \prec_{GS} B$ if and only if $A = \emptyset$ and $B \neq \emptyset$. Thus, any vertex with a numbered neighbor can be numbered next.

The partial label order $\prec_{BFS}$ for *Breadth First Search* (BFS) is defined as follows: $A \prec_{BFS} B$ if and only if $A = \emptyset$ and $B \neq \emptyset$ or $\min(A) > \min(B)$. For the *Lexicographic Breadth First Search* (LBFS) [27] we consider the partial order $\prec_{LBFS}$ with $A \prec_{LBFS} B$ if and only if $A \subsetneq B$ or $\min(A \setminus B) > \min(B \setminus A)$. Both BFS and LBFS are *layered*, i.e., the sets $N_G^{\ell}(r)$ are consecutive within orderings starting in r. We sometimes use the term *layer* if we refer to a set $N_G^{\ell}(r)$.

The partial label order $\prec_{DFS}$ for *Depth First Search* (DFS) is defined as follows: $A \prec_{DFS} B$ if and only if $A = \emptyset$ and $B \neq \emptyset$ or $\max(A) < \max(B)$. For the *Lexicographic Depth First Search* [12] we use the strict partial order $\prec_{LDFS}$ where $A \prec_{LDFS} B$ if and only if $A \subsetneq B$ or $\max(A \setminus B) < \max(B \setminus A)$.

The *Maximum Cardinality Search* (MCS) [32] uses the partial order $\prec_{MCS}$ with $A \prec_{MCS} B$ if and only if $|A| < |B|$. The *Maximal Neighborhood Search* (MNS) [12] is defined using $\prec_{MNS}$ with $A \prec_{MNS} B$ if and only if $A \subsetneq B$. It follows directly from these partial label orders, that any LBFS, LDFS, and MCS ordering is also an MNS ordering. Furthermore, the orderings of all presented searches are GS orderings.

Searches as BFS and DFS are often used to compute corresponding graph search trees. Beisegel et al. [2] formalized the different concepts of search trees as follows.

Definition 2.1 (Beisegel et al. [2]). *Let σ be a GS ordering of a connected graph G. The $\mathcal{F}$-*tree *of σ is the spanning tree of G containing the edge from each vertex v with $\sigma^{-1}(v) > 1$ to its leftmost neighbor in σ.*

*The $\mathcal{L}$-*tree *of σ is the spanning tree containing the edge from each vertex v with $\sigma^{-1}(v) > 1$ to its rightmost neighbor w in σ with $w \prec_\sigma v$.*

In this paper, we consider the leaves of these search trees. For both $\mathcal{F}$-trees and $\mathcal{L}$-trees, we distinguish two different types of leaves.

Definition 2.2. *Let σ be a GS ordering of a connected graph G. A vertex $v \in V(G)$ is an $\mathcal{F}$-*leaf *(*$\mathcal{L}$-leaf*) of σ if v is a leaf in the $\mathcal{F}$-tree ($\mathcal{L}$-tree) of σ. If v is the first vertex of σ, then it is the $\mathcal{F}$-*root leaf *($\mathcal{L}$-*root leaf*) of σ, otherwise it is an $\mathcal{F}$-*branch leaf *($\mathcal{L}$-*branch leaf*) of* σ.

As the graph with exactly one vertex has no leaf in its spanning tree, we will consider only graphs with at least two vertices.

3 Root Leaves

We start this section with the simple observation that $\mathcal{F}$-root leaves of GS orderings of a graph G are quite boring as they are exactly the leaves of G.

Observation 3.1. *Let G be a connected graph with $n(G) \geq 2$. The following conditions are equivalent for a vertex $v \in V(G)$.*

(i) Vertex v is the $\mathcal{F}$-root leaf of some GS ordering of G.
(ii) Vertex v is the $\mathcal{F}$-root leaf of every GS ordering of G starting in v.
(iii) Vertex v is a leaf of G.

Next we consider the $\mathcal{L}$-root leaves of GS, DFS, and MCS. They are exactly those vertices of the graph that are not cut vertices. The same even holds for $\mathcal{F}$-branch leaves and $\mathcal{L}$-branch leaves of GS.

Theorem 3.2. *Let G be a connected graph with $n(G) \geq 2$. The following conditions are equivalent for a vertex $v \in V(G)$.*

(i) Vertex v is the $\mathcal{L}$-root leaf of some DFS ordering of G starting in v.
(ii) Vertex v is the $\mathcal{L}$-root leaf of every DFS ordering of G starting in v.

(iii) Vertex v is the $\mathcal{L}$-root leaf of some MCS ordering of G.
(iv) Vertex v is the $\mathcal{L}$-root leaf of some GS ordering of G.
(v) Vertex v is an $\mathcal{L}$-branch leaf of some GS ordering of G.
(vi) Vertex v is an $\mathcal{F}$-branch leaf of some GS ordering of G.
(vii) Vertex v is the end-vertex of some GS ordering of G.
(viii) Vertex v is not a cut vertex of G.

Note that DFS differs from GS and MCS in this result. While for the latter three searches it is possible that a vertex is not the $\mathcal{L}$-root leaf of a search ordering starting with that vertex, this is not possible for DFS.

Since DFS, LDFS, MCS, and MNS orderings are also GS orderings, Theorem 3.2 directly implies that we can characterize the $\mathcal{L}$-leaves of these orderings.

Theorem 3.3. *For any search $\mathcal{A} \in \{$GS, DFS, LDFS, MCS, MNS$\}$ and any vertex v of a connected graph G with $n(G) \geq 2$, the following statements are equivalent.*

(i) Vertex v is the $\mathcal{L}$-root leaf of some $\mathcal{A}$-ordering of G.
(ii) Vertex v is an $\mathcal{L}$-leaf of some $\mathcal{A}$-ordering of G.
(iii) Vertex v is not a cut vertex of G.

As we can check in linear time whether a vertex is a cut vertex, we can also recognize $\mathcal{L}$-leaves of GS, DFS, LDFS, MCS, and MNS within this time bound. However, we will see in Corollary 4.2 that at least for DFS the recognition of $\mathcal{L}$-branch leaves is NP-complete.

The characterization of $\mathcal{L}$-root leaves given in Theorem 3.2 does not work for BFS as the following theorem shows.

Theorem 3.4. *Let G be a connected graph with $n(G) \geq 2$. A vertex $v \in V(G)$ is the $\mathcal{L}$-root leaf of some BFS ordering of G if and only if $G[N_G(v)]$ is connected.*

4 NP-Hardness of Branch Leaf Recognition

Branch Leaves of DFS. DFS $\mathcal{L}$-trees can be recognized in linear time [17,20]. As we have seen in Theorem 3.3, this also holds for DFS $\mathcal{L}$-leaves. In contrast, recognizing DFS $\mathcal{L}$-branch leaves of a graph is as hard as the recognition of DFS end-vertices since the two concepts are equivalent.

Theorem 4.1. *A vertex $v \in V(G)$ of a graph G is an $\mathcal{L}$-branch leaf of some DFS ordering of G if and only if v is the end-vertex of some DFS ordering of G.*

Charbit et al. [6] gave sufficient conditions on a graph class $\mathcal{G}$ such that the end-vertex problem of DFS is NP-complete on $\mathcal{G}$. Due to Theorem 4.1, we can replace the term end-vertex in their result by the term $\mathcal{L}$-branch leaf.

Corollary 4.2. *Let $\mathcal{G}$ be a graph class that is closed under the insertion of universal vertices. If the Hamiltonian path problem is* NP*-complete on $\mathcal{G}$, then the problem of deciding whether a vertex of a graph $G \in \mathcal{G}$ is an $\mathcal{L}$-branch leaf of some DFS ordering of G is* NP*-complete. In particular, the problem is* NP*-complete on split graphs.*

A similar result can be given for $\mathcal{F}$-branch leaves of DFS. By adapting the proof given in [29] that $\mathcal{F}$-trees of DFS are hard to recognize, we can show that the same holds for $\mathcal{F}$-branch leaves of DFS.

Theorem 4.3. *Let $\mathcal{G}$ be a graph class that is closed under the insertion of universal vertices and leaves. If the Hamiltonian path problem is* NP*-complete on $\mathcal{G}$, then the problem of deciding whether a vertex of a graph $G \in \mathcal{G}$ is an $\mathcal{F}$-branch leaf of some DFS ordering of G is* NP*-complete. In particular, the problem is* NP*-complete on chordal graphs.*

If we compare Corollary 4.2 and Theorem 4.3, then we see that for $\mathcal{L}$-branch leaves it is sufficient that the graph class $\mathcal{G}$ is closed under the addition of universal vertices while for $\mathcal{F}$-branch leaves we have the additional condition that $\mathcal{G}$ is closed under the addition of leaves. We cannot omit this constraint (unless $\mathsf{P} = \mathsf{NP}$) as the $\mathcal{F}$-branch leaf recognition problem of DFS can be solved in polynomial time on split graphs (see Corollary 5.13).

Branch Leaves of BFS. The end-vertex problem of BFS is NP-complete, even if the graph is bipartite and the start vertex of the BFS ordering is fixed [6]. This fact can be used to show that recognizing BFS $\mathcal{F}$-branch leaves is also NP-complete.

Theorem 4.4. *It is* NP*-complete to decide whether a vertex of a bipartite graph G is an $\mathcal{F}$-branch leaf of some BFS ordering of G.*

In contrast to this result, there is a simple characterization of BFS $\mathcal{L}$-branch leaves of bipartite graphs.

Theorem 4.5. *Let G be a connected bipartite graph with $n(G) \geq 2$. A vertex $v \in V(G)$ is an $\mathcal{L}$-branch leaf of some BFS ordering of G if and only if there is an $r \in V(G) \setminus \{v\}$ such that $dist_G(r, w) = dist_{G-v}(r, w)$ for all $w \in V(G) \setminus \{v\}$.*

Proof. Assume that there is a vertex $r \in V(G) \setminus \{v\}$ such that $dist_G(r, w) = dist_{G-v}(r, w)$ for all $w \in V(G) \setminus \{v\}$. Let $(r = w_0, \dots, w_k = v)$ be a shortest path from r to v, i.e., v has distance k to r. It is easy to see that there is a BFS ordering σ of G in which every vertex w_i is the first vertex of the i-th layer. Let T be the $\mathcal{L}$-tree of σ and let x be a vertex in the $(k+1)$-th layer. Due to the condition on r, there is a shortest path from r to x in G that does not use vertex v. Therefore, x has a neighbor y in the k-th layer that is not v. Since $v \prec_\sigma y \prec_\sigma x$, vertex v is not the parent of x in T. Since G is bipartite, the layers of σ are independent sets and, thus, v is neither the parent of any vertex in the k-th layer. Hence, v is a leaf of T.

Now assume that v is a branch leaf of the $\mathcal{L}$-tree T of the BFS ordering σ. Let r be the start vertex of σ. Let w be a vertex different from v and r. Consider the r-w-path P in T. Since G is bipartite, the edges of G and, thus, the edges of T only connect vertices of consecutive layers. Furthermore, every vertex has a neighbor in its preceding layer. Thus, P has $dist_G(r, w)$ edges. Since v is a leaf of T, P does not contain v. Therefore, P is also contained in $G - v$ and $dist_{G-v}(r, w) = dist_G(r, w)$. □

To check whether the condition of Theorem 4.5 is fulfilled, we simply make two all-pair-shortest paths computations and compare the results. This can be done in $\mathcal{O}(n(G) \cdot m(G))$ by using $\mathcal{O}(n(G))$ many BFS computations.

Corollary 4.6. *Given a connected bipartite graph G and a vertex $v \in V(G)$, we can decide in time $\mathcal{O}(n(G) \cdot m(G))$ whether v is the $\mathcal{L}$-branch leaf of some BFS ordering of G.*

The results of Theorem 4.4 and Corollary 4.6 are quite surprising since the $\mathcal{L}$-tree recognition problem of BFS is NP-hard on bipartite graphs [29] while the $\mathcal{F}$-tree recognition problem of BFS can be solved in linear time [18,24].

Branch Leaves of MNS-like Searches. For several subsearches of MNS, the recognition problem of $\mathcal{F}$-branch leaves is NP-complete on weakly chordal graphs.

Theorem 4.7. *Let $\mathcal{A}$ be one of the following searches: LBFS, LDFS, MCS, MNS. It is* NP*-complete to decide whether a vertex of a weakly chordal graph G is an $\mathcal{F}$-branch leaf of some $\mathcal{A}$-ordering.*

Proof. The proof of the theorem is inspired by the NP-completeness proof of the $\mathcal{F}$-tree recognition problem of MNS given by Beisegel et al. [2]. We construct a polynomial-time reduction from 3-SAT. Let $\mathcal{I}$ be an instance of 3-SAT. W.l.o.g. we may assume that $\mathcal{I}$ contains at least two clauses. We construct the corresponding graph $G(\mathcal{I})$ as follows. Let $X = \{x_1, \dots, x_k, \overline{x}_1, \dots, \overline{x}_k\}$ be the set of vertices representing the literals of $\mathcal{I}$. The graph $G(\mathcal{I})[X]$ forms the complement of the matching in which x_i is matched to $\overline{x}_i$ for every $i \in \{1, \dots, k\}$. Let $C = \{c_1, \dots, c_\ell\}$ be the set of vertices representing the clauses of $\mathcal{I}$. The set C forms an independent set in $G(\mathcal{I})$ and every clause vertex c_i is adjacent to each vertex of X whose corresponding literal is contained in the clause associated with c_i. Additionally, we add a universal vertex t.

Assume $G(\mathcal{I})$ has a fulfilling assignment $\mathcal{B}$. Then we create the following $\mathcal{A}$-ordering σ. We first number all literal vertices of literals that are set to true in $\mathcal{B}$ and then we number t. Since these vertices form a clique, this ordering is a prefix of an $\mathcal{A}$-ordering. We number the remaining vertices following an arbitrary $\mathcal{A}$-ordering. As $\mathcal{B}$ is fulfilling, all clause vertices and all literal vertices have a neighbor that is to the left of t in σ. Thus, t is an $\mathcal{F}$-branch leaf of σ.

Now assume that t is an $\mathcal{F}$-branch leaf of the $\mathcal{A}$-ordering σ of $G(\mathcal{I})$. Let S be the set of literal vertices that are to the left of t in σ. Since t is universal and the edges $x_i\overline{x}_i$ are missing, the set S contains at most one literal vertex for every variable. Thus, we can define an assignment $\mathcal{B}$ by giving all literals whose vertices are contained in S the value true. If some variable value is not fixed, then we choose an arbitrary value for the variable. If a clause vertex has a parent in the $\mathcal{F}$-tree T of σ, then this parent is an element of S since t is a leaf in T. If the clause vertex c_i does not have a parent in T, then c_i is the first vertex of σ. Since there are at least two clause vertices, the second vertex of σ is not t but a literal vertex adjacent to c_i. Therefore, every clause vertex has a neighbor in S and, thus, $\mathcal{B}$ is a fulfilling assignment.

To see that $G(\mathcal{I})$ is weakly chordal, we first observe that every pair $(x_i, \overline{x}_i)$ forms a two-pair in $G(\mathcal{I})$. Spinrad and Sritharan [31] showed that the graph that results from the addition of an edge between a two-pair is weakly chordal if and only if the initial graph is weakly chordal. If we add all the edges $x_i\overline{x}_i$, then the resulting graph is a split graphs and, thus, $G(\mathcal{I})$ is weakly chordal. □

5 Branch Leaves and Chordal Graphs

Branch Leaves of MNS-Like Searches. MNS and all of its subsearches compute PEOs of chordal graphs [12,32]. Thus, any $\mathcal{F}$-tree or $\mathcal{L}$-tree of an MNS ordering is also an $\mathcal{F}$-tree or $\mathcal{L}$-tree of some PEO. Beisegel et al. [2] showed that this also holds the other way around for a large family of graph searches including LBFS, LDFS, MCS, and MNS, i.e., the rooted $\mathcal{F}$-trees and rooted $\mathcal{L}$-trees of these searches on chordal graphs are exactly the rooted $\mathcal{F}$-trees and rooted $\mathcal{L}$-trees of PEOs, respectively. Therefore, we will only characterize $\mathcal{F}$-branch leaves and $\mathcal{L}$-branch leaves of PEOs.

We start by showing that the $\mathcal{L}$-branch leaves of PEOs of a chordal graph are exactly the graph's simplicial vertices.

Theorem 5.1. *Let G be a connected chordal graph with $n(G) \geq 2$. A vertex $v \in V(G)$ is an $\mathcal{L}$-branch leaf of some PEO of G if and only if v is simplicial.*

Proof. If v is a simplicial vertex, then there is a PEO σ that ends with v. Vertex v is an $\mathcal{L}$-branch leaf of σ.

For the other direction, let σ be a PEO and let v be a non-simplicial vertex of G. Hence, not all neighbors of v are to the left of v in σ. Let w be the leftmost neighbor of v in σ that is to the right of v in σ. Let x be the parent of w in the $\mathcal{L}$-tree T of σ. If x is not equal v, then it holds $v \prec_\sigma x \prec_\sigma w$. As σ is a PEO and $vw, xw \in E(G)$, the edge vx is also in $E(G)$; a contradiction to the choice of w. Hence, v is the parent of w in T and v is not an $\mathcal{L}$-branch leaf of σ. □

Since we can decide in linear time whether a vertex is simplicial [1], we can recognize $\mathcal{L}$-branch leaves of PEOs in linear time.

Corollary 5.2. *Given a connected chordal graph G and a vertex $v \in V(G)$, we can decide in time $\mathcal{O}(n(G)+m(G))$ whether v is the $\mathcal{L}$-branch leaf of some PEO of G. Therefore, we can also decide in time $\mathcal{O}(n(G)+m(G))$ whether v is the $\mathcal{L}$-branch leaf of some LBFS, LDFS, MCS, or MNS ordering.*

Obviously, simplicial vertices are also $\mathcal{F}$-branch leaves of PEOs. However, there are further $\mathcal{F}$-branch leaves.

Theorem 5.3. *Let G be a connected chordal graph with $n(G) \geq 2$. A vertex $v \in V(G)$ is an $\mathcal{F}$-branch leaf of some PEO of G if and only if the graph $G[N_G(v)]$ has a dominating clique.*

Proof. First assume that $G[N_G(v)]$ has a dominating clique C. It is obvious that there is an LBFS ordering σ of G that starts with the vertices of C. The ordering σ is a PEO. Since all neighbors of v have a neighbor in C or are elements of C, v is an $\mathcal{F}$-branch leaf of σ.

Now let v be an $\mathcal{F}$-branch leaf of the PEO σ. Let S be the set of neighbors of v that are to the left of v in σ. The set S induces a clique of G. Thus, if $S = N_G(v)$, then $G[N_G(v)]$ is a clique and we are done. Hence, we may assume that there is a vertex $w \in N_G(v) \setminus S$. As w is not a child of v in the $\mathcal{F}$-tree of σ, there is a vertex $x \in N_G(w)$ with $x \prec_\sigma v$. Since σ is a PEO, vertex x is a neighbor of v and, thus, $x \in S$. Thus, any neighbor of v that is not in S has a neighbor in S and, hence, S induces a dominating clique of $G[N_G(v)]$. □

To decide the complexity of the $\mathcal{F}$-branch leaf recognition problem of PEOs, we examine the complexity of deciding the existence of a dominating clique in a chordal graph. Kratsch et al. [21] showed that such a clique exists if and only if the diameter of the graph is at most three.

Theorem 5.4 (Kratsch et al. [21]). *A chordal graph G has a dominating clique if and only if the diameter of G is at most three.*

As the diameter of a graph can be determined by computing $n(G)$ many BFS orderings, we can decide the existence of a dominating clique in a chordal graph in polynomial time. Although it is unlikely that the diameter of a chordal graph can be computed in linear time,[1] we can improve our algorithm to decide the existence of a dominating clique in linear time. To this end, we can use the following result of Corneil et al. [9].

Theorem 5.5 (Corneil et al. [9]). *Let G be a chordal graph and let $v \in V(G)$ be the end-vertex of some LBFS ordering of G. If $ecc(v) < diam(G)$, then $ecc(v)$ is even and $ecc(v) = diam(G) - 1$.*

Combining Theorems 5.3 to 5.5, we can give a linear-time recognition algorithm for $\mathcal{F}$-branch leaves of PEOs.

Corollary 5.6. *Given a chordal graph G and a vertex $v \in V(G)$, we can decide in time $\mathcal{O}(n(G) + m(G))$ whether v is an $\mathcal{F}$-branch leaf of some PEO of G. Therefore, we can also decide in time $\mathcal{O}(n(G) + m(G))$, whether v is the $\mathcal{F}$-branch leaf of some LBFS, LDFS, MCS, or MNS ordering.*

[1] Even on split graphs, the diameter cannot be computed in subquadratic time unless the Strong Exponential Time Hypothesis fails [5].

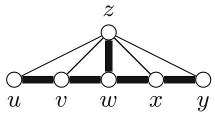


Fig. 1. The given graph G is chordal. There is no dominating clique in the graph $G[N_G(z)]$. However, the given spanning tree is the $\mathcal{F}$-tree of the BFS ordering (w, v, x, z, u, y) and, thus, z is a $\mathcal{F}$-branch leaf of BFS.

Proof. Due to Theorems 5.3 and 5.4, it is sufficient to check whether $G' = G[N_G(v)]$ has diameter 3. We compute an LBFS ordering σ of G' in linear time. Let v be the end-vertex of σ. We compute the eccentricity of v in G' in linear time by starting a BFS in v. If $ecc_{G'}(v) > 3$, then the diameter of G' is larger than 3. If $ecc_{G'}(v) = 3$, then, by Theorem 5.5, $diam(G') = 3$. If $ecc_{G'}(v) < 3$, then $diam(G') \leq ecc_{G'}(v) + 1 \leq 3$, due to Theorem 5.5. □

Branch Leaves of BFS. The condition given in Theorem 5.3 is also sufficient for a vertex to be a BFS $\mathcal{F}$-branch leaf since every LBFS ordering is also a BFS ordering. However, it is not necessary as can be seen in Fig. 1. To characterize BFS $\mathcal{F}$-branch leaves of chordal graphs, we start with the following two lemmas.

Lemma 5.7. *Let G be a chordal graph and let r be a vertex in $V(G)$. Let x and y be two vertices in $N_G^i(r)$. If there is a vertex $z \in N_G^{i+1}(r)$ which is adjacent to both x and y, then $xy \in E(G)$.*

Lemma 5.8. *Let G be a chordal graph and let r be a vertex in $V(G)$. Let x and y be two vertices in $N_G^i(r)$. If $xy \in E(G)$, then $N_G(x) \cap N_G^{i-1}(r) \subseteq N_G(y)$ or $N_G(y) \cap N_G^{i-1}(r) \subseteq N_G(x)$.*

The next lemma makes a statement about the distances of neighbors of a vertex in a chordal graph.

Lemma 5.9. *Let G be a connected chordal graph and let $v \in V(G)$. For any $x, y \in N_G(v)$, the distance between x and y in $G - v$ is equal to the distance between x and y in $G[N_G(v)]$.*

Using Lemmas 5.7 to 5.9, we characterize BFS $\mathcal{F}$-branch leaves of chordal graphs.

Theorem 5.10. *Let G be a connected chordal graph with $n(G) \geq 2$. A vertex $v \in V(G)$ is an $\mathcal{F}$-branch leaf of some BFS ordering of G if and only if the radius of $G[N_G(v)]$ is at most two.*

Proof. First assume that $G[N_G(v)]$ has radius two and let w be a central vertex of $G[N_G(v)]$. There is a BFS ordering σ that starts with w followed by all neighbors of w that are not v. Vertex v is an $\mathcal{F}$-branch leaf of σ since all neighbors of v have some neighbor in $N_G[w] \setminus \{v\}$ or are equal to w.

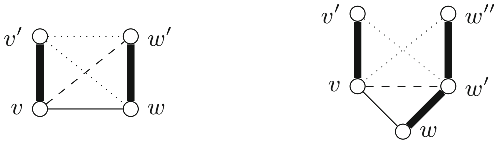


Fig. 2. Two cases of the proof of Theorem 5.10. The vertical arrangement of the vertices represent their layers. Thick edges are edges of the $\mathcal{F}$-tree. Dotted edges are not present. Dashed edges are implied by either Lemma 5.7 or Lemma 5.8.

Now assume that $v \in N_G^i(r)$ is an $\mathcal{F}$-branch leaf of some BFS ordering σ of G starting with r. Let T be the $\mathcal{F}$-tree of σ rooted in r and let v' be the parent of v in T. Since T is an BFS $\mathcal{F}$-tree rooted in r, it holds that $v' \in N_G^{i-1}(r)$.

We claim that in $G-v$ vertex v' has a distance of at most two to every element of $N_G(v)$. Let $w \in N_G(v) \setminus \{v'\}$. If $v'w \in E(G)$, then v' and w have distance one in $G-v$. Therefore, we may assume in the following that $v'w \notin E(G)$. Then Lemma 5.7 implies that $w \notin N_G^{i-1}(r)$. Furthermore, the parent of w in T, say w', is different from v and v'. If $v'w' \in E(G)$, then v' and w have distance two in $G-v$ via the path (v', w', w). Thus, we may also assume that $v'w' \notin E(G)$.

First assume that $w \in N_G^i(r)$ (see left part of Fig. 2). Then $w' \in N_G^{i-1}(r)$. Lemma 5.8 implies that w' is adjacent to v because vw, vv', and $ww' \in E(G)$ but $v'w \notin E(G)$. Now the non-existence of $v'w'$ contradicts Lemma 5.7.

Now assume that $w \in N_G^{i+1}(r)$ (see right part of Fig. 2). Then, due to Lemma 5.7, $vw' \in E(G)$. Since $v'w' \notin E(G)$, the parent of w', say w'', is different from v'. Since w' is the parent of w, it holds that $w' \prec_\sigma v$. This implies that $w'' \prec_\sigma v'$. Therefore, w'' is not adjacent to v since, otherwise, v' would not be the parent of v. The non-existence of both $v'w'$ and vw'' contradicts Lemma 5.8.

Summarizing, v' has distance at most two in $G-v$ to any neighbor of v. By Lemma 5.9, v' has distance at most two in $G[N_G(v)]$ to any neighbor of v in G and, thus, $G[N_G(v)]$ has radius at most two. □

Chepoi and Dragan [7] presented a linear-time algorithm that computes a central vertex of a chordal graph. As the eccentricity of such a vertex can be computed in linear time using BFS, we can compute the radius of a chordal graph and, in particular, of $G[N_G(v)]$ in linear time. Thus, Theorem 5.10 implies a linear-time algorithm for the BFS $\mathcal{F}$-branch leaf recognition on chordal graphs.

Corollary 5.11. *Given a connected chordal graph G and a vertex $v \in V(G)$, we can decide in time $\mathcal{O}(n(G)+m(G))$ whether v is an $\mathcal{F}$-branch leaf of some BFS ordering of G.*

Branch Leaves of DFS. As we have seen in Theorem 4.3, the $\mathcal{F}$-branch leaf recognition problem of DFS is NP-complete on chordal graphs. However, there is a simple characterization of DFS $\mathcal{F}$-branch leaves of split graphs.

Theorem 5.12. *Let G be a connected split graph with $n(G) \geq 2$. A vertex $v \in V(G)$ is an $\mathcal{F}$-branch leaf of some DFS ordering if and only if v is not a cut vertex of G.*

As cut vertices can be identified in linear time, Theorem 5.12 leads directly to a linear-time algorithm for the DFS $\mathcal{F}$-branch leaf recognition on split graphs.

Corollary 5.13. *Given a connected split graph G and a vertex $v \in V(G)$, we can decide in time $\mathcal{O}(n(G) + m(G))$ whether v is an $\mathcal{F}$-branch leaf of some DFS ordering of G.*

In contrast to this result, it is NP-hard to decide whether a vertex of a split graph is a DFS $\mathcal{L}$-branch leaf (see Corollary 4.2). Thus, the $\mathcal{L}$-branch leaf recognition of DFS seems to be harder than the $\mathcal{F}$-branch leaf recognition of DFS, a surprising contrast to the hardness of the DFS $\mathcal{F}$-tree recognition problem [29] and the easiness of the DFS $\mathcal{L}$-tree recognition problem [17,20]. Recall that we have made a similar observation for the complexity of the branch leaf and tree recognition of BFS (see Theorem 4.4 and Corollary 4.6).

References

1. Beisegel, J., et al.: On the end-vertex problem of graph searches. Discret. Math. Theor. Comput. Sci. **21**(1) (2019). https://doi.org/10.23638/DMTCS-21-1-13
2. Beisegel, J., et al.: The recognition problem of graph search trees. SIAM J. Discret. Math. **35**(2), 1418–1446 (2021). https://doi.org/10.1137/20M1313301
3. Beisegel, J., Köhler, E., Scheffler, R., Strehler, M.: Linear time LexDFS on chordal graphs. In: Grandoni, F., Herman, G., Sanders, P. (eds.) 28th Annual European Symposium on Algorithms (ESA 2020). LIPIcs, vol. 173, pp. 13:1–13:13. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, Dagstuhl (2020). https://doi.org/10.4230/LIPIcs.ESA.2020.13
4. Berry, A., Blair, J.R., Heggernes, P., Peyton, B.W.: Maximum cardinality search for computing minimal triangulations of graphs. Algorithmica **39**(4), 287–298 (2004). https://doi.org/10.1007/s00453-004-1084-3
5. Borassi, M., Crescenzi, P., Habib, M.: Into the square: On the complexity of some quadratic-time solvable problems. In: Crescenzi, P., Loreti, M. (eds.) Proceedings of ICTCS 2015, the 16th Italian Conference on Theoretical Computer Science, ENTCS, vol. 322, pp. 51–67 (2016). https://doi.org/10.1016/j.entcs.2016.03.005
6. Charbit, P., Habib, M., Mamcarz, A.: Influence of the tie-break rule on the end-vertex problem. Discrete Math. Theor. Comput. Sci. **16**(2), 57 (2014). https://doi.org/10.46298/dmtcs.2081
7. Chepoi, V., Dragan, F.: A linear-time algorithm for finding a central vertex of a chordal graph. In: van Leeuwen, J. (ed.) ESA 1994. LNCS, vol. 855, pp. 159–170. Springer, Heidelberg (1994). https://doi.org/10.1007/BFb0049406
8. Corneil, D.G., Dalton, B., Habib, M.: LDFS-based certifying algorithm for the minimum path cover problem on cocomparability graphs. SIAM J. Comput. **42**(3), 792–807 (2013). https://doi.org/10.1137/11083856X
9. Corneil, D.G., Dragan, F.F., Habib, M., Paul, C.: Diameter determination on restricted graph families. Discret. Appl. Math. **113**(2), 143–166 (2001). https://doi.org/10.1016/S0166-218X(00)00281-X
10. Corneil, D.G., Dusart, J., Habib, M., Mamcarz, A., De Montgolfier, F.: A tie-break model for graph search. Discret. Appl. Math. **199**, 89–100 (2016). https://doi.org/10.1016/j.dam.2015.06.011

11. Corneil, D.G., Köhler, E., Lanlignel, J.M.: On end-vertices of lexicographic breadth first searches. Discret. Appl. Math. **158**(5), 434–443 (2010). https://doi.org/10.1016/j.dam.2009.10.001
12. Corneil, D.G., Krueger, R.M.: A unified view of graph searching. SIAM J. Discret. Math. **22**(4), 1259–1276 (2008). https://doi.org/10.1137/050623498
13. Corneil, D.G., Olariu, S., Stewart, L.: Linear time algorithms for dominating pairs in asteroidal triple-free graphs. SIAM J. Comput. **28**(4), 1284–1297 (1999). https://doi.org/10.1137/S0097539795282377
14. Corneil, D.G., Olariu, S., Stewart, L.: The LBFS structure and recognition of interval graphs. SIAM J. Discret. Math. **23**(4), 1905–1953 (2009). https://doi.org/10.1137/S0895480100373455
15. Gorzny, J., Huang, J.: End-vertices of LBFS of (AT-free) bigraphs. Discret. Appl. Math. **225**, 87–94 (2017). https://doi.org/10.1016/j.dam.2017.02.027
16. Habib, M., McConnell, R., Paul, C., Viennot, L.: Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. Theoret. Comput. Sci. **234**(1–2), 59–84 (2000). https://doi.org/10.1016/S0304-3975(97)00241-7
17. Hagerup, T.: Biconnected graph assembly and recognition of DFS trees. Technical Report, A 85/03, Universität des Saarlandes (1985). https://doi.org/10.22028/D291-26437
18. Hagerup, T., Nowak, M.: Recognition of spanning trees defined by graph searches. Technical Report, A 85/08, Universität des Saarlandes (1985)
19. Hopcroft, J., Tarjan, R.E.: Efficient planarity testing. J. ACM **21**(4), 549–568 (1974). https://doi.org/10.1145/321850.321852
20. Korach, E., Ostfeld, Z.: DFS tree construction: algorithms and characterizations. In: van Leeuwen, J. (ed.) WG 1988. LNCS, vol. 344, pp. 87–106. Springer, Heidelberg (1989). https://doi.org/10.1007/3-540-50728-0_37
21. Kratsch, D., Damaschke, P., Lubiw, A.: Dominating cliques in chordal graphs. Discret. Math. **128**(1), 269–275 (1994). https://doi.org/10.1016/0012-365X(94)90118-X
22. Kratsch, D., Liedloff, M., Meister, D.: End-vertices of graph search algorithms. In: Paschos, V.T., Widmayer, P. (eds.) CIAC 2015. LNCS, vol. 9079, pp. 300–312. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18173-8_22
23. Krueger, R., Simonet, G., Berry, A.: A general label search to investigate classical graph search algorithms. Discret. Appl. Math. **159**(2–3), 128–142 (2011). https://doi.org/10.1016/j.dam.2010.02.011
24. Manber, U.: Recognizing breadth-first search trees in linear time. Inf. Process. Lett. **34**(4), 167–171 (1990). https://doi.org/10.1016/0020-0190(90)90155-Q
25. Rong, G., Cao, Y., Wang, J., Wang, Z.: Graph searches and their end vertices. Algorithmica **84**, 2642–2666 (2022). https://doi.org/10.1007/s00453-022-00981-5
26. Rose, D.J.: Triangulated graphs and the elimination process. J. Math. Anal. Appl. **32**(3), 597–609 (1970). https://doi.org/10.1016/0022-247X(70)90282-9
27. Rose, D.J., Tarjan, R.E., Lueker, G.S.: Algorithmic aspects of vertex elimination on graphs. SIAM J. Comput. **5**(2), 266–283 (1976). https://doi.org/10.1137/0205021
28. Scheffler, R.: Linearizing partial search orders. In: Bekos, M.A., Kaufmann, M. (eds.) WG 2022. LNCS, vol. 13453, pp. 425–438. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15914-5_31
29. Scheffler, R.: On the recognition of search trees generated by BFS and DFS. Theoret. Comput. Sci. **936**, 116–128 (2022). https://doi.org/10.1016/j.tcs.2022.09.018
30. Scheffler, R.: Graph search trees and their leaves. Preprint on arXiv (2023). https://doi.org/10.48550/arXiv.2307.07279

31. Spinrad, J., Sritharan, R.: Algorithms for weakly triangulated graphs. Discret. Appl. Math. **59**(2), 181–191 (1995). https://doi.org/10.1016/0166-218X(93)E0161-Q
32. Tarjan, R.E., Yannakakis, M.: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. SIAM J. Comput. **13**(3), 566–579 (1984). https://doi.org/10.1137/0213035
33. Zou, M., Wang, Z., Wang, J., Cao, Y.: End vertices of graph searches on bipartite graphs. Inf. Proc. Lett. **173**, 106176 (2022). https://doi.org/10.1016/j.ipl.2021.106176

Author Index

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 477–478, 2023.
https://doi.org/10.1007/978-3-031-43380-1