

**Daya Gaur
Rogers Mathew (Eds.)**

LNCS 15536

Algorithms and Discrete Applied Mathematics

**11th International Conference, CALDAM 2025
Coimbatore, India, February 13–15, 2025
Proceedings**



Springer

Founding Editors

Gerhard Goos

Juris Hartmanis

Editorial Board Members

Elisa Bertino, *Purdue University, West Lafayette, IN, USA*

Wen Gao, *Peking University, Beijing, China*

Bernhard Steffen , *TU Dortmund University, Dortmund, Germany*

Moti Yung , *Columbia University, New York, NY, USA*

The series Lecture Notes in Computer Science (LNCS), including its subseries Lecture Notes in Artificial Intelligence (LNAI) and Lecture Notes in Bioinformatics (LNBI), has established itself as a medium for the publication of new developments in computer science and information technology research, teaching, and education.

LNCS enjoys close cooperation with the computer science R & D community, the series counts many renowned academics among its volume editors and paper authors, and collaborates with prestigious societies. Its mission is to serve this international community by providing an invaluable service, mainly focused on the publication of conference and workshop proceedings and postproceedings. LNCS commenced publication in 1973.

Daya Gaur · Rogers Mathew
Editors

Algorithms and Discrete Applied Mathematics

11th International Conference, CALDAM 2025
Coimbatore, India, February 13–15, 2025
Proceedings

Editors

Daya Gaur 
University of Lethbridge
Lethbridge, AB, Canada

Rogers Mathew 
Indian Institute of Technology Hyderabad
Sangareddy, Telangana, India

ISSN 0302-9743

ISSN 1611-3349 (electronic)

Lecture Notes in Computer Science

ISBN 978-3-031-83437-0

ISBN 978-3-031-83438-7 (eBook)

<https://doi.org/10.1007/978-3-031-83438-7>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2025

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

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

If disposing of this product, please recycle the paper.

Preface

This volume contains the papers presented at CALDAM 2025 (the 11th International Conference on Algorithms and Discrete Applied Mathematics), held on February 13–15, 2025, at the PSG College of Technology, Coimbatore, Tamilnadu, India. CALDAM 2025 was organized by PSG College of Technology and the Association for Computer Science and Discrete Mathematics (ACSDM), India. The program committee comprised 30 highly experienced and active researchers from various countries.

The conference had papers in algorithms and complexity, discrete applied mathematics, computational geometry, graph theory, graph colouring, graph partition, and domination in graphs. We received 89 submissions from all over the world. Each paper was reviewed by 3 program committee members and other experts. The program committee decided to accept 30 papers for presentation and inclusion in the proceedings of CALDAM 2025 published by Springer. The program included three invited talks by Wilfried Imrich (Montanuniversität Leoben - Technical University), Manoj Changat (University of Kerala), and Wolfgang Mulzer (Freie Universität Berlin).

We thank the authors of all submissions for considering CALDAM 2025 as a potential venue for disseminating their work. We are very much indebted to the program committee members and the external reviewers for providing reviews within a short period. We thank Springer for publishing the proceedings in the Lecture Notes in Computer Science series. Our sincere thanks to the invited speakers, Wilfried Imrich, Manoj Changat, and Wolfgang Mulzer, for accepting our invitation to give a talk. We thank the organizing committee chaired by R. S. Lekshmi of PSG College of Technology Coimbatore for the smooth conduct of CALDAM 2025 and PSG College of Technology, for providing the necessary facilities.

We thank the chair of the steering committee, Subir Ghosh, for his help and active support. We thank the chairs of the previous edition, Subrahmanyam Kalyanasundaram and Anil Maheshwari, for assisting with various process details. We also thank Springer for its support for the best paper presentation awards and Google India for support.

February 2025

Daya Gaur
Rogers Mathew

Organization

Steering Committee

Subir Kumar Ghosh (Chair)	Ramakrishna Mission Vivekananda Educational and Research Institute, India
Gyula O. H. Katona János Pach	Hungarian Academy of Sciences, Hungary École Polytechnique Fédérale de Lausanne, Switzerland
Swami Sarvottamananda	Ramakrishna Mission Vivekananda Educational and Research Institute, India
Chee Yap	New York University, USA

Program Committee

Aida Abiad	Eindhoven University of Technology, Netherlands
Akanksha Agrawal	IIT Madras, India
Niranjan Balachandran	IIT Bombay, India
Sayan Bandyapadhyay	Portland State University, USA
Aritra Banik	NISER, India
Ahmad Biniaz	University of Windsor, Canada
Manoj Changat	University of Kerala, India
Sandip Das	Indian Statistical Institute Kolkata, India
Mark de Berg	Eindhoven University of Technology, Netherlands
Jean-Lou De Carufel	University of Ottawa, Canada
Palash Dey	IIT Kharagpur, India
Florent Foucaud	Université Clermont Auvergne, France
Sumit Ganguly	IIT Kanpur, India
Daya Gaur (Co-chair)	University of Lethbridge, Canada
Sathish Govindarajan	Indian Institute of Science, India
Ragesh Jaiswal	IIT Delhi, India
Subrahmanyam Kalyanasundaram	IIT Hyderabad, India
Anil Maheshwari	Carleton University, Canada
Rogers Mathew (Co-chair)	IIT Hyderabad, India
Neeldhara Misra	IIT Gandhinagar, India
Wolfgang Mulzer	Freie Universität Berlin, Germany
Rahul Muthu	DA-IICT, India
Arti Pandey	IIT Ropar, India

Iztok Peterin	University of Maribor, Slovenia
Monika Pilšniak	AGH University of Science and Technology, Poland
Rishi Ranjan Singh	IIT Bhilai, India
Bodhayan Roy	IIT Kharagpur, India
Swagato Sanyal	IIT Kharagpur, India
Sagnik Sen	IIT Dharwad, India

Additional Reviewers

Antony, Dhanyamol	Mande, Nikhil
Ashok, Pradeesha	Mann, Kevin
Bag, Kunal	Manna, Bubai
Bandopadhyay, Susobhan	Marcille, Clara
Basu Roy, Aniket	Mittal, Rajat
Baswana, Surender	Mukherjee, Anish
Bhattacharya, Srimanta	Myint, Zin Mar
Bhyravarapu, Sriram	Nandakumar, Satyadev
Chakraborty, Dibyayan	Narayanaswamy, N. S.
Chakraborty, Sourav	Neuen, Daniel
Chen, Tianzhi	Nguyen, Linh
D. K., Supraja	Pal, Sudebkumar Prasant
Das, Arun Kumar	Paul, Kaustav
Das, Gautam K.	Ramanna, Somindu C.
Das, Soura Sena	Reddy, Sangam Balchandar
Dey, Hiranya	Reddy, Vinod
Dey, Sanjana	S. Taruni
Dhar, Amit Kumar	Sahu, Abhishek
Fernau, Henning	Sankarnarayanan, Brahadheesh
Gahlawat, Harmender	Sanpui, Md Azharuddin
Ghosh, Arijit	Sarkar, Siddhartha
Gorain, Barun	Shankar, Umesh
Inamdar, Tanmay	Tale, Prafullkumar
Jamshidian, Mahya	Tewari, Raghunath
Kumar, Gunjan	Tripathi, Vikash
Kundu, Madhumita	Upasana, Anannya
Mahato, Iswar	Venkitesh, S.

Organizing Committee

L. Gopalakrishnan	PSG College of Technology, India
R. S. Lekshmi (Chair)	PSG College of Technology, India

K. Prakasan
R. Nadarajan
Shina Sheen

PSG College of Technology, India
PSG College of Technology, India
PSG College of Technology, India

Abstracts of Invited Talks

Graph Products, Prime Factorization, Unique Roots and Cancellation

Wilfried Imrich

Montanuniversität Leoben, Austria

This talk combines results about unique prime factorizations of connected graphs, with respect to various graph products, with algebraic results about unique n^{th} roots and cancellation in commutative and non-commutative polynomial rings, and their power series rings.

On the way we pose several problems about prime factorizations, the computation of prime factors, and the role of spectra in this connection.

We begin with products of rooted and unrooted graphs. The products are non-associative, and non-commutative. That is, they have an ordering, or hierarchy, of the factors.

The first product that we consider is the hierarchical product, as introduced by Barrière, Comellas et al. in 2009. It is non-associative, strictly non-commutative, and prime factorization is not unique. Nonetheless, the prime factors of finite connected graphs are uniquely determined as unrooted graphs, and their ordering too.

For this product prime factorization of infinite connected graphs need not be unique, but it is unique for homogeneous trees of finite degree, and for connected graphs without proper fixed-point free auto-endomorphism. One of the consequences is that prime factorization of infinite connected rayless graphs follows the same rules as prime factorization for finite connected graphs.

A similar product is the rooted hierarchical product. It is associative, strictly non-commutative, and prime factorization of finite graphs is unique if they are connected, but not necessarily when they are disconnected.

This is taken up in the second part of the talk, in which we consider disconnected graphs, where each component is a finite rooted graph. We allow that the number of components is infinite, but require that each component has finite multiplicity.

Multiplication of such graphs via the rooted hierarchical product is distribute with respect to the disjoint union of graphs. Hence, these graphs form rings over prime graphs if there are only finitely many components, and power series rings otherwise.

We show that these classes of graphs have unique n^{th} roots and that cancellation holds, although prime factorization of disconnected graphs is not unique.

In the third part we transfer these results to the rooted generalized hierarchical product, the Cartesian product, the Cartesian product with loops, the strong product, and the direct product of non-bipartite graphs with loops.

A comparison with similar older results concludes the talk.

Disk Graphs and Transmission Graphs – Recent Developments

Wolfgang Mulzer

Freie Universität Berlin, Germany

Let S be a set of n points in the plane such that each point p in S has an associated radius $r_p > 0$. Then, S induces a set of planar disks, by drawing around each point p from S a disk of radius r_p . The disk graph for S is the undirected graph with vertex set S in which two points from S are adjacent if and only if their associated disks intersect. The transmission graph for S is the directed graph in which each point p in S has outgoing edges to all the points that lie inside the associated disk for p .

It turns out that the additional structure that is provided by disk graphs and transmission graphs makes it possible to develop algorithms that are simpler and much more efficient than algorithms for general graphs. In my talk, I will survey a few recent results that showcase the power of this approach, and I will present open problems for further research.

Based on joint work with many people, most prominently Alexander Baumann, Haim Kaplan, Katharina Kloß, Kristin Knorr, Liam Roditty, and Paul Seiferth.

Convex Geometries Determined by Betweenness Axioms

Manoj Changat

University of Kerala, India

A *transit function* is a set-valued function defined for every pair of elements u, v in a non-empty set V that returns a subset $R(u, v)$ of V which satisfies the isotone, symmetric and idempotent axioms. The transit functions are introduced to generalize natural **betweenness**, **intervals**, and **convex sets** in an axiomatic setup on several mathematical structures, including vector spaces, graphs, posets, lattices, hypergraphs, and other discrete structures.

Given the function R on V , a set $S \subseteq V$ is said to be R convex if, for $R(u, v) \subseteq S, \forall u, v \in S$ and the family of all R convex sets of V is called R -convexity denoted as \mathcal{C}_R (also known as *interval convexity*) on V . The R -convex hull of a subset S of V is the smallest R -convex subset containing S . For an R -convex set K in an R convexity, an element $k \in K$ is called an *extreme point*, if the set $K \setminus \{k\}$ is also R -convex. R -convexity is a convex geometry if every R -convex set K is the convex hull of extreme points of K .

This paper is motivated by the idea of betweenness through transit functions to capture abstract convex geometries, which are interval convexities. We try to address the following problem.

Problem: Is it possible to characterize convex geometries arising from interval convexities \mathcal{C}_R using the betweenness axioms of the corresponding transit function R ?

Here, we partially solve this problem and provide connections of our results to the well-known instances of convex geometries which are interval convexities. This is joint work with Lekshmi Kamal K. Sheela, Iztok Peterin, and Ameera Vaheeda Shanavas.

Contents

Addressing Bias in Algorithmic Solutions: Exploring Vertex Cover and Feedback Vertex Set	1
<i>Sheikh Shakil Akhtar, Jayakrishnan Madathil, Pranabendu Misra, and Geevarghese Philip</i>	
Graceful Coloring is Computationally Hard	13
<i>Cyriac Antony, Jacob Antony, D. Laavanya, and S. Devi Yamini</i>	
Parameterized Complexity of Coupon Coloring of Graphs	25
<i>Pradeesha Ashok, Pradyun Devarakonda, Shiven Phogat, Swaroop A. Ram Rayala, and J. A. Sherin</i>	
Spectra of Eccentricity Matrices of Product of Graphs	38
<i>S. Balamoorthy and T. Kavaskar</i>	
Almost Empty Monochromatic Polygons in Planar Point Sets	50
<i>Bhaswar B. Bhattacharya, Sandip Das, Sk. Samim Islam, and Saumya Sen</i>	
On the Parameterized Complexity of Odd Coloring	60
<i>Sriram Bhyravarapu, Swati Kumari, and I. Vinod Reddy</i>	
On Full-Separating Sets in Graphs	73
<i>Dipayan Chakraborty and Annegret K. Wagler</i>	
Polynomial Time Algorithms for Hop Domination	85
<i>D. Karthika, R. Muthucumaraswamy, Sriram Bhyravarapu, and Pritesh Kumar</i>	
Charging Station Placement for Limited Energy Robots	97
<i>Arun Kumar Das</i>	
Multipacking in the Euclidean Metric Space	109
<i>Arun Kumar Das, Sandip Das, Sk Samim Islam, Ritam Manna Mitra, and Bodhayan Roy</i>	
Partial Domination in Some Geometric Intersection Graphs	121
<i>Madhura Dutta, Anil Maheshwari, and Subhas C. Nandy</i>	

Generalized Lettericity of Graphs	134
<i>Zhidan Feng, Henning Fernau, Kevin Mann, Indhumathi Raman, and Silas Cato Sacher</i>	
Polynomial-Time Algorithms for PATH COVER on Trees and Graphs of Bounded Treewidth	147
<i>Florent Foucaud, Atrayee Majumder, Tobias Mömke, and Aida Roshany-Tabrizi</i>	
Fast FPT Algorithms for Grundy Number on Dense Graphs	160
<i>Sina Ghasemi Nezhad, Maryam Moghaddas, and Fahad Panolan</i>	
On a Tight Bound for the Maximum Number of Vertices that Belong to Every Metric Basis	173
<i>Anni Hakanen, Ville Junnila, Tero Laihonen, Havu Miikonen, and Ismael G. Yero</i>	
Algorithms and Hardness Results for the (3, 1)-Cover Problem	185
<i>Amirali Madani, Anil Maheshwari, Babak Miraftab, and Bodhayan Roy</i>	
Extension Perfect Roman Domination	197
<i>Kevin Mann and Henning Fernau</i>	
A Sub-quadratic Algorithm for the Minsum One Sink Location Problem on Balanced Binary Tree Networks	210
<i>Jannatul Maowa and Robert Benkoczi</i>	
Two Step Graph Protection Game	222
<i>Aakash Ghosh and Saraswati Girish Nanoti</i>	
Bipartite Domination in Graphs: Complexity and Algorithms	234
<i>Bhawani Sankar Panda, Subhasmita Joshi, and Dalu Jacob</i>	
Packing Sets of Paths, Stars and Triangles: Tractability and Approximability ...	247
<i>Santiago Valdés Ravelo and Flávio K. Miyazawa</i>	
Structural Parameterization of Minus Domination	258
<i>Sangam Balchandar Reddy and Anjeneya Swami Kare</i>	
An Algebraic Characterization of Strong Graphs	270
<i>Pablo Romero</i>	
There are Finitely Many Uniformly Most Reliable Graphs of Corank 5	280
<i>Pablo Romero</i>	

Helly Number, Radon Number and Rank in Δ -Convexity on Graphs	292
<i>Bijo S. Anand, Arun Anil, Manoj Changat, Revathy S. Nair, and Prasanth G. Narasimha-Shenoi</i>	
Forbidden Induced Subgraphs in Iterative Higher Order Line Graphs	307
<i>Aryan Sanghi, Devsi Bantva, and Sudebkumar Prasant Pal</i>	
Total Domination and Open Packing in Two Subclasses of Triangle-Free Graphs	319
<i>M. A. Shalu and V. K. Kirubakaran</i>	
The MASEMPR Problem and Its Applications in Logistics	331
<i>A. Subramani, K. Subramani, Piotr Wojciechowski, and Sangram K. Jena</i>	
Broadcast Graph is NP-Complete	343
<i>Jinghan Xu and Zhiyuan Li</i>	
Maximizing the Maximum Degree in Ordered Nearest Neighbor Graphs	358
<i>Péter Ágoston, Adrian Dumitrescu, Arsenii Sagdeev, Karamjeet Singh, and Ji Zeng</i>	
Author Index	369



Addressing Bias in Algorithmic Solutions: Exploring Vertex Cover and Feedback Vertex Set

Sheikh Shakil Akhtar¹(✉) , Jayakrishnan Madathil² , Pranabendu Misra¹ ,
and Geevarghese Philip¹

¹ Chennai Mathematical Institute, Chennai, India
shakil@cmi.ac.in

² University of Glasgow, Glasgow, Scotland

Abstract. A typical goal of research in combinatorial optimization is to come up with fast algorithms that find optimal solutions to a computational problem. The process that takes a real-world problem and extracts a clean mathematical abstraction of it often throws out a lot of “side information” which is deemed irrelevant. However, the discarded information could be of real significance to the end-user of the algorithm’s output. All solutions of the same cost are not necessarily of equal impact in the real-world; some solutions may be much more desirable than others, even at the expense of additional increase in cost. If the impact, positive or negative, is mostly felt by some specific (minority) subgroups of the population, the population at large will be largely unaware of it.

In this work we ask the question of finding solutions to combinatorial optimization problems that are “unbiased” with respect to a collection of specified subgroups of the total population. We consider a simple model of bias, and study it via two basic optimization problems on graphs: VERTEX COVER and FEEDBACK VERTEX SET, which are both NP-hard. Here, the input is a graph and the solution is a subset of the vertex set. The vertices represent members of a population, and each vertex has been assigned a subset of colors, where each color indicates membership of a specific subgroup. The goal is to find a small-sized solution to the optimization problem in which no color appears more than a specified—per-color—number of times. The colors can be used to model various relevant—economic, political, demographic, or other—classes to which the entities belong, and the variants that we study can then be used to look for small solutions which enforce per-class upper bounds on the number of removed entities. These upper-bounds enforce the constraint that no subclass of the population is over-represented in the solution.

We show the new variants of VERTEX COVER and FEEDBACK VERTEX SET, obtained by adding these additional constraints, are *Fixed-Parameter Tractable*, when parameterized by various combinations of the solution size, the number of colors, and the treewidth of the graph. Our results shows that it is possible to devise fast algorithms to solve these problem in many practical settings.

J. Madathil—This work was done while the author was at the Chennai Mathematical Institute.

Keywords: Parameterized Algorithms · Vertex Cover · Feedback Vertex Set · Fairness · Bias

1 Introduction

Consider a hospital that treats a large number of patients every day. The patients require timely access to a number of diagnostic and medical procedures for making a successful recovery. Since the manpower and equipment at a hospital are shared and in high-demand, scheduling these procedures is a non-trivial task. A simple way to schedule the patients could be as follows: construct a graph G where the vertices $V(G)$ represent the patients, and we have an edge between patient u and patient v if they have a common medical procedure; i.e. both can't be scheduled simultaneously. The objective is to maximize the number of patients that can be scheduled now, or equivalently to minimize the number of patients who will have to wait for later. The second formulation of the problem can be recognized as the classic VERTEX COVER problem, for which efficient Parameterized algorithms [5] and Approximation algorithms [8,9] are well-known.

The known algorithms for VERTEX COVER however aren't designed for being *unbiased* and *impartial*; i.e. it can't be ensured that the patients who are rescheduled are not disproportionately from one subgroup, e.g. economically weaker. Here the subgroups may be social, economic, medical or as per some other relevant criteria. If not addressed, such (unintended) bias can have severe and lasting impact. Moreover, if the adverse effects are limited to some small subgroups, then the majority population will be quite unaware that there is a problem at all. For example, if the local hospital is resource constrained and the delayed patients are largely from some minority social groups, then to the general population it will appear that the hospital is well-functioning and there is no need for any additional funding or resources! This is clearly a serious problem.

It is self-evident that (combinatorial) algorithms themselves are unbiased. The bias in the solutions computed by them appears in other ways. For example, the order in which the input data is presented could influence the output. Sometimes the bias may be inherent in the data itself; e.g. the prevalence and severity of certain medical conditions varies by gender, ethnicity, income, age etc. This means that certain sophisticated medical diagnostics and procedures may correlate with certain subgroups of patients. The algorithm may decide to schedule these patients for later in the pursuit of an optimal solution.

One of the main reasons for this un-indentified bias is that when we reduce a real-world problem to an abstract mathematical problem, we try to simplify it as much as possible and discard a lot of associated information. Furthermore, certain heuristics employed to speed up the algorithm could have unexpected consequences. For example, most algorithms for VERTEX COVER will pick high-degree vertices into the solution. These vertices corresponds to patients who require a number of medical procedures and are likely to be in a critical condition, and must not be delayed!

Other examples arise in the applications of resource allocation using algorithms. Consider the process of setting up sample-collection centers across a city medical-tests during the COVID-19 pandemic [6]. Due to cost reasons, only a limited number of such centers can be opened and the goal will be to place them so as to minimize the maximum travel distance of a citizen to the nearest center. Similar challenges arise in many other public health settings. The problem of finding the optimal location for these centers can be modeled as a *Clustering* problem which is very well-studied (see, e.g. [4,7]). However, consider the subset of people who are all quite far way from the centers, as compared to the rest of the population, e.g. in a remote settlement. They are less likely to travel to the centers compared to the general population. If they happen to be disproportionately from a certain subgroup, which could be case for a historically disadvantaged ethnic subgroup, then the data collected from the tests will fail to properly account for this subgroup. Then, any policy built upon this data will be flawed [1].

The current algorithms for these computational problems are oblivious to the biases in the output solution. If unchecked, such bias can have significant impact on certain groups. We emphasize once more that this is not because the algorithms are themselves biased, but because of the attributes of the problem instance itself, the choice of mathematical abstraction, the presentation of the input data etc. We also remark that simple strategies such as randomizing the order of input data may or may not be helpful; it is unclear without a formal (or experimental) analysis. Moreover, this will not ameliorate every form of bias (such as the consequence of the high-degree rule for VERTEX COVER mentioned earlier). It is therefore essential to consider more concrete ways of addressing the bias in algorithmic solutions.

Motivated by this, we explore the class of computational problems derived from classic NP-complete optimization problems obtained by introducing additional constraints for being unbiased with respect to population subgroups. This new class of problems are natural combinatorial questions that are interesting in their own right. For simplicity, we focus on graph problems where the vertices represent members of the general population, and vertex subsets naturally represent various subgroups of the population. We may also generalize this definition to broader classes of problems such as those modeled by *Constraint Satisfaction Problems(CSPs)* or *Linear Programming*, but at the expense of clarity. Therefore we state them for optimization problems on graphs, which capture a broad class of classic optimization problems. Let Π be a graph problem such that the solution to Π is a subset of vertices of the input graph G . Then, we have the following *unbiased* variant of Π :

UNBIASED Π

Input: (G, c) where G is a graph whose vertices are labeled by *colors* from a set $\{1, 2, \dots, t\}$ via the function $c : V(G) \rightarrow 2^{\{1, 2, \dots, t\}} \setminus \{\emptyset\}$.

Task: Find a solution S of minimum size such that for every $i \in \{1, 2, \dots, t\}$, $\frac{|\{v \in S \mid i \in c(v)\}|}{|S|} = \frac{|\{v \in V(G) \mid i \in c(v)\}|}{|V(G)|}$

Here the colors $\{1, 2, \dots, t\}$ can represent any relevant subgroup of the total population. A single vertex can be a member of more than one subgroup, which is captured by the *coloring function* c ; and note that every vertex is part of at least one subgroup. Our objective is to compute a solution S where the fraction of vertices for each color i is the same as the fraction of the color in the total population. We call such a solution *unbiased*. Naturally, our objective is to compute an unbiased solution of the smallest size.

We remark that the *price of fairness* in a given instance (G, c) of UNBIASED Π could be very high, due to the strict unbiasedness constraints. Indeed, it is possible to construct artificial examples where the difference between the optimal solution to G and the optimal *unbiased* solution to (G, c) is unbounded. Therefore, we define the following more general variant. Here $\alpha \leq 1 \leq \beta$ are real numbers, that set the desired level of *unbiasedness*.

(α, β) -UNBIASED Π

Input: (G, c) where G is a graph whose vertices are labeled by *colors* from a set $\{1, 2, \dots, t\}$ via the function $c : V(G) \rightarrow 2^{\{1, 2, \dots, t\}} \setminus \{\emptyset\}$.

Task: Find a solution S of minimum size such that for every $i \in \{1, 2, \dots, t\}$, we have $\alpha \cdot \frac{|\{v \in V(G) \mid i \in c(v)\}|}{|V(G)|} \leq \frac{|\{v \in S \mid i \in c(v)\}|}{|S|} \leq \beta \cdot \frac{|\{v \in V(G) \mid i \in c(v)\}|}{|V(G)|}$

Observe that for $\alpha = 0, \beta = \infty$ we obtain Π , whereas for $\alpha = \beta = 1$, we obtain UNBIASED Π . We can obtain the desired level of trade-off between the level of bias and the price of fairness by setting α and β .

In this paper we consider these problems in the framework of Parameterized Complexity [5]. Here, the problem instances are parameterized, i.e. they consist of a pair (I, k) , where I is an instance of a problem Π and k is a number called the parameter representing some structural property of I that is typically bounded in practice. A typical parameter is the optimum solution size, and obtaining an FPT algorithm for this parameter means that we can compute an optimal solution efficiently when the solution size is bounded, which is the case in many practical settings, even if the input instance itself is very large. Other examples parameters are treewidth of a graph, the genus of the graph etc. The objective is to obtain *FPT algorithms* that find an optimal solution in time $f(k) \cdot n^{O(1)}$, where f is a function of k alone. FPT algorithms are employed to compute optimal solutions to NP-hard problems in nearly-polynomial time. Another objective is to design *kernelization algorithms*, that given an instance (I, k) , run in polynomial time, and produce an equivalent instance (I', k') such that $|I'| + k' = k^{O(1)}$. (I', k') is called a *polynomial kernel*. Kernelization captures the notion of efficient data-reduction algorithms. We refer to [5] for the details.

We define a slightly different variant of (α, β) -UnBiased Π which naturally fits into this paradigm. Here for each color-class $i \in \{1, \dots, t\}$, we are given a number k_i , and let $\mathbb{T} = (k_i \mid 1 \leq i \leq t)$ denote this tuple of integers. Then we define the following problem.

(α, β) -T-FAIR Π

Input: (G, c) where G is a graph whose vertices are labeled by *colors* from a set $\{1, 2, \dots, t\}$ via the function $c : V(G) \rightarrow 2^{\{1, 2, \dots, t\}} \setminus \{\emptyset\}$, and a t -tuple of integers $\mathbb{T} = (k_1, k_2, \dots, k_t)$.

Task: Decide whether there is a solution S such that $|S| \leq k$ and $\alpha \cdot k_i \leq |\{v \in S \mid i \in c(v)\}| \leq \beta \cdot k_i$ for each $1 \leq i \leq t$. Here $k = \sum_{i=1}^t k_i$.

Observe that, if k were the size of the solution S , then we can set $k_i = k \cdot \frac{|\{v \in V(G) \mid i \in c(v)\}|}{|V(G)|}$ to obtain (α, β) -UNBIASED Π . Also, when $\alpha = \beta = 1$, we call it T-FAIR Π , which corresponds to UNBIASED Π .

For simplicity, we say (G, c) is t -colored if vertices of G are labeled by *colors* from the set $\{1, 2, \dots, t\}$ via the given function $c : V(G) \rightarrow 2^{\{1, 2, \dots, t\}} \setminus \{\emptyset\}$. We say that a solution S is (α, β) -T-fair if it is a solution to Π , $|S| \leq k$ and $\alpha \cdot k_i \leq |\{v \in S \mid i \in c(v)\}| \leq \beta \cdot k_i$ for each $1 \leq i \leq t$. In the strict setting where $\alpha = \beta = 1$, we call it T-fair solution.

In this paper, we initiate the study of these class of problems via two classic graph problems, namely VERTEX COVER and FEEDBACK VERTEX SET, which are defined as follows. Recall that a vertex-subset S is a *vertex cover* of a graph G if $G - S$ has no edges. S is a (α, β) -T-fair vertex cover if it satisfies the constraints imposed by (G, c) and \mathbb{T} . A vertex-subset S is a *feedback vertex set* of a graph G , if $G - S$ is acyclic. We can similarly define a (α, β) -T-fair feedback vertex set.

 (α, β) -T-FAIR VERTEX COVER (T-FAIR VC)

Input: A t -coloured graph (G, c) and a t -tuple of integers $\mathbb{T} = (k_1, k_2, \dots, k_t)$.

Task: Decide whether G has a (α, β) -T-fair vertex cover.

 (α, β) -T-FAIR FEEDBACK VERTEX SET(T-FAIR FVS)

Input: A t -coloured graph (G, c) and a t -tuple of integers $\mathbb{T} = (k_1, k_2, \dots, k_t)$.

Task: Decide whether G has a (α, β) -T-fair feedback vertex set.

We first consider T-FAIR VC and T-FAIR FVS and obtain FPT-algorithms and polynomial kernels for these problems when parameterized by solution-size, number of colors, and the treewidth of the underlying graph. Note that the treewidth of a graph is never larger than the size of a minimum vertex cover, and at most one larger than the size of a minimum feedback vertex set [5]. We then show that these algorithms can be extended to the more general (α, β) -T-FAIR VC and (α, β) -T-FAIR FVS. Formally,

Theorem 1. T-FAIR VERTEX COVER can be solved in $\mathcal{O}(n^2 t (\prod_{i=1}^t k_i^2) \cdot 1.4656^k)$ time where n is the number of vertices in the input graph G , t is the number of colours used by the colouring function c , (k_1, k_2, \dots, k_t) is the colour budget specified in the input, and $k = (\sum_{i=1}^t k_i)$.

Theorem 2. *There is a polynomial time algorithm which outputs a polynomial kernel for T-FAIR VERTEX COVER. If G is the input graph, t is the number of colours used by the colouring function c , and (k_1, k_2, \dots, k_t) is the colour budget specified in the input, then the size of this kernel is quadratic in $k = (\sum_{i=1}^t k_i)$.*

Theorem 3. *Given a graph G , a nice tree decomposition $(\mathcal{T}, (B_x)_{x \in V(\mathcal{T})})$ of G of width t_w and with l nodes, a colouring function c , where t is the number of colours used by the colouring function c , and (k_1, k_2, \dots, k_t) is the colour budget specified in the input, there is an algorithm which solves T-FAIR VERTEX COVER in time $O(t l t_w^2 2^{t_w+1} \prod_{i=1}^t k_i^2)$.*

Theorem 4. *Given a graph G with n vertices, a nice tree decomposition $(\mathcal{T}, (B_x)_{x \in V(\mathcal{T})})$ of G of width t_w and with l nodes, a colouring function c , where t is the number of colours used by the colouring function c , and (k_1, k_2, \dots, k_t) is the colour budget specified in the input, there is an algorithm which solves T-FAIR FEEDBACK VERTEX SET in time $n^{O(1)} 2^{O(t_w)}$.*

Theorem 5. *T-FAIR FEEDBACK VERTEX SET can be solved in $n^{O(1)} 2^{O(k)}$ time where n is the number of vertices in the input graph G , t is the number of colours used by the colouring function c , (k_1, k_2, \dots, k_t) is the colour budget specified in the input, and $k = (\sum_{i=1}^t k_i)$.*

From the above we obtain the following results.

Theorem 6. *For every fixed α, β , the (α, β) -T-FAIR VC problem can be solved in time $\mathcal{O}(n^2 t (\prod_{i=1}^t k_i^2 ((\beta - \alpha) k_i + 1)) \cdot 1.4656^k)$.*

Theorem 7. *For every fixed α, β , the (α, β) -T-FAIR FVS problem can be solved in time $\prod_{i=1}^t ((\beta - \alpha) k_i + 1) \cdot n^{O(1)} 2^{O(k)}$.*

More generally, for any classic computation problem Π such that T-FAIR Π has an algorithm with runtime $g(k_1, k_2, \dots, k_t, n)$, we have the following.

Theorem 8. *For every fixed α, β , the (α, β) -T-FAIR Π problem can be solved in time $\prod_{i \in [t]} ((\beta - \alpha) k_i + 1) \cdot g(\beta k_1, \beta k_2, \dots, \beta k_t, n)$, where $g(k_1, k_2, \dots, k_t, n)$ is the runtime of an algorithm for T-FAIR Π .*

Proofs and other details, including a section on related works, omitted from this extended abstract can be found in the full version included as an appendix.

2 Preliminaries

We use standard notations from graph theory. For an integer $t \in \mathbb{N}^+$ and a graph G , a t -colouring function of G is any function $c : V \rightarrow (2^{[t]} \setminus \{\emptyset\})$ that assigns at least one “colour” from the set $[t]$ to each vertex of G . A t -coloured graph is a pair (G, c) where c is a t -colouring function of G . For a t -coloured graph (G, c) , vertex $v \in V(G)$, and $i \in [t]$, we use $N_i(v)$ to denote the colour- i neighbourhood $\{w \in N(v) \mid i \in c(w)\}$ of the set of all neighbours of v which have been assigned

the colour i by c . For a subset $S \subseteq V(G)$ of vertices we use $c_i(S)$ to denote the number $|\{v \in S | i \in c(v)\}|$ of vertices in S which have been assigned the colour i by c . For a t -tuple of integers $\mathbb{T} = (k_1, k_2, \dots, k_t)$ we say that $S \subseteq V(G)$ is \mathbb{T} -fair if for each $i \in [t]$ it is the case that $c_i(S) = k_i$ holds: the number of vertices in S which have the colour i , is exactly k_i . For a fixed finite alphabet Σ , a *parameterised problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$. For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, k is called the *parameter*. A parameterised problem $L \subseteq \Sigma^* \times \mathbb{N}$ is called *fixed parameter tractable* if there exists an algorithm, say \mathcal{A} , a computable function $f : \mathbb{N} \rightarrow \mathbb{N}$, and a constant c such that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, the algorithm correctly decides whether $(x, k) \in L$ in time bounded by $|(x, k)|^c f(k)$. The complexity class of all fixed-parameter tractable problems is called FPT.

We derive algorithms based on (nice) tree decompositions [3, 5]. If a graph G has a tree decomposition $(\mathcal{T}, (B_x)_{x \in V(\mathcal{T})})$ of width at most t_w , then in time $O(t_w^2 \times \max(|V(\mathcal{T})|, |V(G)|))$ we can obtain a nice tree decomposition of G of width at most t_w that has at most $O(t_w |V(G)|)$ nodes. Also if G is a path or a cycle then one can get a nice tree decomposition of G in time $O(|V(G)|)$ with $O(|V(G)|)$ nodes. Refer to [5] and [3] for more information. For the remainder of the paper, we will assume that a tree decomposition is a nice tree decomposition unless stated otherwise. For a graph G with a nice tree decomposition $(\mathcal{T}, (B_x)_{x \in V(\mathcal{T})})$, we define $G_x = (V_x, E_x)$, the *subgraph of G rooted at x* as follows:

- $V_x = B_x \cup_{y \text{ is a descendant of } x \in V(\mathcal{T})} B_y$
- $E_x = \{\text{All edges introduced at any node in the subtree of } \mathcal{T} \text{ rooted at } x\}$

3 T-FAIR VERTEX COVER

In this section we take up the T-FAIR VERTEX COVER problem. T-FAIR VERTEX COVER is NP-hard by a simple reduction from VERTEX COVER, which we will now sketch. Let (G, k) be an instance of VERTEX COVER. If G has fewer than k vertices then (G, k) is trivially a YES-instance of VERTEX COVER, and we construct and return a trivial YES-instance of T-FAIR VERTEX COVER. So let us assume, without loss of generality, that G has at least k vertices. This implies that G has a vertex cover of size at most k if and only if G has a vertex cover of size exactly k .

We construct a 1-coloured graph (H, c) by setting (i) $H = G$ and (ii) c to be the function that assigns the colour set $\{1\}$ to every vertex of H . Further, we set $\mathbb{T} = (k)$ to be a 1-tuple with its sole element being k . $((H, c), \mathbb{T})$ is the reduced instance of T-FAIR VERTEX COVER. It is easy to see that S is a vertex cover of G of size exactly k if and only if S is a \mathbb{T} -fair vertex cover of H .

We study different parameterizations of T-FAIR VERTEX COVER in this section. In Subsect. 3.1 we set the parameter to be the “total colour budget” $(\sum_{i=1}^t k_i)$. This is a natural generalization of the “standard” parameter k of VERTEX COVER, as suggested by the above reduction. We show—subsection

3.1—that \mathbb{T} -FAIR VERTEX COVER is fixed-parameter tractable with $(\sum_{i=1}^t k_i)$ as the parameter. We take up the kernelization question in Subsect. 3.1, and show that when the number of colours t is a constant then \mathbb{T} -FAIR VERTEX COVER has a kernel of size quadratic in $(\sum_{i=1}^t k_i)$. In Subsect. 3.3 we show that \mathbb{T} -FAIR VERTEX COVER has a single-exponential FPT algorithm when parameterized by the *treewidth* of the input graph G .

3.1 An FPT Algorithm Parameterized by $k = (\sum_{i=1}^t k_i)$

The “total colour budget” $k = (\sum_{i=1}^t k_i)$ is a natural parameter for \mathbb{T} -FAIR VERTEX COVER, since it is an analogue of the vertex cover size of the “plain” VERTEX COVER problem. We show that this analogy carries over, to a good extent, to the parameterized tractability of \mathbb{T} -FAIR VERTEX COVER: we prove

Theorem 1. *\mathbb{T} -FAIR VERTEX COVER can be solved in $\mathcal{O}(n^2 t (\prod_{i=1}^t k_i^2) \cdot 1.4656^k)$ time where n is the number of vertices in the input graph G , t is the number of colours used by the colouring function c , (k_1, k_2, \dots, k_t) is the colour budget specified in the input, and $k = (\sum_{i=1}^t k_i)$.*

Our algorithm is an adaptation of a standard FPT algorithm for VERTEX COVER that branches on vertices of degree at least 3 [2]. The base case—graphs with maximum degree at most 2—is easy to solve in polynomial time for “plain” VERTEX COVER, since these graphs are collections of paths and cycles. But with the added constraint of the colour budget, it is not immediately clear that \mathbb{T} -FAIR VERTEX COVER can be solved in polynomial time on such graphs. We get around this by appealing to our algorithm for \mathbb{T} -FAIR VERTEX COVER on graphs of *bounded treewidth*; see Subsect. 3.3.

Our algorithm uses two No-instance checks, a reduction rule, a branching rule that applies when the graph has at least one vertex of degree 3, and a subroutine for handling the base case when every vertex has degree at most 2. Recall that the input consists of a graph G on n vertices, a colouring function $c : V(G) \rightarrow (2^{[t]} \setminus \{\emptyset\})$, and a t -tuple of integers $\mathbb{T} = (k_1, k_2, \dots, k_t)$. Recall also that for any $S \subseteq V(G)$, the expression $c_i(S)$ denotes the number of vertices in the set S which have been assigned the colour i by the colouring function c .

We are just stating the branching rule here and leaving the details in appendix.

The Branching Rule. This rule applies when G contains at least one vertex of degree at least 3.

Branching Rule. Let v be a vertex of degree at least 3 in G . Let H' be the graph obtained by deleting the vertex v (and its incident edges) from G , let c' be the function obtained by restricting c to $V(H') = (V(G) \setminus \{v\})$, and let $\mathbb{T}' = \{k'_1, k'_2, \dots, k'_t\}$ where for each $i \in [t]$ we have:

- $k'_i = k_i - 1$ if $i \in c(v)$; that is, if colour i is among the set of colours that the t -colouring function c assigns to vertex v , and,
- $k'_i = k_i$ otherwise.

Let H'' be the graph obtained by deleting the open neighbourhood $N(v)$ (and their incident edges) from G , let c'' be the function obtained by restricting c to $V(H'') = (V(G) \setminus N(v))$, and let $\mathbb{T}'' = \{k_1'', k_2'', \dots, k_t''\}$ where for each $i \in [t]$ we have $k_i'' = k_i - c_i(N(v))$.

The branching rule recursively solves the two instances (H', c', \mathbb{T}') and (H'', c'', \mathbb{T}'') . If at least one of these recursive calls returns YES, then the rule returns YES; otherwise it returns NO.

The Subroutine for the Base Case. This subroutine is applied to solve an instance of the form (G, c, \mathbb{T}) , where graph G has no vertex of degree at least 3. Note that such a graph is a disjoint union of paths and cycles. Unlike in the case of VERTEX COVER, we cannot (i) directly delete isolated vertices (paths with no edges) or (ii) greedily solve the problem on paths and cycles. This is because the solution must also respect the colour constraints. However, a graph which is just a disjoint union of paths and cycles has treewidth at most 2. And a nice tree decomposition of G of width 2 can be found in polynomial time. We compute a nice tree decomposition of G and apply the algorithm in Theorem 9. This completes the description of the algorithm.

3.2 Polynomial Kernel for \mathbb{T} -FAIR VERTEX COVER with a Constant Number of Colours

In this section we design a kernelization algorithm for the problem. To design the kernel, given a graph, G , the colouring function, $c : V(G) \rightarrow (\mathcal{P}([t]) \setminus \emptyset)$, a t -tuple of integers, $\mathbb{T} = (k_1, \dots, k_t)$, we apply the following rules.

The Global No-instance Check. We apply this rule once, right at the beginning.

Check 0. If $c_i(V(G)) < k_i$ holds for at least one $i \in [t]$ then return NO.

Setting Aside Isolated Vertices. If the above rule does not return NO, then we know that, for each $i \in [t]$, $c_i(V(G)) \geq k_i$. Let $k_{max} = \max\{k_i | i \in [t]\}$. We define a subset I^* of $V(G)$ as follows. For every $X \subseteq [t]$, if $X \neq \emptyset$, we denote the set $V_X := \{v \in V(G) | c(v) = X \wedge \deg(v) = 0\}$. If $|V_X| > k_{max}$, then keep any k_{max} of them in V_X and remove the rest. We define, $I^* := \bigcup_{X \subseteq [t] \wedge X \neq \emptyset} V_X$.

We then apply the following rules exhaustively (in order as they appear). If at some point, any of them return NO, we terminate the algorithm and return NO for the original input.

The No-instance Check

Check 1. If there is at least one pair $(i, j) \in [t] \times [t]$ such that (i) $k_i = k_j = 0$ and (ii) there is an edge $(u, v) \in E(G)$ for which both $i \in c(u)$ and $j \in c(v)$ hold, then return NO.

The Isolated Vertex Rule

Isolated Vertex Rule. If $v \in V(G)$, such that $\deg(v) = 0$ and $v \notin I^*$, then we return the instance, $(G - v, c', \mathbb{T})$, where c' is the restriction of c on $V(G) \setminus \{v\}$.

The Large Neighbourhood Rule

Large Neighbourhood Rule. If for some $v \in V(G)$ such that for some $i \in [t]$, $|N_i(v)| > k_i$, then we return $(G - v, c', (k'_1, \dots, k'_t))$, where c' is the restriction of c on $V(G) \setminus \{v\}$ and for each $i \in [t]$, $k'_i = k_i - c_i(\{v\})$.

The Final Instance Size.

Return a Kernel. Let (G, c, \mathbb{T}) be an input instance, such that the above rules are no more applicable. Then, if $|V(G)| > (\sum_{i=1}^t k_i)^2 + \sum_{i=1}^t k_i \times (1 + 2^t)$ or $|E(G)| > (\sum_{i=1}^t k_i)^2$, we return No, else we return this instance as our kernel.

We thus have the following result.

Theorem 2. *There is a polynomial time algorithm which outputs a polynomial kernel for \mathbb{T} -FAIR VERTEX COVER. If G is the input graph, t is the number of colours used by the colouring function c , and (k_1, k_2, \dots, k_t) is the colour budget specified in the input, then the size of this kernel is quadratic in $k = (\sum_{i=1}^t k_i)$.*

3.3 Parameterization by Treewidth

In the section, we design a fixed-parameter tractable algorithm for the \mathbb{T} -FAIR VERTEX COVER problem, parameterized by the treewidth of the input graph. We assume that along with the graph G , we are also given a tree decomposition $(\mathcal{T}, (B_x)_{x \in V(\mathcal{T})})$ of width t_w . Specifically, we prove the following theorem

Theorem 9. *There is an algorithm that, given an n -vertex graph G , a colouring function $c : V(G) \rightarrow 2^{[t]} \setminus \{\emptyset\}$, a t -tuple of non-negative integers $(k_i)_{i=1}^t$, and a tree decomposition $(\mathcal{T}, (B_x)_{x \in V(\mathcal{T})})$ of G of width t_w , runs in time $O(t t_w^2 2^{t_w+1} \prod_{i=1}^t k_i^2)$ and decides correctly if G has a $(k_i)_{i=1}^t$ -fair vertex cover.*

To prove Theorem 9, we design a dynamic programming algorithm over the tree decomposition.

Definition of the States of the DP. For a node x of \mathcal{T} , a subset S of B_x , a t -tuple, $(r_i)_{i=1}^t$, where for each $i \in [t]$, $0 \leq r_i \leq k_i$, we define $I_x[S, (r_i)_{i=1}^t]$ to be as follows: $I_x[S, (r_i)_{i=1}^t] = 1$ if G_x has an $(r_i)_{i=1}^t$ -fair vertex cover that intersects B_x at exactly S ; and $I_x[S, (r_i)_{i=1}^t] = 0$ otherwise. For each node x , observe that the number of states $I_x[\cdot, \cdot]$ is at most $2^{|B_x|} \cdot \prod_{i=1}^t (k_i + 1)$.

Our algorithm will fill the DP tables for all choices of x, S and $(r_i)_{i=1}^t$ using the rules above. And once we are done with that, we check $I_x[\emptyset, (k_i)_{i=1}^t]$, where x is the root node of \mathcal{T} . If it has been assigned the value 1, then we return YES else we return No. The details of the computation along with the proofs are provided in the appendix.

4 FEEDBACK VERTEX SET

In this section we study the \mathbb{T} -FAIR FEEDBACK VERTEX SET problem. Just like \mathbb{T} -FAIR VERTEX COVER, \mathbb{T} -FAIR FVS is also NP-hard. And we can show the NP-hardness by a simple reduction from FEEDBACK VERTEX SET. Let (G, k) be an instance of FEEDBACK VERTEX SET. We assume without loss of generality that G has at least k vertices, for otherwise (G, k) is a YES-instance of FEEDBACK VERTEX SET, and in this case, we can return a trivial YES-instance of \mathbb{T} -FAIR FVS. Then G has an fvs of size at most k if and only G has an fvs of size exactly k .

We now construct a 1-coloured graph (H, c) by setting (i) $H = G$ and (ii) c to be the function that assigns the colour set $\{1\}$ to every vertex of H . Further, we set $\mathbb{T} = (k)$. It is easy to see that S is an fvs of G of size exactly k if and only if S is a \mathbb{T} -fair fvs of H .

As we did for \mathbb{T} -FAIR VERTEX COVER, we study two different parameterizations of \mathbb{T} -FAIR FVS—with respect to the treewidth of the input graph and with respect to the total colour budget. With respect to both the parameterizations, we design single-exponential FPT algorithms.

Theorem 4. *Given a graph G with n vertices, a nice tree decomposition $(\mathcal{T}, (B_x)_{x \in V(\mathcal{T})})$ of G of width t_w and with l nodes, a colouring function c , where t is the number of colours used by the colouring function c , and (k_1, k_2, \dots, k_t) is the colour budget specified in the input, there is an algorithm which solves \mathbb{T} -FAIR FEEDBACK VERTEX SET in time $n^{O(1)}2^{O(t_w)}$.*

Theorem 5. *\mathbb{T} -FAIR FEEDBACK VERTEX SET can be solved in $n^{O(1)}2^{O(k)}$ time where n is the number of vertices in the input graph G , t is the number of colours used by the colouring function c , (k_1, k_2, \dots, k_t) is the colour budget specified in the input, and $k = (\sum_{i=1}^t k_i)$.*

The details of the computation along with the proofs of the theorems are provided in the appendix.

5 Conclusion

In this paper, we defined a notion of *unbiased* solutions to combinatorial problems. We introduced a definition to formally derive the unbiased variant of a classic combinatorial problem. We then explored the variants of VERTEX COVER and FEEDBACK VERTEX SET in the paradigm of Parameterized Complexity, and give efficient algorithms for them. The natural next step is to explore the parameterized complexity of the unbiased variant of other well-studied problems, such

as PLANAR DOMINATING SET, ODD CYCLE TRANSVERSAL, MATCHING UNDER PREFERENCES and many others. Another direction worth exploring is the notion of an unbiased approximation solution, which requires some further research.

References

1. Abbasi-Sureshjani, S., Raumanns, R., Michels, B.E.J., Schouten, G., Cheplygina, V.: risk of training diagnostic algorithms on data with demographic bias. In: Cardoso, J., et al. (eds.) IMIMIC/MIL3ID/LABELS -2020. LNCS, vol. 12446, pp. 183–192. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-61166-8_20
2. Balasubramanian, R., Fellows, M.R., Raman, V.: An improved fixed-parameter algorithm for vertex cover. *Inf. Process. Lett.* **65**(3), 163–168 (1998)
3. 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)
4. Charu, C.A., Chandan, K.R.: Data Clustering: Algorithms and Applications. Chapman and Hall/CRC, Boston (2013)
5. Cygan, M., et al.: Parameterized Algorithms. Springer, Cham (2015). <https://doi.org/10.1007/978-3-319-21275-3>
6. Munguía-López, A.D.C., Ponce-Ortega, J.M.: Fair allocation of potential COVID-19 vaccines using an optimization-based strategy. *Process Integr. Optim. Sustain.* **5**(1), 3–12 (2021)
7. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Pearson Education India (2016)
8. Vazirani, V.V.: Approximation Algorithms. Springer, Heidelberg (2001). <http://www.springer.com/computer/theoretical+computer+science/book/978-3-540-65367-7>
9. Williamson, D.P., Shmoys, D.B.: The Design of Approximation Algorithms. Cambridge University Press, Cambridge (2011). http://www.cambridge.org/de/knowledge/isbn/item5759340/?site_locale=de_DE



Graceful Coloring is Computationally Hard

Cyriac Antony¹ , Jacob Antony² , D. Laavanya³ ,
and S. Devi Yamini³

¹ IIT Madras, Chennai, India

`ma23r004@mail.iitm.ac.in`

² Mahatma Gandhi University, Kottayam, India

`phdmath2401@cmscollege.ac.in`

³ Vellore Institute of Technology, Chennai, India

`laavanya.d2020@vitstudent.ac.in, deviyamini.s@vit.ac.in`

Abstract. Given a (proper) vertex coloring f of a graph G , where $f: V(G) \rightarrow \mathbb{N}$, the *difference edge labeling* induced by f is a function $h: E(G) \rightarrow \mathbb{N}$ defined as $h(uv) = |f(u) - f(v)|$ for every edge uv of G . A *graceful coloring* of G is a vertex coloring f of G such that the difference edge labeling h induced by f is a (proper) edge coloring of G . A graceful coloring with co-domain $\{1, 2, \dots, k\}$ is called a graceful k -coloring. The least integer k such that G admits a graceful k -coloring is called the *graceful chromatic number* of G , denoted by $\chi_g(G)$.

We prove that $\chi(G^2) \leq \chi_g(G) \leq a(\chi(G^2))$ for every graph G , where $a(n)$ denotes the n th term of the integer sequence A065825 in OEIS. We also prove that graceful coloring problem is NP-hard for planar bipartite graphs, regular graphs and 2-degenerate graphs. In particular, we show that for each $k \geq 5$, it is NP-complete to check whether a planar bipartite graph of maximum degree $k - 2$ is graceful k -colorable. The complexity of checking whether a planar graph is graceful 4-colorable remains open. We show that graceful 4-colorability of chordal graphs is polynomial-time testable.

1 Introduction

Many branches of mathematics started out as problems in recreational mathematics which are easy to understand, yet challenging to solve. The story of graph theory is no different. The innocuous problem of coloring maps using only four colors gave birth to a thriving area of graph theory named graph coloring. The notion of graph labeling is a generalization of graph coloring. Graph labeling is an area of immense theoretical interest and diverse practical applications, evident from Gallian's dynamic survey [8]. Similar to how attempts to prove the four color conjecture lead to the historical origin and popularity of the area of graph colorings, the study of graceful labeling and harmonious labeling lead to the boom of the area of graph labelings [8].

The definition of graceful labeling requires the notion of difference edge labeling induced by a vertex labeling. Given a vertex labeling f of a graph G , say $f: V(G) \rightarrow \mathbb{N}$, the *difference edge labeling induced by f* is a function

$h: E(G) \rightarrow \mathbb{N} \cup \{0\}$ defined as $h(uv) = |f(u) - f(v)|$ for every edge uv of G . A *graceful labeling* of a graph G on m edges is an injection $f: V(G) \rightarrow \{0, 1, \dots, m\}$ such that the difference edge labeling h induced by f is an injection from $E(G)$ to $\{1, 2, \dots, m\}$ [8].

The most popular problem on graceful labeling is settling the infamous Kötzig-Ringel-Rosa conjecture [15, 16], better known as the graceful tree conjecture, which states that all trees are graceful. The graceful tree conjecture is far from resolved till date. Hence, the practical approaches to the problem includes resolving the conjecture for subclasses of trees on one hand, and resolving weaker versions of the conjecture on the other hand. One way to produce notions weaker than graceful labeling is to impose restrictions locally on vertex neighborhoods rather than globally. This produces the notion of graceful coloring.

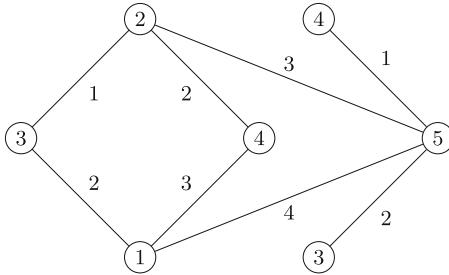


Fig. 1. A graceful 5-coloring of a graph.

A vertex labeling f of a graph G , say $f: V(G) \rightarrow \mathbb{N}$, is a *graceful coloring* of G if (i) f is an injection when limited to each vertex neighborhood, and (ii) the induced difference edge labeling h is an injection when limited to each vertex neighborhood; formally, the restrictions $f|_{N_G[v]}$ and $h|_{\partial_G(v)}$ are injections (here, $N_G[v] = \{v\} \cup \{u \in V(G): uv \in E(G)\}$ and $\partial_G(v) = \{e \in E(G): v \text{ incident on } e\}$). In other words, a graceful coloring of G is a (proper vertex) coloring f of G such that the difference edge labeling h induced by f is a (proper) edge coloring of G [1] (see Fig. 1 for an example).

The notion of graceful coloring was introduced by Chartrand (see [1, 3]), and first appeared in Bi et al. [1]. Graceful coloring is studied for various classes including trees [1, 3, 6, 13], regular graphs [1, 3, 6], complete bipartite graphs [1, 3], generalized Petersen graphs [10], and various graph products [9, 12, 17].

For $k \in \mathbb{N}$, a *graceful k -coloring* of G is a graceful coloring f of G with co-domain $\{1, 2, \dots, k\}$ (i.e., $f: V(G) \rightarrow \{1, 2, \dots, k\}$). The least integer k such that G admits a graceful k -coloring is called the *graceful chromatic number* of G , denoted by $\chi_g(G)$.

It is easy to observe that under a graceful coloring, no two neighbors of a vertex can get the same color. A (proper) coloring of a graph G with this property is called a *distance-two coloring* of G [7]. Hence, every graceful coloring is a distance-two coloring, but the converse is not true. The least number of colors required to produce a distance-two coloring of a graph G is called the *distance-two chromatic number* of G , and is equal to the chromatic number

of the square graph G^2 . We denote the distance-two chromatic number of G by $\chi(G^2)$. Obviously, $\chi(G^2) \leq \chi_g(G)$.

We relate the graceful chromatic number of complete graphs to integer sequences. Throughout this paper, $a(n)$ denotes the n th term of the integer sequence A065825 in OEIS [14]. It is known that $\chi_g(K_n) = a(n)$ [11]. We prove that $\chi(G^2) \leq \chi_g(G) \leq a(\chi(G^2))$ for every graph G .

Since the complexity of graceful labeling remains an open problem, computational hardness results on variations of graceful labeling, such as graceful coloring, are theoretically important. We prove that graceful coloring problem is NP-hard for planar bipartite graphs, regular graphs and 2-degenerate graphs. For $k \in \mathbb{N}$, a graph G is k -degenerate if every subgraph of G has a vertex of degree at most k . We show that (i) for each $k \geq 6$, it is NP-complete to check whether a planar bipartite 3-degenerate graph of maximum degree $k - 2$ is graceful k -colorable, (ii) it is NP-complete to check whether a 3-regular 3-connected planar bipartite graph is graceful 5-colorable, (iii) it is NP-complete to check whether a 2-degenerate graph is graceful 4-colorable, and (iv) it is polynomial-time solvable to check whether a chordal graph is graceful 4-colorable. The complexity of checking whether a planar graph is graceful 4-colorable remains open.

2 Results

2.1 Graceful Coloring and Integer Sequences

By definition, $a(n)$ is the least integer k for which $\{1, 2, \dots, k\}$ contains a subset S of cardinality n such that no three distinct elements i, j, k of S satisfy $|i - j| = |j - k|$. This imply the following.

Theorem 1 ([11]). $\chi_g(K_n) = a(n)$ for every positive integer n . \square

The next theorem shows that the parameters $\chi(G^2)$ and $\chi_g(G)$ bind each other.

Theorem 2. $\chi(G^2) \leq \chi_g(G) \leq a(\chi(G^2))$ for every graph G .

Proof. Let f be an arbitrary distance-two q -coloring of graph G with $q \in \mathbb{N}$, and let h be a graceful coloring of the complete graph K_q with vertex set $\{1, 2, \dots, q\}$. To prove that $\chi_g(G) \leq a(\chi(G^2))$, it suffices to show that $h \circ f$ is a graceful coloring of G .

Consider an arbitrary 3-vertex path u, v, w in G . Note that $f(u), f(v)$ and $f(w)$ are pairwise distinct vertices in K_q (because f is a distance-two coloring of G). Since $f(u), f(v), f(w)$ is a 3-vertex path in K_q and h is a graceful coloring of K_q , we have $|h(f(u)) - h(f(v))| \neq |h(f(v)) - h(f(w))|$. Therefore, $h \circ f$ is a graceful coloring of G . \square

2.2 Complexity of Graceful Coloring

For $k \in \mathbb{N}$, the problem GRACEFUL k -COLORABILITY is defined as follows.

GRACEFUL k -COLORABILITY

Instance: A graph G .

Question: Does G admit a graceful k -coloring?

For a fixed k , GRACEFUL k -COLORABILITY can be expressed in monadic second-order logic, and thus by Courcelle's theorem [2,4], it is polynomial-time solvable for graph classes of bounded treewidth (or cliquewidth) in general, and for the classes of forests and outerplanar graphs in particular (recall that outerplanar graphs have treewidth at most 2).

Observation 1. Let G be a graceful 4-colorable connected chordal graph. Then, either G does not contain any 4-vertex cycle, or G is diamond (i.e., $K_4 - e$). \square

Due to Observation 1, every component H of a graceful 4-colorable chordal graph G is either the diamond graph or does not contain any cycle of length 4 or more, and thus G is outerplanar. As a result, we have the following.

Theorem 3. GRACEFUL 4-COLORABILITY is polynomial-time solvable for the class of chordal graphs. \square

To prove the NP-completeness of GRACEFUL k -COLORABILITY in various graph classes, we employ the following observation.

Observation 2 ([11]). Let G be a graph with a graceful k -coloring f , and let v be a vertex of G with degree $d \geq k/2$. Then, $f(v) \in \{1, \dots, k-d, d+1, \dots, k\}$. \square

Using the following construction, we show that GRACEFUL k -COLORABILITY of planar graphs is NP-complete for all $k \geq 5$.

Construction 1.

Parameter: A fixed integer $k \geq 5$ (not part of input).

Input: A 3-regular graph G .

Output: A graph G' of maximum degree $k-2$.

Guarantee (Lemma 1): G' is graceful k -colorable if and only if G is distance-two 4-colorable.

Steps: Introduce a copy of G . Attach $k-5$ leaf vertices at each vertex of the copy of G .

Lemma 1. Let $k \geq 5$. Let G be a 3-regular graph, and let G' be the output of Construction 1 when G is given as the input. Then, G' is graceful k -colorable if and only if G is distance-two 4-colorable.

Proof. Suppose that G' admits a graceful k -coloring $f': V(G') \rightarrow \{1, 2, \dots, k\}$. By Observation 2, each non-leaf vertex v of G' can get only the colors 1, 2, $k-1$ or k under f' (because $d_{G'}(v) = k-2$). Hence, the restriction of f' to $V(G)$ is a graceful coloring and in particular a distance-two coloring of G that uses only 4 colors (namely, 1, 2, $k-1$ and k). Therefore, G is distance-two 4-colorable.

Conversely, suppose that G is distance-two 4-colorable. Then, there exists a distance-two 4-coloring f of G with color palette $\{1, 2, k-1, k\}$ (i.e., $f: V(G) \rightarrow$

$\{1, 2, k-1, k\}$). We show that f can be extended into a graceful k -coloring f' of G' . For each non-leaf vertex v of G' , define $f'(v) = f(v)$, and color the leaf neighbors of v in C' as follows: if $f'(v) = 2$, then color the leaf neighbors of v with colors $\{4, 5, \dots, k-2\}$ in a bijective fashion; if $f'(v) = k-1$, then color the leaf neighbors of v with colors $\{3, 4, \dots, k-3\}$ in a bijective fashion; if $f'(v) \in \{1, k\}$, then color the leaf neighbors of v with distinct colors from $\{3, 4, \dots, k-2\}$.

Observe that f' is a distance-two k -coloring of G' . Thus, to prove that f' is a graceful k -coloring of G' , it suffices to show that there is no 3-vertex path u, v, w in G' with $f'(v) = (f'(u) + f'(w))/2$. On the contrary, assume that such a 3-vertex path u, v, w exists in G' . Since f' is a distance-two k -coloring of G' and its restriction to $V(G)$ is a graceful coloring of G , v is a non-leaf vertex (in G'), one vertex from $\{u, w\}$ is a non-leaf vertex, and the other vertex from $\{u, w\}$ is a leaf vertex. Without loss of generality, suppose that u is a non-leaf vertex, and w is a leaf vertex. Since $f'(v) = (f'(u) + f'(w))/2$, we have $f'(v) \notin \{1, k\}$; thus, $f'(v) \in \{2, k-1\}$. If $f'(v) = 2$, then $f'(u) = 1$ and $f'(w) = 2f'(v) - f'(u) = 3$, a contradiction because no leaf neighbor at v is colored 3. If $f'(v) = k-1$, then $f'(u) = k$ and $f'(w) = 2f'(v) - f'(u) = k-2$, a contradiction because no leaf neighbor at v is colored $k-2$. By contradiction, f' is indeed a graceful k -coloring of G' . This completes the proof. \square

Observe that for $k = 5$, the output graph in Construction 1 is the same as the input graph (i.e., $G' = G$). Further, the construction obviously takes only polynomial time in the input size. Moreover, observe that Construction 1 preserves planarity and bipartiteness. Feder, Hell and Subi [7] proved that it is NP-complete to check whether a 3-regular 3-connected planar bipartite graph is distance-two 4-colorable. Thanks to Construction 1 and Lemma 1, we have the following theorems.

Theorem 4. GRACEFUL 5-COLORABILITY is NP-complete for 3-regular 3-connected planar bipartite graphs. \square

Theorem 5. For $k \geq 6$, GRACEFUL k -COLORABILITY is NP-complete for planar bipartite 3-degenerate graphs of maximum degree $k-2$. \square

Next, we show that GRACEFUL 4-COLORABILITY is NP-complete by reducing from the following problem.

POSITIVE NOT-ALL-EQUAL 3-SAT E4

Instance: A boolean formula $\mathcal{F} = (X, C)$, where X is a set of variables, C is a set of clauses over X each containing three distinct variables, and each variable appears in exactly four clauses (the formula contains no negations).
Question: Does there exist a truth assignment for X such that each clause $c \in C$ contains at least one true variable and at least one false variable?

Darmann and Döcker [5] demonstrated the NP-completeness of this problem. We employ the following lemmas to construct gadgets. The proofs of Lemma 2 and Lemma 4 are omitted (hint for Lemma 2: consider colors on a, u, v, w, x, y, b).

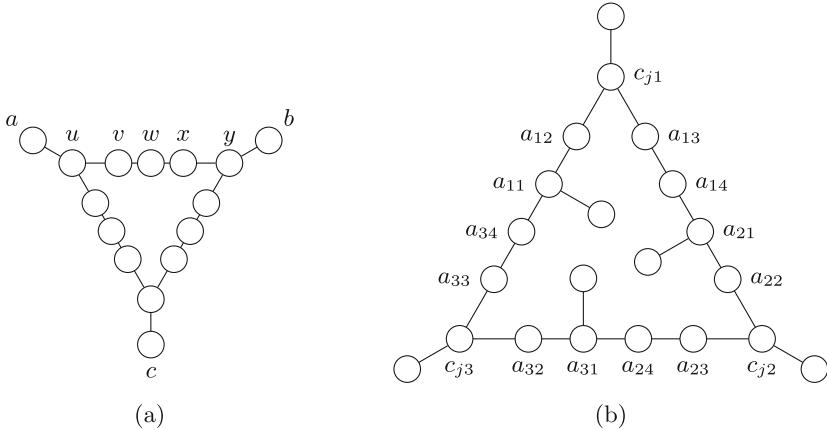


Fig. 2. Two graphs used to construct gadgets.

Lemma 2. Let f be a graceful 4-coloring of the graph in Fig. 2a with $f(a), f(b), f(c) \in \{1, 4\}$. Then, $f(a) = f(b) = f(c)$.

Lemma 3. Let f be a graceful 4-coloring of the graph in Fig. 2b. Then, $f(c_{j1}), f(c_{j2}), f(c_{j3}) \in \{1, 4\}$, and not all of them are equal.

Proof. By Observation 2, $f(c_{jk}), f(a_{k1}) \in \{1, 4\}$ for $k \in \{1, 2, 3\}$. To produce a contradiction, assume that $f(c_{j1}) = f(c_{j2}) = f(c_{j3}) = x$ (where $x \in \{1, 4\}$). Without loss of generality, assume that $x = 1$. For $k \in \{1, 2, 3\}$, vertex a_{k1} is at distance two from c_{jk} and thus $f(a_{k1}) = 4$. For $k \in \{1, 2, 3\}$ and $\ell \in \{2, 3, 4\}$, vertex $a_{k\ell}$ is within distance two from a vertex colored 1 and a vertex colored 4, and thus $f(a_{k\ell}) \in \{2, 3\}$. Since a_{13} is at a distance two from a_{12} , vertex a_{13} should get the opposite color (i.e., $f(a_{13}) = 5 - f(a_{12})$). By the same argument, $f(a_{14}) = 5 - f(a_{13}) = f(a_{12})$ and $f(a_{22}) = 5 - f(a_{14}) = 5 - f(a_{12})$. Since $f(a_{22}) = 5 - f(a_{12})$, by rotational symmetry, $f(a_{32}) = 5 - f(a_{22})$ and $f(a_{12}) = 5 - f(a_{32})$. That is, $f(a_{32}) = 5 - f(a_{22}) = f(a_{12})$ and $f(a_{12}) = 5 - f(a_{32}) = 5 - f(a_{12})$. Thus, a_{12} should receive the color opposite to its own color, a contradiction. \square

An *edge cut* $(S, V \setminus S)$ of a graph $G(V, E)$ is the set of edges in G with one endpoint in S and the other endpoint in $V \setminus S$, where $S \subseteq V$.

Lemma 4. Let $G = (V, E)$ be a graph with an edge cut $M = (S, V \setminus S)$ (where $S \subseteq V$). Let V_M denote the union of end points of edges in M . Let G_1 be the subgraph of G induced by $S \cup V_M$, and let G_2 be the subgraph of G induced by $(V \setminus S) \cup V_M$. Let f_1 and f_2 be graceful 4-colorings of G_1 and G_2 respectively such that $f_1(v) = f_2(v) \in \{1, 4\}$ for all $v \in V_M$. Then, combining f_1 and f_2 produces a graceful 4-coloring of G (i.e., the function $f: V(G) \rightarrow \{1, 2, 3, 4\}$ defined as $f(u) = f_1(u)$ for $u \in V(G_1)$ and $f(u) = f_2(u)$ for $u \in V(G_2)$ is a graceful 4-coloring of G).

Theorem 6. GRACEFUL 4-COLORABILITY is NP-complete for 2-degenerate graphs.

Proof. Let the boolean formula $\mathcal{F} = (X, C)$ be a given instance of POSITIVE NOT-ALL-EQUAL 3-SAT E4, where $X = \{x_1, x_2, \dots, x_n\}$ and $C = \{c_1, c_2, \dots, c_m\}$. For each clause c_j , fix an ordering of members of c_j . For each variable x_i , fix an ordering of clauses in which x_i appears. We construct a graph G by introducing a variable gadget (Fig. 3) for each variable x_i , introducing a clause gadget (Fig. 4) for each clause c_j , and then identifying vertex x_{ik} with vertex $c_{j\ell}$, vertex z_{ik} with vertex $d_{j\ell}$, and edge $x_{ik}z_{ik}$ with edge $c_{j\ell}d_{j\ell}$ whenever the following hold: (i) c_j is the k th clause in which x_i appears, and (ii) x_i appears as the ℓ th member of clause c_j . See Fig. 6 for an example. Clearly, G can be constructed in time polynomial in $m + n$.

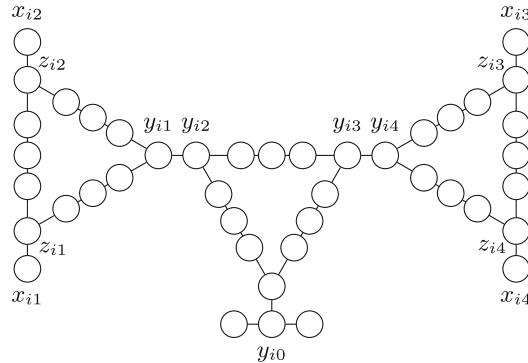


Fig. 3. Variable gadget for variable x_i .

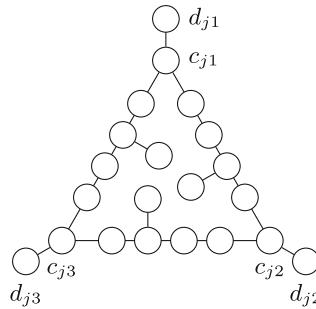


Fig. 4. Clause gadget for clause c_j . It is isomorphic to the graph in Fig. 2b.

We prove that G admits a graceful 4-coloring if and only if \mathcal{F} is a yes instance of POSITIVE NOT-ALL-EQUAL 3-SAT E4.

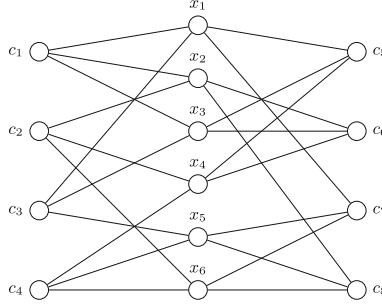


Fig. 5. Variable-clause incidence graph of the formula $\mathcal{F} = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_4 \vee x_6) \wedge (x_1 \vee x_3 \vee x_5) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee x_5 \vee x_6) \wedge (x_2 \vee x_5 \vee x_6)$.

Suppose that G admits a graceful 4-coloring f . For $i \in \{1, 2, \dots, n\}$, $j \in \{1, 2, \dots, m\}$, $k \in \{1, 2, 3, 4\}$ and $\ell \in \{1, 2, 3\}$, we have $f(x_{ik}), f(y_{ik}), f(c_{j\ell}) \in \{1, 4\}$ by Observation 2. Define a truth assignment T of X as $T(x_i) = \text{True}$ if $f(x_{i1}) = 1$, and $T(x_i) = \text{False}$ if $f(x_{i1}) = 4$. We claim that under T , each clause $c_j \in C$ contains at least one true variable and at least one false variable. Let $c_j = \{x_k, x_\ell, x_p\}$ be an arbitrary clause in C , where $k, \ell, p \in \{1, 2, \dots, n\}$. Then, in G , we have $c_{j1} = x_{kq}$, $c_{j2} = x_{\ell r}$ and $c_{j3} = x_{ps}$ for some $q, r, s \in \{1, 2, 3, 4\}$. Since the clause gadget is isomorphic to the graph in Fig. 2b, $f(c_{j1}), f(c_{j2}), f(c_{j3}) \in \{1, 4\}$, and not all of them are equal by Lemma 3. That is, at least one vertex among c_{j1}, c_{j2} and c_{j3} is colored 1 by f , and at least one of them is colored 4 by f . Thus, at least one vertex among $x_{kq}, x_{\ell r}$ and x_{ps} is colored 1 by f , and at least one of them is colored 4 by f . Thus, to prove that at least of x_k, x_ℓ and x_p is assigned *True* by T and at least one of them is assigned *False* by T , it suffices to show that $f(x_{kq}) = f(x_{k1})$, $f(x_{\ell r}) = f(x_{\ell 1})$, and $f(x_{ps}) = f(x_{p1})$. We prove this by showing that $f(x_{i1}) = f(x_{i2}) = f(x_{i3}) = f(x_{i4})$ for $i \in \{1, 2, \dots, n\}$. Note that the variable gadget contains three distinct copies of the graph in Fig. 2a. Moreover, $f(x_{ik}), f(y_{ik}), f(y_{i0}) \in \{1, 4\}$ for $i \in \{1, 2, \dots, n\}$ and $k \in \{1, 2, 3, 4\}$ by Observation 2. Hence, by Lemma 2, for each $i \in \{1, 2, \dots, n\}$,

$$f(x_{i1}) = f(x_{i2}) = f(y_{i2}), \quad f(y_{i1}) = f(y_{i4}), \quad \text{and} \quad f(y_{i3}) = f(x_{i3}) = f(x_{i4}) \quad (1)$$

Since y_{i1} is adjacent to y_{i2} , we have $f(y_{i1}) = 5 - f(y_{i2}) = 5 - f(x_{i2})$. Similarly, $f(y_{i4}) = 5 - f(y_{i3}) = 5 - f(x_{i3})$. Hence, $5 - f(x_{i2}) = f(y_{i1}) = f(y_{i4}) = 5 - f(x_{i3})$. Thus, $f(x_{i2}) = f(x_{i3})$. Therefore, by Eq. (1), we have $f(x_{i1}) = f(x_{i2}) = f(x_{i3}) = f(x_{i4})$. Consequently, \mathcal{F} is a yes instance of POSITIVE NOT-ALL-EQUAL 3-SAT E4.

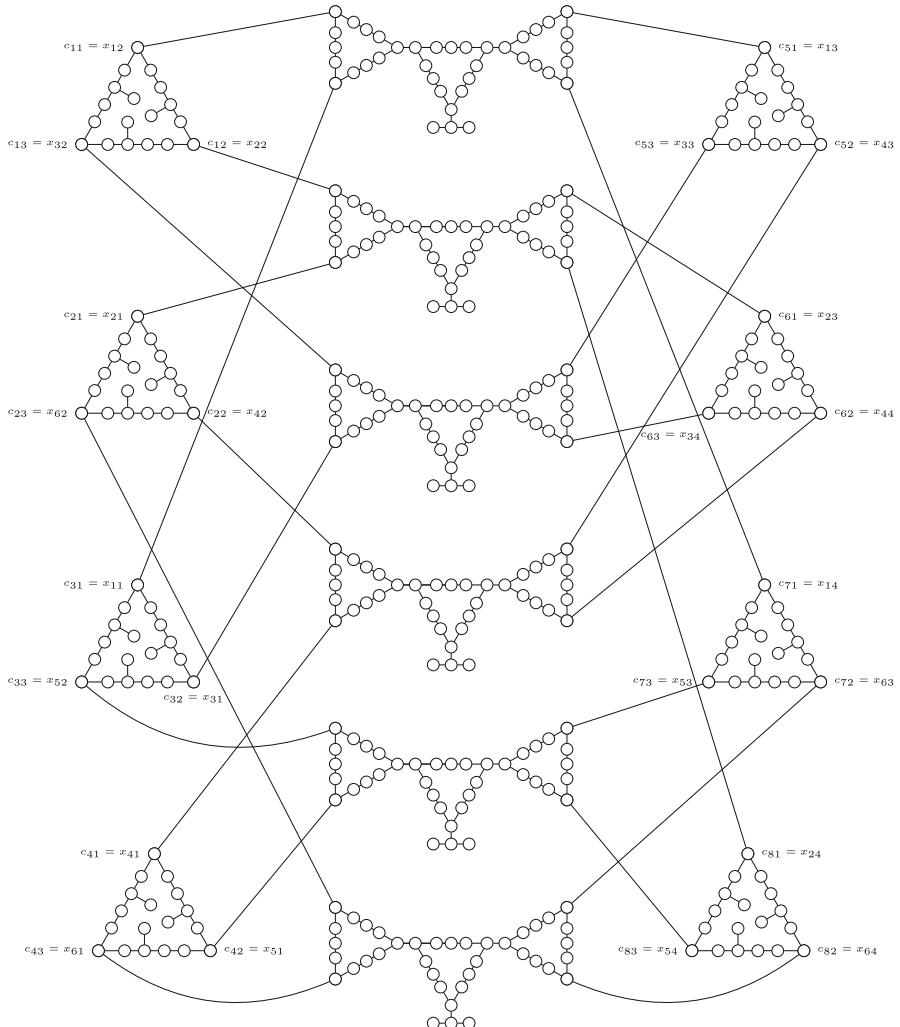


Fig. 6. An example for the construction of graph G . This graph is for the formula $(x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_4 \vee x_6) \wedge (x_1 \vee x_3 \vee x_5) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee x_5 \vee x_6) \wedge (x_2 \vee x_5 \vee x_6)$. The variable-clause incidence graph of this formula is shown in Fig. 5.

Conversely, suppose that there is a truth assignment T on X such that each clause contains a variable assigned *True* by T and a variable assigned *False* by T . If variable x_i is assigned *True* under T , then color the variable gadget for x_i by the coloring ψ shown in Fig. 7. If $T(x_i) = \text{False}$, then color the variable gadget for x_i using the ‘opposite’ of colors shown in Fig. 7 (i.e., color each vertex v in the gadget using the color $5 - \psi(v)$). This can be extended to a graceful 4-coloring of G by coloring each clause gadget by one of the colorings in Fig. 8 or a suitable angular rotation of them. The coloring obtained this way is a graceful 4-coloring of G due to Lemma 4 (see Fig. 9 for an example). \square

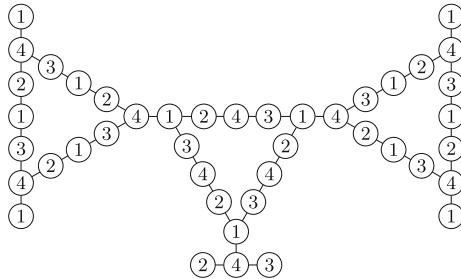


Fig. 7. A graceful 4-coloring ψ of the variable gadget.

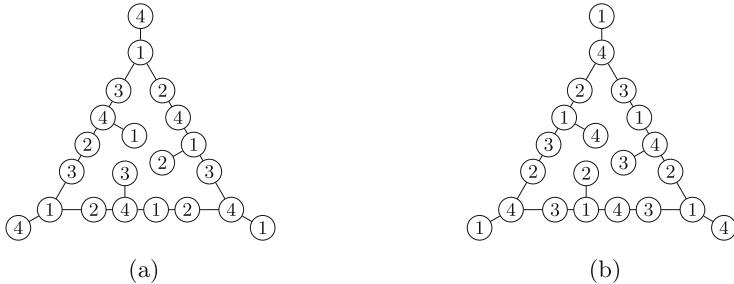


Fig. 8. Two graceful 4-colorings of the clause gadget.

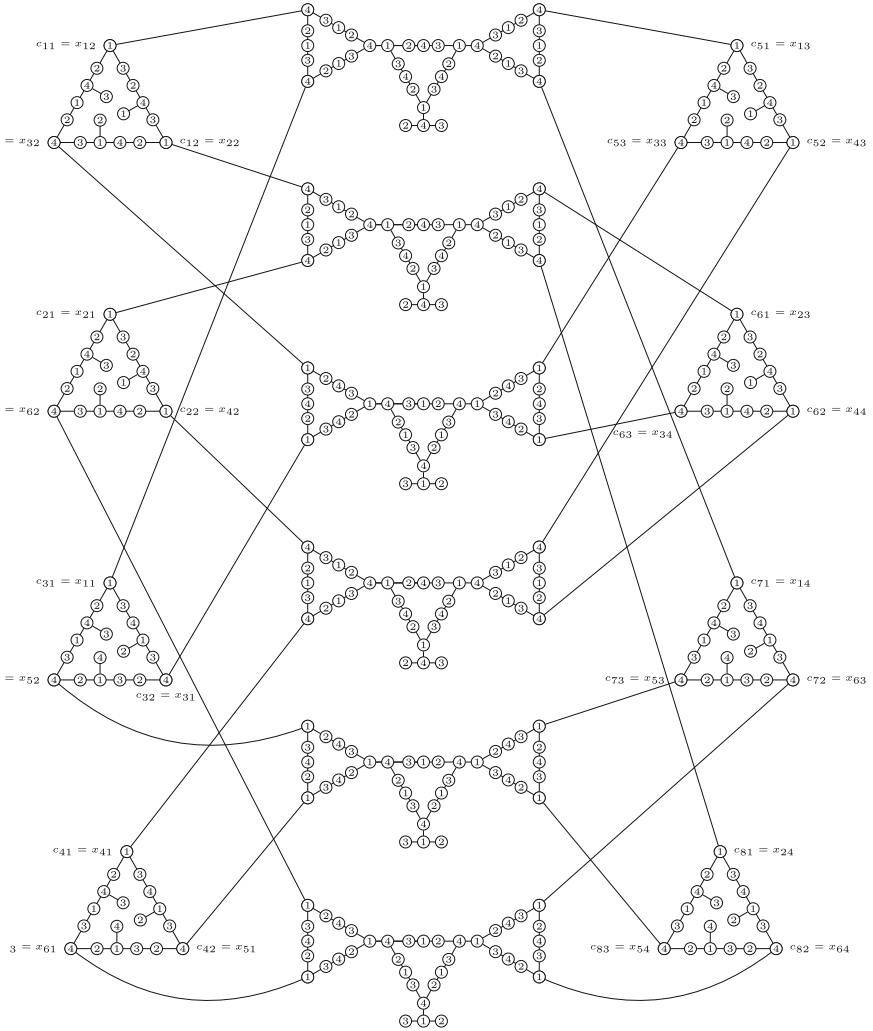


Fig. 9. A graceful 4-coloring of the graph in Fig. 6. This graceful 4-coloring of G corresponds to the truth assignment $x_1 \leftarrow \text{True}$, $x_2 \leftarrow \text{True}$, $x_3 \leftarrow \text{False}$, $x_4 \leftarrow \text{True}$, $x_5 \leftarrow \text{False}$, and $x_6 \leftarrow \text{False}$.

Acknowledgement. We thank three anonymous referees for their valuable suggestions.

Future Directions. In this article, we proved that graceful coloring is NP-hard even when the given graph is planar bipartite, regular or 2-degenerate. The complexity of checking whether a planar graph is graceful 4-colorable remains open. Providing an explicit algorithm to check whether a chordal graph is graceful 4-colorable is another future direction we are interested in.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bi, Z., Byers, A., English, S., Laforge, E., Zhang, P.: Graceful colorings of graphs. *JCMCC J. Comb. Math. Comb. Comput.* **101**, 101–119 (2017)
2. Borie, R.B., Parker, R.G., Tovey, C.A.: Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica* **7**(1–6), 555–581 (1992). <https://doi.org/10.1007/BF01758777>
3. Byers, A.D.: Graceful colorings and connection in graphs. Ph.D. thesis, Western Michigan University (2018)
4. 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](https://doi.org/10.1016/0890-5401(90)90043-H)
5. Darmann, A., Döcker, J.: On a simple hard variant of not-all-equal 3-sat. *Theoret. Comput. Sci.* **815**, 147–152 (2020). <https://doi.org/10.1016/j.tcs.2020.02.010>
6. English, S., Zhang, P.: On graceful colorings of trees. *Mathematica Bohemica* **142**(1), 57–73 (2017). <https://doi.org/10.21136/MB.2017.0035-15>
7. Feder, T., Hell, P., Subi, C.: Distance-two colourings of Barnette graphs. *Eur. J. Comb.* **91**, 15 (2021). <https://doi.org/10.1016/j.ejc.2020.103210>. id/No 103210
8. Gallian, J.A.: A dynamic survey of graph labeling. *Electron. J. Comb.* **1**(DynamicSurveys), DS6 (2018). 10.37236/27
9. Khoirunnisa, S., Dafik, Kristiana, A.I., Alfarisi, R., Albirri, E.R.: On graceful chromatic number of comb product of ladder graph. *J. Phys. Conf. Ser.* **1836**(1), 012027 (2021). <https://doi.org/10.1088/1742-6596/1836/1/012027>
10. Kristiana, A., Setyawan, D., Albirri, E., Prihandini, R., Alfarisi, R.: On graceful coloring of generalized Petersen graphs. *Discrete Math. Algorithms Appl.* **16**(07), 2350097 (2024). <https://doi.org/10.1142/S1793830923500970>
11. Laavanya, D.: Graceful colorings of bipartite graph subclasses and graphs from operations: A structural and algorithmic perspective. Ph.D. thesis, Vellore Institute of Technology, Chennai, India (2024)
12. Laavanya, D., Devi Yamini, S.: Graceful coloring of some corona graphs - An algorithmic approach. *Commun. Comb. Optim.* (2024). <https://doi.org/10.22049/cco.2024.29155.1866>
13. Laavanya, D., S., D.Y.: A structural approach to the graceful coloring of a subclass of trees. *Heliyon* **9**(9) (2023). <https://doi.org/10.1016/j.heliyon.2023.e19563>
14. OEIS Foundation Inc.: Entry A065825 in the on-line encyclopedia of integer sequences. <https://oeis.org/A065825>
15. Ringel, G.: Theory of graphs and its applications. In: Proceedings of the Symposium Smolenice, p. 162 (1963)
16. Rosa, A.: On certain valuations of the vertices of a graph. In: Theory of Graphs, International Symposium, Rome 1966, pp. 349–355 (1967)
17. Suparta, I.N., Venkatachalam, M., Gunadi, I.G.A., Pratama, P.A.C.: Graceful chromatic number of some cartesian product graphs. *Ural Math. J.* **9**(2 (17)), 193–208 (2023)



Parameterized Complexity of Coupon Coloring of Graphs

Pradeesha Ashok¹(✉), Pradyun Devarakonda¹, Shiven Phogat¹,
Swaroop A. Ram Rayala¹, and J. A. Sherin²

¹ International Institute of Information Technology Bangalore, Bengaluru, India

{pradeesha,devarakonda,pradyun,shiven.phogat,
swaroop.ramrayala}@iitb.ac.in

² PSG College of Technology, Coimbatore, India

21pt28@psgtech.ac.in

Abstract. Given a graph $G(V, E)$, a q -coupon coloring of G refers to a coloring $f : V \rightarrow [q]$ such that the following is true for all $v \in V$: for all $i \in [q]$, there exists $u \in N(v)$ such that $f(u) = i$. Given a graph G , the q -COUPON COLORING problem is to decide whether G admits a q -coupon coloring. The q -COUPON COLORING problem is shown to be NP -complete. We initiate the study of parameterized complexity of the q -COUPON COLORING problem. It is implied by existing results that parameterization by q is unlikely to admit FPT algorithms. We study the q -COUPON COLORING problem parameterized by structural parameters including neighborhood diversity, twin cover, distance to clique and treewidth of the graph. We show FPT algorithms when the parameter is neighborhood diversity, twin cover and distance to clique and prove tight lower bounds when the parameter is treewidth.

Keywords: Coupon Coloring · Total Domatic Number · Parameterized Complexity · Fixed Parameter Tractable

1 Introduction

Given a graph $G(V, E)$, a (vertex) q -coloring of G is a function $f : V \rightarrow [q]$. A subset of vertices that are assigned the same *color* is referred to as a *color class*. The well studied *proper coloring* is a vertex coloring such that every color class induces an independent set. Graph Coloring is an important and well-studied topic in Graph Theory. Many variants of coloring like list coloring [10], conflict free coloring [5, 13], $L(h, k)$ -coloring [6] etc. are studied. The aim of the aforementioned coloring problems is to find a coloring that minimizes the number of colors used. However, there is also a class of coloring problems where the objective is to maximize the number of colors used. For example, the *b-coloring* [20] problem asks for a proper vertex coloring $f : V \rightarrow [q]$ such that for every color $c \in [q]$, there exists a vertex v such that $f(v) = c$ and for all $c_i \in [q] \setminus \{c\}$, v has at least one neighbor of color c_i . For every graph G , there

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2025

D. Gaur and R. Mathe (Eds.): CALDAM 2025, LNCS 15536, pp. 25–37, 2025.

https://doi.org/10.1007/978-3-031-83438-7_3

exists a b -coloring that uses $\chi(G)$ colors, where $\chi(G)$ is the minimum number of colors needed to proper color G .

Hence, the interesting question is to determine the maximum number of colors that can be used in a b -coloring. A closely related question is the *Fall Coloring* [11] problem which is to find a proper coloring such that every vertex v is adjacent to at least one vertex of every color class other than its own.

We study a problem called *Coupon Coloring*.

Definition 1. *Given a graph $G(V, E)$, a q -coupon coloring of G is a coloring $f : V \rightarrow [q]$ such that for all $v \in V$, for all $i \in [q]$, there exists $u \in N(v)$ such that $f(u) = i$.*

Clearly, a coupon coloring is not a proper coloring since the color assigned to a vertex needs to be repeated in its open neighborhood. It is also clear that assigning the same color to all vertices is a valid coupon coloring. Hence, the interesting question to study here is how to maximize q such that a q -coupon coloring exists. We study the following decision version, called the *q -COUPON COLORING* problem.

q -COUPON COLORING

Input: A graph $G(V, E)$ and an integer q

Question: Does G admit a q -coupon coloring?

The coupon coloring is also connected to graph domination, another well-studied area in Graph Theory. Given a graph $G(V, E)$, a subset $D \subseteq V$ is called a *total dominating set* of G , if the union of open neighborhoods of vertices in D gives V . We can easily see that any color class in a coupon coloring of G is a total dominating set of G . Thus coupon coloring can be seen as a partition of the vertex set such that every part is a total dominating set. Now, for a graph G , the maximum value of q such that there exists a partitioning of V into q total dominating sets is called the *total domatic number* of G , denoted as $d_t(G)$. Thus, for a graph G and integer q , checking whether the total domatic number of G is at least q is equivalent to the *q -COUPON COLORING* problem.

The concept of coupon coloring also has real-world applications in network science, especially in the context of large-scale multi-robot networks [33]. We present a slight variation of an application involving the deployment of a group of robots to a remote location for environmental monitoring, as described in [1]. This system can be represented as a graph where the vertices correspond to the robots, and an edge exists between two vertices if the corresponding robots can communicate. Each robot in the network is responsible for collecting different types of data (e.g., temperature, humidity) but, due to power limitations, is only equipped with a single sensor (such as a thermometer or barometer). To access other types of data, each robot must communicate with its neighboring robots. This approach is designed to account for power constraints or potential system failures, ensuring that no robot relies solely on its own sensor but instead can

obtain the full range of data through collaboration with neighboring robots. The objective in this scenario is to maximize the variety of sensors distributed across the network, ensuring that each robot has access to all the necessary types of data, even if it depends on its neighbors to acquire all the information. In this context, the set of all vertices that correspond to a sensor or instrument of a specific type, forms a total dominating set. It is possible to distribute q different instruments across the network if and only if the corresponding graph admits a q -coupon coloring.

Coupon Coloring and Total Domatic number have been extensively studied. The term *coupon coloring* was introduced in [7], while Goddard and Henning [16] referred to such a coloring as *thoroughly distributed*. In a work by Heggernes and Telle [17], the authors demonstrated that determining whether a given graph G has a total domatic number $d_t(G) \geq 2$ is NP-complete, even for bipartite graphs. Zelinka [34] proved that no minimum degree condition suffices to guarantee $d_t(G) = 2$. Similarly, [22] shows that determining whether a bipartite planar graph has $d_t(G) \geq 3$ is NP-complete. This builds on the work of Goddard and Henning [16], who proved that no planar graph G can have a total domatic number $d_t(G) > 4$. Furthermore, it is NP-complete to decide if a split graph has $d_t(G) \geq k$ for any $k \geq 2$ [22]. However, for threshold graphs, the total domatic number can be computed in polynomial time. The coupon coloring of special graph classes has been explored in [14, 28, 29, 31]. Specifically, Chen et al. demonstrated in [8] that the coupon coloring problem can be computed in polynomial time for cographs. The coupon coloring problem for hypercubes is strongly connected to problems in coding theory, as discussed by Östergård [27]. Specifically, for $k = 2$, this problem is equivalent to the well-known Property B of hypergraphs, which was studied by Erdős [12]. A concept related to coupon coloring in hypergraphs, where every color is present in each hyperedge, is known as *panchromatic coloring* [32]. Additionally, [22] presents fast exponential time algorithms for solving this problem on general graphs. The notions of total domination and coupon coloring have been thoroughly investigated for regular graphs. Several researchers, such as Aram et al. [4], studied the total domatic number of random r -regular graphs, Chen et al. [7] showed that k -regular graphs have a total domatic number of at least $(1 - o(1)) \frac{k}{\ln k}$. Additionally, Akbari et al. [3] provided a characterization of 3-regular graphs with a total domatic number of at least two, revealing that these graphs specifically exclude a certain tree with a maximum degree of 3 as an induced subgraph. Furthermore, the findings of Henning and Yeo [18] indicate that all k -regular graphs, for $k \geq 4$, have $d_t(G) \geq 2$. The study of 2-coupon coloring in cubic graphs containing 3-cycles or 4-cycles is addressed in [2].

1.1 Our Results

We initiate the study of the parameterized complexity of q -COUPON COLORING problem. Since the 2-coupon coloring problem is known to be NP-complete, it follows that the q -coupon coloring problem parameterized by q is para-NP-hard, implying it is unlikely to admit FPT algorithms or even XP-algorithms.

Therefore, we focus on structural parameters in our study. (In each result, k denotes the corresponding parameter).

1. Firstly, we consider the parameter **neighborhood diversity**. We represent q -COUPON COLORING problem parameterized by neighborhood diversity as an *ILP* formulation. By utilizing known algorithms to solve the *ILP*, we derive an algorithm that runs in $\mathcal{O}(k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)})$ time.
2. We investigate the parameter **twin cover** number and reduce the problem to a bounded number of sub-problems each of which can be solved using a dynamic programming approach. Consequently, we obtain an algorithm for q -COUPON COLORING problem, parameterized by twin cover number, which runs in $2^{\mathcal{O}(k^2)} \cdot n^{\mathcal{O}(1)}$ time.
3. Next, we explore the parameter **distance to clique**. We present an algorithm for the q -COUPON COLORING problem, parameterized by distance to clique, which runs in $2^{\mathcal{O}(2^k)} \cdot n^{\mathcal{O}(1)}$ time.
4. Lastly, we investigate the parameter **treewidth**. Using existing results, we observe that q -COUPON COLORING problem, parameterized by treewidth admits an FPT algorithm with a runtime of $\mathcal{O}(2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)})$. Moreover, we demonstrate that this bound is essentially tight by proving that the problem cannot be solved in $2^{o(k \log k)} \cdot n^{\mathcal{O}(1)}$ time, unless ETH fails.

Organization of the Paper: Sect. 2 provides notations and definitions that will be used in subsequent sections. In Sect. 3, we present an algorithm for q -COUPON COLORING parameterized by neighborhood diversity. In Sect. 4, we describe an algorithm for q -COUPON COLORING parameterized by twin cover. In Sect. 5, we present an algorithm for q -COUPON COLORING parameterized by distance to clique. In Sect. 6, we discuss the q -COUPON COLORING problem, parameterized by treewidth.

2 Preliminaries

In this section, we discuss some concepts, notations and results that will be used in subsequent sections. For $n \in \mathbb{N}$, the set $\{1, 2, \dots, n\}$ is denoted by $[n]$. We consider the graph $G = (V, E)$ to be simple, finite, and undirected throughout this paper. The open and closed neighborhood of a vertex v is denoted as $N(v)$ and $N[v]$ respectively.

Fixed Parameter Tractability (FPT) [9]: A parameterized problem (Π, k) , with $|\Pi| = n$, is said to be *fixed parameter tractable* (FPT) if there exists an algorithm with running time $f(k) \cdot n^c$, for a constant c and a computable function f .

Locally Checkable Vertex Partitioning Problems: [30] A degree constraint matrix D_q is a $q \times q$ matrix where each entry is a subset of natural numbers $\{0, 1, \dots\}$. LOCALLY CHECKABLE VERTEX PARTITIONING problems in a graph G involves finding a partition of the vertex set $V(G)$ into q subsets V_1, V_2, \dots, V_q ,

such that the following is true : For any vertex v in subset V_i , the number of neighbors it has in subset V_j , $|N_G(v) \cap V_j|$, belongs to the set $D_q[i, j]$, for all $1 \leq i, j \leq q$.

Exponential Time Hypothesis (ETH): The Exponential Time Hypothesis, introduced by Impagliazzo and Paturi [19], states that the 3-SAT problem cannot be solved in $2^{o(n)}$ time, where n is the number of variables in the input formula.

Integer Linear Programming [23]: The Integer Linear Programming problem is defined as follows:

$$\min \mathbf{w}^T \mathbf{x} : A\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq 0, \quad \mathbf{x} \in \mathbb{Z}^d \quad (\text{ILP})$$

where $\mathbf{w} \in \mathbb{Z}^d$, $A \in \mathbb{Z}^{r \times d}$, and $\mathbf{b} \in \mathbb{Z}^r$. Let $\Delta = \|A\|_\infty = \max_{i,j} |A_{ij}|$ be the largest coefficient in A in absolute value. We know the following.

Proposition 1. [21] *ILP can be solved in time $\mathcal{O}(\sqrt{r}\Delta)^{2r} \log \|b\|_\infty + \mathcal{O}(rd)$.*

Feedback Vertex Set: For a given graph G , a set of vertices C is called a *feedback vertex set* if, after removing all vertices in C , the resulting graph is acyclic.

Neighborhood Diversity: [24]: For a given graph G , let P_1, \dots, P_k be a partition of $V(G)$, where each P_i represents an equivalence class of the following equivalence relation: Two vertices $u, v \in V(G)$ are considered equivalent if they have the same neighborhoods except possibly themselves, i.e., $u \sim v \iff N(v) \setminus \{u\} = N(u) \setminus \{v\}$.

Such a vertex partitioning is called as an *nd-decomposition*. The neighborhood diversity of a graph G , denoted $\text{nd}(G)$, is defined as the smallest integer k for which there exists an *nd-decomposition* of G with exactly k partitions.

The neighborhood diversity of G and can be computed in polynomial time [25]. It can be seen that each P_i either induces a clique or an independent set, and between any two parts P_i and P_j , there is either a complete bipartite graph or no edges at all.

Let $T(G)$ be a weighted multigraph constructed based on the minimal *nd-decomposition* of G . The vertices of $T(G)$ are v_1, \dots, v_k , with each vertex v_i having a weight n_i equal to $|P_i|$. An edge exists between v_i and v_j (for $i \neq j$) if the edges connecting P_i and P_j form a complete bipartite graph in G . Additionally, there is a loop at v_i if P_i forms a clique in G . This multigraph $T(G)$ is referred to as the *type graph* of G . The encoding length of $T(G)$ is $\mathcal{O}(k^2 \log n)$, and it can be constructed from G in $\mathcal{O}(n + m)$ time [23].

Twin Cover: Two vertices u and v are considered (true) twins if they share the same closed neighborhood, i.e., $N[u] = N[v]$. An edge $\{u, v\}$ in the edge set E is referred to as a twin edge if u and v are true twins. A vertex set S is called a *twin cover* if, for every edge $\{u, v\} \in E$, either the edge is a twin edge or at least one of the vertices u or v is included in S . The smallest size of a twin cover in G

is known as the *twin cover number* of G . Alternatively, the twin cover number of a graph G is the size of a smallest subset $S \subseteq V(G)$, such that every two adjacent vertices in $G \setminus S$ have the same closed neighborhood in G . In Theorem 4 of [15], it is proven that if a graph G has a minimum twin cover of size at most k , then a twin cover of size k can be found in $\mathcal{O}(|E| + k|V| + 1.2738^k)$ time.

Distance to Clique: For a graph $G = (V, E)$, the parameter *distance to clique* refers to the size of the smallest subset $D \subseteq V$ such that the remaining vertices, $V \setminus D$, induce a clique.

3 Neighborhood Diversity

Let G be a graph with neighborhood diversity, k , and let $T(G)$ be its type graph. Assume an *nd-decomposition* P_1, P_2, \dots, P_k and the corresponding type graph $T(G)$ is given. Let $n_i = |P_i|$ for each $i \in [k]$.

Theorem 1. *q -COUPON COLORING admits an FPT algorithm parameterized by neighborhood diversity k , that runs in $O(k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)})$.*

Proof. We define $\mathcal{I}(G) = 2^{V(T(G))}$ to be the power set of $V(T(G))$, which consists of all possible subsets of $V(T(G))$.

We now construct an instance of Integer Linear Programming whose optimal solution will encode a q -coupon coloring of G . We define a variable x_I for each $I \in \mathcal{I}(G)$, that intuitively stores the number of color classes that intersect exactly the types present in I . The intuition underlying the ILP formulation is as follows: First of all, the total number of color classes is q . Also, the number of colors intersecting any type does not exceed the size of that type. In the case where a type v_i in I corresponds to a part P_i that induces a clique and none of the neighbors of v_i are included in I , then for any color c that intersects exactly the types in I , at least two vertices within P_i are assigned the color c in any valid coupon coloring. Otherwise, if v_i has at least one neighbor in I , at least one vertex in P_i is assigned the color c . Each type's neighborhood must include all q colors, ensuring that the coupon coloring property is satisfied. The constraints are as follows:

$$\sum_{I \in \mathcal{I}(G)} x_I = q \quad (1)$$

$$\sum_{\substack{I \in \mathcal{I}(G) \\ v_i \in I}} x_I \leq |P_i| \quad \forall P_i \text{ where } i \in [k] \quad (2)$$

$$\sum_{\substack{I \in \mathcal{I}(G) \\ v_i \in I \\ I \cap N_T(v_i) = \emptyset}} 2 \cdot x_I + \sum_{\substack{I' \in \mathcal{I}(G) \\ v_i \in I' \\ I' \cap N_T(v_i) \neq \emptyset}} x_{I'} \leq |P_i| \quad i \in [k] \text{ s.t. } G[P_i] \text{ is a clique} \quad (3)$$

$$\sum_{\substack{I \in \mathcal{I}(G) \\ v_i \in I \\ I \cap N_T(v_i) \neq \emptyset}} x_I = q \quad \forall P_i \text{ where } i \in [k] \quad (4)$$

Consider an optimal solution $(x'_{I_1}, x'_{I_2}, \dots, x'_{I_{2^k}})$ of (1)–(4). We now construct a valid q -coupon coloring of G using Algorithm 1.

It is easy to verify that Algorithm 1 returns a valid q -coupon coloring.

Let $f : V \rightarrow [q]$ be a valid q -coupon coloring of G . We show that there exists a solution of (1)–(4) that corresponds to f . Let V' initially be an empty set. For every vertex $v \in V$, repeat the following : For every color $i \in [q]$, check if V' contains a vertex $u \in N(v)$ such that $f(u) = i$. If not, select an arbitrary vertex $u \in N(v)$ such that $f(u) = i$ and add u to V' . Note that such a vertex u always exists since f is a valid q -coupon coloring. Let G' be the sub-graph of G induced by V' . Let f' be the restriction of f to V' . We now iterate through each subset I in $\mathcal{I}(G)$. Let $V'_i = P_i \cap V'$. We now examine the coloring f' and count how many of the color classes intersect precisely with I , i.e., the color class that intersects with every V'_i , $v_i \in I$ and does not intersect with any V'_j , $v_j \in I$. Now we set x_I to this count. It is easy to verify that it is a solution.

Algorithm 1

```

1: Let  $c$  be a variable ranging from 1 to  $q$ .
2: for each  $I \in \mathcal{I}(G)$  do
3:   for  $x'_I$  times do
4:     for each  $v_i \in \mathcal{I}(G)$  do
5:       if  $G[P_i]$  is an independent set then
6:         Color an arbitrary uncolored vertex with color  $c$ .
7:       else if  $G[P_i]$  is a clique then
8:         if there exists  $P'_i \in N_T(v_i) \cap I$  then
9:           Color one uncolored vertex with  $c$ .
10:        else
11:          Color two uncolored vertices with  $c$ .
12:        end if
13:      end if
14:    end for
15:  end for
16:   $c := c + 1$ 
17: end for
18: if any uncolored vertices remain then
19:   Color them arbitrarily with any color from  $[q]$ .
20: end if
```

Time Complexity Analysis : ILP (1)–(4) can be solved using Proposition 1 in time $\mathcal{O}(\sqrt{r}\Delta)^{2r} \log \|b\|_\infty + \mathcal{O}(rd)$. Here, $r \leq 1 + 3k \implies r \leq \mathcal{O}(k)$, $d = |\mathcal{I}(G)| = 2^k$, $\Delta = 2$, and $\log \|b\|_\infty \leq \log n$. Also, Algorithm 1 runs in $\mathcal{O}(2^k)$ time. Thus the resulting complexity is $\mathcal{O}(k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)})$. \square

4 Twin Cover

Let $G = (V, E)$ be a graph with a twin cover $X \subseteq V$ of size k and let X be given. Let $C_1, C_2, \dots, C_{n'}$ be the cliques in $G \setminus X$. Note that two vertices in the same clique are twins. We use the notation $N(C_i)$ to denote $X \cap N(v)$, for $v \in C_i$.

Theorem 2. *q -COUPON COLORING admits an FPT algorithm parameterized by twin cover number k , that runs in $2^{\mathcal{O}(k^2)} \cdot n^{\mathcal{O}(1)}$ time.*

Proof. We give an algorithm that returns a q -coupon coloring χ' of G , if it exists. Assume, without loss of generality, that χ' uses only colors from $[k]$ on the vertices in X . We repeat the following for all possible colorings $\chi : X \rightarrow [k]$.

For each clique C_i , perform the following: For each $k+1 \leq j \leq q$, assign the color j to two arbitrarily selected uncolored vertices in C_i . Next, for all $j \in [1, k]$ such that C_i does not have a neighbor with color j in X , assign j to two arbitrarily selected uncolored vertices within C_i . If, at any point, it becomes impossible to find such uncolored vertices for the assignment, we return *NO*.

We denote the partial coloring obtained until now as χ . Now we try to extend the coloring χ to get a valid q -coupon coloring of G . For this, we define a function $CC_\chi(f, j)$, where $f : X \rightarrow 2^{[k]}$ and $1 \leq j \leq n'$. $CC_\chi(f, j)$ returns *True* if and only if there exists a coupon coloring of G that extends the coloring χ such that $\forall v \in X$ and $\forall c \in f(v)$, the neighbors of v that are assigned color c belongs to at least one of the cliques C_1, C_2, \dots, C_j . Let the *deficiency function* $def : X \rightarrow 2^{[k]}$ be defined as follows:

$$def(v) = [k] \setminus \bigcup_{u \in N(v)} \chi(u)$$

Then there exists a valid q -coupon coloring of G that extends χ if $CC_\chi(def, n')$ returns *True*. Now, we give a dynamic programming based algorithm to solve $CC_\chi(f, j)$. We give a recurrence for $CC_\chi(f, j)$ as follows:

$$CC_\chi(f, j) = \bigvee_{S \subseteq [k]} CC_\chi(f'_S, j-1) \quad (5)$$

where f'_S is defined as follows: $\forall x \in N(C_j), f'_S(x) = f(x) \setminus S$ and $\forall x \notin N(C_j), f'_S(x) = f(x)$.

Now, we prove the correctness of this recurrence. Assume $CC_\chi(f, j)$ holds true for some f and j . This implies that a valid coloring exists that extends the coloring χ such that $\forall v \in X$ and $\forall c \in f(v)$, the neighbors of v that are assigned color c belongs to at least one of the cliques C_1, C_2, \dots, C_j . Let S be the subset of colors that are assigned to vertices of C_j in this coloring. Since this is a valid coloring, it is trivial that $CC_\chi(f'_S, j-1)$ must also hold true. For a fixed f and j , computing the recurrence takes $\mathcal{O}(2^k)$ time. The number of distinct values of f is $\mathcal{O}(2^{k^2})$ and that for j is n' . Thus computing $CC_\chi(def, n')$ takes $\mathcal{O}(2^{\mathcal{O}(k^2)} n^{\mathcal{O}(1)})$ time. Moreover, this is repeated at most k^k times. Thus the total running time is $2^{\mathcal{O}(k^2)} \cdot n^{\mathcal{O}(1)}$. \square

5 Distance to Clique

Let $G = (V, E)$ be a graph with a set $X \subseteq V$ of size k , such that $G[V \setminus X]$ is a clique. X can be computed in $\mathcal{O}(2^k \cdot n^{\mathcal{O}(1)})$ time.

Theorem 3. *q -COUPON COLORING admits an FPT algorithm parameterized by distance to clique that runs in $2^{\mathcal{O}(2^k)} \cdot n^{\mathcal{O}(1)}$ time.*

Proof. We partition the clique into subsets V_1, \dots, V_{2^k} such that for all i where $1 \leq i \leq 2^k$, we have $N(u) \cap X = N(v) \cap X, \forall u, v \in V_i$.

Let $\{x_1, x_2, \dots, x_k\}$ be the vertices in X . It is easy to see that the partitions $\{V_1, \dots, V_{2^k}, \{x_1\}, \{x_2\}, \dots, \{x_k\}\}$ of $V(G)$ is a valid nd-decomposition. Thus, the neighborhood diversity of G can be at most $2^k + k$. Now the result follows from Theorem 1. \square

6 Treewidth

The q -COUPON COLORING problem is a special case of LOCALLY CHECKABLE VERTEX PARTITIONING problem (refer page 4 for definition), with q color classes, a degree constraint matrix D_q such that for all $1 \leq i, j \leq q$, $D_q[i, j] = \mathbb{N} \setminus \{0\}$. By Theorem 5.7 in [30], we can see the following result.

Lemma 1. *q -COUPON COLORING problem parameterized by treewidth k , can be solved in time $\mathcal{O}(q^k n^{\mathcal{O}(1)})$ time.*

We know that if $q > \delta(G)$, where $\delta(G)$ is the minimum degree of G , then the instance is trivially NO. Hence, we assume $q \leq \delta(G)$. It is also known that $\delta(G) \leq k$. Thus the following result follows.

Theorem 4. *q -COUPON COLORING problem parameterized by treewidth k , can be solved in time $\mathcal{O}(2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)})$.*

Now, we show that this result is essentially tight, assuming ETH. This follows from the following theorem.

Theorem 5. *Assuming ETH, there is no $2^{o(s \log s)} \cdot n^{\mathcal{O}(1)}$ time algorithm for q -COUPON COLORING where, s is the size of the feedback vertex set of the input graph.*

Proof. We give a reduction from the $2k \times 2k$ BIPARTITE PERMUTATION INDEPENDENT SET problem to the q -COUPON COLORING problem.

$2k \times 2k$ BIPARTITE PERMUTATION INDEPENDENT SET

Input: A graph G with vertex set $[2k] \times [2k]$ where every edge connects vertices between $I_1 = \{(i, j) \mid i, j \leq k\}$ and $I_2 = \{(i, j) \mid i, j \geq k+1\}$.

Parameter: k

Question: Does there exist an independent set $\{(1, \rho(1)), \dots, (2k, \rho(2k))\} \subseteq I_1 \cup I_2$ in G for some permutation ρ of $[2k]$?

The following result is known. \square

Proposition 2. [26] Assuming ETH, there is no $2^{o(k \log k)}$ time algorithm for the $2k \times 2k$ BIPARTITE PERMUTATION INDEPENDENT SET.

Consider an instance (H, k) of the $2k \times 2k$ BIPARTITE PERMUTATION INDEPENDENT SET problem. We construct a graph G as follows.

We add four cliques: C_a , C_b , C'_a , and C'_b , each with $2k$ vertices. The vertex set of each C_x is $\{x_1, \dots, x_{2k}\}$, while the vertex set of each C'_x is $\{x'_1, x'_2, \dots, x'_{2k}\}$, for $x \in \{a, b\}$. Further, for every $i \in [2k]$, we add an edge between x_i and x'_i for $x \in \{a, b\}$. For every $i, j, x, y \in [2k]$, where $i \neq j$ and $x \neq y$, if the vertices (i, x) and (j, y) are adjacent in H , we introduce two new vertices: w_{xy}^{ij} and $w_{xy}^{ij'}$. We then add edges between w_{xy}^{ij} and every vertex in the set $\{a_1, \dots, a_{2k}\} \setminus \{a_x, a_y\}$ and between $w_{xy}^{ij'}$ and every vertex in the set $\{b_1, \dots, b_{2k}\} \setminus \{b_i, b_j\}$. Similarly, we add edges between $w_{xy}^{ij'}$ and every vertex in the set $\{a_1, \dots, a_{2k}\} \setminus \{a_x, a_y\}$ and between $w_{xy}^{ij'}$ and every vertex in the set $\{b_1, \dots, b_{2k}\} \setminus \{b_i, b_j\}$. Additionally, we add an edge between w_{xy}^{ij} and $w_{xy}^{ij'}$. (See Fig. 1). We refer to the set that contains all vertices w_{xy}^{ij} as W , and the set that contains all vertices $w_{xy}^{ij'}$ as W' . This completes the construction of the graph G . It is easy to see that the vertices of C_a , C_b , C'_a , and C'_b together form a feedback vertex set for G of size $8k$. We now show that H has a bipartite permutation independent set if and only if G admits a $2k$ -coupon coloring. Assume H has a bipartite permutation independent set, I . Let the vertices in I be represented as $(1, \rho(1)), \dots, (2k, \rho(2k))$. We now give a $2k$ -coupon coloring for G . For each $i \in [2k]$, we color the vertices $a_i, b_{\rho(i)}, a'_i$ and $b'_{\rho(i)}$ with the same color c_i . At this point, every vertex in C_a , C_b , C'_a , and C'_b has all $2k$ colors in its open neighborhood. We refer to the partial coloring obtained thus far as χ . We now extend the coloring χ to ensure that all vertices in W and W' have all $2k$ colors in its open neighborhood. Consider any vertex w_{xy}^{ij} in W , at least one of the following conditions must be true: $\chi(a_x) \neq \chi(b_i)$ or $\chi(a_y) \neq \chi(b_j)$. Assume, for contradiction, that $\chi(a_x) = \chi(b_i)$ and $\chi(a_y) = \chi(b_j)$. This implies that $i = \rho(x)$ and $j = \rho(y)$, that implies that the vertices (i, x) and (j, y) are part of the independent set I . This is a contradiction since the construction implies they are adjacent.

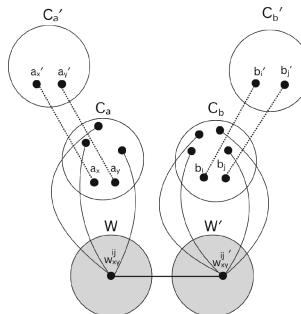


Fig. 1. The sub-graph corresponding to the edge between (i, x) and (j, y)

Thus, it follows that the neighbors of w_{xy}^{ij} in the cliques C_a and C_b are colored using at least $2k - 1$ distinct colors. The color that is possibly missing from the open neighborhood of w_{xy}^{ij} under χ can be assigned to w_{xy}^{ij}' , thus making sure that the open neighborhood of w_{xy}^{ij} has all the $2k$ colors. Similar arguments hold for vertices in W' also. Thus G admits a $2k$ -coupon coloring. The other direction follows by symmetric arguments.

The result now follows from Proposition 2.

Based on the above discussion and the fact that treewidth of G is at most one more than the feedback vertex set of G . we derive the following corollary.

Corollary 1. *Assuming ETH, there is no $2^{o(k \log k)} \cdot n^{\mathcal{O}(1)}$ time algorithm for q -COUPON COLORING, where k is the treewidth of the input graph.*

Acknowledgement. This work has been supported by the Anusandhan National Research Foundation under grant number MTR/2023/001078.

References

1. Abbas, W., Egerstedt, M., Liu, C.-H., Thomas, R., Whalen, P.: Deploying robots with two sensors in $k_{1,6}$ -free graphs. *J. Graph Theory* **82**(3), 236–252 (2016). <https://doi.org/10.1002/jgt.21898>
2. Akbari, S., Azimian, M., Fazli Khani, A., Samimi, B., Zahiri, E.: 2-coupon coloring of cubic graphs containing 3-cycle or 4-cycle. *Discrete Appl. Math.* **351**, 105–110 (2024). <https://doi.org/10.1016/j.dam.2024.03.012>
3. Akbari, S., Motiei, M., Mozaffari, S., Yazdanbod, S.: Cubic graphs with total domatic number at least two. *Discussiones Mathematicae Graph Theory* **38**, 75–82 (2015)
4. Aram, H., Sheikholeslami, S.M., Volkmann, L.: On the total domatic number of regular graphs. *Trans. Comb.* **1**(1), 45–51 (2012). <https://doi.org/10.22108/toc.2012.760>. ISSN 2251–8657
5. Ashok, P., Bhargava, R., Gupta, N., Khalid, M., Yadav, D.: Structural parameterization for minimum conflict-free colouring. *Discret. Appl. Math.* **319**, 239–253 (2022). <https://doi.org/10.1016/j.dam.2021.12.026>
6. Calamoneri, T.: The l (h, k)-labelling problem: a survey and annotated bibliography. *Comput. J.* **49**(5), 585–608 (2006). <https://doi.org/10.1093/comjnl/bxl018>
7. Chen, B., Kim, J.H., Tait, M., Verstraete, J.: On coupon colorings of graphs. *Discrete Appl. Math.* **193**, 94–101 (2015). <https://doi.org/10.1016/j.dam.2015.04.026>
8. Chen, H., Jin, Z.: Coupon coloring of cographs. *Appl. Math. Comput.* **308**, 90–95 (2017). <https://doi.org/10.1016/j.amc.2017.03.023>
9. Cygan, M., Fomin, F.V., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer, Cham (2015). <https://doi.org/10.1007/978-3-319-21275-3>
10. Donner, Q.: On the number of list-colorings. *J. Graph Theory* **16**(3), 239–245 (1992). <https://doi.org/10.1002/jgt.3190160307>

11. Dunbar, J., et al.: Fall colorings of graphs. *J. Comb. Math. Comb. Comput.* **33**, 257–273 (2000)
12. Erdős, P.: On a combinatorial problem. *Nordisk Matematisk Tidskrift* **11**(1), 5–10 (1963). ISSN 00291412
13. Even, G., Lotker, Z., Ron, D., Smorodinsky, S.: Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks. *SIAM J. Comput.* **33**(1), 94–136 (2003)
14. Francis, P., Rajendraprasad, D.: On coupon coloring of cartesian product of some graphs. In: Mudgal, A., Subramanian, C.R. (eds.) CALDAM 2021. LNCS, vol. 12601, pp. 309–316. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-67899-9_25
15. Ganian, R.: Improving vertex cover as a graph parameter. *Discrete Math. Theor. Comput. Sci.* **17**(Discrete Algorithms) (2015)
16. Goddard, W., Henning, M.A.: Thoroughly distributed colorings. arXiv preprint [arXiv:1609.09684](https://arxiv.org/pdf/1609.09684.pdf) (2016). [https://arxiv.org/pdf/1609.09684](https://arxiv.org/pdf/1609.09684.pdf)
17. Heggernes, P., Telle, J.A.: Partitioning graphs into generalized dominating sets. *Nordic J. Comput.* **5**(2), 128–142 (1998). ISSN 1236-6064
18. Henning, M.A., Yeo, A.: 2-colorings in k-regular k-uniform hypergraphs. *Eur. J. Combinatorics* **34**(7), 1192–1202 (2013)
19. Impagliazzo, R., Paturi, R.: On the complexity of k-sat. *J. Comput. Syst. Sci.* **62**(2), 367–375 (2001)
20. Irving, R.W., Manlove, D.F.: The b-chromatic number of a graph. *Discrete Appl. Math.* **91**(1–3), 127–141 (1999)
21. Jansen, K., Rohwedder, L.: On integer programming, discrepancy, and convolution. *Math. Oper. Res.* **48**(3), 1481–1495 (2023)
22. Koivisto, M., Laakkonen, P., Lauri, J.: Np-completeness results for partitioning a graph into total dominating sets. *Theoret. Comput. Sci.* **818**, 22–31 (2020)
23. Koutecký, M.: A note on coloring $(4k_1, c_4, c_6)$ -free graphs with a c_7 . *38*(5) (2022). <https://doi.org/10.1007/s00373-022-02553-4>. ISSN 0911-0119
24. Koutecký, M.: Solving hard problems on neighborhood diversity. Master's thesis (2013)
25. Lampis, M.: Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica* **64**, 19–37 (2012). <https://doi.org/10.1007/s00453-011-9554-x>
26. Lokshtanov, D., Marx, D., Saurabh, S.: Slightly superexponential parameterized problems. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 760–776. SIAM (2011). <https://doi.org/10.1137/16M1104834>
27. Östergård, P.R.J.: On a hypercube coloring problem. *J. Comb. Theory, Ser. A* **108**(2), 199–204 (2004). <https://doi.org/10.1016/j.jcta.2004.06.010>
28. Shadravan, M., Borzooei, R.A.: Coupon coloring of kneser graph k (n, 2). *Discrete Math. Algorithms Appl.* **16**(03), 2350020 (2024)
29. Shi, Y., Wei, M., Yue, J., Zhao, Y.: Coupon coloring of some special graphs. *J. Comb. Optim.* **33**(1), 156–164 (2015). <https://doi.org/10.1007/s10878-015-9942-2>
30. Telle, J.A., Proskurowski, A.: Algorithms for vertex partitioning problems on partial k-trees. *SIAM J. Discret. Math.* **10**(4), 529–550 (1997). <https://doi.org/10.1137/S0895480194275825>
31. Thankachan, R., Rajamani, P.: On coupon coloring of Cayley graphs. In: Bagchi, A., Muthu, R. (eds.) CALDAM 2023. LNCS, vol. 13947, pp. 184–191. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-25211-2_14

32. Kostochka, A.V., Woodall, D.R.: Density conditions for panchromatic colourings of hypergraphs. *Combinatorica* **21**(4), 515–541 (2001). <https://doi.org/10.1007/s004930100011>
33. Zavlanos, M.M.: Distributed control of robotic networks. PhD thesis, University of Pennsylvania (2008)
34. Zelinka, B.: Total domatic number and degrees of vertices of a graph. *Math. Slovaca* **39**(1), 7–11 (1989)



Spectra of Eccentricity Matrices of Product of Graphs

S. Balamoorthy^(✉) and T. Kavaskar

Central University of Tamil Nadu, Thiruvarur 610 005, India
moorthybala545@gmail.com, t_kavaskar@yahoo.com

Abstract. In this paper, we obtain the eccentricity spectrum and irreducibility of eccentricity matrix of the generalized friendship graph which is a generalization of the results in [12]. Also, we study the irreducibility of eccentricity matrix the H -join of graphs. As a consequence, we deduce a result in [12]. Further, we characterize graphs for which the eccentricity matrix of the generalized corona product of them is irreducible. As a result, we deduce a result in [11].

Keywords: Eccentricity matrix · Spectra of graphs · Energy of graphs · Spectral radius

1 Introduction

We consider only finite, simple, and undirected graph. Let $G = (V(G), E(G))$ be a graph. For any $v \in V(G)$, $\deg_G(v)$ denotes the degree of v and $\Delta(G)$ denotes the maximum degree of G . For a subset U of $V(G)$, $\langle U \rangle$ denotes the subgraph of G induced by U . A graph G is k -regular (where k is a non-negative integer) if $\deg_G(v) = k$, for all $v \in V(G)$. For a graph G , the complement of G , denoted by \overline{G} , is the graph with $V(\overline{G}) = V(G)$ and $E(\overline{G}) = \{uv : uv \notin E(G)\}$.

Let H be a graph with $V(H) = \{1, 2, \dots, k\}$ and let $\mathcal{F} = \{G_1, G_2, \dots, G_k\}$ be a family of graphs. The H -join operation of the graphs G_1, G_2, \dots, G_k , denoted by $H[G_1, G_2, \dots, G_k]$, is obtained by replacing the vertex i of H by the graph G_i for $1 \leq i \leq k$ and every vertex of G_i is made adjacent with every vertex of G_j , whenever i is adjacent to j in H , that is, the vertex

$$\text{set } V(H[G_1, G_2, \dots, G_k]) = \bigcup_{\ell=1}^k V(G_\ell) \text{ and edge set } E(H[G_1, G_2, \dots, G_k]) = \left(\bigcup_{\ell=1}^k E(G_\ell) \right) \cup \left(\bigcup_{ij \in E(H)} \{uv : u \in V(G_i), v \in V(G_j)\} \right) \text{ (see [14]).}$$

If $G \cong G_i$, for $1 \leq i \leq k$, then $H[G_1, G_2, \dots, G_k] \cong H[G]$, the *lexicographic product* of H and G . If $H = K_2$, then $K_2[G_1, G_2]$ is the join, denoted by $G_1 \vee G_2$, of G_1 and G_2 . The authors studied the spectrum of eccentricity matrix of H -join of graphs in [4]. Several results have been done on H -join of graphs, see [1–4] and [14].

Consider two disjoint graphs G_1 and G_2 with distinguished vertices v_1 and v_2 , respectively. The coalescence of two disjoint graphs $G_1 * G_2 = (G_1, v_1) * (G_2, v_2)$ is the graph obtained by gluing G_1 and G_2 at the distinguished vertices (see [12]).

In [8], H. Fernau et al. defined the *generalized friendship graph* as follows: For the positive integers $m \geq 3$, $n \geq 2$ the generalized friendship graph $f_{m,n}$ is a collection of n cycles (all of same order m), meeting at a common vertex. Note that, $f_{m,n} = \underbrace{C_m * \dots * C_m}_{n \text{ times}}$. If $m = 3$, then $f_{3,n}$ is the friendship graph.

Given graphs H, G_1, G_2, \dots, G_k , where $k = |V(H)|$, the generalized corona product denoted by $H \tilde{\diamond} \bigwedge_{i=1}^k G_i$, is the graph obtained by taking one copy of graphs H, G_1, G_2, \dots, G_k and joining the i^{th} vertex of H to every vertex of G_i . In particular, if $G_i \cong G$, for $1 \leq i \leq k$, the graph $H \tilde{\diamond} \bigwedge_{i=1}^k G_i$ is called simply corona of H and G , denoted by $H \circ G$ (see [14]).

For a connected graph G and $u, v \in V(G)$, the distance between u and v in G , denoted by $d_G(u, v)$, is the length of a shortest path between them in G and the eccentricity of u , denoted by $e_G(u)$, is defined as $e_G(u) = \max\{d_G(u, v) : v \in V(G)\}$. The radius of G , denoted by $\text{rad}(G)$, is the minimum eccentricity of all vertices of G . The diameter of G , denoted by $\text{diam}(G)$, is the maximum eccentricity of all vertices of G . A graph G is said to be self-centered if $\text{rad}(G) = \text{diam}(G)$. Let G be a graph with diameter d . If there exists a partition of the vertex set into classes (called the fibers of G) with the property that two distinct vertices are in the same class if and only if they are at distance d , G is called antipodal graph. If all the fibers have the same cardinality, say a , we say that G is an a -antipodal graph,

For a real symmetry $n \times n$ matrix B with real eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$, denote $\phi(\lambda, B) = \det(\lambda I - B)$ the characteristic polynomial of B and $\mathcal{E}(B) = \sum_{i=1}^n |\lambda_i|$ the energy of B . The spectral radius of B is defined as $\rho(B) = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } B\}$. If $\lambda_1, \lambda_2, \dots, \lambda_s$ are the only distinct eigenvalues of B , then the spectrum of B , denoted by $\text{Spec}(B)$, is defined as follows

$$\text{Spec}(B) = \left\{ \begin{matrix} \lambda_1 & \lambda_2 & \dots & \lambda_s \\ m_1 & m_2 & \dots & m_s \end{matrix} \right\},$$

where m_i is the algebraic multiplicity of the eigenvalue λ_i , for $1 \leq i \leq s$. Note that, if B is a real symmetric matrix, then all the eigenvalues of B are real. The inertia of B is characterized by the triplet of integers $(N_+(B), N_0(B), N_-(B))$, where $N_+(B)$, $N_0(B)$, and $N_-(B)$ denote the counts of positive, null, and negative eigenvalues of B , respectively. For a block matrix B_{ij} of B , denote $B_{ij}(r, s)$ the rs^{th} entry of B_{ij} .

For a graph G with vertex set $V(G) = \{v_1, v_2, \dots, v_n\}$, the adjacency matrix of G is $A(G) = (b_{ij})$, where

$$b_{ij} = \begin{cases} 1 & \text{if } v_i v_j \in E(G), \\ 0 & \text{otherwise.} \end{cases}$$

Randic proposed the eccentricity matrix [13] as DMax-matrix in 2013 and later it was renamed as Ecc-matrix by Wang et al. [15] in 2018. The eccen-

tricity matrix of a connected graph G with $V(G) = \{v_1, v_2, \dots, v_n\}$ is $\varepsilon(G) = (a_{ij})$, where

$$a_{ij} = \begin{cases} d_G(v_i, v_j) & \text{if } d_G(v_i, v_j) = \min\{e_G(v_i), e_G(v_j)\} \\ 0 & \text{otherwise.} \end{cases}$$

A non-negative square matrix M is said to be reducible if there exists a permutation matrix P such that $PMP^T = \begin{bmatrix} S & R \\ \mathbf{0} & N \end{bmatrix}$, where S and N are square matrices. If no such permutation matrix P exists, then M is said to be irreducible.

To check the irreducibility of a given non-negative square matrix is a challenging problem. We restrict our attention to the matrices obtained from the connected graphs. But for a connected graph, its adjacent and distance matrices are irreducible and it is not true for the case of eccentricity matrix of a connected graph, see for examples, $\varepsilon(C_n)$ is irreducible if and only if n is odd and the eccentricity matirx of tree with at least two vertices is irreducible as shown in [15]. In this direction, Wang et al., in [15] raised the following question.

Problem ([15]). Which connected graphs exhibit either reducible or irreducible eccentricity matrices?

In [17], Zhou et al., have proved that irreducibility of eccentricity matrix for r -uniform hypertrees. In [7], Divyadevi et al., have proved the irreducibility of eccentricity matrix of a subclass of bi-block graphs. In [11], Pandey et al., have characterized irreduciblity of eccentricity matrix of corona product of a graph and self-centered graph. This result, in fact, is an immediate consequence of our result, see Corollary 12 in Sect. 3.

Given an $n \times n$ non-negative symmetric matrix A , associate a graph $\Gamma(A)$ on n vertices, say $\{v_1, \dots, v_n\}$ such that there is an edge between the vertices v_i and v_j in $\Gamma(A)$ if and only if the $(i, j)^{\text{th}}$ entry of A is nonzero. An $n \times n$ symmetric non-negative matrix A is irreducible if and only if $\Gamma(A)$ is connected. The eccentric graph of a graph G , denoted by G^e , has the same set of vertices as G with two vertices x and y being adjacent in G^e if and only if $d_G(x, y) = \min\{e_G(x), e_G(y)\}$, that is $G^e = \Gamma(\varepsilon(G))$.

Wang et al. [16] introduced the concept of eccentricity energy, named as ε -energy, for a graph G with n vertices. This is formally defined as

$$\mathcal{E}(\varepsilon(G)) = \sum_{i=1}^n |\lambda_i|,$$

where $\lambda_1, \lambda_2, \dots, \lambda_n$ are the eigenvalues of $\varepsilon(G)$. Here, the characteristic polynomial, spectrum, energy, spectral radius and inertia of G represent, the characteristic polynomial, spectrum, energy, spectral radius and inertia, respectively of its eccentricity matrix $\varepsilon(G)$.

In [10], Mahato and Kannan examined the inertia of the eccentricity matrices of trees and characterized those trees that have exactly three distinct eccentricity

eigenvalues. Additionally, Divyadevi and Jeyaraman [7] investigated the eccentricity matrices of a subclass \mathcal{B} of bi-block graphs. They also proved that if the diameter of $G \in \mathcal{B}$ is greater than three, then the eigenvalues of the eccentricity matrix of G are symmetric with respect to the origin if and only if the diameter of G is odd. Further, they prove that the eccentricity matrices of graphs in \mathcal{B} are irreducible. There are several papers on the eccentricity energy and eccentricity eigenvalues of graphs. See [4, 7, 10–13, 15, 16].

We state the following results in [4–6, 9, 14] which will be used throughout this paper.

Lemma 1 ([4]). *Let H be a connected graph with $V(H) = \{1, 2, \dots, k\}$. Let G_1, G_2, \dots, G_k be a family of graphs with $|V(G_i)| = n_i$, for $i = 1, 2, \dots, k$ and $G = H[G_1, G_2, \dots, G_k]$. Then*

1. *If $i, j \in V(H)$ with $i \neq j$ and $x, y \in V(G)$ with $x \neq y$, then*

$$d_G(x, y) = \begin{cases} 2 & \text{if } x, y \in V(G_i) \text{ with } xy \notin E(G_i) \\ d_H(i, j) & \text{if } x \in V(G_i) \text{ and } y \in V(G_j). \end{cases}$$

2. *If $i \in V(H)$, then $e_H(i) \leq e_G(x)$, for all $x \in V(G_i)$.*

3. *If $e_H(i) \geq 2$, then $e_H(i) = e_G(x)$, for all $x \in V(G_i)$.*

4. *If $e_H(i) = 1$ and $x \in V(G_i)$, then*

$$e_G(x) = \begin{cases} 1 & \text{if } \deg_{G_i}(x) = n_i - 1, \\ 2 & \text{otherwise.} \end{cases}$$

Lemma 2 ([5]). *Let A, B, C and D be matrices such that $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$. If A is invertible, then $\det(M) = \det(A) \det(D - CA^{-1}B)$.*

Lemma 3 ([6]). *Let N_1, N_2, N_3, N_4 be $n \times n$ matrices. If N_1 is invertible and $N_1N_3 = N_3N_1$, then $\begin{vmatrix} N_1 & N_2 \\ N_3 & N_4 \end{vmatrix} = |N_1N_4 - N_3N_2|$.*

The paper is organized as follows.

In Sect. 2, we study the spectrum and irreducibility of $\varepsilon(f_{m,n})$, for all $m \geq 3$, $n \geq 2$. As a result, we deduce the results in [12].

In Sect. 3, we study the irreducibility of eccentricity of H -join of graphs. As a result, we deduce a result in [12] and we characterize graphs for which the eccentricity matrix of the generalized corona product of them is irreducible which is a generalization of a result in [11].

2 Spectrum of Generalized Friendship Graphs

In [12], the authors obtained the characteristic polynomial, spectrum, rank, inertia and irreducibility of eccentricity matrix of coalescence of two cycles with same length.

In this section, we generalize these results in much more general that is, we find the eccentric spectrum and irreducibility of generalized friendship graphs.

Theorem 1. For $m \geq 3$ and $n \geq 2$ and $s = \frac{14k^3 - 15k^2 + k}{6}$, where $k = \lfloor \frac{m}{2} \rfloor$,

1. If m is even, then

$$(i) \phi(\lambda, \varepsilon(f_{m,n})) =$$

$$\lambda^{n(m-3)+1} [\lambda^2 - 2k(n-1)\lambda - (2n^2 - 4n + 2)s - nk^2] (\lambda^2 + 2k\lambda - 2s)^{n-1}.$$

$$(ii) \text{Spec}(\varepsilon(f_{m,n})) = \left\{ \begin{matrix} k(n-1) + a & 0 & k(n-1) - a \\ 1 & n-1 & 1 \\ n(m-3) + 1 & n-1 & n-1 \end{matrix} \right\},$$

$$\text{where } a = \sqrt{\frac{k((14n^2 - 28n + 14)k^2 - (12n^2 - 30n + 18)k + (n^2 - 2n + 1))}{3}} \text{ and } b = \sqrt{\frac{k(14k^2 - 12k + 1)}{3}}.$$

2. If m is odd, then

$$(i) \phi(\lambda, \varepsilon(f_{m,n})) =$$

$$\lambda^{n(m-3)+1} [\lambda^2 - 4k(n-1)\lambda - (4n^2 - 8n + 4)s - 2nk^2] (\lambda^2 + 4k\lambda - 4s)^{n-1}.$$

$$(ii) \text{Spec}(\varepsilon(f_{m,n})) =$$

$$\left\{ \begin{matrix} 2k(n-1) + c & 0 & -2k + d & -2k - d \\ 1 & 1 & n(m-3) + 1 & n-1 & n-1 \end{matrix} \right\},$$

$$\text{where } c = \sqrt{\frac{2k((14n^2 - 28n + 14)k^2 - (9n^2 - 21n + 9)k + (n^2 - 2n + 1))}{3}} \text{ and } d = \sqrt{\frac{2k(14k^2 - 9k + 1)}{3}}.$$

3. $f_{m,n}$ is irreducible.

4. The rank of $\varepsilon(f_{m,n})$ is $2n$.

Proof. Let $V(f_{m,n}) = \{v\} \cup \{v_{ij} : 1 \leq i \leq n \text{ and } 1 \leq j \leq m-1\}$ and $E(f_{m,n}) = \bigcup_{i=1}^n \{vv_{i1}, vv_{im-1}, v_{ij}v_{ij+1} : 1 \leq j \leq m-2\}$. We now consider the following cases.

Case 1. $m = 2k$ ($k \geq 2$) is even.

If $i, \ell \in \{1, 2, \dots, n\}$ with $i \neq \ell$, then for $j \in \{1, 2, \dots, k-1\}$, we have $e_{f_{m,n}}(v_{ij}) = e_{f_{m,n}}(v_{im-j}) = d_{f_{m,n}}(v_{ij}, v_{\ell k}) = d_{f_{m,n}}(v_{im-j}, v_{\ell k}) = k+j$ and $e_{f_{m,n}}(v) = d_{f_{m,n}}(v, v_{ik}) = k$. Also, $e_{f_{m,n}}(v_{\ell k}) = d_{f_{m,n}}(v_{\ell k}, v_{ik}) = 2k$. We now label the vertices of $f_{m,n}$ as follows: $v, v_{11}, v_{21}, \dots, v_{n1}, v_{1m-1}, v_{2m-1}, \dots, v_{nm-1}, v_{12}, v_{22}, \dots, v_{n2}, v_{1m-2}, v_{2m-2}, \dots, v_{nm-2}, \dots, v_{1k}, v_{2k}, \dots, v_{nk}$. With this labeling of $f_{m,n}$, the eccentricity matrix of $f_{m,n}$ is given by

$$\varepsilon(f_{m,n}) = \begin{bmatrix} \mathbf{0}_{1 \times 1} & \mathbf{0}_{1 \times (m-2)n} & kJ_{1 \times n} \\ \mathbf{0}_{(m-2)n \times 1} & \mathbf{0}_{(m-2)n \times (m-2)n} & M \\ kJ_{n \times 1} & M^T & 2k(J-I)_{n \times n} \end{bmatrix}, \quad (1)$$

$$\text{where } M = \begin{bmatrix} J_{2 \times 1} \otimes (k+1) (J-I)_{n \times n} \\ J_{2 \times 1} \otimes (k+2) (J-I)_{n \times n} \\ \vdots \\ J_{2 \times 1} \otimes (k+(k-1)) (J-I)_{n \times n} \end{bmatrix}_{m-2 \times n}.$$

Then

$$\phi(\lambda, \varepsilon(f_{m,n})) = \begin{vmatrix} \lambda I_{1 \times 1} & \mathbf{0}_{1 \times (m-2)n} & -k J_{1 \times n} \\ \mathbf{0}_{(m-2)n \times 1} & \lambda I_{(m-2)n \times (m-2)n} & -M \\ -k J_{n \times 1} & -M^T & (\lambda I - 2k(J-I))_{n \times n} \end{vmatrix}.$$

Let $A = \lambda I_{1 \times 1}$, $B = -[\mathbf{0}_{1 \times (m-2)n} \ k J_{1 \times n}]$, $C = B^T$ and

$$D = \begin{bmatrix} \lambda I_{(m-2)n \times (m-2)n} & -M \\ -M^T & (\lambda I - 2k(J-I))_{n \times n} \end{bmatrix}.$$

Applying Lemma 2, we have

$$\phi(\lambda, \varepsilon(f_{m,n})) = \lambda \begin{vmatrix} \lambda I_{(m-2)n \times (m-2)n} & -M \\ -M^T & (\lambda I - 2k(J-I)) - \frac{k^2}{\lambda} J \end{vmatrix}.$$

Again applying Lemma 2, we have

$$\phi(\lambda, \varepsilon(f_{m,n})) = \lambda^{n(m-2)+1} \left| (\lambda I - 2k(J-I)) - \frac{k^2}{\lambda} J - \frac{1}{\lambda} M^T M \right|$$

As $M^T M = (2n-4)sJ + 2sI$, where $s = \sum_{i=1}^{k-1} (k+i)^2 = \frac{14k^3 - 15k^2 + k}{6}$, we have

$$\begin{aligned} \phi(\lambda, \varepsilon(f_{m,n})) &= \lambda^{n(m-2)+1} \left| \left(-\frac{k^2 + (2n-4)s + 2k\lambda}{\lambda} J + \frac{\lambda^2 + 2k\lambda - 2s}{\lambda} I \right) \right| \\ &= \lambda^{n(m-3)+1} \left| -(k^2 + (2n-4)s + 2k\lambda)J + (\lambda^2 + 2k\lambda - 2s)I \right| \\ &= \lambda^{n(m-3)+1} [(\lambda^2 + 2k\lambda - 2s)^{n-1} (-nk^2 - n(2n-4)s - n2k\lambda + \lambda^2 + 2k\lambda - 2s)] \end{aligned}$$

Thus,

$$\phi(\lambda, \varepsilon(f_{m,n})) = \lambda^{n(m-3)+1} [(\lambda^2 + 2k\lambda - 2s)^{n-1} (\lambda^2 - 2k(n-1)\lambda - (2n^2 - 4n + 2)s - nk^2)].$$

So, $\text{Spec}(\varepsilon(G)) = \left\{ \begin{array}{ccccc} k(n-1) + a & -k+b & 0 & k(n-1) - a & -k-b \\ 1 & n-1 & n(m-3)+1 & 1 & n-1 \end{array} \right\}$, where $a = \sqrt{\frac{k((14n^2 - 28n + 14)k^2 - (12n^2 - 30n + 18)k + (n^2 - 2n + 1))}{3}}$ and $b = \sqrt{\frac{k(14k^2 - 12k + 1)}{3}}$.

Next, we prove that $\varepsilon(f_{m,n})$ is irreducible. It is enough to prove that the graph $f_{m,n}^e$ is connected. From Eq. 1, we see that

(i) v is adjacent to v_{ik} in $f_{m,n}^e$, for $1 \leq i \leq n$,

(ii) v_{ik} is adjacent to $v_{j\ell}$ in $f_{m,n}^e$, for $1 \leq i \neq j \leq n$ and $1 \leq \ell \leq m$.

From (i) and (ii), we see that if $i, j \in \{1, 2, \dots, n\}$ with $i \neq j$ and $\ell, \ell' \in \{1, 2, \dots, m\} \setminus \{k\}$, then $v_{i\ell}$ is adjacent to v_{jk} , v_{jk} is adjacent to v , v is adjacent to v_{ik} is adjacent to $v_{j\ell'}$ in $f_{m,n}^e$ and there is a path in $f_{m,n}^e$ between $v_{i\ell}$ and $v_{j\ell'}$.

Also, we see that from (i) and (ii), there is a path between v and $v_{j\ell}$ in $f_{m,n}^e$. Hence $f_{m,n}^e$ is connected and thus $\varepsilon(f_{m,n})$ is irreducible.

Case 2. $m = 2k + 1$ ($k \geq 1$) is odd.

If $i, \ell \in \{1, 2, \dots, n\}$ with $i \neq \ell$, then for $j \in \{1, 2, \dots, k - 1\}$ and $g \in \{k, k + 1\}$, we have $e_{f_{m,n}}(v_{ij}) = e_{f_{m,n}}(v_{im-j}) = d_{f_{m,n}}(v_{ij}, v_{\ell g}) = d_{f_{m,n}}(v_{im-j}, v_{\ell g}) = k + j$ and $e_{f_{m,n}}(v) = d_{f_{m,n}}(v, v_{ig}) = k$. Also, $e_{f_{m,n}}(v_{\ell g}) = d_{f_{m,n}}(v_{\ell g}, v_{ig}) = 2k$. We label the vertices of $f_{m,n}$ as similar in Case 1 as follows: $v, v_{11}, v_{21}, \dots, v_{n1}, v_{1m-1}, v_{2m-1}, \dots, v_{nm-1}, v_{12}, v_{22}, \dots, v_{n2}, v_{1m-2}, v_{2m-2}, \dots, v_{nm-2}, \dots, v_{1k}, v_{2k}, \dots, v_{nk}, v_{1k+1}, v_{2k+1}, \dots, v_{nk+1}$. Then the eccentricity matrix of $f_{m,n}$ is given by

$$\varepsilon(f_{m,n}) = \begin{bmatrix} \mathbf{0}_{1 \times 1} & \mathbf{0}_{1 \times (m-3)n} & kJ_{1 \times 2n} \\ \mathbf{0}_{(m-3)n \times 1} & \mathbf{0}_{(m-3)n \times (m-3)n} & M \\ kJ_{2n \times 1} & M^T & J_{2 \times 2} \otimes 2k(J - I)_{n \times n} \end{bmatrix}, \quad (2)$$

$$\text{where } M = \begin{bmatrix} J_{2 \times 2} \otimes (k+1)(J - I)_{n \times n} \\ J_{2 \times 2} \otimes (k+2)(J - I)_{n \times n} \\ \vdots \\ J_{2 \times 2} \otimes (k+(k-1))(J - I)_{n \times n} \end{bmatrix}_{(m-3)n \times 2n}.$$

Then

$$\phi(\lambda, \varepsilon(f_{m,n})) = \begin{vmatrix} \lambda I_{1 \times 1} & -\mathbf{0}_{1 \times (m-3)n} & -kJ_{1 \times 2n} \\ -\mathbf{0}_{(m-3)n \times 1} & \lambda I_{(m-3)n \times (m-3)n} & -M \\ -kJ_{2n \times 1} & -M^T & \lambda I_{2n \times 2n} - (J_{2 \times 2} \otimes 2k(J - I)_{n \times n}) \end{vmatrix}.$$

As similar in the Case 1, let us take $A = \lambda I_{1 \times 1}$, $B = -[\mathbf{0}_{1 \times (m-3)n} \ kJ_{1 \times 2n}]$, $C = B^T$ and

$$D = \begin{bmatrix} \lambda I_{(m-3)n \times (m-3)n} & -M \\ -M^T & \lambda I_{2n \times 2n} - (J_{2 \times 2} \otimes 2k(J - I)_{n \times n}) \end{bmatrix}.$$

Applying Lemma 2, we have

$$\phi(\lambda, \varepsilon(f_{m,n})) = \lambda \begin{vmatrix} \lambda I_{(m-3)n \times (m-3)n} & -M \\ -M^T & \lambda I_{2n \times 2n} - (J_{2 \times 2} \otimes (2k(J - I) + \frac{k^2}{\lambda} J)_{n \times n}) \end{vmatrix}.$$

Again applying Lemma 2, we have

$$\phi(\lambda, \varepsilon(f_{m,n})) = \lambda^{n(m-3)+1} \left| \lambda I_{2n \times 2n} - (J_{2 \times 2} \otimes (2k(J - I) + \frac{k^2}{\lambda} J)_{n \times n}) - \frac{1}{\lambda} M^T M \right|,$$

as $M^T M = J_{2 \times 2} \otimes (2(n-2)s(J - I) + 2(n-1)sI)$, where $s = \sum_{i=1}^{k-1} (k+i)^2 = \frac{14k^3 - 15k^2 + k}{6}$.

Therefore

$$\begin{aligned} \phi(\lambda, \varepsilon(f_{m,n})) &= \lambda^{n(m-3)+1} \left| \lambda I - (J_{2 \times 2} \otimes (2k(J - I) + \frac{k^2}{\lambda} J)_{n \times n}) - \frac{1}{\lambda} M^T M \right|_{2n \times 2n} \\ &= \lambda^{n(m-3)+1} \begin{vmatrix} \lambda I - B & -B \\ -B & \lambda I - B \end{vmatrix}_{2n \times 2n}, \end{aligned}$$

where $B = B_{n \times n} = (2k(J - I) + \frac{k^2}{\lambda}J + \frac{1}{\lambda}(2(n-2)s(J - I) + 2(n-1)sI))$. Let $N_1 = \lambda I - B = N_4$ and $N_2 = N_3 = -B$.

Applying Lemma 3, we have

$$\begin{aligned}\phi(\lambda, \varepsilon(f_{m,n})) &= \lambda^{n(m-3)+1} |\lambda^2 I - 2B\lambda| \\ &= \lambda^{n(m-3)+1} \lambda^n |\lambda I - 2B| \\ &= \lambda^{n(m-3)+1} \lambda^n \left| \frac{-2(2k\lambda + k^2 + 2(n-2)s)}{\lambda} J + \frac{(\lambda^2 + 4k\lambda - 4s)}{\lambda} I \right| \\ &= \lambda^{n(m-3)+1} |-2(2k\lambda + k^2 + 2(n-2)s)J + (\lambda^2 + 4k\lambda - 4s)I|.\end{aligned}$$

Hence,

$$\phi(\lambda, \varepsilon(f_{m,n})) = \lambda^{n(m-3)+1} (\lambda^2 - 4k(n-1)\lambda - (4n^2 - 8n + 4)s - 2nk^2)(\lambda^2 + 4k\lambda - 4s)^{n-1}.$$

So, $\text{Spec}(\varepsilon(f_{m,n})) =$

$$\left\{ \begin{matrix} 2k(n-1) + c & 2k(n-1) - c & 0 & -2k + d & -2k - d \\ 1 & 1 & n(m-3) + 1 & n-1 & n-1 \end{matrix} \right\},$$

where $c = \sqrt{\frac{2k((14n^2 - 28n + 14)k^2 - (9n^2 - 21n + 9)k + (n^2 - 2n + 1))}{3}}$ and $d = \sqrt{\frac{2k(14k^2 - 9k + 1)}{3}}$.

The argument similar to case 1, we see that $\varepsilon(f_{m,n})$ is irreducible, using Eq. 2.

The proof of (4) follows from 1(ii) and 2(ii) and $\varepsilon(f_{m,n})$ is symmetric because the rank of the real symmetric matrix is equal to the number of its non-zero eigenvalues.

As an immediate consequence of Theorem 1, we deduce the following results in [12].

Corollary 1 (Lemma 4.1, [12]). *Let $C_m * C_m$ be the coalescence of two cycles of C_m .*

1. If $m = 2k(k \geq 2)$, then

$$\text{Spec}(\varepsilon(C_m * C_m)) = \left\{ \begin{matrix} k+1 & -k+b & 0 & k-a & -k-b \\ 1 & 1 & 4k-5 & 1 & 1 \end{matrix} \right\},$$

where $a = \sqrt{\frac{k(14k^2 - 6k + 1)}{3}}$ and $b = \sqrt{\frac{k(14k^2 - 12k + 1)}{3}}$.

2. If $m = 2k + 1(k \geq 1)$, then

$$\text{Spec}(\varepsilon(C_m * C_m)) = \left\{ \begin{matrix} 2k+c & -2k+d & 0 & 2k-c & -2k-d \\ 1 & 1 & 4k-3 & 1 & 1 \end{matrix} \right\},$$

where $c = \sqrt{\frac{2k(14k^2 - 3k + 1)}{3}}$ and $d = \sqrt{\frac{2k(14k^2 - 9k + 1)}{3}}$.

Proof. When $n = 2$ in both 1(ii) and 2(ii) in Theorem 1, we obtain the result.

In Theorem 1, take $n = 2$ we deduce the following results in [12].

Corollary 2 (Theorem 4.1, [12]). *The rank of $\varepsilon(C_m * C_m)$ is four, for all $m \geq 3$.*

Corollary 3 (Theorem 4.2, [12]). *Let m be an integer with $m \geq 3$.*

1. $\mathcal{E}(\varepsilon(C_m * C_m)) = 2\sqrt{\frac{k}{3}}(\sqrt{14k^2 - 6k + 1} + \sqrt{14k^2 - 12k + 1})$ if m is even;
2. $\mathcal{E}(\varepsilon(C_m * C_m)) = 2\sqrt{\frac{2k}{3}}(\sqrt{14k^2 - 3k + 1} + \sqrt{14k^2 - 9k + 1})$ if m is odd.

Corollary 4 (Corollary 4.2.1, [12]). *Let m be an integer with $m \geq 3$. Then*

1. $N_+(\varepsilon(C_m * C_m)) = 2$.
2. $N_0(\varepsilon(C_m * C_m)) = 2m - 5$.
3. $N_-(\varepsilon(C_m * C_m)) = 2$.

The following result are immediate consequence of Theorem 1.

Corollary 5. *For $n \geq 2$, $\varepsilon(f_{3,n})$ is irreducible.*

Corollary 6. *For $m \geq 3$, $\varepsilon(C_m * C_m)$ is irreducible.*

3 Irreducibility of the Eccentricity Matrix of H -join of graphs

In [4], the present authors have given complete structure of the eccentricity matrix of H -join graphs. Using this result, they could not find the characteristic polynomial, spectrum, inertia of eccentricity matrix of arbitrary H -join graphs, but they obtained these results for the H -join of some special families of graphs, see [4].

Let us now begin this section with the following observations.

Observation 1. *If G is a graph with $\text{rad}(G) = 1$, then G^e is connected and hence $\varepsilon(G)$ is irreducible.*

Observation 2. *If $\text{rad}(G) = \text{diam}(G) = 2$, then $\varepsilon(G)$ is irreducible if and only if \overline{G} is connected.*

Using these observations and Lemma 1, we prove the following result.

Theorem 2. *Let H be a connected graph with $V(H) = \{1, 2, \dots, k\}$ and $V_1 = \{i \in V(H) : \deg_H(i) = k - 1\}$ and $V_2 = V(H) \setminus V_1$. Let G_1, G_2, \dots, G_k be a collection of graphs and $G = H[G_1, G_2, \dots, G_k]$.*

1. *If $\text{rad}(H) = 1$, then G^e is connected if and only if there exists $i \in V_1$ such that $|V(G_i)| = 1$ or $\text{rad}(G_i) = 1$, in other words, $\Delta(G_i) = |V(G_i)| - 1$. In particular, $\varepsilon(G)$ is irreducible if and only if there exists $i \in V_1$ such that $|V(G_i)| = 1$ or $\Delta(G_i) = |V(G_i)| - 1$.*

2. If $\text{rad}(H) \geq 2$, then G^e is connected if and only if H^e is connected. In particular, $\varepsilon(G)$ is irreducible if and only if $\varepsilon(H)$ is irreducible.

The following results are consequence of the Theorem 2.

Corollary 7. *If G is a graph and H is a connected graph, then*

1. *If $\text{rad}(H) = 1$, then $\varepsilon(H[G])$ is irreducible if and only if $\Delta(G) = |V(G)| - 1$.*
2. *If $\text{rad}(H) \geq 2$, then $\varepsilon(H[G])$ is irreducible if and only if $\varepsilon(H)$ is irreducible.*

We recall the following relation on $V(G)$ (see [3]). For any two vertices $u, v \in V(G)$, define $u \sim_G v$ if and only if $N_G(u) = N_G(v)$. It is easy to see that, the relation \sim_G is an equivalence relation on $V(G)$. Let $[u]$ be the equivalence class which contains u and $S = \{\langle [u_1] \rangle, \langle [u_2] \rangle, \dots, \langle [u_k] \rangle\}$ be the set of all equivalence classes of this relation \sim_G . The reduced graph G_r of G is defined as follows. The *reduced graph* G_r of G is the graph with vertex set $V(G_r) = \{1, 2, \dots, k\}$ and two distinct vertices i and j are adjacent in G_r if and only if u_i and u_j are adjacent in G . Note that, G is a G_r -join of $\langle [u_1] \rangle, \langle [u_2] \rangle, \dots, \langle [u_k] \rangle$, that is, $G \cong G_r[\langle [u_1] \rangle, \langle [u_2] \rangle, \dots, \langle [u_k] \rangle]$, where $\langle [u] \rangle$ denote the subgraph induced by $[u]$ and each $[u_i]$ is an independent subset of G . Clearly, G_r is isomorphic to a subgraph of G induced by $\{u_1, u_2, \dots, u_k\}$. Note that, G is connected if and only if the reduced graph G_r of G is connected.

The next result gives a necessary and sufficient conditions for irreducibility of $\varepsilon(G)$ in terms of $\varepsilon(G_r)$.

Corollary 8. *If G is a connected graph, then*

1. *If $\text{rad}(G_r) = 1$, then $\varepsilon(G)$ is irreducible if and only if there exists $[u_i] \in V(G_r)$ such that $|\langle [u_i] \rangle| = 1$ and $d_{G_r}([u_i]) = |V(G_r)| - 1$.*
2. *If $\text{rad}(G_r) \geq 2$, then $\varepsilon(G)$ is irreducible if and only if $\varepsilon(G_r)$ is irreducible.*

As an immediate consequence of Corollary 8, we have

Corollary 9. $\varepsilon(K_{n_1, n_2, \dots, n_r})$ is irreducible if and only if $n_i = 1$, for some $i \in \{1, 2, \dots, r\}$.

As a consequence of Theorem 2, we deduce the following result in [12].

Corollary 10 (Theorem 3.1, [12]). *Let $G = K_{a_1} * K_{a_2} * \dots * K_{a_m}$ be a graph of order n obtained by coalescence of $m \geq 2$ disjoint complete graph of orders $a_1, a_2, \dots, a_m \geq 2$. Then $\varepsilon(G)$ is irreducible.*

Proof. Clearly, $G \cong K_{1,m}[K_1, K_{a_1-1}, K_{a_2-1}, \dots, K_{a_m-1}]$. Since $\text{rad}(K_{1,m}) = 1$, by Theorem 2 (i), the result follows.

Next, let us prove when the generalized corona product graph whose eccentricity matrix is irreducible. For this, we first prove the following lemma.

Lemma 4. *If G is a connected graph and $H = G \circ K_1$, then G^e is connected if and only if H^e is connected.*

One can see that, the generalized corona product of H with G_1, G_2, \dots, G_k can be realized as $(H \circ K_1)$ -join of $H_1, H_2, \dots, H_k, G_1, G_2, \dots, G_k$, where $H_i \cong K_1$ for $1 \leq i \leq k$, see [4] and [14]. Using this observation and by Theorem 2 and Lemma 4, we prove the following result.

Theorem 3. *Let H be a connected graph with vertex set $V(H) = \{v_1, v_2, \dots, v_k\}$ and $G = H \circ \wedge_{i=1}^k G_i$, where G_i 's are arbitrary k graphs. Then H^e is connected if and only if G^e is connected.*

As a consequence of Theorem 3, we have

Corollary 11. *If $G = H \circ \wedge_{i=1}^k G_i$, where G_i 's are arbitrary k graphs, then $\varepsilon(G)$ is irreducible if and only if $\varepsilon(H)$ is irreducible. In particular, $\varepsilon(H \circ G)$ is irreducible if and only if $\varepsilon(H)$ is irreducible.*

As an immediate consequence of Corollary 11, we deduce the following main result in [11]

Corollary 12 (Theorem 4.2, [11]). *Let G and H be two graphs, where H is a self-centered graph. Then $\varepsilon(H \circ G)$ is irreducible if and only if $\varepsilon(H)$ is irreducible.*

Conclusion and Scope

In this paper, we obtained the eccentricity spectrum of generalized friendship graph. Also, we characterize graphs for which the eccentricity matrix of the generalized corona product of them is irreducible.

Note that, the generalized friendship graph $f_{m,n}$ is the coalescence of n copies of C_m . In the proof of Theorem 1, we computed the characteristic polynomial of $\varepsilon(f_{m,n})$ using same size of block matrices. But this technic is not working in case of the coalescence of different cycles and it is difficult to find its characteristic polynomial $\varepsilon(C_{m_1} * C_{m_2} * \dots * C_{m_n})$. So, we post the problem.

Problem. Find the characteristic polynomial, spectrum, energy, spectral radius and inertia of $\varepsilon(C_{m_1} * C_{m_2} * \dots * C_{m_n})$, where m_i 's are positive integers such that $m_i \geq 3$ for $1 \leq i \leq n$.

Funding Information. The first author received support from the CSIR-UGC Junior Research Fellowship (UGC Ref. No.: 1085/(CSIR-UGC NET JUNE 2019)). The second author, support came from the National Board of Higher Mathematics (NBHM), Government of India, under grant No. 02011/50/2023 NBHM dated October 19, 2023.

References

1. Balamoorthy, S., Bharanedhar, S.V.: Group vertex magicness of H -join and generalized friendship graphs. Electron. J. Graph Theory Appl. **12**(2), 315–328 (2024)
2. Balamoorthy, S., Kavaskar, T., Vinothkumar, K.: Harary and hyper-Wiener indices of some graph operations. J. Inequal. Appl. **2024**, 34 (2024)

3. Balamoorthy, S., Kavaskar, T., Vinothkumar, K.: Wiener index of an ideal-based zero-divisor graph of commutative ring with unity. *AKCE Int. J. Graphs Comb.* **21**(2), 111–119 (2024)
4. Balamoorthy, S., Kavaskar T.: Spectra of eccentricity matrices of H-join of graphs. <https://doi.org/10.48550/arXiv.2410.13382>
5. Cvetković, D., Doob, M., Sachs, H.: *Spectra of Graphs: Theory and Application*. Johann Ambrosius Barth Verlag, Heidelberg-Leipzig (1995)
6. Cvetković, D., Doob, M., Sachs, H.: *Spectral of Graphs Theory and Applications*. Academic Press, New York (1980)
7. Divyadevi, T., Jeyaraman, J.: On the eccentricity matrices of certain bi-block graphs. *Bull. Malays. Math. Sci. Soc.* **47**, 92 (2024)
8. Fernaua, H., Ryanb, J.F., Sugeng, K.A.: A sum labelling for the generalised friendship graph. *Discrete Math.* **308**, 734–740 (2008)
9. Horn, R.A., Johnson, C.R.: *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, 1994, p. viii+607, Corrected reprint of the 1991 original
10. Mahato, I., Kannan, M.R.: On the eccentricity matrices of trees: Inertia and spectral symmetry. *Discrete Math.* **345**, 113067 (2022)
11. Pandey, S., Selvaganesh, L., Pervin, J.: Eccentricity matrix of corona of two graphs. *Discrete Appl. Math.* **359**, 354–363 (2024)
12. Patel, A.K., Selvaganesh, L., Pandey, S.K.: Energy and inertia of the eccentricity matrix of coalescence of graphs. *Discrete Math.* **344**, 112591 (2021)
13. Randić, M.: DMAX-matrix of dominant distances in a graph. *MATCH Commun. Math. Comput. Chem.* **70**, 221–238 (2013)
14. Saravanan, M., Murugan, S.P., Arunkumar, G.: A generalization of Fiedler's Lemma and the spectra of H-join of graphs. *Lin. Alg. Appl.* **625**, 20–43 (2021)
15. Wang, J., Lu, M., Belardo, F., Randić, M.: The anti-adjacency matrix of a graph: eccentricity matrix. *Discrete Appl. Math.* **251**, 299–309 (2018)
16. Wang, J., Lu, M., Belardo, F., Randić, M., Li, G.Z.: Graph energy based on the eccentricity matrix. *Discrete Math.* **342**, 2636–2646 (2019)
17. Zhou, J., Zhu, Z.: Some properties on eccentricity matrices of uniform hypertrees. *Discrete Appl. Math.* **349**, 86–95 (2024)



Almost Empty Monochromatic Polygons in Planar Point Sets

Bhaswar B. Bhattacharya¹, Sandip Das², Sk. Samim Islam²,
and Saumya Sen^{2(✉)}

¹ University of Pennsylvania, Philadelphia, USA

² Indian Statistical Institute, Kolkata, Kolkata, India
sauomyasen72@gmail.com

Abstract. For integers $k \geq 3$, $c \geq 2$, and $s \geq 0$, let $M_k^*(c, s)$ be the least integer such that any set of at least $M_k^*(c, s)$ points in the plane in general position, colored with c colors, contains a monochromatic k -gon (not necessarily convex) with at most s interior points. Denote by $\lambda_k^*(c)$ the least integer such that $M_k^*(c, \lambda_k^*(c)) < \infty$. It follows from results in [3, 5] that

$$\left\lfloor \frac{c-1}{2} \right\rfloor \leq \lambda_3^*(c) \leq c-3.$$

In this paper, we extend this result to $k \geq 4$. Specifically, we show that, for $c \geq 3$,

$$2 \left\lfloor \frac{c-1}{2} \right\rfloor \leq \lambda_4^*(c) \leq 2c-4.$$

Moreover, for $k \geq 5$ and $c \geq 2$, we show that

$$(k-2) \left\lfloor \frac{c-1}{2} \right\rfloor \leq \lambda_k^*(c) \leq (k-2)c - (k-1).$$

Keywords: Discrete geometry · Colored point sets · Empty polygons · Erdős-Szekeres theorem

1 Introduction

The celebrated Erdős-Szekeres theorem asserts that for every positive integer $m \geq 3$, there exists a smallest integer $ES(m)$, such that any set of at least $ES(m)$ points in the plane in general position (that is, no three on a line), contains m points which lie on the vertices of a convex polygon. In particular, Erdős and Szekeres established the following bounds [8, 9]:

$$2^{m-2} + 1 \leq ES(m) \leq \binom{2m-4}{m-2} + 1.$$

They also famously conjectured that the lower bound is sharp. After a series of improvements, in a major breakthrough Suk [22] almost resolved this conjecture by showing that

$$ES(m) = 2^{m+o(m)}.$$

A refinement was later obtained by Holmsen et al. [14]. The conjecture of Erdős and Szekeres has been also verified for a few small values of m . In particular, it is known that $ES(3) = 3$, $ES(4) = 5$ [16], $ES(5) = 9$ [16] and $ES(6) = 17$ [23].

In 1978, Erdős [7] asked the following stronger question: Does every sufficient large set of points in the plane in general position contains a k -hole, that is, k -points which lie on the vertices of a convex polygon whose interior contains no points of the set? Denote by $H(k)$, if it exists, the smallest integer such that every set of at least $H(k)$ points in the plane in general position contains a k -hole. Esther Klein proved $H(4) = 5$, and Harborth [12] proved $H(5) = 10$. A few years later, Horton [15] showed that it is possible to construct an arbitrarily large set of points without a 7-hole, which shows $H(k)$ does not exist for $k \geq 7$. The existence of $H(6)$ remained open for several years, until Gerken [10] and Nicolás [19] independently established the finiteness of $H(6)$ (see also [24]). In particular, Gerken [10] showed that $H(6) \leq ES(9)$, which was later improved to $H(6) \leq \max\{ES(8), 400\}$ by Koshelev [17]. For the lower bound, Overmars [21] constructed a set of 29 points without any 6-hole, which implies $H(6) \geq 30$. In a recent breakthrough, Heule and Scheucher [13] proved that $H(6) = 30$, finally resolving the 6-hole problem.

The study of *almost empty* polygons, that is, convex polygons with few interior points was initiated in [20]. Specifically, given integers $k \geq 3$ and $s \geq 0$, denote by $H(k, s)$ be the minimum integer such that any set of at least $H(k, s)$ points in the plane in general position contains a convex k -gon with at most s interior points. Obviously, $ES(k) \leq H(k, s) \leq H(k, 0) = H(k)$. Using a Horton set construction, Nyklová [20] showed that $H(k, s)$ does not exist when $s \leq 2^{(k+6)/4} - k - 3$. In another direction, Devillers et al. [6] considered colored variants of Erdős-Szekeres problem. In such problems, a set of points is partitioned into $r \geq 2$ color classes, and a polygon spanned by points in the set is called *monochromatic* if all of its vertices are of the same color. In this setting, Grima et al. [11] showed that any set of 9 points colored arbitrarily with 2 colors contains a monochromatic 3-hole. On the other hand, Devillers et al. [6] showed that there exists arbitrarily large 3-colored point sets with no monochromatic 3-hole and there exists arbitrarily large 2-colored point sets with no monochromatic 5-hole. Devillers et al. [6] also conjectured that any sufficiently large 2-colored point set contains a 4-hole. While this conjecture remains open, a relaxation which does not require the 4-hole to be convex has been established by Aichholzer et al. [2].

A few years ago, Basu et al. [3] combined the notions of almost empty polygons and monochromatic holes and initiated the study of almost empty monochromatic polygons. Formally, given integers $k \geq 3$, $c \geq 2$, and $s \geq 0$, define $M_k(c, s)$ to be the least integer such that any set of at least $M_k(c, s)$ points in the plane colored with c colors contains a monochromatic convex k -gon with at most s interior points. Moreover, denote by $\lambda_k(c)$ the least integer such that $M_k(c, \lambda_k(c)) < \infty$. Basu et al. [3] considered the case $k = 3$ and proved that, for any $c \geq 2$,

$$\left\lfloor \frac{c-1}{2} \right\rfloor \leq \lambda_3(c) \leq c-2.$$

For $c \geq 3$, the upper bound was improved to $c - 3$ by Cravioto et al. [5]. They also showed that $\lambda_4(c) \leq 2c - 3$, for $c \geq 2$ (see [5, Theorem 3]). Relaxing the convexity requirement, one can define $M_k^*(c, s)$ to be the least integer such that any set of at least $M_k^*(c, s)$ points in the plane colored with c colors contains a monochromatic k -gon (not necessarily convex) with at most s interior points. Further, denote by $\lambda_k^*(c)$ the least integer such that $M_k^*(c, \lambda_k^*(c)) < \infty$. Clearly, $\lambda_3(c) = \lambda_3^*(c)$. Liu and Zhang [18] proved that for $c \geq 3$,

$$2 \left\lfloor \frac{c-1}{2} \right\rfloor \leq \lambda_4^*(c) \leq 2c - 3.$$

Our first result in this paper is to improve the upper on $\lambda_4^*(c)$ by a factor of 1, when $c \geq 3$.

Theorem 1. *For $c \geq 3$, $\lambda_4^*(c) \leq 2c - 4$.*

The proof of Theorem 1 is given in Sect. 2. The proof relies on a result about quadrangulations of planar point sets and an iterative counting idea from [5] combined with an induction argument.

Remark 1. Although our proof of Theorem 1 requires $c \geq 3$, the result continues to hold for $c = 2$. This is because we know from [2] that any sufficiently large bichromatic point set contains an empty monochromatic quadrilateral (not necessarily convex), which implies $\lambda_4^*(2) = 0$. Further, the result in Theorem 1 for $c = 3$ also follows from [18, Theorem 3], where it is shown that $M_4^*(3, 2) \leq 120$.

To the best of our knowledge, properties of $\lambda_k^*(c)$, for $k \geq 5$, have not been explored before. In this paper, we take the first step in this direction.

Theorem 2. *For $k \geq 5$ and $c \geq 2$,*

$$(k-2) \left\lfloor \frac{c-1}{2} \right\rfloor \leq \lambda_k^*(c) \leq (k-2)c - (k-1). \quad (1)$$

The proof of Theorem 2 is given in Sect. 3. The upper bound relies on a result about k -angulations of planar point sets [1] and the lower bound uses a Horton set construction from [3].

2 Proof of Theorem 1

Let S be a set of points in the plane, with $|S| = n$, in general position, that is, no three points of the set S are collinear. We denote the convex hull of S by $CH(S)$. The boundary vertices of $CH(S)$ and the interior points of $CH(S)$ are denoted by $V(CH(S))$ and $I(CH(S))$, respectively. A *polygonal partition* $\mathcal{P}(S)$ of S is a partition of $CH(S)$ into simple polygons (faces) such that the vertex set of $\mathcal{P}(S)$ is S and each edge of $\mathcal{P}(S)$ which is not an edge of $CH(S)$ is common to exactly two faces. A polygonal partition $\mathcal{P}(S)$ is said to be a *quadrangulation* if every face of $\mathcal{P}(S)$ is a quadrilateral (non-necessarily convex). It is known that a point-set

S admits a quadrangulation if and only if $|V(CH(S))|$ is even (see [4, Theorem 3.1]). Moreover, if $|V(CH(S))|$ is odd, one can construct a polygonal partition where one face is a triangle and the remaining faces are quadrilaterals, which, with slight abuse of terminology, we will also refer to this as a quadrangulation. Note that the faces of a quadrangulation are empty quadrilaterals which are interior disjoint. This leads to the following result:

Lemma 1 (Aichholzer et al. [1]). *Suppose S is a set of points in the plane in general position, with $|S| = n$ and $|V(CH(S))| = h$. Then any quadrangulation of S contains at least $n - \lceil h/2 \rceil - 1$ interior disjoint empty quadrilaterals.*

Remark 2. Since $h \leq n$, the above lemma implies that any set S , with $|S| = n$, contains at least $\lfloor n/2 \rfloor - 1$ empty quadrilaterals.

We now proceed with the proof of Theorem 1. Towards this fix $c \geq 3$ and let R be an integer (depending on c) that will be chosen later. We will show that

$$M_4^*(c, 2c - 4) \leq cES(2R).$$

For this, consider a set S of points in the plane in general position, with $|S| = cES(2R)$, colored with c colors such that every monochromatic quadrilateral in S contains at least $2c - 3$ interior points. Then some color class contains at least $ES(2R)$ points. Hence, by the Erdős-Szekeres theorem S contains a monochromatic convex $2R$ -gon. Denote by T a monochromatic convex $2R$ -gon in S which has the smallest number of points in the interior. Without loss of generality suppose the color of T is 1. Define

$$T_1 = CH(T) \cap S_1,$$

where S_1 is the set of points with color 1. Note that $|V(CH(T_1))| = 2R$ and $|T_1| = 2R + m_1$, where m_1 is the number of points of color 1 inside T . Hence, by Lemma 1 there is a quadrangulation $\mathcal{Q}(T_1)$ of T_1 which contains at least $m_1 + R - 1$ interior disjoint quadrilaterals of color 1 which have no points of color 1 in their interiors. Moreover, by assumption, each of these quadrilaterals contain at least $(2c - 3)$ interior points (all of which have colors different than 1). Hence, $CH(T_1)$ has at least $(2c - 3)(m_1 + R - 1)$ interior points with color different from 1. Without loss of generality, suppose the points colored 2 have the largest size among the points in $CH(T_1)$ which have color different from 1. Denote this set of points by T_2 . Note that

$$t_2 := |T_2| \geq \left\lceil \frac{(2c - 3)(m_1 + R - 1)}{c - 1} \right\rceil. \quad (2)$$

By minimality, $|V(CH(T_2))| < 2R$. Hence, by Lemma 1 T_2 has a quadrangulation with at least $t_2 - R - 1$ interior disjoint quadrilaterals of color 2 which have no points of color 2 in their interiors. Hence, by assumption, there are at least $(2c - 3)(t_2 - R - 1)$ points with color different from 2 in the interior of $CH(T_2)$. Also, note that the number of points inside $CH(T_1)$ with color different

from 1 and 2 is at most $(c - 2)t_2$. Therefore, we have the following bound on the number of points of color 1 inside $CH(T_2)$:

$$\begin{aligned}
|CH(T_2) \cap S_1| &\geq (2c - 3)(t_2 - R - 1) - (c - 2)t_2 \\
&= (c - 1)t_2 - (2c - 3)(R + 1) \\
&\geq (c - 1) \left\lceil \frac{(2c - 3)(m_1 + R - 1)}{c - 1} \right\rceil - (2c - 3)(R + 1) \quad (\text{by (2)}), \\
&\geq (2c - 3)(m_1 + R - 1) - (2c - 3)(R + 1) \quad (\text{since } \lceil x \rceil \geq x) \\
&= (2c - 3)(m_1 - 2). \tag{3}
\end{aligned}$$

Now, we consider the following cases:

Case 1: Suppose $c \geq 4$. Then, if $m_1 \geq 3$, from (3), we have

$$|CH(T_2) \cap S_1| \geq 5(m_1 - 2) \geq m_1 + 1,$$

which is a contradiction, since $|CH(T_2) \cap S_1| \leq m_1$. Hence, suppose $m_1 \leq 2$. This means T is a $2R$ -gon with color 1 that has at most 2 points of color 1 in the interior. Then as before $CH(T_1)$ has $m_1 + R - 1$ interior disjoint quadrilaterals that have no points of color 1 in their interiors. Hence, $CH(T_1)$ has at least $(2c - 3)(m + R - 1)$ points with color other than 1. Now, assume $M_4^*(c - 1, 2c - 6) < \infty$ and choose

$$R \geq \frac{M_4^*(c - 1, 2c - 6)}{2c - 3} + 1.$$

Then $CH(T_1)$ has at least $M_4^*(c - 1, 2c - 6)$ points in the interior with color different from 1. Hence, by definition of M_4^* , there is a monochromatic quadrilateral with at most $2c - 6$ interior points with color different from 1 among the points inside $CH(T_1)$. Denote this monochromatic quadrilateral by W . Note that W can have at most 2 interior points of color 1, hence, the total number of points inside W is at most $2c - 4$, which is a contradiction. This implies, when $c \geq 4$,

$$M_4^*(c, 2c - 4) \leq cES \left(2 \left(\frac{M_4^*(c - 1, 2c - 6)}{2c - 3} + 1 \right) \right), \tag{4}$$

whenever $M_4^*(c - 1, 2c - 6) < \infty$. Note that if we assume $M_4^*(3, 2) < \infty$, this implies, (4) holds for $c = 4$. Hence, by induction (4) holds for all $c \geq 4$. The proof of $M_4^*(3, 2) < \infty$ is discussed in the next case.

Case 2: Suppose $c = 3$. In this case, we will directly argue that $M_4^*(3, 2) < \infty$. (This also follows from [18, Theorem 3], where it is shown that $M_4^*(3, 2) \leq 120$ (recall Remark 1). Here, we include a shorter, self-contained proof, but with a looser bound on $M_4^*(3, 2)$.) To this end, choose $R \geq \frac{2}{3}M_4^*(2, 1)$. (Note that $M_4^*(2, 1) < \infty$ by [2, Theorem 1]). Then from (3), we have

$$|CH(T_2) \cap S_1| \geq 3(m_1 - 2) \geq m_1 + 1,$$

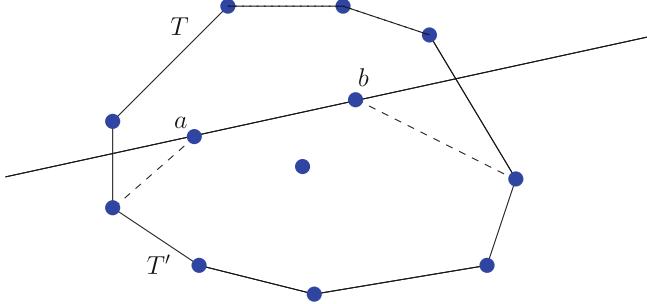


Fig. 1. A convex monochromatic $2R$ -gon of color 1 (blue) with 3 interior points of color 1.

whenever $m_1 \geq 4$, which is a contradiction. Hence, it suffices to assume that $m_1 \leq 3$. This means T is a convex $2R$ -gon of color 1 with at most 3 points of color 1 in the interior (see Fig. 1). Let $a, b \in I(CH(T)) \cap S_1$. The line passing through a and b splits the plane into 2 disjoint half planes. One of these half planes will contain at least R points of T . Hence, we can obtain a convex R -gon T' with color 1 which has at most 1 point of color 1 in the interior. Then applying Lemma 2 there are at least $\frac{R}{2}$ area disjoint quadrilaterals of color 1 inside $CH(T')$ which do not contain any point of color 1 in the interior. If any one of these quadrilaterals contain at most 2 interior points, then we are done. Otherwise, $CH(T')$ contains at least $\frac{3}{2}R$ points of color 2 or 3. Since $R \geq \frac{2}{3}M_4^*(2, 1)$, among the points of colors 2 and 3 inside $CH(T')$ there is a monochromatic quadrilateral U which has at most 1 interior point (of color 2 or 3). Note that U can have at most one interior point of color 1, hence, the number of points inside U is at most 2. This shows that

$$M_4^*(3, 2) \leq 3ES\left(\frac{4}{3}M_4^*(2, 1)\right).$$

This completes the proof of Theorem 1. □

3 Proof of Theorem 2

The proof of the upper bound in (1) is given in Sect. 3.1. The proof of the lower bound is given in Sect. 3.2.

3.1 Proof of the Upper Bound

Let S be a set of points in the plane, with $|S| = n$, in general position. Fix $k \geq 3$. A polygonal partition $\mathcal{P}(S)$ of S is said to be a k -angulation if every face of $\mathcal{P}(S)$ is a k -gon (non-necessarily convex). Counting the number of faces in a k -angulation gives the following result:

Lemma 2 (Aichholzer et al. [1]). Suppose S is a set of points in the plane in general position, with $|S| = n$ and $|V(CH(S))| = h$. Then S contains $\lfloor \frac{2n-h-2}{k-2} \rfloor$ interior disjoint empty k -gons.

Remark 3. Since $h \leq n$, the above lemma implies that any set S , with $|S| = n$, contains at least $\lfloor \frac{n-2}{k-2} \rfloor$ interior disjoint empty k -gons.

Note that for the upper bound in Theorem 2 we will show that, for $k \geq 5$ and $c \geq 2$,

$$M_k^*(c, (k-2)c - (k-1)) \leq cES(R),$$

for some R (depending on c and k) that will be chosen later. For this consider a set S of points in the plane in general position, with $|S| = cES(R)$, colored with c colors such that every monochromatic k -gon in S contains at least $(k-2)(c-1)$ interior points. Then some color class contains at least $ES(R)$ points. Hence, by the Erdős-Szekeres theorem S contains a monochromatic convex R -gon. Denote by T a monochromatic convex R -gon in S which has the smallest number of points in the interior. Without loss of generality, suppose the color of T is 1. Define

$$T_1 = CH(T) \cap S_1,$$

where S_1 is the set of points with color 1. Note that $|V(CH(T_1))| = R$ and $|T_1| = R + m_1$, where m_1 is the number of points of color 1 inside T . Hence, by Remark 3 there is a k -angulation $\mathcal{P}(T_1)$ of T_1 which contains at least

$$\left\lfloor \frac{(m_1 + R) - 2}{k - 2} \right\rfloor$$

interior disjoint k -gons with no point of color 1 in their interiors. Note, by assumption, each of these k -gons contain at least $(k-2)(c-1)$ interior points. Hence, $CH(T_1)$ has at least

$$(k-2)(c-1) \left\lfloor \frac{(m_1 + R) - 2}{k - 2} \right\rfloor \geq (k-2)(c-1) \frac{(m_1 + R - k)}{k - 2}.$$

interior points with color different from 1. Without loss of generality, suppose the points colored 2 have the largest size among the points in $CH(T_1)$ which have color from different 1. Denote this set of points by T_2 . Note that

$$t_2 := |T_2| \geq \left\lceil \frac{(k-2)(c-1) \frac{(m_1 + R - k)}{k-2}}{c-1} \right\rceil = m_1 + R - k. \quad (5)$$

By Remark 3, T_2 has a k -angulation with at least

$$\left\lfloor \frac{t_2 - 2}{k - 2} \right\rfloor$$

interior disjoint k -gons of color 2. Hence, $CH(T_1)$ contains at least

$$(k-2)(c-1) \left\lfloor \frac{t_2 - 2}{k - 2} \right\rfloor \geq (k-2)(c-1) \frac{t_2 - k}{k - 2}$$

points of color other than 2. Also, note that the number of points of color different from 1 or 2 inside $CH(T_1)$ is at most $(c - 2)t_2$. Therefore, the number of points of color 1 inside $CH(T_2)$ satisfies the following:

$$\begin{aligned}
 & |CH(T_2) \cap S_1| \\
 & \geq (k - 2)(c - 1) \frac{t_2 - k}{k - 2} - (c - 2)t_2 \\
 & \geq ((k - 2)(c - 1)) \frac{(m_1 + R - 2k)}{k - 2} - (c - 2)(m_1 + R - k) \text{ (by (5))} \\
 & = m_1 + R - kc.
 \end{aligned} \tag{6}$$

Note that if $R \geq kc + 1$, then from (6),

$$|CH(T_2) \cap S_1| \geq m_1 + R - kc \geq m_1 + 1,$$

which is a contradiction, since $|CH(T_2) \cap S_1| \leq m_1$. Hence, we have showed that

$$M_k^*(c, (k - 2)c - (k - 1)) \leq cES(kc + 1),$$

which implies $\lambda_4^*(c) \leq (k - 2)c - (k - 1)$, thus proving the upper bound. \square

3.2 Proof of the Lower Bound

In this section we prove the lower bound in Theorem 2 by appropriately coloring a Horton set with c colors. A *Horton set* is a set H of n points sorted by x -coordinates: $h_1 <_x h_2 <_x h_3 <_x \dots <_x h_n$, such that the set $H^+ := \{h_2, h_4, \dots\}$ of the even points and the set $H^- := \{h_1, h_3, \dots\}$ of the odd points are Horton sets and any line through two even points (the *upper set*) leaves all odd points below and any line through two odd points (the *lower set*) leaves all even points above. A Horton set of size n can be recursively obtained by adding a large vertical separation after intertwining in the x -direction an upper Horton set H^+ of size $\lfloor n/2 \rfloor$ and a lower set H^- of size $\lceil n/2 \rceil$ [15].

To begin with, suppose $c = 2q + 1$ be an odd number. The c colors are indexed by $\{1, 2, \dots, c\}$. Consider a Horton set H of size n and arrange the points h_1, h_2, \dots, h_n of H in increasing order of x -coordinates. Color H with these c colors as follows:

- The points $h_1, h_{2q+2}, h_{4q+3}, \dots$ are colored by color 1.
- The points $h_2, h_{2q+3}, h_{4q+4}, \dots$ are colored by color 2.
- In general, the points $h_i, h_{2q+i+1}, h_{4q+i+2}, \dots$ by color i , for $i \in \{1, 2, \dots, c\}$.

As $c = 2q + 1$ is odd, this coloring splits in a similar pattern in H^+ and H^- , that is, H^- is colored in cyclical manner as $1, 3, 5, \dots, 2q+1, 2, 4, \dots, 2q, \dots$ and so on, and H^+ is colored in cyclical manner as $2, 4, \dots, 2q, 1, 3, 5, \dots, 2q+1, \dots$ and so on. If $c = 2q + 2$ is an even number, and a Horton set H of size n is given, then color the rightmost point of H with color $2q + 2$. The remaining $n - 1$ points are colored with $c - 1$ colors as before.

Hereafter, we assume that c is odd. The proof for the case c is even can be done similarly. Let $\Delta = \{h_{i_1}, h_{i_2}, \dots, h_{i_k}\}$ be a monochromatic k -gon in a Horton set H , colored with $2q + 1$ colors as above. Consider a triangulation of Δ which decomposes Δ into $k - 2$ interior disjoint monochromatic triangles. The proof of the lower bound in [3, Theorem 1.2] shows that for the coloring scheme described above, any monochromatic triangle in H contains at least q interior points. Hence, Δ contains at least $(k - 2)q$ interior points. This shows $M_k^*(2q + 1, (k - 2)q - 1) = \infty$, which means,

$$\lambda_k^*(c) \geq (k - 2)q \geq (k - 2) \left\lfloor \frac{c - 1}{2} \right\rfloor.$$

This completes the proof of the lower bound in Theorem 2. \square

4 Conclusions

In this paper, we study the existence of monochromatic empty polygons with few interior points in planar point sets. We prove that any large enough c -colored point set contains a monochromatic quadrilateral with at most $2c - 4$ interior points, for $c \geq 3$. More generally, for $k \geq 5$ and $c \geq 2$ we show that any large enough c -colored point set contains a monochromatic k -gon with at most $(k - 2)c - (k - 1)$ interior points. Using a Horton set construction, we also show that for any $c \geq 2$, there exist arbitrarily large c -colored point sets in which every monochromatic k -gon contains at least $(k - 2) \lfloor \frac{c-1}{2} \rfloor$ interior points. Improving the bounds on $\lambda_k^*(c)$ would be an interesting future direction.

References

1. Aichholzer, O., et al.: Matching edges and faces in polygonal partitions. *Comput. Geom.* **39**(2), 134–141 (2008)
2. Aichholzer, O., Hackl, T., Huemer, C., Hurtado, F., Vogtenhuber, B.: Large bichromatic point sets admit empty monochromatic 4-gons. *SIAM J. Discret. Math.* **23**(4), 2147–2155 (2010)
3. Basu, D., Basu, K., Bhattacharya, B.B., Das, S.: Almost empty monochromatic triangles in planar point sets. *Discrete Appl. Math.* **210**, 207–213 (2016)
4. Bose, P., Toussaint, G.: Characterizing and efficiently computing quadrangulations of planar point sets. *Comput. Aided Geom. Des.* **14**(8), 763–785 (1997)
5. Cravio-Lagos, J., González-Martínez, A.C., Sakai, T., Urrutia, J.: On almost empty monochromatic triangles and convex quadrilaterals in colored point sets. *Graphs Combinatorics* **35**(6), 1475–1493 (2019)
6. Devillers, O., Hurtado, F., Károlyi, G., Seara, C.: Chromatic variants of the Erdős-Szekeres theorem on points in convex position. *Comput. Geom.* **26**(3), 193–208 (2003)
7. Erdős, P.: Some more problems on elementary geometry. *Austral. Math. Soc. Gaz.* **5**(2), 52–54 (1978)
8. Erdős, P., Szekeres, G.: A combinatorial problem in geometry. *Compos. Math.* **2**, 463–470 (1935)

9. Erdős, P., Szekeres, G.: On some extremum problems in elementary geometry. *Ann. Univ. Sci. Budapest. Eötvös Sect. Math.*, 3(4), 53–62 (1960)
10. Gerken, T.: Empty convex hexagons in planar point sets. *Discrete Comput. Geom.* **39**, 239–272 (2008)
11. Grima, C., del Carmen Hernando Martín, M., Huemer, C., Hurtado Díaz, F.A.: On some partitioning problems for two-colored point sets. *XIII Encuentros de Geometría Computacional*, 221–228 (2009)
12. Harborth, H.: Konvexe fünfecke in ebenen punktmengen. *Elem. Math.* **33**, 116–118 (1978)
13. Heule, M.J.H., Scheucher, M.: Happy ending: an empty hexagon in every set of 30 points. In: Finkbeiner, B., Kovács, L. (eds.) TACAS 2024. LNCS, vol. 14570, pp. 61–80. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-57246-3_5
14. Holmsen, A.F., Mojarrad, H.N., Pach, J., Tardos, G.: Two extensions of the Erdős–Szekeres problem. *J. Eur. Math. Soc.* **22**(12) (2020)
15. Horton, J.D.: Sets with no empty convex 7-Gons. *Can. Math. Bull.* **26**(4), 482–484 (1983)
16. Kalbfleisch, J.D.: A combinatorial problem on convex regions. In: 1970 Proceedings of Louisiana Conference on Combinatorics, Graph Theory and Computing, pp. 180–188 (1970)
17. Koshelev, V.A.: On the Erdős-Szekeres problem. In *Doklady Mathematics*, vol. 76, pp. 603–605. Springer (2007)
18. Liu, L., Zhang, Y.: Almost empty monochromatic quadrilaterals in planar point sets. *Math. Notes* **103**, 415–429 (2018)
19. Nicolás, C.M.: The empty hexagon theorem. *Discrete Comput. Geom.* **38**, 389–397 (2007)
20. Nyklová, H.: Almost empty polygons. *Stud. Sci. Math. Hung.* **40**(3), 269–286 (2003)
21. Overmars, M.: Finding sets of points without empty convex 6-Gons. *Discrete Comput. Geom.* **29**, 153–158 (2002)
22. Suk, A.: On the Erdős-Szekeres convex polygon problem. *J. Am. Math. Soc.* **30**(4), 1047–1053 (2017)
23. Szekeres, G., Peters, L.: Computer solution to the 17-point erdős-szekeres problem. *ANZIAM J.* **48**(2), 151–164 (2006)
24. Valtr, P.: On empty hexagons. *Contemp. Math.* **453**, 433–442 (2008)



On the Parameterized Complexity of Odd Coloring

Sriram Bhyravarapu^{1(✉)}, Swati Kumari², and I. Vinod Reddy²

¹ The Institute of Mathematical Sciences, HBNI, Chennai, India
`sriramb@imsc.res.in`

² Department of Computer Science and Engineering, IIT Bhilai, Bhilai, India
`{swatik,vinod}@iitbhilai.ac.in`

Abstract. A proper vertex coloring of a connected graph G is called an odd coloring if, for every vertex v of G there is a color that appears odd number of times in the open neighborhood of v . The minimum number of colors required to obtain an odd coloring of G is called the *odd chromatic number* of G and it is denoted by $\chi_o(G)$. The problem of determining $\chi_o(G)$ is NP-hard. Given a graph G and an integer k , the ODD COLORING problem is to decide whether $\chi_o(G)$ is at most k . In this paper, we study the problem from the viewpoint of parameterized complexity. In particular, we study the problem with respect to structural graph parameters. We prove that the problem admits a polynomial kernel when parameterized by distance to clique. On the other hand, we show that the problem cannot have a polynomial kernel when parameterized by vertex cover number unless $\text{NP} \subseteq \text{Co-NP/poly}$. We show that the problem is fixed-parameter tractable when parameterized by distance to cluster, distance to co-cluster, or neighborhood diversity. We show that the problem is W[1]-hard parameterized by clique-width. Finally, we study the problem on restricted graph classes. We show that the problem can be solved in polynomial time on cographs and split graphs.

1 Introduction

All graphs considered in this paper are finite, connected, undirected and simple. For a graph $G = (V, E)$, the vertex set and edge set of G are denoted by $V(G)$ and $E(G)$ respectively. For a positive integer k , a k -coloring of a graph G is a mapping $f : V(G) \rightarrow \{1, 2, \dots, k\}$. A k -coloring of a graph G is called a *proper k -coloring*, if $f(u) \neq f(v)$ for every edge uv of G . The chromatic number $\chi(G)$ of a graph G is the minimum k for which there is a proper k -coloring of G . A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is a generalization of a graph, where hyper-edges are subsets of \mathcal{V} of arbitrary positive size. Several notions of vertex coloring have been studied on hypergraphs [20, 21]. One such coloring is conflict-free (CF) coloring of hypergraphs introduced by Even et al. [14] in geometric setting motivated by its applications in frequency assignment problems on cellular networks. A coloring of a hypergraph \mathcal{H} is called *conflict-free* if, for every hyper-edge e , there is a color that occurs exactly once on the vertices of e . The minimum number

of colors needed to conflict-free color the vertices of a hypergraph \mathcal{H} is called the conflict-free chromatic number of \mathcal{H} . Conflict-free coloring of hypergraphs is well studied on rectangles [3], intervals [4], unit disks [18] etc.

Cheilaris [6] studied the CF coloring of graphs with respect to neighborhoods. A coloring f (may not be proper) of a graph G is called *conflict-free* (CF) if for every vertex v there is a color that appears exactly once in the open neighborhood of v . For more details on Conflict-free coloring of graphs, see, e.g., [1, 16].

Another variant of conflict-free coloring called *proper conflict-free* (PCF) coloring was introduced by Fabrici et al. [15]. A proper k -coloring of a graph G is called a proper conflict-free k -coloring if each vertex has a color appearing exactly once in its open neighborhood. It was shown [15] that a planar graph has a proper conflict-free 8-coloring and also constructed a planar graph with no proper conflict-free 5-coloring. This problem was studied in several graph classes, such as minor-closed families, bounded layered treewidth [19], and graphs with bounded expansion [17].

The concept of odd coloring was introduced by Petruševski and Škrekovski [23], which is a generalization of proper conflict-free coloring. A proper k -coloring of a graph G is an *odd k -coloring* if every vertex has some color that appears an odd number of times in its open neighborhood. The *odd chromatic number* of a graph G denoted $\chi_o(G)$ is the minimum integer k such that G has an odd k -coloring. Given a graph G and an integer k , the ODD COLORING problem is to decide whether $\chi_o(G)$ is at most k .

The ODD COLORING problem is well studied on planar graphs [5, 7, 11, 22, 23]. Petruševski and Škrekovski [23] showed that $\chi_o(G) \leq 9$ for every connected planar graph G and they conjectured that for all planar graphs G , $\chi_o(G) \leq 5$. Towards resolving this conjecture, Petr and Portier [22] showed that $\chi_o(G) \leq 8$ for every planar graph G . Cranston et al. [11] proved that every 1-planar graph G , $\chi_o(G) \leq 23$. Caro et al. [5] showed that for every outerplanar graph G , $\chi_o(G) \leq 5$. They also showed that determining $\chi_o(G)$ of a graph G is NP-hard. Ahn et al. [2] proved that it is NP-complete to decide whether $\chi_o(G) \leq k$ for $k \geq 3$, even if G is bipartite.

ODD COLORING

Input: A graph $G = (V, E)$ and integer k .

Question: Does there exist an odd coloring $f : V(G) \rightarrow \{1, \dots, k\}$ of G ?

In this paper, we study the parameterized complexity of ODD COLORING. In parameterized complexity, the running time of an algorithm is measured as a function of input size and a secondary measure called a parameter. A parameterized problem is called fixed-parameter tractable (FPT) with respect to a parameter k , if the problem can be solved in $f(k)n^{O(1)}$ time, where f is a computable function independent of the input size n and k is a parameter associated with the input. For more details on parameterized complexity, we refer the reader to the texts [12, 13]. The obvious natural parameter is k , the number of colors. Since the problem is NP-hard even when $k = 2$, the problem becomes Para-NP-hard

when parameterized by number of colors k . Therefore, we study the problem with respect to structural graph parameters.

The graph parameter tree-width is a well-known structural graph parameter. ODD COLORING is expressible in counting monadic second-order logic (MSO). Therefore, the problem is FPT from the Courcelle [9] meta theorem. Next, we consider the parameter clique-width [10], which is a generalization of tree-width. We show that the problem is W[1]-hard parameterized by the clique-width.

Next, we consider the parameters distance to cluster, distance to co-cluster and neighborhood diversity. These are intermediate parameters between vertex cover and clique-width. We show that the problem is FPT when parameterized by (a) distance to cluster, (b) distance to co-cluster, or (c) neighborhood diversity. We also study the kernelization complexity of the problem. We show that ODD COLORING admits a polynomial kernel when parameterized by distance to clique. We show that ODD COLORING does not admit a polynomial kernel when parameterized by vertex cover number unless $\text{NP} \subseteq \text{Co-NP/poly}$. Next, we consider the parameters distance to split graphs and distance to cographs. Both these parameters are generalizations of vertex cover and distance to clique. However, the complexity of ODD COLORING is not known on split graphs and cographs. Hence, we study ODD COLORING on cographs and split graphs. We show that the problem is polynomial time solvable on cographs and split graphs.

Organization of the Paper. In Sect. 3, we present results of ODD COLORING related to parameterized complexity. In Sects. 4 and 5, we show that ODD COLORING can be solved in polynomial time on cographs and split graphs respectively. Due to space limitations, the proofs of the theorems marked with the (\star) sign are presented in the full version of the paper.

2 Preliminaries

For $k \in \mathbb{N}$, we use $[k]$ to denote the set $\{1, 2, \dots, k\}$. We use n and m to denote the number of vertices and the number of edges of the graph. For simplicity, an edge between vertices x and y is denoted as xy . For a subset $X \subseteq V(G)$, the graph $G[X]$ denotes the subgraph of G induced by vertices of X . For a vertex set $X \subseteq V(G)$, we denote $G - X$, the graph obtained from G by deleting all vertices of X and their incident edges. For a subset $X \subseteq V(G)$, $E[X]$ denotes the set of edges having both end vertices in the set X in G . For subsets $X, Y \subseteq V(G)$, $E[X, Y]$ denotes the set of edges connecting X and Y in G .

Let $f : V(G) \rightarrow [k]$ be a coloring of a graph G . We say that, a vertex $v \in V(G)$ has a color $c \in [k]$ as its *odd color* with respect to f , if the number of neighbors of v that are assigned the color c is odd, that is $|f^{-1}(c) \cap N(v)|$ is odd.

Parameterized Complexity and Kernelization: A parameterized problem L is a set of instances $(x, k) \in \Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet and $k \in \mathbb{N}$ is a parameter. We say that a parameterized problem L is *fixed-parameter tractable* (FPT) if there exists an algorithm and a computable function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that given $(x, k) \in \Sigma^* \times \mathbb{N}$ the algorithm correctly decides whether $(x, k) \in L$ in $f(k) \cdot |(x, k)|^{O(1)}$ time.

A *kernelization* algorithm is a polynomial time algorithm that takes an instance (x, k) of a parameterized problem L as input and outputs an equivalent instance (x', k') of L such that $|x'| \leq h(k)$ for some computable function h and $k' \leq k$. If h is a polynomial, we say that (x', k') is a polynomial kernel.

3 Results: Parameterized Complexity

For a graph G , the distance to clique of G is the smallest number d such that there is a set $X \subseteq V(G)$ of size d with $G - X$ being a clique. In this section, we present a polynomial kernel for the problem parameterized by distance to clique.

Theorem 1. ODD COLORING parameterized by the distance to clique d admits a kernel with $O(d^3)$ vertices.

Proof. Let (G, X, d, k) be an instance of ODD COLORING, where $X \subseteq V(G)$ of size d be such that the graph induced on $C = V(G) \setminus X$ is a clique of size $n-d$. We assume that $k \geq |C|$, otherwise the given instance is a trivial no-instance as we need $|C|$ colors to color the clique $G[C]$. Further, we assume that $|C| > d^2 + d + 1$, otherwise we trivially get a kernel of size at most $d^2 + 2d + 1$.

Let $C_N = \{v \in C \mid N(v) \cap X = \emptyset\}$. If $|C_N| \geq d$, then the given instance is a yes-instance and we get a trivial kernel. Therefore, we assume that $|C_N| < d$. We now partition the vertices of X based on their degree in the clique $G[C]$. Let $X_\ell = \{x \in X : |N(x) \cap C| \leq d-1\}$, $X_m = \{x \in X : d \leq |N(x) \cap C| \leq n-d^2-d-1\}$ and $X_h = \{x \in X : |N(x) \cap C| \geq n-d^2-d\}$. Let $X_\ell^m = \{x \in X_\ell \mid N(x) \cap X_m \neq \emptyset\}$.

Reduction Rule 1. Delete the vertices of X_m from G and for each vertex $x \in X_\ell^m$ add a pendant vertex x' adjacent to x . The new instance is $(G + X'_\ell - X_m, (X \cup X'_\ell) \setminus X_m, d - |X_m| + |X'_\ell|, k)$, where X'_ℓ denotes the set of pendant vertices.

Lemma 1. Reduction Rule 1 is safe.

Proof. *Forward direction.* Let f be a k -odd coloring of G . Let $G' = G + X'_\ell - X_m$. We define a k -coloring g of G' as follows:

- for each $v \in X_\ell \cup X_h \cup C$, we assign $g(v) = f(v)$, and
- for each $v \in X'_\ell$, we assign $g(v) = c$, where c is some arbitrary color from $[k] - f(N[u])$, where u is the only neighbor of v in G' .

Clearly, g is a proper k -coloring of G' . Next we argue that g is an odd coloring of G' . Consider a vertex $v \in X_h$. As v has at least d neighbors in C , at least one color in $g(N(v))$ appears exactly once in the neighborhood of v , and v has an odd color with respect to g . Consider a vertex $v \in C$. As $|C| > d^2 + 1$, v has an odd color with respect to g . Consider a vertex $v \in X_\ell$. If $v \in X_\ell^m$, then the color of pendant vertex adjacent to v appears exactly once in the neighborhood of v and v has an odd color with respect to g . Else if $v \notin X_\ell^m$, we have that $f(N(v)) = g(N(v))$, and v has an odd color with respect to g . Finally, as each

vertex of X'_ℓ has degree one, they trivially have an odd color. Therefore g is an odd coloring of G' .

Reverse Direction. Let g be a k -odd coloring of G' . We define a k -coloring f of G as follows:

- for each $v \in X_\ell \cup X_h \cup C$, we assign $f(v) = g(v)$,
- for $v \in X_m$, we have $|N(v) \cap C| \leq n - d^2 - d - 1$. That is, each vertex $v \in X_m$ is not adjacent to at least $d^2 + 1$ vertices in C . As $|X_\ell| + |X_h| + |N(X_\ell) \cap C| \leq (d - |X_m|) + d(d - 1)$, we color the vertices of X_m with $|X_m|$ many distinct colors, which are different from the colors used in $X_\ell \cup X_h \cup (N(X_\ell) \cap C) \cup (N(X_m) \cap C)$ in f .

For each $v \in X_m$, we have $|N(v) \cap C| \geq d$, therefore at least one color used for vertices of C appears exactly once in $N(v)$. Hence v has an odd color. For each $v \in X_m^m$, each color of $f(N(v) \cap X_m)$ acts as a odd color. For each $v \in X_\ell \setminus X_m^m$, $f(N(v)) = g(N(v))$, therefore v retains its odd color. \square

We assume that $X_h \neq \emptyset$; otherwise, using the above reduction rule and the property of X_ℓ , we trivially get a kernel of size at most $d(d - 1) + d$.

Let $D_h = \bigcap_{x \in X_h} (N(x) \cap C)$ and $D_\ell = \bigcup_{x \in X_\ell} (N(x) \cap C)$. It is easy to see that $|D_\ell| \leq d(d - 1)$. Let $C_1 = D_h \setminus D_\ell$.

Lemma 2. *The size of D_h is at least $n - (|X_h|d + 1)d$.*

Proof. As each vertex of X_h has at least $n - d^2 - d$ neighbors in C , the number of edges $|E[X_h, C]|$ between X_h and C is at least $|X_h|(n - d^2 - d)$. Suppose $|D_h| = n - (|X_h|d + 1)d - r$ for some $r \geq 1$. Then the number of edges $|E[X_h, D_h]|$ between X_h and D_h equal to $|X_h|(n - (|X_h|d + 1)d - r)$. As each vertex in $C \setminus D_h$ only has at most $|X_h| - 1$ neighbors in X_h , the number of edges $|E[X_h, C \setminus D_h]|$ between X_h and $C \setminus D_h$ is at most $(|X_h| - 1)(|X_h|d^2 + r)$. Hence, we have

$$\begin{aligned} |E[X_h, C]| &\leq |E[X_h, D_h]| + |E[X_h, C \setminus D_h]| \\ &\leq |X_h|(n - (|X_h|d + 1)d - r) + (|X_h| - 1)(|X_h|d^2 + r) \\ &= |X_h|(n - d^2 - d) - r \end{aligned}$$

which is contradiction to the fact that $|E[X_h, C]| \geq |X_h|(n - d^2 - d)$. \square

Therefore, we have $|D_h| \geq n - (|X_h|d + 1)d$. Hence $|C_1| \geq n - (|X_h|d + 1)d - d(d - 1) \geq n - (|X_h| + 1)d^2$. Let $C_2 \subseteq C_1$ be a set of $d + 1$ arbitrary vertices. Let $C' = C_1 \setminus C_2$.

Reduction Rule 2. *Delete the vertices in C' from G and decrease k by $|C'|$. The new instance is $(G - C', X, d, k - |C'|)$.*

Lemma 3. *Reduction Rule 2 is safe.*

Proof. *Reverse Direction.* If g is an odd $(k - |C'|)$ -coloring of $G - C'$ then we can obtain an odd k -coloring f of G from g by coloring C' with $|C'|$ many new distinct colors.

For the other direction, let f be an odd k -coloring of G . We assume that $f(X_\ell) \cap f(C') = \emptyset$. If not, we recolor the vertices of X_ℓ with $|X_\ell|$ many distinct colors from $f(C_2)$. Observe that $N(C') = X_h \cup (C \setminus C')$, that is, in the coloring f of G each vertex of C' receives a color different from each color in $X_\ell \cup X_h \cup (C \setminus C')$. Therefore f restricted to $G - C'$ is a proper $(k - |C'|)$ -coloring. We now argue that f is also an odd coloring of $G - C'$. Consider a vertex $v \in V(G) \setminus C'$, if $v \in X_\ell$ then the odd color of v in $G - C'$ is same as the odd color of v in G . If $v \in (X_h \cup (C \setminus C'))$, then as C_2 has $d + 1$ vertices and at least one of the vertex color c of C_2 is not used in $X_\ell \cup X_h$ by the coloring f . Hence, the color c acts as a odd color for the vertex v . \square

It is easy to see that both reduction rules are applicable in polynomial time. Next, we show that the size of the reduced instance is at most $d^3 + 2d^2$.

Lemma 4. *Let (G, X, d, k) be an instance of ODD COLORING obtained after applying the above reduction rules. If it is a yes-instance then $|V(G)| \leq d^3 + 2d^2$.*

Proof. We know that the size of C' is at least $n - (|X_h| + 1)d^2 - d$. Therefore the number of vertices in $X_h \cup X_\ell \cup D_\ell \cup (C \setminus C')$ is at most $d + d(d - 1) + (|X_h|d^2 + d^2)$, which is at most $d^3 + 2d^2$. \square

Theorem 2 (★). ODD COLORING parameterized by vertex cover number does not admit a polynomial kernel, unless $\text{NP} \subseteq \text{Co-NP/poly}$.

Theorem 3 (★). ODD COLORING is FPT parameterized by (a) distance to cluster, (b) distance to co-cluster, and (c) neighborhood diversity.

Theorem 4 (★). ODD COLORING is W[1]-hard parameterized by clique-width.

4 Cographs

In this section, we give a linear time algorithm for cographs. A *cograph* is a graph with no induced P_4 .

Lemma 5 ([8]). *If G is a cograph then one of the following holds.*

1. G has at most one vertex.
2. G is the union of two cographs G_1 and G_2 , i.e., $G = G_1 \cup G_2$.
3. G is the join of two cographs G_1 and G_2 , i.e., $G = G_1 \vee G_2$.

A cotree T_G of a cograph G is a rooted tree in which each internal vertex is either of \cup (union) type or \vee (join) type. The leaves of T_G are precisely the vertices of the cograph G .

Lemma 6. *For any connected cograph G , we have $\chi(G) \leq \chi_o(G) \leq \chi(G) + 2$.*

Proof. As every odd coloring of G is a proper coloring, we get $\chi(G) \leq \chi_o(G)$. Since G is a connected cograph, let $G = G_1 \vee G_2$. Let f be a proper coloring of G with colors $\{1, 2, \dots, \chi(G)\}$. We obtain an odd coloring g of G from f by changing the color of two arbitrary vertices u from G_1 and v from G_2 with colors $\chi(G) + 1$ and $\chi(G) + 2$ respectively. The new colors $\chi(G) + 1$ and $\chi(G) + 2$ are odd colors of vertices of G_2 and G_1 respectively. \square

Next, we introduce the notion of strong proper-coloring and strong odd-coloring.

Definition 1 (Strong proper-coloring). A strong proper-coloring of a graph G is a proper-coloring $f : V(G) \rightarrow [k]$ such that $|f^{-1}(i)|$ is odd for some $i \in [k]$. The smallest integer k such that G has a strong proper-coloring is called strong chromatic number of G , denoted by $\tilde{\chi}(G)$.

Definition 2 (Strong odd-coloring). A strong odd-coloring of a graph G is an odd-coloring $f : V(G) \rightarrow [k]$ such that $|f^{-1}(i)|$ is odd for some $i \in [k]$. The smallest integer k such that G has a strong odd-coloring is called strong odd chromatic number of G , denoted by $\tilde{\chi}_o(G)$.

Lemma 7. For any graph G , we have

1. $\tilde{\chi}(G) \leq \chi(G) + 1$
2. $\tilde{\chi}_o(G) \leq \chi_o(G) + 1$

Proof. Given a proper coloring f of G using $\chi(G)$ colors, we can obtain a strong proper coloring g of G from f using $\chi(G) + 1$ colors by changing the color of an arbitrary vertex with the new color $\chi(G) + 1$. Therefore we get $\tilde{\chi}(G) \leq \chi(G) + 1$. Analogously, we can prove the second part of the lemma. \square

Lemma 8. Let G_1 and G_2 be two graphs and $G = G_1 \cup G_2$. Then $\chi_o(G) = \max\{\chi_o(G_1), \chi_o(G_2)\}$

Proof. As there are no edges from vertices of G_1 to the vertices of G_2 in G , it follows that $\chi_o(G) = \max\{\chi_o(G_1), \chi_o(G_2)\}$. \square

Lemma 9. Let G_1 and G_2 be two graphs and $G = G_1 \vee G_2$. Then

$$\chi_o(G) = \min\{\chi_o(G_1) + \chi_o(G_2), \tilde{\chi}(G_1) + \tilde{\chi}(G_2), \tilde{\chi}_o(G_1) + \chi(G_2), \chi(G_1) + \tilde{\chi}_o(G_2)\}$$

Proof. Let f_i be an odd coloring of G_i with colors $\{1, 2, \dots, \chi_o(G_i)\}$, for each $i \in \{1, 2\}$. Then we define an odd coloring $f : V(G) \rightarrow \{1, 2, \dots, \chi_o(G_1), \chi_o(G_1) + 1, \dots, \chi_o(G_1) + \chi_o(G_2)\}$ as follows.

$$f(u) = \begin{cases} f_1(u) & \text{if } u \in V(G_1); \\ \chi_o(G_1) + f_2(u) & \text{if } u \in V(G_2); \end{cases}$$

It is easy to see that f is an odd coloring of G using $\chi_o(G_1) + \chi_o(G_2)$ colors. Therefore, $\chi_o(G) \leq \chi_o(G_1) + \chi_o(G_2)$.

Similarly we can show that (a) $\chi_o(G) \leq \tilde{\chi}(G_1) + \tilde{\chi}(G_2)$, (b) $\chi_o(G) \leq \widetilde{\chi}_o(G_1) + \chi(G_2)$ and (c) $\chi_o(G) \leq \chi(G_1) + \widetilde{\chi}_o(G_2)$. Using the four inequalities, we get the following.

$$\chi_o(G) \leq \min\{\chi_o(G_1) + \chi_o(G_2), \tilde{\chi}(G_1) + \tilde{\chi}(G_2), \widetilde{\chi}_o(G_1) + \chi(G_2), \chi(G_1) + \widetilde{\chi}_o(G_2)\} \quad (1)$$

Next we show the lower bound. Let $\chi_o(G) = k$ and f be an odd coloring of G using k colors. For $i \in \{1, 2\}$, let f_i be the coloring of G_i obtained by restricting f to the vertices of G_i . Observe that $f_1(V(G_1)) \cap f_2(V(G_2)) = \emptyset$ as there are all possible edges between G_1 and G_2 in G . Let $|f_1(V(G_1))| = k_1$ and $|f_2(V(G_2))| = k_2$, where $k_1 + k_2 = k$. As f is an odd coloring of G , both f_1 and f_2 are proper colorings of G_1 and G_2 respectively. Therefore, we have $\chi(G_1) \leq k_1$ and $\chi(G_2) \leq k_2$. We divide the proof into several cases.

– **Case 1.** $\widetilde{\chi}_o(G_1) \leq k_1$.

In this case we get

$$\chi_o(G) = k = k_1 + k_2 \geq \widetilde{\chi}_o(G_1) + \chi(G_2)$$

– **Case 2.** $\widetilde{\chi}_o(G_1) > k_1$, $\chi_o(G_1) = k_1$ and $\tilde{\chi}(G_1) > k_1$.

Then f_2 must be an odd k_2 -coloring of G_2 . Otherwise some vertices of G_2 in the coloring f will not have any odd color which contradicts the fact that f is an odd coloring of G . Therefore, in this case we have $\chi_o(G_1) \leq k_1$ and $\chi_o(G_2) \leq k_2$. Therefore, we get,

$$\chi_o(G) = k = k_1 + k_2 \geq \chi_o(G_1) + \chi_o(G_2)$$

– **Case 3.** $\widetilde{\chi}_o(G_1) > k_1$, $\chi_o(G_1) > k_1$ and $\tilde{\chi}(G_1) = k_1$.

Then f_2 must be a strong proper k_2 -coloring of G_2 . Otherwise some vertices of G_1 in the coloring f will not have any odd color which contradicts the fact that f is an odd coloring of G . Therefore, in this case we have $\tilde{\chi}(G_1) \leq k_1$ and $\tilde{\chi}(G_2) \leq k_2$. Therefore we get,

$$\chi_o(G) = k = k_1 + k_2 \geq \tilde{\chi}(G_1) + \tilde{\chi}(G_2)$$

– **Case 4.** $\chi_o(G_1) > k_1$ and $\tilde{\chi}(G_1) > k_1$.

Then f_2 must be a strong odd k_2 -coloring of G_2 . Otherwise some vertices of G_1 in the coloring f will not have any odd color which contradicts the fact that f is an odd coloring of G . Therefore we get

$$\chi_o(G) = k = k_1 + k_2 \geq \chi(G_1) + \widetilde{\chi}_o(G_2)$$

Therefore, combining all the above we get

$$\chi_o(G) \geq \min\{\chi_o(G_1) + \chi_o(G_2), \tilde{\chi}(G_1) + \tilde{\chi}(G_2), \widetilde{\chi}_o(G_1) + \chi(G_2), \chi(G_1) + \widetilde{\chi}_o(G_2)\} \quad (2)$$

From Eqs. 1 and 2 the result follows. \square

Lemma 10. Let G_1 and G_2 be two graphs and let $G = G_1 \vee G_2$ then

1. $\chi(G) = \chi(G_1) + \chi(G_2)$
2. $\chi_o(G) = \min\{\chi_o(G_1) + \chi_o(G_2), \widetilde{\chi}_o(G_1) + \chi(G_2), \chi(G_1) + \widetilde{\chi}_o(G_2)\}$
3. $\widetilde{\chi}(G) = \min\{\widetilde{\chi}(G_1) + \chi(G_2), \chi(G_1) + \widetilde{\chi}(G_2)\}$
4. $\widetilde{\chi}_o(G) = \min\{\widetilde{\chi}_o(G_1) + \chi(G_2), \chi(G_1) + \widetilde{\chi}_o(G_2)\}$

Proof. Follows from the Lemmas 7 and 9. \square

Theorem 5. Given a cograph G , we can compute $\chi_o(G)$ in linear time.

Proof. Follows from the Lemmas 5, 8 and 10. \square

5 Split Graphs

In this section, we show that ODD COLORING can be solved in polynomial time on split graphs. A graph is a split graph if its vertices can be partitioned into a (maximal) clique and an independent set. We denote a split graph with $G = (K, I)$ where K and I denote the partition of $V(G)$ into a clique and an independent set. As K is maximal, there is no vertex $v \in I$ such that $N(v) = K$. For a subset $Y \subseteq K$, we use $T^Y = \{v \in I \mid N(v) = Y\}$ to denote the subset of vertices in I , whose neighborhood in K is exactly Y . Throughout this section we assume that $K = \{v_1, v_2, \dots, v_k\}$, $|K| = k$ and $I = \{u_1, u_2, \dots, u_\ell\}$, $|I| = \ell$.

Theorem 6. Let $G = (K, I)$ be a split graph. Then $k \leq \chi_o(G) \leq k + 1$.

Proof. As every odd coloring of G is a proper coloring, we have $\chi_o(G) \geq \chi(G) \geq k$. We can obtain an odd $(k+1)$ -coloring of G by coloring the vertices of K with k distinct colors and all vertices I with color $k+1$. Hence, $\chi_o(G) \leq k+1$. \square

Next, we give a characterization of split graphs having odd chromatic number two.

Lemma 11. Let $G = (K, I)$ be a split graph with $K = \{v_1, v_2\}$. Then, $\chi_o(G) = 2$ if and only if $I_i = |N(v_i) \cap I|$ is even for each $i \in \{1, 2\}$.

Proof. Forward Direction. Let $\chi_o(G) = 2$ and let f be an odd 2-coloring of G with colors $\{1, 2\}$. Without loss of generality assume that $f(v_i) = i$ for $i \in \{1, 2\}$. Then we must have $f(I_1) = \{2\}$ and $f(I_2) = \{1\}$. As f is an odd 2-coloring of G , both $|I_1|$ and $|I_2|$ must be even.

Reverse Direction. As there is no vertex $u_j \in I$ such that $N(u_j) = K$ we have $I_1 \cap I_2 = \emptyset$. If both $|I_1|$ and $|I_2|$ are even then odd coloring of G is obtained by coloring v_1 and every vertex of I_2 with color one and v_2 and every vertex of I_1 with color two. Hence, $\chi_o(G) = 2$. \square

For the rest of this section we assume that $|K| \geq 3$. Notice that in any proper coloring f of a split graph $G = (K, I)$, it is easy to see that every vertex of I has an odd color with respect to f . Hence, in all the proofs presented in the rest of this section, we omit the proof of showing a vertex in I has an odd color with respect to a proper coloring.

Lemma 12. Let $G = (K, I)$ be a split graph. If there exists a vertex v_p in K such that $N(v_p) \cap I = \emptyset$ then $\chi_o(G) = k$.

Proof. Let $f : V(G) \rightarrow [k]$ be a proper coloring of G defined as follows. For each $v_i \in K$, $f(v_i) = i$ and $f(u_j) = p$ for every $u_j \in I$. As $|K| \geq 3$, every vertex v_i of K have color q as an odd color with respect to f , where $q \in [k] \setminus \{i, p\}$. \square

For the rest of this section we assume for each $v_i \in K$, we have $N(v_i) \cap I \neq \emptyset$.

Theorem 7. Let $G = (K, I)$ be a split graph. Then, $\chi_o(G) = |K| + 1$ if and only if there exists a vertex $v \in K$ such that for every $w \in K - \{v\}$, $|T^{K-\{w\}}|$ is odd and $N(v) \cap I = \bigcup_{w \in K \setminus \{v\}} T^{K-\{w\}}$.

Proof. Reverse Direction. Let $|K| = k$. If there exists a vertex $v \in K$ satisfying the conditions of the premise, then we show that $\chi_o(G) = k + 1$. Suppose that $\chi_o(G) = k$. Let $f : V(G) \rightarrow [k]$ be an odd k -coloring of G . Since f is a proper coloring, no two vertices of K are assigned the same color, that is $f(v_i) = i$ for $v_i \in K$. It is easy to see that for every vertex $w \in K - \{v\}$, all the vertices of $T^{K-\{w\}}$ are colored with the color $f(w)$. As $|T^{K-\{w\}}|$ is odd for every vertex $w \in K - \{v\}$, and $N(v) \cap I = \bigcup_{w \in K \setminus \{v\}} T^{K-\{w\}}$, each color from the set $[k] \setminus \{f(v)\}$

appears even number of times in the neighborhood of v , which is a contradiction to the fact that v has an odd color with respect to f . Therefore $\chi_o(G) = k + 1$.

Forward Direction. We show this by contradiction. That is, if there is no vertex $v \in K$ satisfying the conditions of the premise, then $\chi_o(G) = k$. We construct a coloring $f : V(G) \rightarrow [k]$ as follows. We first color the vertices of K with k distinct colors, then each vertex in I has an odd color with respect to f . We now explain the procedure to extend the partial coloring f to the vertices of I so that every vertex of K has an odd color with respect to f .

Let v be an arbitrary vertex of K , then at least one of the following holds.

1. $|T^{K-\{w\}}|$ is even, for some $w \in K - \{v\}$.
2. $(N(v) \cap I) - \bigcup_{w \in K \setminus \{v\}} T^{K-\{w\}} \neq \emptyset$.

– **Case 1.** $|T^{K-\{w\}}|$ is even for some $w \in K - \{v\}$

It is the case that either $|T^{K-\{w\}}| = 0$ or $|T^{K-\{w\}}| = 2p$ for some $p \in \mathbb{N}$. For the latter case, we color each vertex of $T^{K-\{w\}}$ with $f(w)$. As $|T^{K-\{w\}}|$ is even, the color $f(w)$ acts as an odd color (in the partial coloring that we have so far) for every vertex of K except for w .

We now have two cases based on $N(w) \cap I$.

- **Case 1a.** $|T^{K-\{z\}}|$ is even for some $z \in K - \{w\}$

If $|T^{K-\{z\}}| > 0$, we color each vertex of $T^{K-\{z\}}$ with the color of z . As $|T^{K-\{z\}}|$ is even, the color $f(z)$ acts as an odd color for w (this holds for the case even when $|T^{K-\{z\}}| = 0$). Now each vertex in K has an odd color. We assign colors to the remaining vertices of I in such a way that the odd colors for the vertices in K remain the same.

Algorithm 1: An odd coloring of $G = (K, I)$ satisfying the assumptions of Case 2.

Input: Split graph $G = (K, I)$, a partial coloring f of G
Output: An odd coloring f of G using k colors

- 1 $K' = K$ and $I' = I - \bigcup_{w \in K} T^{K-\{w\}}$
- 2 $j = 0$, $Q = \emptyset$, $R = \emptyset$
- 3 **while** $K' \neq \emptyset$ **do**
- 4 Let $u \in I'$ be the highest degree vertex in $G' = (K', I')$
- 5 $j = j + 1$
- 6 $p_j = u$, $Q_j = N(u)$, $R_j = \{v \in I' \mid N_{G'}(v) \subseteq N_{G'}(u)\}$
- 7 $K' = K' - Q_j$, $I' = I' - (\{p_j\} \cup R_j)$
- 8 $\ell = \max\{j : Q_j \neq \emptyset\}$
- 9 /* The partial coloring f is extended to uncolored independent set vertices as follows. */
- 10 $f(p_1) = c_1$, where $c_1 \in f(Q_\ell)$.
- 11 **for each** $j = 2$ to ℓ **do**
- 12 $f(p_j) = c_j$, where $c_j \in f(Q_{j-1}) \setminus f(N(p_j))$
- 13 **for each** $j = 1$ to ℓ **do**
- 14 For each $v \in R_j$, $f(v) = c'_j$, where $c'_j = [k] \setminus (f(N(v)) \cup \{f(p_1), \dots, f(p_j)\})$

14 return (coloring f)

We assign each vertex of $|T^{K-\{y\}}|$, where $y \in K \setminus \{w, z\}$, the color $f(y)$. The remaining uncolored vertices in I have degree at most $k - 2$. We color every uncolored vertex of $N(w)$ with any color other than color of z . Color rest of the uncolored vertices with a color other than $f(w)$. It is easy to see that f is an odd coloring.

- **Case 1b.** $|T^{K-\{z\}}|$ is odd for every $z \in K - \{w\}$ and $(N(w) \cap I) - \bigcup_{z \in K \setminus \{w\}} T^{K-\{z\}} \neq \emptyset$.

For every $w \in K$, we color the vertices in $T^{K \setminus \{w\}}$ with the color $f(w)$. Let $D_w = (N(w) \cap I) - \bigcup_{z \in K \setminus \{w\}} T^{K-\{z\}}$. Observe that each vertex in D_w

has degree at most $k - 2$. We choose an arbitrary vertex $w' \in D_w$ and assign a color $c_w \in [k] - f(N(w'))$ and color the remaining vertices of D_w with any available color other than c_w . Each uncolored vertex of $I \setminus D_w$ is assigned a color other than $f(w)$. In this case c_w is an odd color for w and $f(w)$ is an odd color for every vertex $u \in K - \{w\}$. Thus f is an odd coloring.

- **Case 2.** For every $v \in K$, (i) $|T^{K-\{w\}}|$ is odd for every $w \in K - \{v\}$, and (ii) $(N(v) \cap I) - \bigcup_w T^{K-\{w\}} \neq \emptyset$.

For every $w \in K$, each vertex in $T^{K \setminus \{w\}}$ is assigned the color $f(w)$. Thus every color of $[k]$ appears even number of times in the open neighborhood of each vertex $w \in K$, in the partial coloring f constructed so far. We now assign colors to the remaining uncolored vertices of I such that each vertex in K has

an odd color. We extend f to an odd k -coloring of G using Algorithm 1 which takes as input a split graph $G = (K, I)$ and a partial coloring f as described above. \square

References

1. Abel, Z., et al.: Conflict-free coloring of graphs. *SIAM J. Discrete Math.* **32**(4), 2675–2702 (2018)
2. Ahn, J., Im, S., il Oum, S.: The proper conflict-free k -coloring problem and the odd k -coloring problem are np-complete on bipartite graphs. arXiv preprint [arXiv:2208.08330](https://arxiv.org/abs/2208.08330) (2022)
3. Ajwani, D., Elbassioni, K., Govindarajan, S., Ray, S.: Conflict-free coloring for rectangle ranges using $O(n^{0.382})$ colors. In: Proceedings of the Nineteenth Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 181–187. ACM (2007)
4. Bar-Noy, A., Cheilaris, P., Smorodinsky, S.: Deterministic conflict-free coloring for intervals: from offline to online. *ACM Trans. Algorithms (TALG)* **4**(4), 44 (2008)
5. Caro, Y., Petruševski, M., Škrekovski, R.: Remarks on odd colorings of graphs. *Discret. Appl. Math.* **321**, 392–401 (2022)
6. Cheilaris, P.: Conflict-free coloring (Ph.D. thesis). City University of New York (2009)
7. Cho, E.-K., Choi, I., Kwon, H., Park, B.: Odd coloring of sparse graphs and planar graphs. *Discret. Math.* **346**(5), 113305 (2023)
8. Corneil, D.G., Perl, Y., Stewart, L.K.: A linear recognition algorithm for cographs. *SIAM J. Comput.* **14**(4), 926–934 (1985)
9. Courcelle, B.: The monadic second-order logic of graphs III: tree-decompositions, minors and complexity issues. *RAIRO-Theor. Inform. Appl.* **26**(3), 257–286 (1992)
10. Courcelle, B., Olariu, S.: Upper bounds to the clique width of graphs. *Discret. Appl. Math.* **101**(1–3), 77–114 (2000)
11. Cranston, D.W., Lafferty, M., Song, Z.-X.: A note on odd colorings of 1-planar graphs. *Discrete Appl. Math.* **330**, 112–117 (2023)
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.: Parameterized Complexity, vol. 3. Springer, Heidelberg (1999). <https://doi.org/10.1007/978-1-4612-0515-9>
14. Even, G., Lotker, Z., Ron, D., Smorodinsky, S.: Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks. *SIAM J. Comput.* **33**(1), 94–136 (2003)
15. Fabrici, I., Lužar, B., Rindošová, S., Soták, R.: Proper conflict-free and unique-maximum colorings of planar graphs with respect to neighborhoods. *Discret. Appl. Math.* **324**, 80–92 (2023)
16. Gargano, L., Rescigno, A.A.: Complexity of conflict-free colorings of graphs. *Theor. Comput. Sci.* **566**, 39–49 (2015)
17. Hickinbotham, R.: Odd colourings, conflict-free colourings and strong colouring numbers. arXiv preprint [arXiv:2203.10402](https://arxiv.org/abs/2203.10402) (2022)
18. Lev-Tov, N., Peleg, D.: Conflict-free coloring of unit disks. *Discret. Appl. Math.* **157**(7), 1521–1532 (2009)

19. Liu, C-H.: Proper conflict-free list-coloring, subdivisions, and layered treewidth. arXiv preprint [arXiv:2203.12248](https://arxiv.org/abs/2203.12248) (2022)
20. Nagy-György, J., Imreh, C.: Online hypergraph coloring. Inf. Process. Lett. **109**(1), 23–26 (2008)
21. Pach, J., Tardos, G.: Conflict-free colourings of graphs and hypergraphs. Comb. Probab. Comput. **18**(5), 819–834 (2009)
22. Petr, J., Portier, J.: The odd chromatic number of a planar graph is at most 8. Graphs Comb. **39**(2), 28 (2023)
23. Petruševski, M., Škrekovski, R.: Colorings with neighborhood parity condition. Discret. Appl. Math. **321**, 385–391 (2022)



On Full-Separating Sets in Graphs

Dipayan Chakraborty^{1,2} and Annegret K. Wagler¹

¹ Université Clermont-Auvergne, CNRS, Mines de Saint-Étienne,
Clermont-Auvergne-INP, LIMOS, 63000 Clermont-Ferrand, France
{dipayan.chakraborty,annegret.wagler}@uca.fr

² Department of Mathematics and Applied Mathematics, University of
Johannesburg, Auckland Park 2006, South Africa

Abstract. Several different types of identification problems have been studied in the literature, where the objective is to distinguish any two vertices of a graph by their unique neighborhoods in a suitably chosen dominating or total-dominating set, often referred to as a *code*. To study such problems under a unifying point of view, reformulations in terms of covering problems in suitably constructed hypergraphs have been provided. Analyzing these hypergraph representations, we introduce a new separation property, called *full-separation*, which has not yet been considered in the literature so far. We study it in combination with both domination and total-domination, and call the resulting codes *full-separating dominating codes* (or *FD-codes* for short) and *full-separating total-dominating codes* (or *FTD-codes* for short), respectively. We address the conditions for the existence of FD- and FTD-codes, bounds for their size, their relation to codes of the other types and present some extremal cases for these bounds and relations. We further show that the problems of determining an FD- or an FTD-code of minimum cardinality in a graph is NP-hard. We also show that the cardinalities of minimum FD- and FTD-codes differ by at most one, but that it is NP-hard to decide if they are equal for a given graph in general.

Keywords: full-separation · open-separation · closed-separation · domination · total-domination · NP-completeness · hypergraph

1 Introduction

In the domain of identification problems, the objective is typically to distinguish any two vertices of a graph by the unique intersection of their neighborhoods with a suitably chosen dominating or total-dominating set of the graph. Depending on the types of dominating sets and separation properties used, various problems arise under various names in the literature.

More precisely, consider a graph $G = (V, E)$ and denote by $N(v) = \{u \in V : uv \in E\}$ (respectively, $N[v] = N(v) \cup \{v\}$) the open (respectively, closed) neighborhood of a vertex $v \in V$. On the one hand, a subset $C \subseteq V$ is *dominating* (respectively, *total-dominating*) if the set $N[v] \cap C$ (respectively, $N(v) \cap C$) is non-empty for each $v \in V$. On the other hand, a subset $C \subseteq V$ is *closed-separating*

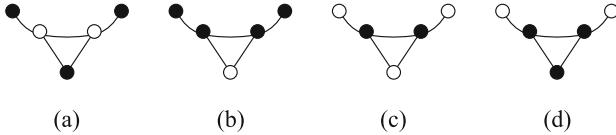


Fig. 1. Minimum X-codes in a graph (the black vertices belong to the code), where (a) is an ID-code, (b) an ITD-code, (c) both an LD- and LTD-code, (d) both an OD- and OTD-code.

(respectively, *open-separating*) if the set $N[v] \cap C$ (respectively, $N(v) \cap C$) is unique for each $v \in V$. Moreover, C is *locating* if the set $N(v) \cap C$ is unique for each $v \in V \setminus C$. So far, the following combinations of location/separation and domination properties have been studied in the literature.

- closed-separation with domination and total-domination leading to *identifying codes* (*ID-codes* for short) and *identifying total-dominating codes*¹ (*ITD-codes* for short), see [11] and [8], respectively;
- location with domination and total-domination leading to *locating dominating codes* (*LD-codes* for short) and *locating total-dominating codes* (*LTD-codes* for short), see [13] and [8], respectively;
- open-separation with domination and total-domination leading to *open-separating dominating codes* (*OD-codes* for short) and *open-separating total-dominating codes*² (*OTD-codes* for short), see [4] and [9, 12], respectively.

Figure 1 illustrates examples of such codes in a small graph. Given a graph $G = (V, E)$, for $X \in \{\text{ID, ITD, LD, LTD, OD, OTD}\}$, the X-problem on G is the problem of finding an X-code of minimum cardinality $\gamma^X(G)$ in G .

Problems of this type are intensively studied in the literature (see, for example, the internet bibliography containing over 500 articles around these topics maintained by Jean and Lobstein [10]) and have manifold applications, for example, in locating threats/ intruders in facilities using sensor networks [14]. To model, for example, such monitoring problems, the rooms of a building are represented by vertices of a graph, doors between rooms as edges, and the task is to place monitoring devices at some vertices of the graph (a dominating or total-dominating set) such that the intruder (like a fire, a thief, or a saboteur) can be either detected directly in the room where the device is present or can be located with the help of a unique pattern of alerts sent out by other devices in some of the adjacent rooms. In the literature, several adaptations have been discussed to tackle further practical challenges, whereby some of the monitoring devices may become faulty over time:

¹ ITD-codes had been introduced in [8] under the name *differentiating total-dominating codes*. Due to consistency in notation, we call them ITD-codes in this article.

² OTD-codes were introduced independently in [9] and in [12] under the names of *strongly ($t, \leq l$)-identifying codes* and *open neighborhood locating-dominating sets* (or *OLD-sets*), respectively. However, due to consistency in naming that specifies the separation and the domination property, we prefer to call them open-separating total-dominating codes in this article.

- Fault type 1: A monitoring device may lose the ability to distinguish the presence of an intruder in the room it is installed in or in an adjacent room.
- Fault type 2: An intruder may disable the monitoring device in a room.

Monitoring without faulty or disabled devices leads to location [13], tackling faults of type 1 corresponds to closed-separation [11], whereas tackling faults of type 2 corresponds to open-separation [9, 12]. However, to the best of our knowledge, the literature still lacks the model of a monitoring system where faults of both type 1 and 2 may occur simultaneously. The purpose of this article is to introduce such a separation property.

To study various X-problems from a unifying point of view, reformulations of the above six X-problems in terms of covering problems in suitably constructed hypergraphs have been considered, for example, in [1–4]. Given a graph $G = (V, E)$ and an X-problem, we look for a hypergraph $\mathcal{H}_X(G) = (V, \mathcal{F}_X)$ so that $C \subseteq V$ is an X-code of G if and only if C is a *cover* of $\mathcal{H}_X(G)$, that is, a subset $C \subseteq V$ satisfying $C \cap F \neq \emptyset$ for all $F \in \mathcal{F}_X$. Then the *covering number* $\tau(\mathcal{H}_X(G))$, defined as the minimum cardinality of a cover of $\mathcal{H}_X(G)$, equals by construction the X-number $\gamma^X(G)$. The hypergraph $\mathcal{H}_X(G)$ is called the *X-hypergraph* of G .

It is a simple observation that for an X-problem involving domination (respectively, total-domination), \mathcal{F}_X needs to contain the closed (respectively, open) neighborhoods of all vertices of G . In order to encode the separation properties, that is, the fact that the intersection of an X-code with the neighborhood of each vertex is *unique*, it was suggested in [1, 2] to use the symmetric differences of the neighborhoods. Here, given two sets A and B , their *symmetric difference* is defined by $A \Delta B = (A \setminus B) \cup (B \setminus A)$. In fact, it has been shown in [1, 2] that a code C of a graph G is

- closed-separating if and only if $(N[u] \Delta N[v]) \cap C \neq \emptyset$ for all distinct $u, v \in V$,
- open-separating if and only if $(N(u) \Delta N(v)) \cap C \neq \emptyset$ for all distinct $u, v \in V$,
- locating if and only if $(N(u) \Delta N(v)) \cap C \neq \emptyset$ for all $uv \in E$ and $(N[u] \Delta N[v]) \cap C \neq \emptyset$ for all $uv \notin E$, where $u, v \in V$ are distinct.

Accordingly, for each of the already studied X-problems, the hyperedge set \mathcal{F}_X of $\mathcal{H}_X(G)$ consists of exactly 3 different subsets as shown in Table 1, where

- $N[G]$ (respectively, $N(G)$) denotes the set of all closed (respectively, open) neighborhoods of vertices in G ,
- $\Delta_a[G]$ (respectively, $\Delta_a(G)$) the set of symmetric differences of closed (respectively, open) neighborhoods of all pairs of *adjacent* vertices in G ,
- $\Delta_n[G]$ (respectively, $\Delta_n(G)$) the set of symmetric differences of closed (respectively, open) neighborhoods of all pairs of *non-adjacent* vertices in G .

Analyzing this summary of hypergraph representations of the six already established X-problems, we observe that one separation property has not yet been considered in the literature, namely the one involving $\Delta_a[G]$ and $\Delta_n(G)$. The aim of this paper is to introduce the corresponding separation property, called *full-separation*, and to study it in combination with both domination and total-domination leading to *full-separating dominating codes* (or *FD-codes* for short) and *full-separating total-dominating codes* (or *FTD-codes* for short), respectively.

Table 1. The X-hypergraphs $\mathcal{H}_X(G) = (V, \mathcal{F}_X)$ for the already considered X-problems, listing column-wise the 3 needed subsets of hyperedges

X	ID	ITD	LD	LTD	OD	OTD
\mathcal{F}_X	$N[G]$	$N(G)$	$N[G]$	$N(G)$	$N[G]$	$N(G)$
	$\Delta_a[G]$	$\Delta_a[G]$	$\Delta_a(G)$	$\Delta_a(G)$	$\Delta_a(G)$	$\Delta_a(G)$
	$\Delta_n[G]$	$\Delta_n[G]$	$\Delta_n[G]$	$\Delta_n[G]$	$\Delta_n(G)$	$\Delta_n(G)$

Note that not all graphs admit codes of all studied types. We accordingly address the conditions for the existence of FD- and FTD-codes, their relation to codes of the other types, bounds on $\gamma^{\text{FD}}(G)$ and $\gamma^{\text{FTD}}(G)$ as well as some extremal examples for their order in Sect. 2. In particular, we show that $\gamma^{\text{FD}}(G)$ and $\gamma^{\text{FTD}}(G)$ differ by at most one. Moreover, the problems of determining an X-code of minimum cardinality $\gamma^X(G)$ in a graph G have been shown to be NP-hard for all the previously studied X-problems [4–6, 12]. We show the same for FD- and FTD-codes in Sect. 3. Furthermore, we show that deciding whether the minimum FD- and FTD-codes of a graph have equal cardinality is NP-hard.

2 Full-Separation and Related Codes

In this section, we formally introduce the concept of full-separation, the related full-separating codes and address fundamental questions concerning the existence of these codes and bounds for their cardinalities.

Full-Separation and Related Codes. Let $G = (V, E)$ be a graph. A subset $C \subseteq V$ is a *full-separating set* of G if, for each pair of distinct vertices $u, v \in V$, we have

$$(N(v) \cap C) \setminus \{u\} \neq (N(u) \cap C) \setminus \{v\}.$$

Moreover, a full-separating set $C \subseteq V$ is a *fullarating dominating code* (FD-code for short) (respectively, a *full-separating total-dominating code* (FTD-code for short)) if C is also a dominating (respectively, total-dominating) set of G . The ITD-code from Fig. 1 is also both a minimum FD- and FTD-code of the graph, Fig. 2 illustrates further examples of such codes in a small graph.

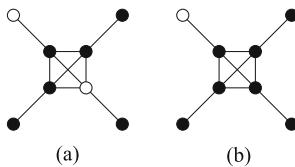


Fig. 2. Minimum X-codes in a graph (the black vertices belong to the code), where (a) is an FD-code, (b) an FTD-code.

In the following, we show some characterizations of full-separating sets.

Theorem 1. Let $G = (V, E)$ be a graph. For a subset $C \subseteq V$, the following assertions are equivalent.

- (a) C is full-separating.
- (b) C has a non-empty intersection with $(N(u) \Delta N(v)) \setminus \{u, v\}$ for all distinct $u, v \in V$.
- (c) C has a non-empty intersection with
 - $N[u] \Delta N[v]$ for all pairs of adjacent vertices $u, v \in V$, and
 - $N(u) \Delta N(v)$ for all pairs of non-adjacent vertices $u, v \in V$.
- (d) $N[u] \cap C \neq N[v] \cap C$ and $N(u) \cap C \neq N(v) \cap C$ for each pair of distinct vertices $u, v \in V$.
- (e) C is both a closed-separating and an open-separating set of G .

Accordingly, by Theorem 1, the hyperedge sets of both $\mathcal{H}_{FD}(G)$ and $\mathcal{H}_{FTD}(G)$ are composed of $N[G]$ for domination and $N(G)$ for total-domination, respectively, as well as $\Delta_a[G]$ and $\Delta_n(G)$ in both cases. Moreover, it turns out that a full-separating set incorporates both elements of closed-separation and open-separation. Thus, Theorem 1 also justifies how the full-separation property models monitoring systems that tackle both faults types 1 and 2 simultaneously.

Existence of FD- and FTD-Codes. It is known from the literature that the already studied X-codes may not exist in all graphs, see for example [4, 8, 11, 12]. Considering the existence of X-codes from the point of view as covers of the X-hypergraphs $\mathcal{H}_X(G) = (V, \mathcal{F}_X)$, we immediately see that G has no X-code if and only if $\mathcal{H}_X(G)$ has no cover if and only if $\emptyset \in \mathcal{F}_X$. In this context, it has been observed in [1, 2] that there is no X-code in a graph G with $\mathcal{H}_X(G)$ involving

- $N(G)$ if G has *isolated vertices*, that is, vertices v with $N(v) = \emptyset$;
- $\Delta_a[G]$ if G has *closed twins*, that is, adjacent vertices u, v with $N[u] = N[v]$ (and thus $N[u] \Delta N[v] = \emptyset$);
- $\Delta_n(G)$ if G has *open twins*, that is, non-adjacent vertices u, v with $N(u) = N(v)$ (and thus $N(u) \Delta N(v) = \emptyset$).

Table 2. The X-hypergraphs $\mathcal{H}_X(G) = (V, \mathcal{F}_X)$, listing in boldface the subsets of hyperedges which may contain an empty hyperedge and hence, may lead to non-existence of X-codes.

X	LD	LTD	ID	ITD	OD	OTD	FD	FTD
\mathcal{F}_X	$N[G]$	$N(\mathbf{G})$	$N[G]$	$N(\mathbf{G})$	$N[G]$	$N(\mathbf{G})$	$N[G]$	$N(\mathbf{G})$
	$\Delta_a(G)$	$\Delta_a(G)$	$\Delta_a[\mathbf{G}]$	$\Delta_a[\mathbf{G}]$	$\Delta_a(G)$	$\Delta_a(G)$	$\Delta_a[\mathbf{G}]$	$\Delta_a[\mathbf{G}]$
	$\Delta_n[G]$	$\Delta_n[G]$	$\Delta_n[G]$	$\Delta_n[G]$	$\Delta_n(\mathbf{G})$	$\Delta_n(\mathbf{G})$	$\Delta_n(\mathbf{G})$	$\Delta_n(\mathbf{G})$

Table 2 illustrates which X-problems are concerned. Calling a graph G *X-admissible* if G has an X-code, we see, for example, from Table 2 that every graph G is LD-admissible, whereas a graph G is OTD-admissible if and only if G has neither isolated vertices nor open twins. Moreover, we call a graph *twin-free* if it has neither closed nor open twins. Accordingly, we conclude the following regarding the existence of FD- and FTD-codes in graphs:

Corollary 1. *A graph G is*

- *FD-admissible if and only if G is twin-free;*
- *FTD-admissible if and only if G is twin-free and has no isolated vertex.*

Since any two distinct isolated vertices of a graph are open twins with the empty set as both their open neighborhoods, Corollary 1 further implies the following.

Corollary 2. *An FD-admissible graph has at most one isolated vertex.*

Relations of X-Numbers and Resulting Bounds on FD- and FTD-Numbers. Consider the set $\mathcal{H}^*(V)$ of all hypergraphs $\mathcal{H} = (V, \mathcal{F})$ on the same vertex set V . We define a relation \prec on $\mathcal{H}^*(V) \times \mathcal{H}^*(V)$ by $\mathcal{H} \prec \mathcal{H}'$ if, for every hyperedge F' of \mathcal{H}' , there exists a hyperedge F of \mathcal{H} such that $F \subseteq F'$. We can show:

Lemma 1. *If $\mathcal{H}, \mathcal{H}' \in \mathcal{H}^*(V)$ with $\mathcal{H} \prec \mathcal{H}'$, then any cover of \mathcal{H} is also a cover of \mathcal{H}' . In particular, we have $\tau(\mathcal{H}') \leq \tau(\mathcal{H})$.*

Proof. Let $\mathcal{H}, \mathcal{H}' \in \mathcal{H}^*(V)$ with $\mathcal{H} = (V, \mathcal{F})$, $\mathcal{H}' = (V, \mathcal{F}')$ and assume that $\mathcal{H} \prec \mathcal{H}'$. Moreover, let $C \subseteq V$ be a cover of \mathcal{H} . Now, let $F' \in \mathcal{F}'$ be any hyperedge. Since $\mathcal{H} \prec \mathcal{H}'$, there exists a hyperedge F of \mathcal{H} such that $F \subseteq F'$. Then $C \cap F \neq \emptyset$, as C is a cover of \mathcal{H} . This implies that $C \cap F' \neq \emptyset$ as well. Hence, C intersects every hyperedge of \mathcal{H}' and thus, is also a cover of \mathcal{H}' . The second statement follows by the fact that, if C is minimum cover of \mathcal{H} , then C is also a cover of \mathcal{H}' and hence, $\tau(\mathcal{H}') \leq |C| = \tau(\mathcal{H})$. \square

Let $\text{CODES} = \{\text{LD, LTD, ID, ITD, OD, OTD, FD, FTD}\}$. It is easy to see that for every graph $G = (V, E)$, the X-hypergraphs $\mathcal{H}_X(G) = (V, \mathcal{F}_X)$ belong to $\mathcal{H}^*(V)$ for all $X \in \text{CODES}$. Furthermore, it is clear that

- $N[v] = N(v) \cup \{v\}$ for all vertices $v \in V$;
- $N(u) \Delta N(v) = (N[u] \Delta N[v]) \cup \{u, v\}$ for all adjacent vertices $u, v \in V$;
- $N[u] \Delta N[v] = (N(u) \Delta N(v)) \cup \{u, v\}$ for all non-adjacent vertices $u, v \in V$.

This observation combined with the above lemma implies:

Corollary 3. *If for a graph $G = (V, E)$ and two problems $X, X' \in \text{CODES}$, the hyperedges of $\mathcal{H}_X(G) = (V, \mathcal{F}_X)$ and $\mathcal{H}_{X'}(G) = (V, \mathcal{F}_{X'})$ only differ in the*

- (a) *neighborhoods such that $N(G) \subset \mathcal{F}_X$ and $N[G] \subset \mathcal{F}_{X'}$, or*
- (b) *symmetric differences of neighborhoods of adjacent vertices such that $\Delta_a[G] \subset \mathcal{F}_X$ and $\Delta_a(G) \subset \mathcal{F}_{X'}$, or*
- (c) *symmetric differences of neighborhoods of non-adjacent vertices such that $\Delta_n(G) \subset \mathcal{F}_X$ and $\Delta_n[G] \subset \mathcal{F}_{X'}$,*

then we have $\mathcal{H}_X(G) \prec \mathcal{H}_{X'}(G)$ in all the above cases which implies in particular $\gamma^{X'}(G) \leq \gamma^X(G)$.

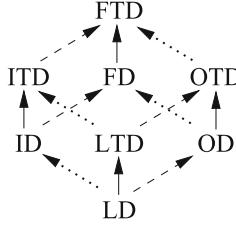


Fig. 3. The relations between the X-numbers for all $X \in \text{CODES}$, where $X' \rightarrow X$ stands for $\gamma^{X'}(G) \leq \gamma^X(G)$ and solid (respectively, dotted and dashed) arrows refer to case (a) (respectively, (b) and (c)) of Corollary 3.

From Corollary 3 combined with the hypergraph representations from Table 2, we immediately obtain the relations between the X-numbers for all $X \in \text{CODES}$ shown in Fig. 3. Note that most of the relations not involving FD- or FTD-numbers were already known before, see, for example, [2, 4]. However, from the global point of view on all X-numbers given in Fig. 3, we see in particular that $\gamma^{\text{LD}}(G)$ is a lower bound for all other X-numbers, whereas $\gamma^{\text{FTD}}(G)$ is an upper bound for all other X-numbers in FTD-admissible graphs. Moreover, we can establish the following relation of FD- and FTD-numbers:

Theorem 2. *Let G be an FD-admissible graph. If G is a disjoint union of a graph G' and an isolated vertex, then we have $\gamma^{\text{FD}}(G) = \gamma^{\text{FTD}}(G') + 1$; otherwise, $\gamma^{\text{FTD}}(G) - 1 \leq \gamma^{\text{FD}}(G) \leq \gamma^{\text{FTD}}(G)$ holds.*

Proof Ideas for Theorem 2: In the case that G is the disjoint union of a graph G' and an isolated vertex v , we can show that $\mathcal{H}_{\text{FD}}(G) = (V, \mathcal{F}_{\text{FD}})$ can be reduced to the hyperedge set $\mathcal{F}_{\text{FTD}} \cup \{v\}$ which implies $\gamma^{\text{FD}}(G) = \gamma^{\text{FTD}}(G') + 1$. If G has no isolated vertices, then G is also FTD-admissible and Corollary 3(a) shows $\gamma^{\text{FD}}(G) \leq \gamma^{\text{FTD}}(G)$. To prove the other inequality, observe that a minimum FD-code C of G is also an FTD-code provided that C is also a total-dominating set of G . One can show that this fails only if one vertex v_C of G satisfies $N(v_C) \cap C = \emptyset$. Including a neighbor of v_C in C results in a minimum FTD-code of G and we, thus, have $\gamma^{\text{FTD}}(G) \leq |C| + 1 = \gamma^{\text{FD}}(G) + 1$. \square

Combining the previous results, we further conclude the following lower bounds on FD- and FTD-numbers.

Corollary 4. *Let G be an FTD-admissible graph. We have*

$$\begin{aligned} \gamma^{\text{FTD}}(G) &\geq \max(\gamma^{\text{ITD}}(G), \gamma^{\text{OTD}}(G), \gamma^{\text{FD}}(G)), \\ \gamma^{\text{FD}}(G) &\geq \max(\gamma^{\text{ID}}(G), \gamma^{\text{OD}}(G), \gamma^{\text{LTD}}(G) - 1, \gamma^{\text{ITD}}(G) - 1, \gamma^{\text{OTD}}(G) - 1). \end{aligned}$$

Some Extremal Cases for Lower and Upper Bounds on FD- and FTD-Numbers. To exhibit some examples of extremal graphs for the bounds on FD- and FTD-numbers established in Theorem 2 and Corollary 4, let us consider the following family of bipartite graphs. A graph $G = (U \cup W, E)$ is *bipartite* if its vertex set can be partitioned into two stable sets U and W so that every edge of G

has one endpoint in U and the other in W . For any integer $k \geq 1$, the *half-graph* $B_k = (U \cup W, E)$ is the bipartite graph with vertices in $U = \{u_1, \dots, u_k\}$, $W = \{w_1, \dots, w_k\}$ and edges $u_i w_j$ if and only if $i \leq j$ (see Fig. 4). In particular, we have $B_1 = K_2$ and $B_2 = P_4$. Moreover, we clearly see that half-graphs are connected and twin-free for all $k \geq 2$ and hence, are both FD- and FTD-admissible in this case.

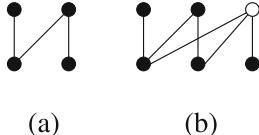


Fig. 4. Minimum FD-codes in half-graphs (the black vertices belong to the code), where (a) is $B_2 = P_4$ and (b) is B_3 .

In [7] it was shown that the only graphs whose OTD-numbers equal the order of the graph are the disjoint unions of half-graphs. In particular, we have $\gamma^{\text{OTD}}(B_k) = 2k$. Combining this result with Corollary 3(b) showing that $\gamma^{\text{OTD}}(G) \leq \gamma^{\text{FTD}}(G)$ holds for all graphs and Theorem 2 showing $\gamma^{\text{FD}}(G + K_1) = \gamma^{\text{FTD}}(G) + 1$ together yields:

Corollary 5. *For a graph $G = (V, E)$ of order n being the disjoint union of half-graphs, we have $\gamma^{\text{FTD}}(G) = n$ and $\gamma^{\text{FD}}(G + K_1) = \gamma^{\text{FTD}}(G) + 1 = n + 1$.*

Hence, there are graphs where the FD- and FTD-numbers equal the order of the graph. A further example is $B_2 = P_4$ with $\gamma^{\text{FD}}(P_4) = 4$. Conversely, we have:

Theorem 3. *For a half-graph B_k with $k \geq 3$, we have $\gamma^{\text{FD}}(B_k) = 2k - 1$.*

Hence, for half-graphs B_k with $k \geq 3$, FD- and FTD-numbers differ. This result combined with Theorem 2 shows that $G = B_k + K_1$ for $k \geq 3$ are examples of graphs where $\gamma^{\text{FD}}(G)$ is larger than the sum of the FD-numbers of its components. So far, this behavior was only observed for OD-numbers, see [4].

Moreover, half-graphs are extremal graphs for the lower bounds $\gamma^{\text{OTD}}(G) \leq \gamma^{\text{FTD}}(G)$ and $\gamma^{\text{OD}}(G), \gamma^{\text{OTD}}(G) - 1 \leq \gamma^{\text{FD}}(G)$ (the latter can be deduced by combining Theorem 3 with the results from [7] showing $\gamma^{\text{OTD}}(B_k) = 2k$ and from [4] showing $\gamma^{\text{OD}}(B_k) = 2k - 1$, respectively, for all $k \geq 1$).

3 Complexity of the FD and FTD-Problems

In this section, we address the complexity of the FD- and FTD-problems. The decision versions of these two problems are formally stated as follows.

FD

Input: (G, k) : An FD-admissible graph G and a positive integer k .

Question: Does there exist an FD-code C of G such that $|C| \leq k$?

FTD

Input: (G, k) : An FTD-admissible graph G and a positive integer k .

Question: Does there exist an FTD-code C of G such that $|C| \leq k$?

We prove the above two problems to be NP-complete. Moreover, despite a difference of at most one as established in Theorem 2, we show that it is hard to decide if the FD- and FTD-numbers of a given input graph are the same or different. The decision version of this problem is defined as follows.

FD \neq FTD

Input: An FTD-admissible graph G and an integer k .

Question: Is $\gamma^{\text{FTD}}(G) = k$ and $\gamma^{\text{FD}}(G) = k - 1$? That is, are the following assertions true?

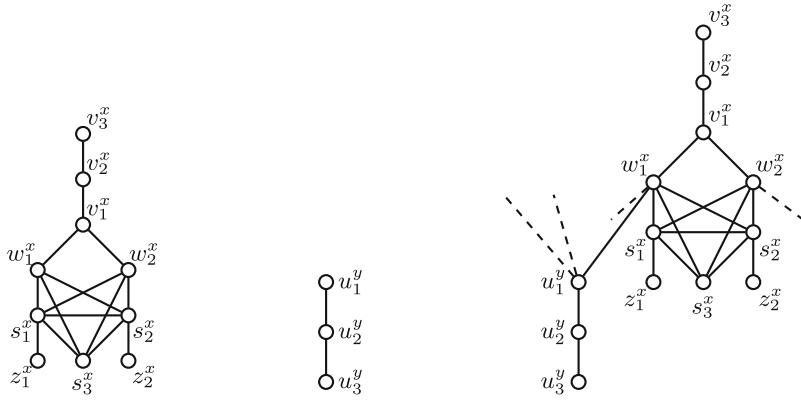
- (a) There exists an FTD-code S of G such that $|S| = k$.
- (b) There exists an FD-code S' of G such that $|S'| = k - 1$.
- (c) For any vertex subset S'' of G , if $|S''| < |S|$, then S'' is not an FTD-code of G ; and if $|S''| < |S'|$, then S'' is not an FD-code of G .

As can be noticed in FD \neq FTD, given a vertex subset S (with $|S| = k$) of an input graph G on n vertices, to check condition (c), one has to consider all subsets of $V(G)$ of order at most $k - 1$ (which admits $\mathcal{O}(n^{k-1})$ as a worst-case running time) and check if S is an FTD-code (which admits $\mathcal{O}(n^2)$ as a running time). In other words, verifying a certificate for FD \neq FTD can potentially take upto a running time of order $\mathcal{O}(n^{k-1}) \cdot \mathcal{O}(n^2) \subseteq \mathcal{O}(n^{k+1})$ which may not be polynomial in $n + k$. Moreover, since $|S| \in \mathcal{O}(n)$, it implies that verifying a certificate for this problem can not necessarily be done by a polynomial-time algorithm. This implies that, FD \neq FTD does not necessarily belong to the class NP. However, as we shall see in this section, the problem still remains hard.

To prove the hardness results for the above three problems, we use one single reduction from an instance ψ of 3-SAT to an instance (G^ψ, k) of FD, FTD and FD \neq FTD.

Reduction 1. *The reduction takes as input an instance $\psi = (A, B)$ of 3-SAT, where A denotes the set of variables and B denotes the set of clauses in ψ . Moreover, let $n = |A|$ and $m = |B|$. The reduction constructs a graph G^ψ on $10n + 3m$ vertices as follows. For every variable $x \in A$, we create the variable gadget G^x as shown in Fig. 5(a)); and for every clause $y \in B$, we create a clause gadget P^y as shown in Fig. 5(b)). Finally, for all variables $x \in A$ and all clauses $y \in B$, if the literal x is in a clause y , then add the edge $u_1^y w_1^x$; and if the literal $\neg x$ is in a clause y , then add the edge $u_1^y w_2^x$ (see Fig. 5(c)).*

It can be verified that Reduction 1 is carried out in time polynomial in the inputs m and n . The hardness proofs of the above three problems comprise of first proving the following lemmas.



(a) The variable gadget G^x corresponding to a variable $x \in A$.

(b) The clause gadget P^y corresponding to a clause $y \in B$.

(c) The graph G^ψ constructed from the instance ψ of 3-SAT in Reduction 1.

Fig. 5. Components of Reduction 1.

Lemma 2. *Let ψ have a satisfying assignment. Then there exists an FTD-code C of G^ψ such that $|C| = 7n + 2m$ and an FD-code C' of G^ψ such that $|C'| = 7n + 2m - 1$. In particular, we have $\gamma^{\text{FD}}(G^\psi) = 7n + 2m - 1$ and $\gamma^{\text{FTD}}(G^\psi) = 7n + 2m$.*

Lemma 3. *If there exists an FTD-code C (respectively, an FD-code C') of G^ψ such that $|C| \leq 7n + 2m$ (respectively, $|C'| \leq 7n + 2m - 1$), then ψ has a satisfying assignment. In particular, if $\gamma^{\text{FD}}(G^\psi) = 7n + 2m - 1$ and $\gamma^{\text{FTD}}(G^\psi) = 7n + 2m$, then ψ has a satisfying assignment.*

Proof Ideas of Lemmas 2 and 3. We give the proof ideas by arguing how many vertices must be necessarily included in an FTD-code S of G^ψ . The arguments for an FD-code are similar. Since u_3^y, v_3^x, z_1^x and z_2^x are pendant vertices, all their neighbors must be in S . Moreover, the vertices u_1^y, v_1^x, z_1^x and z_2^x must be in S to full-separate the pairs $(u_2^y, u_3^y), (v_2^x, v_3^x), (s_1^x, s_3^x)$ and (s_2^x, s_3^x) , respectively. Finally, at least one of w_1^x and w_2^x must be in S to full-separate the pair (v_1^x, v_3^x) . This implies that $|S \cap V(G^x)| \geq 7$ and $|S \cap V(P^y)| \geq 2$. Hence, any FTD-code S must have at least $7n + 2m$ vertices.

Now, to prove Lemma 2 for FTD-codes, given a satisfying assignment ψ , in order to construct an FTD-code S of G^ψ , the choice to include either w_1^x or w_2^x in S is made according to which of x and $\neg x$, respectively, is assigned to 1. Then it can be checked that S , along with all the other previously argued vertices that must be included in it, is an FTD-code of order $7n + 2m$. On the other hand, having chosen a minimum FTD-code S of order $7n + 2m$, we define an assignment ψ on 3-SAT by designating a variable x to 1 or 0 according to whether w_1^x or w_2^x is in S . Again, it can be checked that ψ defines a satisfying assignment since for each clause y , at least one of w_1^x or w_2^x is in S in order to full-separate (u_1^y, u_3^y) , where x is a variable in the clause y . This proves Lemma 3 for FTD-codes. \square

Theorem 4. FTD is NP-complete.

Proof. The problem FTD clearly belongs to the class NP. Moreover, FTD is NP-hard since, by Lemmas 2 and 3, ψ is a satisfying assignment if and only if there exists an FTD-code C of G^ψ such that $|C| \leq 7n + 2m$. \square

Theorem 5. FD is NP-complete.

Proof. The problem FD clearly belongs to the class NP. Moreover, FD is NP-hard since, by Lemmas 2 and 3, ψ is a satisfying assignment if and only if there exists an FD-code C of G^ψ such that $|C| \leq 7n + 2m - 1$. \square

Theorem 6. $\text{FD} \neq \text{FTD}$ is NP-hard.

Proof. The problem $\text{FD} \neq \text{FTD}$ is NP-hard since, by the last statements of Lemma 2 and 3, ψ is a satisfying assignment if and only if $\gamma^{\text{FD}}(G^\psi) = 7n + 2m - 1 = \gamma^{\text{FTD}}(G^\psi) - 1$. \square

4 Concluding Remarks

In this paper, we studied problems in the domain of identification problems. The contribution of this paper is to introduce a new separation property, called full-separation, and to study it in combination with both domination and total-domination. We characterized full-separating sets in different ways to show that they combine the requirements of both closed- and open-separation which justifies in particular how the full-separation property models monitoring systems that tackle the two studied fault types simultaneously. We addressed questions concerning the existence of FD- and FTD-codes in graphs, the relations between all X-numbers and particularly between the FD- and FTD-numbers. We also provided some extremal examples of graphs (in the form of half-graphs and their disjoint unions) with respect to upper bounds on the FD- and FTD-numbers. We further showed that deciding if the FD- or FTD-number of a graph is below a given value is NP-complete. Moreover, it is in general NP-hard to decide if the two numbers differ on a graph.

Our lines of future research include to pursue these studies of the FD- or FTD-codes by involving more graph families, for example, to find more examples of graphs where the FD- or FTD-numbers are not close to the order of the graph. In general it would be interesting to identify or even characterize the extremal cases, that is, graphs for which the FD- or FTD-numbers equal the upper bound (the order of the graph) or one of its lower bounds in terms of other X-numbers.

Finally, it would be interesting to study a broader set of problem variants in the area of domination/separation from a hypergraph point of view towards a more general classification of such problems. In this context, it is clear that the hypergraphs encoding domination (respectively, total-domination) are only composed of closed (respectively, open) neighborhoods, whereas hypergraphs encoding only the separation properties only contain the corresponding symmetric differences of neighborhoods for all pairs of vertices. Here, it would be interesting to see if other restrictions (for example, only to $\Delta_a[G]$ or only to $N(G)$ and $\Delta_a(G)$) encode further studied problems from the literature and then similar relations as in Fig. 3 can be deduced to a large set of graph parameters.

Acknowledgments. This research was financed by a public grant overseen by the French National Research Agency as part of the “Investissements d’Avenir” through the IMobS3 Laboratory of Excellence (ANR-10-LABX-0016), by the French government IDEX-ISITE initiative 16-IDEX-0001 (CAP 20-25) and the International Research Center “Innovation Transportation and Production Systems” of the I-SITE CAP 20-25.

References

1. Argiroffo, G., Bianchi, S., Lucarini, Y., Wagler, A.K.: Polyhedra associated with identifying codes in graphs. *Discret. Appl. Math.* **245**, 16–27 (2018)
2. Argiroffo, G., Bianchi, S., Lucarini, Y., Wagler, A.K.: Polyhedra associated with locating-dominating, open locating-dominating and locating total-dominating sets in graphs. *Discret. Appl. Math.* **322**, 465–480 (2022)
3. Argiroffo, G., Bianchi, S., Wagler, A.K.: Progress on the description of identifying code polyhedra for some families of split graphs. *Discret. Optim.* **22**, 225–240 (2016)
4. Chakraborty, D., Wagler, A.K.: Open-Separating Dominating Codes in Graphs. In: Basu, A., Mahjoub, A.R., Salazar González, J.J. (eds.) ISCO 2024. LNCS, vol. 14594, pp. 137–151. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-60924-4_11
5. Charon, I., Hudry, O., Lobstein, A.: Minimizing the size of an identifying or locating-dominating code in a graph is NP-hard. *Theoret. Comput. Sci.* **290**, 2109–2120 (2003)
6. Colburn, C., Slater, P.J., Stewart, L.K.: Locating-dominating sets in series-parallel networks. *Congr. Numer.* **56**, 135–162 (1987)
7. Foucaud, F., Ghareghani, N., Roshani-Tabrizi, A., Sharifani, P.: Characterizing extremal graphs for open neighbourhood location-domination. *Discret. Appl. Math.* **302**, 76–79 (2021)
8. Haynes, T.W., Henning, M.A., Howard, J.: Locating and total-dominating sets in trees. *Discret. Appl. Math.* **154**, 1293–1300 (2006)
9. Honkala, I., Laihonen, T., Ranto, S.: On strongly identifying codes. *Discret. Math.* **254**(1–3), 191–205 (2002)
10. Jean, J., Lobstein, A.: Watching systems, identifying, locating-dominating and discriminating codes in graphs, <https://dragazo.github.io/bibdom/main.pdf>
11. Karpovsky, M.G., Chakrabarty, K., Levitin, L.B.: On a new class of codes for identifying vertices in graphs. *IEEE Trans. Inf. Theory* **44**(2), 599–611 (1998)
12. Seo, S.J., Slater, P.J.: Open neighborhood locating dominating sets. *Australas. J. Comb.* **46**, 109–119 (2010)
13. Slater, P.J.: Dominating and reference sets in a graph. *J. Math. Phys. Sci.* **22**, 445–455 (1988)
14. Ungrangsi, R., Trachtenberg, A., Starobinski, D.: An implementation of indoor location detection systems based on identifying codes. In: Aagesen, F.A., Anutariya, C., Wuwongse, V. (eds.) INTELLCOMM 2004. LNCS, vol. 3283, pp. 175–189. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30179-0_16



Polynomial Time Algorithms for Hop Domination

D. Karthika¹(✉), R. Muthucumaraswamy¹, Sriram Bhyravarapu²,
and Pritesh Kumar²

¹ Department of Mathematics, Sri Venkateswara College of Engineering,
Sriperumbudur, Kanchipuram 602117, India
{2022pm0001,msamy}@svce.ac.in

² The Institute of Mathematical Sciences, HBNI, Chennai, India
{sriramb,priteshk}@imsc.res.in

Abstract. A set $S \subseteq V(G)$ is said to be a hop dominating set if every vertex $u \in V(G) \setminus S$, there exists a vertex $v \in S$ such that $d(u, v) = 2$ where $d(u, v)$ represents the distance between u and v in G . The minimum k for which there exists a hop dominating set of size k is called the *hop domination number* denoted by $\gamma_h(G)$. Henning et al. (Inf. Process. Lett. 2020) showed that HOP DOMINATION is NP-hard for bipartite graphs and chordal graphs. The following are the results of this paper.

- Henning et al. presented a linear time algorithm for solving HOP DOMINATION on bipartite permutation graphs which is a proper subset of biconvex bipartite graphs. In this paper we present a polynomial algorithm for the problem on biconvex bipartite graphs, a superclass of bipartite permutation graphs.
- We show that HOP DOMINATION is polynomial time solvable on interval graphs which are a subclass of chordal graphs.
- We initiate the study on this problem from the parameterized complexity perspective. We show that the decision version of HOP DOMINATION is W[1]-hard when parameterized by solution size.

Keywords: domination • hop domination • interval graphs • biconvex bipartite graphs

1 Introduction

Given a graph $G = (V, E)$, a set $D \subseteq V(G)$ is said to be a *dominating set* if for every vertex v in G , either $v \in D$ or v has at least one neighbor in D . The minimum cardinality of a dominating set is called the *domination number* of G and is denoted by $\gamma(G)$. Two vertices u and v in a graph G are said to k -step dominate each other if $d_G(u, v) = k$, where $k \geq 1$ is an integer. A set $S \subseteq V(G)$ is called a k -step dominating set of G if every vertex of G is k -step dominated by some vertex in S . The k -step dominating set problem asks to find a minimum size of a k -step dominating set. Chartrand et al. (1995) introduced and studied 2-step domination in graphs [1]. A generalization of the k -step domination problem

called the *step domination* problem has been studied where the input is a graph G , and integer k_i associated with each vertex $v_i \in V(G)$. The objective is to find a set $S \subseteq V(G)$ of minimum size such that $N_{k_i}(S)$ forms a partition of $V(G)$, where $N_k(v)$ denotes the set of vertices at distance k from v . Hersh [2] generalized to n steps the notion of exact 2-step domination introduced by Chartrand et al. [1].

A set $S \subseteq V(G)$ is said to be a *hop dominating set* if every vertex $u \in V(G) \setminus S$, there exists a vertex in $v \in S$ such that u is 2-step dominated by v . Given a graph G , the HOP DOMINATION problem asks to find the minimum k for which there exists a hop dominating set of size k , called the *hop domination number* denoted by $\gamma_h(G)$. Several bounds on this problem have been studied [3–5].

Henning et al. [5] presented upper bounds for the hop domination number of a graph. Further, they showed that the decision version of HOP DOMINATION is NP-complete for planar bipartite graphs and planar chordal graphs. Henning et al. [6] presented a linear time algorithm to compute the minimum hop domination number in bipartite permutation graphs. Further, they showed that the decision version of HOP DOMINATION is NP-complete for perfect elimination graphs. Also, they showed that the hop domination number could not be approximated within a factor of $(1 - \epsilon) \log |V|$, unless P=NP and gave an approximation algorithm with a factor of $1 + \log(\Delta(\Delta-1)+1)$, where Δ is the maximum degree in G . Further, they showed that HOP DOMINATION is APX-complete for bipartite graphs of maximum degree 3.

Our results and Discussion. In this paper we extend the study on HOP DOMINATION. Henning et al. [6] showed that HOP DOMINATION is NP-hard on chordal graphs and bipartite graphs. We show that HOP DOMINATION is polynomial time solvable on interval graphs. The main ingredient of the algorithm is the structural property admitted by interval graphs called “multi-chain ordering”. This result is presented in Sect. 4. Henning et al. [6] presented a linear time algorithm to compute the minimum hop domination number in bipartite permutation graphs. We consider biconvex bipartite graphs a superclass of bipartite permutation graphs and give a polynomial time algorithm. This result is presented in Sect. 5.

We initiate the study of parameterized complexity on this problem. We show that HOP DOMINATION is W[1]-hard parameterized by solution size and this result is presented in Sect. 3.

As open questions and future directions, one could explore the parameterized complexity of the problem on graphs of bounded treewidth, modular width, distance to cluster and cliquewidth. Another direction could be to obtain polynomial kernels for the problem with respect to various structural parameters. It will also be interesting to find the complexity of the problem on split graphs, distance hereditary graphs, etc.

2 Preliminaries

Let $G = (V, E)$ be a graph with vertex set $V = V(G)$ and edge set $E = E(G)$. A set of vertices $I \subseteq V(G)$ is said to be *independent* if no two vertices in I are adjacent to each other. A *bipartite graph* $G = (X, Y, E)$ is a graph in which the vertex set $V(G)$ can be partitioned into two sets X and Y such that every edge connects a vertex in X to a vertex in Y . The distance between two vertices u and v in a graph G is denoted by $d(u, v)$ which is the number of edges on a shortest path between them. The diameter $\text{diam}(G)$ of a graph G is the maximum distance over all pairs of vertices $u, v \in V(G)$. We denote the set $\{1, 2, \dots, p\}$ with $[p]$ and the set $\{a, a+1, \dots, b\}$ with $[a, b]$. A vertex $v \in V(G)$ is said to be *universal* if for every $u \in V(G) \setminus \{v\}$, $uv \in E(G)$.

In parameterized complexity, in addition to the input we are given a parameter k and the objective is to obtain an algorithm running in time $f(k)n^{O(1)}$ where f is some computable function dependent only on k and n is the size of the input. We say a problem belongs to the class *fixed parameter tractable* if there exists an algorithm running in above mentioned time. Otherwise, the problem belongs to the class W-hard. For more details the reader is directed to [7]. The proofs of the results marked (\star) are presented in the full version of the paper. We now look at the following observation, the proof of which is trivial.

Observation 1. *If u is universal in G , then any hop dominating set contains u .*

3 W[1]-Hard Parameterized by Solution Size

In this section, we initiate the study of the problem from a parameterized complexity perspective. As a first step, we study the problem with respect to the natural parameter, the solution size. We show that HOP DOMINATION is W[1]-hard when parameterized by solution size. Towards this, we show a reduction from MULTICOLORED INDEPENDENT SET which takes as input a graph G , an integer k , and a partition of $V(G)$ into k sets V_1, V_2, \dots, V_k and the objective is to find an independent set $S \subseteq V(G)$ of size k such that $|S \cap V_i| = 1$ for each i .

Theorem 1. *HOP DOMINATION is W[1]-hard parameterized by solution size.*

Proof. We give a reduction from MULTICOLORED INDEPENDENT SET to HOP DOMINATION. Let $G = (V, E)$ and a partition V_1, V_2, \dots, V_k of $V(G)$ be an instance of MULTICOLORED INDEPENDENT SET. We assume that $k \geq 3$ and each V_i is an independent set, for all i . We construct a graph G' from G as follows. An illustration of the gadget is given in Fig. 1.

1. Add a copy of V_1, V_2, \dots, V_k to G' .
2. For each pair i and j , $i \neq j$, each vertex in V_i is adjacent to every vertex in V_j .
3. For each i , add a vertex u_i and make it adjacent to all vertices in V_i .

4. For each i , add two vertices u_i^1 and u_i^2 , and add the edges $u_i u_i^1$, $u_i u_i^2$ and $u_i^1 u_i^2$.
5. For each pair of vertices $u \in V_i$ and $v \in V_j$ such that $uv \in E(G)$, we add a vertex e_{uv} and make it adjacent to all vertices in $(V_i \cup V_j) \setminus \{u, v\}$.
6. For each vertex e_{uv} , add a vertex e'_{uv} and make it adjacent to e_{uv} .

Claim 1. Let D be a hop dominating set of G' of size k then $|D \cap \{V_i, u_i, u_i^1, u_i^2\}| = 1$ for all i . Further $|D \cap V_i| = 1$, for all i .

Proof. Let $A_i = \{V_i, u_i, u_i^1, u_i^2\}$ for all i . Consider a partition V_i . If D hop dominates the vertices u_i^1 and u_i^2 , it is the case that either $\{u_i^1, u_i^2\} \subseteq D$ or there exist $q_i \in D \cap V_i$.

Suppose $\{u_i^1, u_i^2\} \subseteq D$. Since cardinality of D is k , there exists some j such that $|D \cap \{V_j, u_j, u_j^1, u_j^2\}| = \emptyset$. The vertices u_j^1 and u_j^2 are at distance at least three from $V(G) \setminus A_j$ and thus are not hop dominated which is a contradiction. Thus it is the case that $q_i \in D \cap V_i$.

Suppose $|D \cap V_i| > 1$. Since D is of size k , there exist some j such that $V_j \cap D = \emptyset$ and the vertices u_j^1 and u_j^2 are not hop dominated which is a contradiction. \square

Claim 2. Let D be a hop dominating set of G' of size k . Then for each $uv \in E(G)$, $|D \cap \{u, v\}| \leq 1$.

Proof. From Claim 1, it is clear that $|D \cap V_i| = 1$ for each i . Let $u \in V_i$ and $v \in V_j$. Consider the vertex e_{uv} . Suppose for a contradiction that $\{u, v\} \subseteq D$. Then the vertex e'_{uv} is not hop dominated by any of the vertices in D . \square

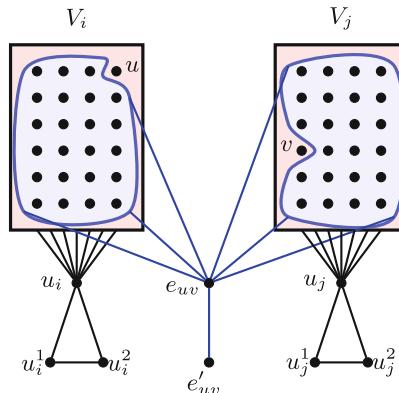


Fig. 1. An illustration of the edges between a pair of sets V_i and V_j . The edges between V_i and V_j are not indicated to avoid clutter.

Lemma 1. *G has a multicolored independent set of size k if and only if G' has a hop dominating set of size k.*

Proof. Suppose G has a multicolored independent set S of size k . Let $S = \{q_1, q_2, \dots, q_k\}$ where $q_i \in V_i$ for all i . We claim that S is a hop dominating set for G' . The vertices u_i^1 and u_i^2 are hop dominated by q_i because u_i is universal to V_i and $q_i \in V_i$. Each vertex u_i is hop dominated by q_j for any $j \neq i$ because $u_i q_j \in E(G')$ and q_j is universal to V_j . For each vertex $w \in V_i \setminus \{q_i\}$, we have that w is hop dominated by q_i because $w u_i \in E(G')$ and u_i is universal to V_i . For each e_{uv} , where $u \in V_i$ and $v \in V_j$, we have that q_j hop dominates e_{uv} where $i \neq j' \neq j$, as e_{uv} is at a distance two from q_j . The existence of j' is guaranteed because $k \geq 3$. Now consider the vertex e'_{uv} . Since S is an independent set of G , it is the case that $|S \cap \{u, v\}| \leq 1$. Without loss of generality let $v \notin S$. Then e'_{uv} is hop dominated by q_j because e'_{uv} is adjacent to e_{uv} and e_{uv} is universal to V_j except v , in particular q_j .

We now prove the converse. Suppose D is a hop dominating set for G' of size k . We show that D is a k -multicolored independent set for G . It is sufficient to show that for all pairs of vertices u and v in D either e_{uv} does not exist in G' or $|\{u, v\} \setminus (D \cap \{u, v\})| \geq 1$. The former case indicates that $uv \notin E(G)$ while the latter case indicates that at most one of u and v are in D . If $e_{uv} \notin V(G')$, then we are done. Suppose that $e_{uv} \in V(G')$. From Claims 1 and 2, we get that at most one of u and v are in D and thus D is an independent set in G . \square

From Lemma 1, and the fact that MULTICOLORED INDEPENDENT SET is W[1]-complete [7], we obtain our desired result. \square

4 Interval Graphs

In this section, we show that HOP DOMINATION can be solved in polynomial time on interval graphs. Before we look at the algorithm, we look at the definition of multi-chain ordering which is crucial to our algorithm.

A bipartite graph $G = (A \cup B, E)$ is called a *chain graph* if for every two vertices $u_1, u_2 \in A$ we have either $N(u_1) \subseteq N(u_2)$ or $N(u_2) \subseteq N(u_1)$. We can see that each part of a chain graph can be linearly ordered under the inclusion of their neighborhoods.

Definition 1 (Multi-chain ordering [8]). *Let $G = (V, E)$ be a connected graph and a partition of $V(G)$ say L_0, L_1, \dots, L_{q-1} be distance layers of $V(G)$ starting from a vertex $v_0 \in V(G)$ where $L_0 = \{v_0\}$. The layer L_i , where $i \in [q-1]$, denotes the set of vertices that are at a distance i from v_0 . Note that q here denotes the largest integer such that L_{q-1} is non-empty. We say that these layers form a multi-chain ordering of G if for every two consecutive layers L_i and L_{i+1} , where $i \in \{0, 1, \dots, q-1\}$, we have that the vertices in L_i and L_{i+1} , and the edges connecting these layers form a chain graph.*

As a consequence of the above definition, we have the following. For each $i \in [q-2]$, since $G[L_i \cup L_{i-1}]$ and $G[L_i \cup L_{i+1}]$ induce a chain graph, there are two orderings one with respect to L_{i-1} and another with respect to L_{i+1} . In the former ordering (resp. latter case), there exists a vertex, say $\alpha_i \in L_i$ (resp. $\beta_i \in L_i$) such that $N(\alpha_i)$ (resp. $N(\beta_i)$) contains every vertex in L_{i-1} (resp. L_{i+1}) that has an edge to a vertex in L_i . Notice that α_i and β_i could be the same vertex. Since L_0 and L_{q-1} are the first and the last layers there exists only one ordering for each of them and thus we β_0 and α_{q-1} respectively. It is known that all connected interval graphs admit multi-chain ordering [8]. Thus we assume that G admits a multi-chain ordering $\mathcal{L} = \{L_0, L_1, \dots, L_{q-1}\}$ on q layers that can be computed in polynomial time.

Definition 2 (Interval graph). A graph $G = (V, E)$ is an interval graph if there exists a set \mathcal{I} of closed intervals on the real line such that there is a bijection $f : V \rightarrow \mathcal{I}$ such that $uv \in E(G)$ if and only if $f(u) \cap f(v) \neq \emptyset$.

For each interval $I \in \mathcal{I}$, we use $L(I)$ and $R(I)$ to denote the left and right endpoints of I respectively. Without loss of generality, we can assume that no two intervals share the same endpoint. We designate two special intervals I_ℓ and I_r where $R(I_\ell) < R(I')$, for every $I' \in \mathcal{I} \setminus I_\ell$ and $L(I_r) > L(I')$, for every $I' \in \mathcal{I} \setminus I_r$. That is I_ℓ is the interval with the least right endpoint and I_r is the interval that has the largest left endpoint.

Lemma 2. Let G be an interval graph with $\text{diam}(G)=3$. Then there exists a hop dominating set $S \subseteq V(G)$ such that $|S| \leq 4$.

Proof. Let G be a connected interval graph and \mathcal{I} be an interval representation of G . Recall that $R(I_\ell) < R(I')$, for every $I' \in \mathcal{I} \setminus I_\ell$ and $L(I_r) > L(I')$, for every $I' \in \mathcal{I} \setminus I_r$. By the definition of I_ℓ , I_r and $\text{diam}(G)=3$, we have $d(I_\ell, I_r) = 3$. Then there exist two intervals I_1 and I_2 such that the graph induced on the intervals I_ℓ, I_1, I_2, I_r is a path.

Let **class- a** be the set of intervals in \mathcal{I} that are adjacent to I_ℓ , **class- b** be the set of intervals in \mathcal{I} that are not adjacent to I_ℓ but are adjacent to I_1 , and **class- c** be the set of intervals in \mathcal{I} that are not adjacent to I_ℓ and I_1 . An illustration of the classes is given in Fig. 2. It is easy that any interval other than I_ℓ, I_1, I_2 and I_r belongs to one of these classes.

Claim. Let $S = \{I_\ell, I_1, I_2, I_r\}$. Then S is a hop dominating set.

Proof. Consider an interval $I \in \mathcal{I} \setminus \{S\}$. We now argue that I is hop dominated based on the class it belongs to.

- I belongs to **class- a** .

We first consider when I does not intersect with I_2 . From the definition of I_ℓ and the case assumption, we have that I is adjacent to I_ℓ and I_1 . Thus I is hop dominated by I_2 . Now we consider the case when I intersects with I_2 .

Notice that I does not intersect with I_r otherwise $\text{diam}(G)=2$. Therefore I is hop dominated by I_r . Every interval of **class- a** is hop dominated by either I_2 or I_r .

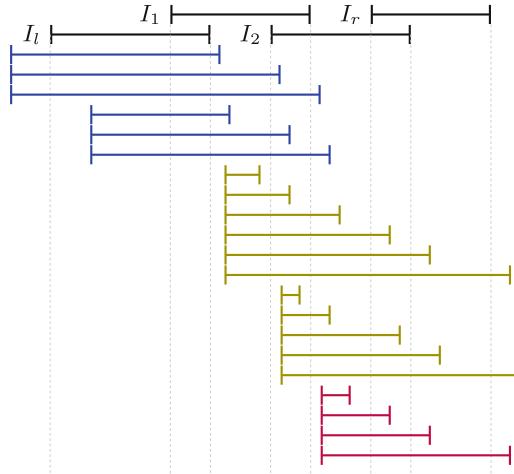


Fig. 2. Interval representation of an interval graph G with $\text{diam}(G) = 3$. The intervals colored in blue, olive and purple denote the intervals of class- a , class- b and class- c respectively.

– I belongs to class- b .

From the definition of class- b the interval I does not intersect with I_ℓ and intersects I_1 . Therefore I is hop dominated by I_ℓ .

– I belongs to class- c .

From the definition of class- c the interval I does not intersect with I_ℓ and I_1 but intersects with I_2 . Therefore I is hop dominated by I_1 .

Hence the claim holds. \square

We consider the vertices of G corresponding to the set of intervals in S and such a vertex set is a hop dominating set of G of size at most four. \square

Lemma 3 (★). *Let G an interval graph with $\text{diam}(G)=2$. Then there exists a hop dominating set $S \subseteq V(G)$ such that $|S| \leq |U| + 2$, where U is the set of all universal vertices of G .*

Observation 2 (★). *Let G be an interval graph that admits a multi-chain ordering $\mathcal{L} = \{L_0, L_1, \dots, L_{q-1}\}$ on $q \leq 3$ layers. Then $\text{diam}(G) \leq 3$.*

Lemma 4 (★). *Let G be an interval graph that admits a multi-chain ordering $\mathcal{L} = \{L_0, L_1, \dots, L_3\}$ on four layers. Then there exists a hop dominating set S such that $|S| \leq 4$.*

Lemma 5. *Let G be an interval graph that admits a multi-chain ordering $\mathcal{L} = \{L_0, L_1, \dots, L_{q-1}\}$ on q layers. Let S be a hop dominating set of G . Then there exists a hop dominating set S' of G of size at most $|S|$ such that $|(L_i \cup L_{i+1} \cup L_{i+2} \cup L_{i+3} \cup L_{i+4}) \cap S'| \leq 8$ for all $i \in \{0, 1, \dots, q-5\}$.*

Proof. If S satisfies the desired constraints then the claim trivially holds. Therefore we assume that S does not satisfy the constraints of the lemma. We construct another hop dominating set S' from S satisfying the constraints of the lemma. Let $S' = S$ and $\mathcal{L}_i = L_i \cup L_{i+1} \cup L_{i+2} \cup L_{i+3} \cup L_{i+4}$ for each $i \in [0, q-5]$ denote the set of vertices in the layers L_j where $j \in [i, i+4]$.

We process the layers in chunks of five starting from the last layer L_{q-1} . Consider the largest $i \in [0, q-5]$ such that $|\mathcal{L}_i \cap S| > 8$. We have two cases depending on whether $i = 0$ or not.

Case: $i \neq 0$.

Let $S' = (S' \setminus \mathcal{L}_i) \cup S_i$ where $S_i = \{\beta_{i-1}, \alpha_i, \beta_i, \alpha_{i+1}, \beta_{i+1}, \alpha_{i+2}, \beta_{i+2}, \beta_{i+3}, \beta_{i+4}\}$. Notice that $|S'| \leq |S|$. We now show the following claim.

Claim. S' is a hop dominating set of G .

Proof. Consider a vertex $v \in L_j \setminus S'$ where $j \in [0, i-3] \cup [i+7, q-1]$. Since the distance between v and the vertices in \mathcal{L}_i is at least three, the vertex v is hop dominated by $S' \setminus \mathcal{L}_i$. Now we consider the vertices in the layers L_j where $j \in [i-2, i+6]$ that do not belong to the set S' . If $v \in L_j$ for all $j \in [i+1, i+6]$, then the vertex β_{j-2} hop dominates v as $\beta_{j-2} \in S'$.

– $v \in L_{i-2}$.

If v was hop dominated by a vertex in $S' \setminus \mathcal{L}_i$, it continues to be hop dominated by the same vertex. Otherwise, v is hop dominated by a vertex say $w \in L_i$. Therefore exists a vertex $z \in L_{i-1}$ such that $vz \in E(G)$ and $zw \in E(G)$. Since α_i is adjacent to z and $\alpha_i \in S'$ we have that v is hop dominated by α_i .

– $v \in L_{i-1}$.

We argue this based on the following sub-cases. If $v\beta_{i-1} \notin E(G)$ then it is hop dominated by β_{i-1} because β_{i-1} is adjacent to β_{i-2} , and β_{i-2} is adjacent to v . Since $\beta_{i-1} \in S'$ and $v \notin S'$ we have that $|L_{i-1}| \geq 2$ and thus L_{i-1} is not the layer L_0 . This exhibits the existence of the vertex β_{i-2} .

Else if $v\beta_{i-1} \in E(G)$ and $v\beta_i \notin E(G)$ then v is hop dominated by β_i . Otherwise v is hop dominated by α_{i+1} , as α_{i+1} is adjacent to β_i .

– $v \in L_i$.

We argue this based on the following sub-cases. If $v\beta_i \notin E(G)$ then v is hop dominated by β_i because β_i is adjacent to β_{i-1} , and β_{i-1} is adjacent to v . Else if $v\beta_i \in E(G)$ and $v\beta_{i+1} \notin E(G)$ then v is hop dominated by β_{i+1} . Otherwise v is hop dominated by α_{i+2} , as α_{i+2} is adjacent to β_{i+1} .

□

Case: $i = 0$.

Let $S' = (S' \setminus \mathcal{L}_0) \cup S_0$ where $S_0 = \{\beta_0, \alpha_1, \beta_1, \alpha_2, \beta_2, \alpha_3, \beta_3, \beta_4\}$. Using similar arguments as before, we now show that S' is a hop dominating set.

Consider a vertex $v \in L_j \setminus S'$ where $j \in [7, q-1]$. Since the distance between v and the vertices in \mathcal{L}_0 is at least three, the vertex v is hop dominated by $S' \setminus \mathcal{L}_0$. Now we consider the vertices in the layers L_j where $j \in [2, 6]$. Each vertex in this set is hop dominated by β_{j-2} . The vertex in $L_0 \in S'$. Now consider a vertex

$v \in L_1 \setminus S'$. If $v\beta_1 \notin E(G)$, then v is hop dominated by β_1 . Else if $v\beta_1 \in E(G)$ and $v\beta_2 \notin E(G)$, then v is hop dominated by β_2 . Otherwise v is hop dominated by α_3 . \square

Theorem 2. HOP DOMINATION is polynomial time solvable on interval graphs.

Proof. Let G be an interval graph and $\mathcal{L} = \{L_0, L_1, \dots, L_{q-1}\}$ be a multi-chain ordering of G with q layers. Let S be the minimum hop dominating set for G . We design an algorithm based on the value of q .

Case: $q \leq 3$.

From Observation 2, we get that the diameter of G is at most three. Now we have the following cases. When $\text{diam}(G) = 1$ then G is clique and thus from Observation 1 we have that $S = V(G)$.

Consider $\text{diam}(G) = 2$ and let U be the set of universal vertices of G . From Observation 1 we have that $U \subseteq S$. From Lemma 3 there exists a hop dominating set of size at most $|U| + 2$. Thus any minimum hop dominating set contains at most two vertices from $V(G) \setminus U$. We go over all subsets of $V(G) \setminus U$ of size at most two and check whether it is a hop dominating set.

Consider $\text{diam}(G) = 3$. From Lemma 2 we get that any minimum hop dominating set S has $|S| \leq 4$. We go all subsets of $V(G)$ of size at most four and check whether it is a hop dominating set, and obtain our desired set.

Case: $q = 4$.

From Lemma 4, we get that the size of a minimum hop dominating set is at most four. We consider all subsets of $V(G)$ of size at most four and check whether it is a hop dominating set of G , and obtain our desired set.

Case: $q \geq 5$.

Let S be a minimum hop dominating set of G . Then by Lemma 5, there exists a hop dominating set S' of size at most S such that $|(\cup L_i \cup L_{i+1} \cup L_{i+2} \cup L_{i+3} \cup L_{i+4}) \cap S'| \leq 8$ for each $i \in [0, q-5]$ and thus we assume $S = S'$. Notice that the vertices in L_i can be hop dominated only by the vertices from \mathcal{M}_i . Therefore, for each $i \in [2, q-3]$, we define a set of tuples $\mathcal{T}_i = \{(t_{i-2}, t_{i-1}, t_i, t_{i+1}, t_{i+2}) \mid t_j \in [0, 8] \text{ for each } j \in [i-2, i+2]\}$ which captures the number of solution vertices from the layer L_j for each $j \in [i-2, i+2]$.

Now for the layers L_0 , L_1 , L_{q-2} and L_{q-1} , we define a set of tuples $\mathcal{T}_0 = \{(t_0, t_1, t_2) \mid t_j \in [0, 8], \forall j \in [0, 2]\}$, $\mathcal{T}_1 = \{(t_0, t_1, t_2, t_3) \mid t_j \in [0, 8], \forall j \in [0, 3]\}$, $\mathcal{T}_{q-2} = \{(t_{q-4}, t_{q-3}, t_{q-2}, t_{q-1}) \mid t_j \in [0, 8], \forall j \in [q-4, q-1]\}$ and $\mathcal{T}_{q-1} = \{(t_{q-3}, t_{q-2}, t_{q-1}) \mid t_j \in [0, 8], \forall j \in [q-3, q-1]\}$ respectively.

For each tuple $T = (t_{i-2}, t_{i-1}, t_i, t_{i+1}, t_{i+2}) \in \mathcal{T}_i$ where $i \in [2, q-3]$, we have set of tuples \mathcal{B}_T defined as $\mathcal{B}_T = \{(B_{i-2}, B_{i-1}, B_i, B_{i+1}, B_{i+2}) \mid B_j \subseteq L_j \text{ and } |B_j| = t_j \text{ for all } j \in [i-2, i+2]\}$.

Similarly for a tuple $T = (t_0, t_1, t_2) \in \mathcal{T}_0$ we define $\mathcal{B}_T = \{(B_0, B_1, B_2) \mid B_j \subseteq L_j \text{ and } |B_j| = t_j \text{ for all } j \in [0, 2]\}$. For a tuple $T = (t_0, t_1, t_2, t_3) \in \mathcal{T}_1$ we define $\mathcal{B}_T = \{(B_0, B_1, B_2, B_3) \mid B_j \subseteq L_j \text{ and } |B_j| = t_j \text{ for all } j \in [0, 3]\}$. For a tuple $T = (t_{q-4}, t_{q-3}, t_{q-2}, t_{q-1}) \in \mathcal{T}_{q-2}$ we define $\mathcal{B}_T = \{(B_{q-4}, B_{q-3}, B_{q-2}, B_{q-1}) \mid B_j \subseteq L_j \text{ and } |B_j| = t_j \text{ for all } j \in [q-4, q-1]\}$.

For a tuple $T = (t_{q-3}, t_{q-2}, t_{q-1}) \in \mathcal{T}_{q-1}$ we define $\mathcal{B}_T = \{(B_{q-3}, B_{q-2}, B_{q-1}) \mid B_j \subseteq L_j \text{ and } |B_j| = t_j \text{ for all } j \in [q-3, q-1]\}$.

We are now ready to describe our dynamic programming table based on the layers.

Case: For the layer L_i where $i \in [2, q-3]$.

For a fixed $T = (t_{i-2}, t_{i-1}, t_i, t_{i+1}, t_{i+2}) \in \mathcal{T}_i$ and a fixed $B = (B_{i-2}, B_{i-1}, B_i, B_{i+1}, B_{i+2}) \in \mathcal{B}_T$, the entry $h[i, T, B]$ stores the minimum size of a set $S \subseteq L_0 \cup L_1 \cup \dots \cup L_{i+2}$ such that every vertex in $L_0 \cup L_1 \cup \dots \cup L_i$ is hop dominated, and $|S \cap L_j| = t_j$ and $S \cap L_j = B_j$, for all $j \in [i-2, i+2]$.

Case: For layer L_0 .

For a fixed $T = (t_0, t_1, t_2) \in \mathcal{T}_0$ and a fixed $B = (B_0, B_1, B_2) \in \mathcal{B}_T$, the entry $h[0, T, B]$ stores the minimum size of a set $S \subseteq L_0 \cup L_1 \cup L_2$ such that every vertex in L_0 is hop dominated and $|S \cap L_j| = t_j$ and $S \cap L_j = B_j$, for all $j \in [0, 2]$.

Case: For layer L_1 .

For a fixed $T = (t_0, t_1, t_2, t_3) \in \mathcal{T}_1$ and a fixed tuple $B = (B_0, B_1, B_2, B_3) \in \mathcal{B}_T$, the entry $h[1, T, B]$ stores the minimum size of a set $S \subseteq L_0 \cup \dots \cup L_3$ such that every vertex in $L_0 \cup L_1$ is hop dominated and $|S \cap L_j| = t_j$ and $S \cap L_j = B_j$, for all $j \in [0, 3]$.

Case: For layer L_{q-2} .

For a fixed $T = (t_{q-4}, t_{q-3}, t_{q-2}, t_{q-1}) \in \mathcal{T}_{q-2}$ and a fixed $B = (B_{q-4}, B_{q-3}, B_{q-2}, B_{q-1}) \in \mathcal{B}_T$, the entry $h[q-2, T, B]$ stores the minimum size of a set $S \subseteq L_0 \cup L_1 \cup \dots \cup L_{q-2}$ such that every vertex in $L_0 \cup L_1 \cup \dots \cup L_{q-2}$ is hop dominated, and $|S \cap L_j| = t_j$ and $S \cap L_j = B_j$, for all $j \in [q-4, q-1]$.

Case: For layer L_{q-1} .

For a fixed $T = (t_{q-3}, t_{q-2}, t_{q-1}) \in \mathcal{T}_{q-1}$ and a fixed $B = (B_{q-3}, B_{q-2}, B_{q-1}) \in \mathcal{B}_T$, the entry $h[q-1, T, B]$ stores the minimum size of a set $S \subseteq L_0 \cup L_1 \cup \dots \cup L_{q-1}$ such that every vertex in $L_0 \cup L_1 \cup \dots \cup L_{q-1}$ is hop dominated, and $|S \cap L_j| = t_j$ and $S \cap L_j = B_j$, for all $j \in [q-3, q-1]$.

We now give the procedure to calculate the values of these entries.

Base case: The base cases are when $i = 0$ and $i = 1$. Let $L_0 = \{v_0\}$. For a fixed T and B ,

$$h(0, T, B) = \begin{cases} t_0 + t_1 + t_2 & \text{if } B_0 \cup B_1 \cup B_2 \text{ hop dominates } v_0 \\ \infty & \text{otherwise.} \end{cases}$$

$$h(1, T, B) = \begin{cases} t_0 + t_1 + t_2 + t_3 & \text{if } B_0 \cup B_1 \cup B_2 \cup B_3 \text{ hop dominates } L_0 \cup L_1 \\ \infty & \text{otherwise.} \end{cases}$$

We say a set X *hop dominates* Y if for each $y \in Y$ either $y \in X$ or there exists a vertex $x \in X$ such that $d(x, y) = 2$.

Recursive Step: For each $i \in [2, q-3]$, a fixed $T = (t_{i-2}, t_{i-1}, t_i, t_{i+1}, t_{i+2})$ and a fixed $B = (B_{i-2}, B_{i-1}, B_i, B_{i+1}, B_{i+2})$, we set

$$h(i, T, B) = t_{i+2} + \min_{\substack{T' \in \mathcal{T}_{i-1} \text{ respects } T \\ B' \in \mathcal{B}_{T'} \text{ respects } B}} h(i-1, T', B')$$

We say a tuple $T' = (t'_{i-3}, t'_{i-2}, t'_{i-1}, t'_i, t'_{i+1}) \in \mathcal{T}_{i-1}$ respects T if $t'_j = t_j$ for all $j \in [i-2, i+1]$. Similarly, we say a tuple $B' = (B'_{i-3}, B'_{i-2}, B'_{i-1}, B'_i, B'_{i+1}) \in \mathcal{B}_{T'}$ respects B if $B'_j = B_j$ for all $j \in [i-2, i+1]$.

For a fixed $T = (t_{q-4}, t_{q-3}, t_{q-2}, t_{q-1})$ and a fixed $B = (B_{q-4}, B_{q-3}, B_{q-2}, B_{q-1})$, we set

$$h(q-2, T, B) = \min_{\substack{T' \in \mathcal{T}_{i-1} \text{ respects } T \\ B' \in \mathcal{B}_{T'} \text{ respects } B}} h(q-3, T', B')$$

For a fixed $T = (t_{q-3}, t_{q-2}, t_{q-1})$ and $B = (B_{q-3}, B_{q-2}, B_{q-1})$, we set

$$h(q-1, T, B) = \min_{\substack{T' \in \mathcal{T}_{i-1} \text{ respects } T \\ B' \in \mathcal{B}_{T'} \text{ respects } B}} h(q-2, T', B')$$

Answer: We need a smallest sized set $S \subseteq V(G)$ such that every vertex in $V(G) \setminus S$ is hop dominated by some vertex in S . This value is attained by $\min_{T \in \mathcal{T}_{q-1}, B \in \mathcal{B}_T} h(q-1, T, B)$.

The correctness of the algorithm follows from the description of the algorithm. The number of entries in the table is $O(n^9)$ and each entry can be computed in $O(1)$ time. Due to space constraints, the detailed running time of the algorithm is presented in the full version of the paper. \square

5 Biconvex Bipartite Graphs

In this section, we show that HOP DOMINATION is polynomial time solvable on biconvex bipartite graphs which generalizes the result on bipartite permutation graphs [6].

Definition 3 ([9]). A bipartite graph $G = (X, Y, E)$ satisfies the adjacency property if for every vertex $y \in Y$, the neighborhood $N(y)$ is a set of vertices that are consecutive in the ordering σ of X . A bipartite graph (X, Y, E) is biconvex if there are orderings of X (with respect to Y) and Y (with respect to X) that fulfill the adjacency property.

It is known [10] that all connected biconvex bipartite graphs admit multi-chain ordering. Notice that since G is a bipartite graph the graph induced on each layer in a multi-chain ordering is an independent set. To prove this theorem, we will have the following lemma which is similar to Lemma 5 for biconvex bipartite graph.

Lemma 6. Let G be an biconvex bipartite graph that admits a multi-chain ordering $\mathcal{L} = \{L_0, L_1, \dots, L_{q-1}\}$ on q layers where $q \leq 4$. Then there is a hop dominating set S of size at most two.

Proof. If $q = 1$ then $S = L_0$ and the claim holds trivially. Else if $q = 2$ then we have $\mathcal{L} = \{L_0, L_1\}$. Notice that the vertex in L_0 is adjacent to each vertex in L_1 . Consider the set $S = \{\beta_0, w\}$ where w is an arbitrary vertex in L_1 . Each vertex in $L_1 \setminus \{w\}$ is hop dominated by w . Else if $q = 3$, we have $\mathcal{L} = \{L_0, L_1, L_2\}$. The set $S = \{\beta_0, \beta_1\}$ is a hop dominating set of G as each vertex in $L_1 \setminus \{\beta_1\}$ (resp. L_2) is hop dominated by β_1 (resp. β_0). Otherwise $q = 4$ and we have $\mathcal{L} = \{L_0, L_1, L_2, L_3\}$. The set $S = \{\beta_0, \beta_1\}$ is a hop dominating set of G as each vertex in $L_1 \cup L_3 \setminus \{\beta_1\}$ is hop dominated by β_1 and the vertices in L_2 are hop dominated by β_0 . \square

Lemma 7 (★). *Let G be an biconvex bipartite graph that admits a multi-chain ordering $\mathcal{L} = \{L_0, L_1, \dots, L_{q-1}\}$ on q layers. Let S be a hop dominating set of G . Then there exists a hop dominating set S' of G of size at most $|S|$ such that $|(L_i \cup L_{i+1} \cup L_{i+2} \cup L_{i+3} \cup L_{i+4}) \cap S'| \leq 7$ for all $i \in \{0, 1, \dots, q-5\}$.*

Theorem 3. HOP DOMINATION is polynomial time solvable on biconvex bipartite graphs.

Proof. Let G be a biconvex bipartite graph and $\mathcal{L} = \{L_0, L_1, \dots, L_{q-1}\}$ be a multi-chain ordering of G with q layers. If $q \leq 4$ then by Lemma 6, the size of a minimum hop dominating set is at most two. We guess all subsets of $V(G)$ of size at most two and verify whether it is a hop dominating set to obtain our desired set.

Consider the case when $q \geq 5$. From Lemma 7, there exists a hop dominating set S' of size at most S such that every chunk of five layers contains at most seven vertices. The rest of the proof is similar to the proof of Theorem 2. \square

References

- Chartrand, G., Harary, F., Hossain, M., Schultz, K.: Exact 2-step domination in graphs. *Math. Bohem.* **120**(2), 125–134 (1995)
- Hersh, P.: On exact n-step domination. *Discret. Math.* **205**(1–3), 235–239 (1999)
- Chen, X., Wang, Y.: On total domination and hop domination in diamond-free graphs. *Bull. Malaysian Math. Sci. Soc.* **43**(2), 1885–1891 (2020)
- Natarajan, C., Ayyaswamy, S.K.: Hop domination in graphs-II. *Analele Stiintifice ale Universitatii Ovidius Constanta, Seria Matematica* **23**, 187–199 (2015)
- Henning, M.A., Rad, N.J.: On 2-step and hop dominating sets in graphs. *Graphs Comb.* **33**(4), 913–927 (2017)
- Henning, M.A., Pal, S., Pradhan, D.: Algorithm and hardness results on hop domination in graphs. *Inf. Process. Lett.* **153**, 105872 (2020)
- Cygan, M., et al.: Parameterized Algorithms. Springer, Marcjan Pilipczuk (2015)
- Enright, J., Stewart, L., Tardos, G.: On list coloring and list homomorphism of permutation and interval graphs. *SIAM J. Discret. Math.* **28**(4), 1675–1685 (2014)
- Bhyravarapu, S., Kalyanasundaram, S., Mathew, R.: Conflict-free coloring on claw-free graphs and interval graphs. In: 47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022), Schloss-Dagstuhl-Leibniz Zentrum für Informatik (2022)
- Díaz, J., Diner, Ö.Y., Serna, M.J., Serra, O.: On list k-coloring convex bipartite graphs. *CoRR*, abs/2002.02729 (2020)



Charging Station Placement for Limited Energy Robots

Arun Kumar Das^(✉)

School of Computer and Information Sciences, University of Hyderabad, Hyderabad,
India
arund426@gmail.com

Abstract. We address the *minimum charging station* (MIN-STATION) placement problem for unlabelled robots. The input consists of a graph $G = (V, E)$, with given starting and target positions $S \subset V$ and $T \subset V$, respectively, for a set of m robots. A robot moves one step by transitioning from one vertex to an adjacent vertex in G and has a movement capacity of r steps, meaning it can move up to r steps without recharging from either its starting position or a charging station. Each target position must be occupied by exactly one robot to complete the motion. The objective is to determine the minimum number of charging stations needed to ensure all target positions are reached. We prove that this problem is NP-hard, even for bounded degree graphs with a maximum degree of 6. However, on the positive side, we present linear and quadratic time algorithms for the MIN-STATION problem on paths and cycles.

Keywords: Minimum station placement · Robot motion planning · Computational geometry

1 Introduction

Robot motion planning [11] has emerged as a prominent area of research, driven by its fundamental significance in the advancement of robotics and automation technologies. Multiple mobile robots move together in order to complete one or multiple tasks in warehouses management [12], airport operation [13], agricultural fields [14], military areas [7], and medical facilities [6]. The theoretical aspects of robot motion planning consists of several factors like *make span minimization* [5], *obstacle avoidance* [3], and *charger placement* [10]. In the classical model of robot motion planning, the robots commence their operation from a set of predetermined initial positions and navigate to some specified target locations to execute the designated tasks assigned to them. However, in practical scenario, robots commonly operate with limited (battery) energy capacity, necessitating periodic recharging to maintain functionality. Thus, the strategic placement of charging stations within the operational environment is essential, and it is crucial to minimize the number of such stations to achieve cost efficiency. In recent years, numerous approaches have been studied by researchers to mitigate energy

constraints using docking based recharging [15], solar charging [19], and continuous tethering [8].

In this article, we consider the variant of the *minimum charging station placement* problem with unlabeled robots. The starting and target positions for a set of m robots on the vertices of a graph G are given as a part of the input, along with a positive integer r . A robot takes one step by moving from one vertex to an adjacent vertex in G . Each robot has a *movement capacity* of r steps. This means it can move r steps without recharging when starting from its starting position or a charging station. To complete the task assigned to the robots, each target position needs to be occupied by exactly one robot. We say that the *motion of the system* is completed when each target position is occupied by a robot. The goal of the problem is to compute the minimum number of charging stations required to complete the motion of the system, formally stated as follows.

Problem 1. MIN-STATION

Input: A graph $G = (V, E)$, a set $S \subset V$ as initial positions of m robots, a set $T \subset V$ as target positions of the robots, such that $|S| = |T| = m$, and a positive integer r , as the movement capacity of the robots.

Output: A minimum cardinality subset C of V such that the robots can complete the motion of the system by reaching T from S , as defined earlier.

Since the robots are unlabeled, they are considered to be identical. So, there is no fixed matching from the starting position and the target positions of the robots. At the completion of the motion of the system, exactly one robot must occupy each of the target positions. During the motion, multiple robots may arrive at one vertex simultaneously, allowing them to recharge concurrently at a single charging station. A vertex $v \in G$ is called a *terminal* if $v \in S \cup T$.

The literature addressing the recharge station placements for mobile robots has a pool of results involving different variants of the problems. Strimel and Veloso [16] considered the problem of placing minimum number of charging stations for a single robot movement with a limited battery. Kundu and Saha [10] studied the problem of finding the minimum number of charging stations for a group of mobile robots with battery capacity d where the *motion graph* of the robots are given as a part of the input. The motion graph is a directed graph which is the union of the directed paths traversed by the robot during the motion. Thus the problem converts to compute the minimum number of vertices in a directed graph such that each vertex is within a distance d from the computed set of vertices. They have also addressed the problem of minimizing d when the positions of a set of stations are given on the graph. They proved the NP-hardness of both problems and presented approximation algorithms. This model relates with the *center finding* problem which is widely studied in the *facility location* [4] problems. Beside theoretical results, heuristic algorithms [9] are also devised to address the problem of locating optimal set of charging stations on the motion graph of the robots. However, the models assume the paths of the

robots to be the part of input when computing the optimal allocation of the charging stations.

We recall, in MIN-STATION, neither the target position for a specific robot nor its path are given as a part of the input. A robot may need to travel beyond its shortest paths from the starting and the target position. To the best of our knowledge, no result is present in the literature addressing MIN-STATION problem. We prove that MIN-STATION is NP-hard even for bounded degree graphs, where the maximum degree of the graph is bounded by 6. Then we present a linear time algorithm for MIN-STATION on paths followed by a quadratic time algorithm for cycles.

The article is organized in the following manner. We describe the necessary definitions in Sect. 2. Section 3 contains the NP-hardness result for MIN-STATION. Then we present two algorithms for MIN-STATION on paths and cycles in Sect. 4. Finally the article is concluded in Sect. 5 pointing some immediate open problems arising from the work.

2 Preliminaries

We define some necessary terminologies in this section. We use most of the standard graph theoretic terminologies [18]. A graph $G = (V, E)$ is called a *path* if the vertices of G can be listed in order $V = \{v_1, v_2, \dots, v_n\}$ and the edges are (v_i, v_{i+1}) , for all $i = \{1, 2, \dots, n-1\}$. The *distance* between two vertices v_i and v_j of a path P consisting of the vertices $\{v_1, v_2, \dots, v_n\}$ is the absolute value of $(i - j)$.

A *cycle* is a path with $v_1 = v_n$. A cycle with n vertices is called an n -cycle. The distance between two vertices on a cycle is the number of edges present on the *shortest path* between the two vertices.

An *interval* $[x, y]$ of a path or a cycle is a set of consecutive vertices going from x to y in the vertex list of the path or cycle. We say that the interval is *spanned* by x and y . The *length of the interval* is the number of edges present in the subgraph *induced* by the vertices of the interval, denoted by $|x - y|$.

3 NP-Hardness

We prove the NP-hardness of MIN-STATION by reducing the well-known NP-complete problem 3,3-SAT [17] into it. In this variant of the classical 3-SAT [1] formula each variable appears in at most three clauses either as a positive or as a negative literal.

Let us consider a 3,3-SAT formula ϕ consisting of n variables and m clauses. We note that in the problem definition, we use n, m as the number of vertices of the graph and the number of robots, respectively. However, in this section, these numbers do not conflict and are clear from the context.

We construct a graph G_ϕ with $(4n + 2m)$ vertices and $(m + n)$ robots such that MIN-STATION problem on G_ϕ has a solution of size n if and only if ϕ is satisfiable. We describe the construction of G_ϕ (depicted partially in Fig. 1) as follows.

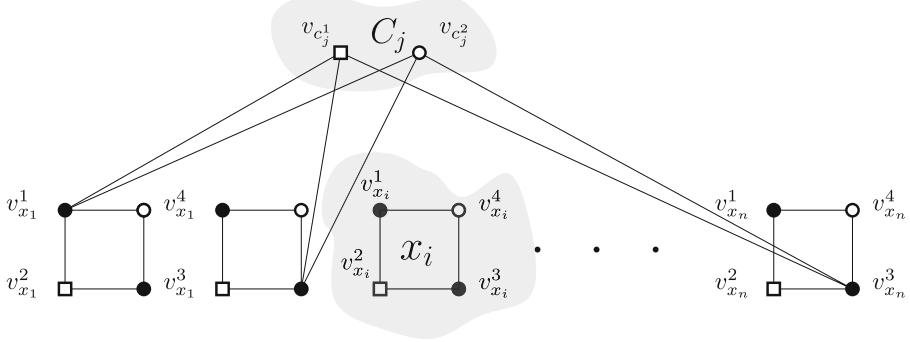


Fig. 1. Formation of G_ϕ from a given formula ϕ . The connection of $C_j = (x_1 \vee \overline{x}_2 \vee \overline{x}_n)$ is depicted. The hollow squares and circles indicate the starting and target positions of the robots, respectively.

1. We introduce a 4-cycle for every variable x_i in ϕ , namely $v_{x_i}^1, v_{x_i}^2, v_{x_i}^3, v_{x_i}^4$. These cycles are referred to as *variable gadgets*.
2. We introduce two vertices $v_{c_j^1}$ and $v_{c_j^2}$ for each clause C_j in ϕ . These vertices are referred to as *clause gadgets*.
3. We introduce edges from $v_{c_j^1}$ and $v_{c_j^2}$ to $v_{x_i}^1$ or $v_{x_i}^3$ if C_j contains x_i as a positive literal or a negative literal, respectively.

Then we form an instance of MIN-STATION on G_ϕ where the set S of starting position of $(n+m)$ robots is $S = (\cup_{j=1}^m v_{c_j}^1) \cup (\cup_{i=1}^n v_{x_i}^2)$. The set T of target positions is $T = (\cup_{j=1}^m v_{c_j}^2) \cup (\cup_{i=1}^n v_{x_i}^4)$. The movement capacity of each robot is 1. Then we have the following lemma.

Lemma 1. *The solution of the MIN-STATION problem, with the input described above, is n on G_ϕ , if and only ϕ is satisfiable.*

Proof. First, let us assume that ϕ is satisfiable. Then at least one literal in each clause is assigned to be true. We can place the n stations in the vertices corresponding to the literals in the variable gadget. Then, for every clause C_j where $j = \{1, 2, \dots, m\}$, the robots starting in $v_{c_j^1}$ reach $v_{c_j^2}$ via the station placed at the vertex corresponding to the literal which is assigned as true. Since every variable gadget contains one station either at $v_{x_i}^1$ or at $v_{x_i}^3$, the robots starting at $v_{x_i}^2$, for all $i = \{1, 2, \dots, n\}$ finish their motion at $v_{x_i}^4$. Since at least one station is necessary inside every variable gadget for the robots to move from $v_{x_i}^2$, for all i , the solution MIN-STATION problem on G_ϕ is n .

Now we assume that the MIN-STATION problem on G_ϕ has output n . We note again that it is necessary to place at least one station in every variable gadget to make a robot reach the target vertices in the variable gadgets. Thus at most one station can be placed in every variable gadget. On the other hand, at least one station must be adjacent to the vertices present in each clause gadget to finish the motion of the robots present in the clause gadgets. Thus the motion

can only be completed when at least one vertex corresponding to the literals of each clause has a station on them, implying the literal is assigned true for the clause. Hence, the lemma follows. \square

Combining Lemma 1 together with the maximum degree of G_ϕ , we get the following theorem.

Theorem 1. *MIN-STATION problem is NP-hard for graphs with maximum degree at most 6.*

Proof. The degree of each vertex present in the clause gadgets is 3. The degree of the terminal vertices in the variable gadgets is 2. The non-terminal vertices within the variable gadgets are adjacent to exactly two vertices from the variable gadget and, at most, four vertices from two clause gadgets. This follows from the fact that, in an instance of (3,3)-SAT, each literal occurs at most two times. If a variable has only positive or negative occurrences, it can be disregarded in the instance by assigning it a true value. Thus the maximum degree of a vertex in G_ϕ is at most 6. This fact along with Lemma 1 completes the proof. \square

4 Algorithms for Paths and Cycles

We present Algorithm 1 and 2 to solve the MIN-STATION problem on paths and cycles, respectively.

4.1 Algorithm for MIN-STATION on a Path

We begin with an observation on the robots' motion on a path $P = \{v_1, v_2, \dots, v_n\}$. We assume the vertices are considered from left to right on the path for calculative brevity. Further, we assume that the initial and target positions of the robots are $\{s_1, s_2, \dots, s_m\}$ and $\{t_1, t_2, \dots, t_m\}$, respectively, from left to right. We recall that the movement capacity of each robot is denoted by r . We state the following lemma, which specifies the minimum number of stations required for a group of robots traveling in the same direction.

Lemma 2. *Let $1, 2, \dots, k$ be a group of k robots starting from $s_1, s_2, \dots, s_k \in P$, sorted from left to right, and all of them need to reach a vertex $x \in P \setminus [s_1, s_k]$. Considering there is no preexisting station in the interval $[s_1, x]$, $\lfloor \frac{|x-s_1|}{r} \rfloor$ stations are necessary and sufficient to complete the motion.*

Proof. We consider dividing the interval $[s_1, x]$ into $\lfloor \frac{|x-s_1|}{r} \rfloor$ portions starting from s_1 , to prove the necessary part of the claim. We note that the robot starting at s_1 must reach a station in every division of the interval $[s_1, x]$. This addresses the minimum number of stations required for the motion.

To prove the sufficiency of these stations to complete the motion, we consider the placement of the stations at the vertices $s_1 + r, s_1 + 2r, \dots, s_1 + \lfloor \frac{|x-s_1|}{r} \rfloor r$. We note that each robot has the capacity of r and as a result each of them can reach the immediate station on its right and hence the next station towards the right. Thus the motion is completed. \square

Lemma 2 helps to determine the minimum number of stations required when the paths of a group of robots intersect while they travel in a same direction. Here the *path of a robot* refers to the interval it travels on P from its starting position to its target position. Now we state a result when the paths of two robots intersect while they are moving in opposite directions of each other.

Lemma 3. *Let the paths of two robots intersect while they are moving in opposite directions of each other. Then, the number of stations required for this motion is greater than or equal to the number of stations required in another motion where their paths do not intersect.*

Proof. Let the two robots starting at s_1 and s_2 be finishing at t_1 and t_2 , respectively, with s_1 being on the left of s_2 . Two intervals $[s_1, t_1]$ and $[s_2, t_2]$ can intersect each other in two ways: Either they intersect partially or one completely contains the other. We deal with both cases in the proof.

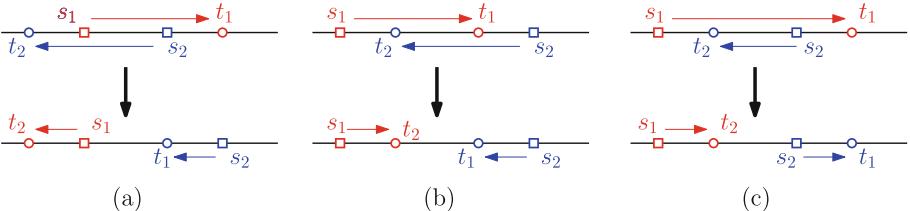


Fig. 2. Target swapping, when two paths intersect and the robots are traveling in opposite directions of each other.

- **Case 1:** Let the intervals intersect partially. Let t_1 lie on the right of s_2 and t_2 lie on the left of s_1 , as depicted in Fig. 2(a). Since each interval of length r starting from t_2 to t_1 must contain a station for the robots to reach the destinations, the minimum number of stations required to finish the motion of these two robots is at least $\lfloor \frac{|t_2-t_1|}{r} \rfloor$. However, when we consider the motion where the robots starting at s_1 and s_2 finishes at t_2 and t_1 respectively, the number of station required is $\lfloor \frac{|s_1-t_2|}{r} \rfloor + \lfloor \frac{|t_1-s_2|}{r} \rfloor \leq \lfloor \frac{|s_1-t_2|}{r} \rfloor + \lfloor \frac{|t_1-s_2|}{r} \rfloor + \lfloor \frac{|s_2-s_1|}{r} \rfloor \leq \lfloor \frac{|t_2-t_1|}{r} \rfloor$. A similar argument holds when t_1 lies on the left of s_2 , depicted in Fig. 2(b).
- **Case 2:** Without loss of generality, let $[s_2, t_2]$ be completely contained inside $[s_1, t_1]$, as depicted in Fig. 2(c). Then the minimum number of stations required for the robot at s_1 to reach t_1 is $\lfloor \frac{|t_1-s_1|}{r} \rfloor$. However, when we swap the target position of the robots, the number of stations required is $\lfloor \frac{|t_2-s_1|}{r} \rfloor + \lfloor \frac{|t_1-s_2|}{r} \rfloor \leq \lfloor \frac{|t_1-s_1|}{r} \rfloor$.

Combining these two cases the lemma holds. \square

Lemma 3 shows that in an optimal solution MIN-STATION problem on a path P , no two robots cross each other while moving in an opposite direction. Now we prove the following lemma that states the property of the motion that completes with the minimum number of charging stations.

Lemma 4. *The number of stations in MIN-STATION problem on a path P is minimized when the robot starting at s_i finishes its motion at t_i , for all $i \in \{1, 2, \dots, m\}$.*

Proof. We prove this lemma by induction on i . We begin with $i = 1$ as the induction base. Let the robot starting at s_1 do not finish its motion in t_1 but in some other target position t_o . Then some other robot finishes at t_1 to complete the motion of the system. Let s_o denote the starting position of this robot. We show that the number of stations required for this motion is higher than the number of stations required when the robots start at s_1 and s_o and finish at t_1 and t_o , respectively. Now depending on the positions of the four vertices s_1, t_1, s_o and t_o on the path P , the following six cases arise.

- **Case 1:** The positions of the four vertices on P are t_1, t_o, s_1, s_o from the left to right on P . Then the minimum number of stations required to finish their motion is $\lfloor \frac{|s_o - t_1|}{r} \rfloor$, by virtue of Lemma 2. Now, this number of stations is also sufficient for the robots starting at s_o and s_1 to reach their targets at t_o and t_1 respectively. Thus the minimum number of stations required in the later motion is less than or equal to our assumed motion.
- **Case 2:** The positions of the four vertices on P are t_1, s_1, t_o, s_o from the left to right on P . If the robots starting at s_1 and s_o are finishing their motion at t_o and t_1 , respectively, then their target positions can be swapped by the Case 2 of Lemma 3.
- **Case 3:** The positions of the four vertices on P are t_1, s_1, s_o, t_o from the left to right on P . This case is addressed by the Case 1 of Lemma 3 as the paths intersect each other while the robots travel in an opposite direction.
- The remaining cases when the positions of the four vertices on P are (s_1, t_1, s_o, t_o) , (s_1, s_o, t_1, t_o) , and (s_1, t_1, t_o, s_o) from left to right. These cases are similar to Case 1, Case 2, and Case 3 of this lemma, stated above.

Now we consider the robot starting at s_{i+1} on P . The robots starting at s_1, s_2, \dots, s_i are finishing their motion at t_1, t_2, \dots, t_i , respectively, by the induction hypothesis. The induction step follows from the following facts.

1. All the robots are of the same capacity. If there is any station placed on the right of s_{i+1} and a robot starting on the left of s_{i+1} , then the robot starting at s_{i+1} can reach the stations on its right.
2. If the robot starting at s_{i+1} reaches a target position t_o , then t_o lies on the right of t_{i+1} . If we denote by s_o , the starting position of the robot finishing at t_{i+1} , the motion of these two robots can be treated in a way similar to the induction base.

Input:

1. A path P of n vertices $\{v_1, v_2, \dots, v_n\}$ ordered from left to right.
2. A set of m starting and target positions $S = \{s_1, s_2, \dots, s_m\}$ and $T = \{t_1, t_2, \dots, t_m\}$, respectively, ordered from left to right.
3. A positive integer r .

Output: A minimum cardinality set $C \subset P$ of charging stations.

```

 $C = \emptyset;$ 
VCOUNTER=0, TCOUNTER=0;
for every  $i = 1, 2, \dots, n$  do
  if  $v_i \in S$  then
    | TCOUNTER++;
  end
  if  $v_i \in T$  then
    | TCOUNTER--;
  end
  if TCOUNTER!=0 then
    | VCOUNTER++;
    | if VCOUNTER==r then
      |   |  $C = C \cup v_i;$ 
      |   | VCOUNTER=0;
    end
  end
end
return  $C$ ;

```

Algorithm 1: MIN-STATION-ON-PATH()

Thus the result follows. □

Now we devise an algorithm for the MIN-STATION problem on a path P with n vertices. Given the terminal positions of the robots from left to right on the path, we compute the minimum number of stations required for each robot using Lemma 4. However, calculating the value for each robot may require $O(n)$ time. Therefore, if we compute the values separately, the overall running time could increase to $O(n^2)$. Instead, we keep two counters from the starting and finishing positions of the robots while scanning the vertices of the path P from left to right. Thus we compute the number of stations required along with their positions on P with one single pass on the vertices of P . The steps are formally stated in Algorithm 1. We keep a terminal counter (TCOUNTER) for the starting and target positions of the robots. We increase or decrease it by 1 when a starting or target position is scanned, respectively. The stations are placed in the interval spanned by two terminal vertices when the distance becomes equal to r . We keep track of this distance while scanning the vertices by another vertex counter(VCOUNTER).

Theorem 2. *MIN-STATION problem can be solved in $O(n)$ time for a path with n vertices and m robots, where $m \leq n$.*

Proof. Algorithm 1 returns a solution for the MIN-STATION problem and the correctness follows from Lemma 4. We note that the *for loop* in the algorithm processes each vertex on the path only once and the counters update the solution in constant time. Thus the running time of the algorithm is $O(n)$. \square

4.2 Algorithm for MIN-STATION for a Cycle

We now design an algorithm for MIN-STATION problem for a cycle with n vertices. The property of an optimal motion of a MIN-STATION instance is as follows.

Lemma 5. *There exists at least one edge of the cycle in an optimal solution of a MIN-STATION instance, that is not traversed by any robot.*

Proof. We prove the lemma by showing that each motion addressing a MIN-STATION instance where each edge of the cycle is traversed by at least one robot, can be improved to another motion where some edge is not traversed and the number of stations required in the later one is lesser than the former one. First, we consider the case when every edge is traversed by a robot and there is no path of the robots intersecting with each other. We note that this motion is redundant as every robot is starting from a target position. Hence, there is no need to move any of the robots to occupy different target positions.

Now we consider the case where the paths of the robots are intersecting and every edge of the cycle is traversed by a robot. If there exist two paths intersecting each other where the robots are traveling in an opposite direction, then by Lemma 3 we can swap their target positions and there will be at least one edge with is not traversed by any robots.

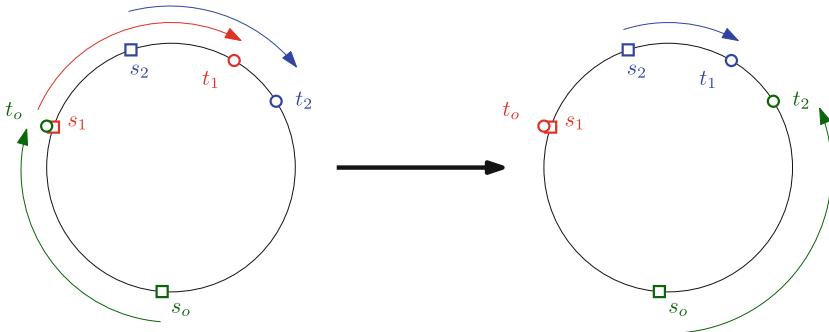


Fig. 3. Finding an edge in a cycle that is not traversed by any robot.

So, we investigate the case when the paths intersect each other and the robots are traveling in a clockwise (say) direction. It is worth noting that in our assumption, $\lfloor \frac{n}{r} \rfloor$ stations are required as every edge is being traversed by some robot. Let us consider the paths of two robots starting at s_i, s_{i+1} are

intersecting with each other while they travel in the same direction towards their target positions t_i and t_{i+1} , respectively. Since all the edges of the cycle are traversed by some robot, s_i is also a target position t_o (say) for some other robot starting at s_o traveling clockwise. We improve the motion of these robots as follows, depicted in Fig. 3.

- The robot starting at s_i remains at s_i occupying the target t_o .
- The robot starting at s_{i+1} moves to t_i . Hence the edges between s_i and s_{i+1} are not being traversed by any robots.
- The robot starting at s_o changes its direction and moves to t_{i+1} . This is possible with the existing stations in the interval from s_o to t_{i+1} in the anti-clockwise direction as all of these edges are traversed by some robots.

Furthermore, we can argue in a similar way when more than two paths intersect. We consider two of them at a time and continue unless we get an edge of the cycle that is not traversed by a robot. This completes the proof by showing there is always one such path in an optimal solution of the MIN-STATION problem for a cycle. \square

Input:

1. A cycle G of n vertices $\{v_0, v_1, \dots, v_{n-1}\}$ sorted in clockwise order.
2. A set of m starting and target positions $S = \{s_1, s_2, \dots, s_m\}$ and $T = \{t_1, t_2, \dots, t_m\}$, respectively.
3. A positive integer r .

Output: A minimum cardinality set $C \subset G$ of charging stations.

$C = \{v_0, v_1, \dots, v_{n-1}\}$;

for every $i = 0, 1, 2, \dots, n - 1$ **do**

$P = G \setminus \{\text{the edge between } v_i \text{ and } v_{(i+1)\%n}\}$;

$X = \text{MIN-STATION-ON-PATH}(P)$;

if cardinality of $X \leq$ cardinality of C **then**

| $C = X$;

end

end

return C ;

Algorithm 2: MIN-STATION-ON-CYCLE()

Using Lemma 5 we devise Algorithm 2 for MIN-STATION problem for a cycle. The algorithm guesses an edge that is not traversed by any robots in an optimal solution and computes the MIN-STATION on the path constructed by removing the edge from the cycle. Then it returns the solution set C of the charging stations with the minimum cardinality among all the candidates. Since there are n edges in the cycle, Algorithm 2 returns an optimal solution in $O(n^2)$ time. Thus we have the following theorem.

Theorem 3. *MIN-STATION problem can be solved in $O(n^2)$ time for a cycle with n vertices.*

5 Conclusion

We conclude the article by pointing out some open problems arising from the work. We presented two algorithms for solving the MIN-STATION problem on paths and cycles and the problem remains NP-hard for general graphs. The immediate question that arises from the work is the complexity of the MIN-STATION problem for *path alike* graph classes such as trees or grids.

Question 1. What is the complexity of the MIN-STATION problem for trees, grids, and planar graphs?

Another avenue in this direction is considering the practical graph parameters [2] like pathwidth, treewidth, etc. It is relevant to study the fixed-parameter tractability of the problem under different parameters.

Question 2. Is it possible to devise fixed-parameter tractable algorithms for MIN-STATION problem for graphs parameterized by the number of terminals, number of leaves, pathwidth, treewidth, etc.?

References

1. Biere, A., Heule, M., van Maaren, H.: *Handbook of Satisfiability*, vol. 185. IOS Press (2009)
2. Cygan, M., et al.: *Parameterized Algorithms*, vol. 5. Springer (2015)
3. Demaine, E.D., Fekete, S.P., Keldenich, P., Meijer, H., Scheffer, C.: Coordinated motion planning: reconfiguring a swarm of labeled robots with bounded stretch. *SIAM J. Comput.* **48**(6), 1727–1762 (2019)
4. Drezner, Z., Hamacher, H.W.: *Facility location: applications and theory*. Springer (2004)
5. Fioravantes, F., Knop, D., Křištan, J.M., Melissinos, N., Opler, M.: Exact algorithms and lowerbounds for multiagent path finding: Power of treelike topology. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 17380–17388 (2024)
6. Fried, I.: Towards Robot Autonomy in Medical Procedures via Visual Localization and Motion Planning. Ph.D. thesis, The University of North Carolina at Chapel Hill (2023)
7. Huang, L., Zhou, M., Hao, K., Hou, E.: A survey of multi-robot regular and adversarial patrolling. *IEEE/CAA J. Automatica Sinica* **6**(4), 894–903 (2019)
8. Kim, S., Bhattacharya, S., Kumar, V.: Path planning for a tethered mobile robot. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1132–1139. IEEE (2014)
9. Kundu, T., Saha, I.: Charging station placement for indoor robotic applications. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3029–3036. IEEE (2018)
10. Kundu, T., Saha, I.: Approximation algorithms for charging station placement for mobile robots. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4770–4776. IEEE (2023)
11. Latombe, J.C.: *Robot Motion Planning*, vol. 124. Springer (2012)

12. Li, J., Tinka, A., Kiesel, S., Durham, J.W., Kumar, T.S., Koenig, S.: Lifelong multi-agent path finding in large-scale warehouses. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 11272–11281 (2021)
13. Morris, R., et al.: Planning, scheduling and monitoring for airport surface operations. In: Workshops at the Thirtieth AAAI Conference on Artificial Intelligence (2016)
14. Pak, J., Kim, J., Park, Y., Son, H.I.: Field evaluation of path-planning algorithms for autonomous mobile robot in smart farms. *IEEE Access* **10**, 60253–60266 (2022)
15. Shnaps, I., Rimon, E.: Online coverage of planar environments by a battery powered autonomous mobile robot. *IEEE Trans. Autom. Sci. Eng.* **13**(2), 425–436 (2016)
16. Strimel, G.P., Veloso, M.M.: Coverage planning with finite resources. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2950–2956. IEEE (2014)
17. Tovey, C.A.: A simplified np-complete satisfiability problem. *Discret. Appl. Math.* **8**(1), 85–89 (1984)
18. West, D.B.: Introduction to graph theory (2001)
19. Wettergreen, D., Tompkins, P., Urmson, C., Wagner, M., Whittaker, W.: Sun-synchronous robotic exploration: technical description and field experimentation. *Int. J. Robot. Res.* **24**(1), 3–30 (2005)



Multipacking in the Euclidean Metric Space

Arun Kumar Das¹, Sandip Das², Sk Samim Islam^{2(✉)}, Ritam Manna Mitra²,
and Bodhanay Roy³

¹ Czech Technical University in Prague, Prague, Czech Republic

² Indian Statistical Institute, Kolkata, India

samimislam08@gmail.com

³ Indian Institute of Technology, Kharagpur, India

Abstract. We initiate the study of multipacking problems for geometric point sets with respect to their Euclidean distances. We consider a set of n points P and define $N_s[v]$ as the subset of P that includes the s nearest points of $v \in P$ and the point v itself. We assume that the s -th neighbor of each point is unique, for every $s \in \{0, 1, 2, \dots, n-1\}$. For a natural number $r \leq n$, an r -multipacking is a set $M \subseteq P$ such that for each point $v \in P$ and for every integer $1 \leq s \leq r$, $|N_s[v] \cap M| \leq (s+1)/2$. The r -multipacking number of P is the maximum cardinality of an r -multipacking of P and is denoted by $\text{mp}_r(P)$. For $r = n-1$, an r -multipacking is called a multipacking and r -multipacking number is called as multipacking number. We study the problem of computing a maximum r -multipacking for point sets in \mathbb{R}^2 . We show that a maximum 1-multipacking can be computed in polynomial time but computing a maximum 2-multipacking is NP complete. Further, we provide approximation and parameterized solutions to the 2-multipacking problem.

Keywords: Geometric packing · Nearest neighbor graph · Multipacking

1 Introduction

The problem of locating facilities, both desirable and undesirable, is of utmost importance due to its huge number of applications in the real world [24]. Allocation of restricted resources within an arena has been extensively studied across various models and methodologies since the 1980s. Drezner and Wesolowsky did a “maximin” problem where a certain facility must serve a given set of users but must be placed in such a manner that its minimum distance from the users is maximized (see [13, 14]). Rodriguez et al. [22] gave a generic model for the

Some of the proofs and figures are omitted in this conference version of the paper. The full version of the paper is available in arXiv [8].

B. Roy—Supported by the Science and Engineering Research Board (SERB) via the project MTR/2021/000474.

obnoxious facility location problem inside a given polygonal region. Rakas et al. [21] devised a multi-objective model to determine the optimal locations for undesirable facilities. The book by Daskin [11] contains a nice collection of problems, models, and algorithms regarding obnoxious facility location problems in the domain of supply chain networks. In this paper, we initiate a novel study of the *Multipacking* [23] problem in Euclidean space, which resembles the obnoxious facility location problem in computational geometry.

Given a planar point set $P \subseteq \mathbb{R}^2$ of size n , the *neighborhood* of size r of a point $v \in P$ is the subset of P that consists of the r nearest points of v and v itself. This subset is denoted by $N_r[v]$. A natural and realistic assumption here is the distances between all pairs of points are distinct. Thus the r -th neighbor of every point is unique. This assumption is easy to achieve through some form of controlled perturbation (see [18]). For a natural number $r \leq n$, an r -*multipacking* is a subset M of P , such that for each point $v \in P$ and for every integer $1 \leq s \leq r$, $|N_s[v] \cap M| \leq (s+1)/2$, i.e. M contains at most half of the number of elements of $N_s[v]$. The r -*multipacking number* of P is the maximum cardinality of an r -multipacking of P and is denoted by $\text{mp}_r(P)$.

An $(n-1)$ -multipacking M of P is referred to as *multipacking* of P . The *multipacking number* of P , denoted by $\text{mp}(P)$, refers to the cardinality of a maximum multipacking of P . Computing maximum multipacking involves selecting the maximum number of points from the given point set satisfying the constraints of multipacking. This variant of facility location problem is interesting due to their wide application in operation research [22], resource allocation [20], VLSI design [1], network planning [6, 21], and clustering [17].

To the best of our knowledge, no significant work on maximum multipacking has been done on planar point sets concerning their Euclidean distances. Whereas the literature contains numerous intriguing findings for this problem in graph theory. Multipacking was first discussed by Teshima [23]. Other related works can be found in [2, 7, 9, 10, 19]. However, as of now, no polynomial-time algorithm is known for finding a maximum multipacking of general graphs, the problem is also not known to be NP-hard. Nonetheless, polynomial-time algorithms exist for trees and strongly chordal graphs [5]. For further references on algorithmic results regarding multipacking problem on graphs, we refer to the paper [16].

Further, we define the function $\text{MP}_r(t)$ to be the smallest number such that any point set $P \subset \mathbb{R}^2$ of size $\text{MP}_r(t)$ admits a maximum r -multipacking of size exactly t . We denote $\text{MP}_{n-1}(t)$ as $\text{MP}(t)$, i.e. $\text{MP}(t)$ is the smallest number such that any point set $P \subset \mathbb{R}^2$ of size $\text{MP}(t)$ admits a *maximum multipacking* of size exactly t . It can be followed that $\text{MP}(1) = 1$ from the definition of multipacking. No general solution has been found for this problem yet.

Our Contribution: In Sect. 2, we provide an algorithm to find a maximum multipacking of a point set on a line and we prove some bounds on the multipacking number on a line. In Sect. 3, we prove that $\text{MP}(2) = 6$ using some geometric observations. Until now, there is no known polynomial-time algorithm to find a maximum multipacking for a given point set $P \subset \mathbb{R}^2$, and the problem is also

not known to be NP-hard. In this paper we make an important step towards solving the problem by studying the 1-multipacking and 2-multipacking in \mathbb{R}^2 . In Sect. 4, we considered the NNG (Nearest-Neighbor-Graph) G of $P \subset \mathbb{R}^2$. Using the observation that the maximum independent sets of G are the maximum 1-multipackings of P , we conclude that a maximum 1-multipacking can be computed in polynomial time. However the 2-multipacking problem in \mathbb{R}^2 is much more complicated. In Sect. 5, we study the hardness of the 2-multipacking problem in \mathbb{R}^2 . We prove that the problem is NP-complete by reducing the well-known NP-complete problem Planar Rectilinear Monotone 3-SAT [12] to the 2-multipacking problem. Moreover, we provide approximation and parameterized solutions to this problem.

2 Maximum Multipacking in \mathbb{R}^1

2.1 Bound of Maximum Multipacking in \mathbb{R}^1

We start by bounding the multipacking number of a point set in the following lemma.

Lemma 1. *Let P be a point set on \mathbb{R}^1 . Then $\lfloor \frac{n}{3} \rfloor \leq \text{mp}(P) \leq \lfloor \frac{n}{2} \rfloor$.*

Proof. In order to show the lower bound, we show that for any point set on \mathbb{R}^1 of size n we can always find a multipacking of size $\lfloor \frac{n}{3} \rfloor$. Let $P = \{p_1, p_2, \dots, p_n\}$ be n points on \mathbb{R}^1 sorted with respect to the ascending order of their x -coordinates. Construct a set M as such, $M = \{p_1, p_4, p_7, \dots, p_{3k+1}\}$ (where $k \in \mathbb{N}$ and $3k+1 \leq n < 3(k+1)+1$) is a multipacking of P . Note that M is a multipacking since for any $1 \leq s \leq n-1$ and $p_i \in P$, $N_s[p_i]$ contains at most $\lfloor \frac{s+1}{2} \rfloor$ points from M . This shows the lower bound of the lemma. The upper bound follows from the definition of multipacking. \square

Tight Bound Example for the Upper Bound: For the upper bound of Lemma 1, we consider the point set $P = \{p_1, \dots, p_n\}$ where,

$$p_i = \begin{cases} 0 & \text{if } i = 0 \\ \frac{4}{3}(2^{i-1} - 1) - \frac{i-1}{2} & \text{if } i \text{ is odd} \\ \frac{4}{3}(2^i - 1) - \frac{i}{2} - 2^{i-1} + 1 & \text{if } i \text{ is even, } i > 0 \end{cases}$$

Note that p_i 's in P are in ascending order. From definition of this example, $p_{2k+2} - p_{2k+1} > p_{2k+1} - p_1$ and $p_{2k+2} - p_{2k+1} > p_{2k+3} - p_{2k+2}$ for each k . From this, we conclude that the set $M = \{p_1, p_4, p_6, p_8, \dots\}$ is a maximum multipacking and $|M| = \lfloor \frac{n}{2} \rfloor$.

Tight Bound Example for the Lower Bound: For the lower bound of Lemma 1, we consider the point set $P = \{p_1, \dots, p_n\}$ where $p_i = 2^i$.

Note that $p_{i-1} - p_1 < p_i - p_{i-1}$ holds for each p_i in P . From this, we conclude that the set $M = \{p_1, p_4, p_7, \dots\}$ is a maximum multipacking with $|M| = \lfloor \frac{n}{3} \rfloor$.

Combining these arguments together with Lemma 1 we have the following theorem.

Theorem 1. *Let P be a point set on \mathbb{R}^1 . Then $\lfloor \frac{n}{3} \rfloor \leq \text{mp}(P) \leq \lfloor \frac{n}{2} \rfloor$. Both the bounds are tight.*

2.2 Algorithm to Find a Maximum Multipacking in \mathbb{R}^1

We devise algorithm 2 that checks possible violations for every member of P and selects the eligible candidates as multipacking accordingly. It takes a point set $P \subset \mathbb{R}^1$ sorted in ascending order with respect to their x -coordinates as input and outputs a maximum multipacking M where $M \subseteq P$.

Algorithm 1 ($CMP_r(P, M)$) is designed to check the eligibility of a member of P as a point in multipacking and executed as a subroutine in each iteration of algorithm 2. $CMP_r(P, M)$ (Algorithm 1) takes a point set $M \subseteq P$ as input and outputs 1 if M is a multipacking and 0 if not. Given a point set $P = \{p_1, p_2, \dots, p_n\}$, at the i^{th} iteration, p_i is included in M and $CMP_r(P, M)$ is called as a subroutine. Depending on the eligibility check done by this subroutine M is formed. The algorithm terminates when P is exhausted and returns M as output. The correctness of the algorithm is presented in [8].

Algorithm 1: Check r-multipacking - $CMP_r(P, M)$

Input: A point set P in \mathbb{R} and $M \subseteq P$ and the value of r .
Output: if M is a valid multipacking on P then return 1, else return 0

```

for  $p_i$  do
    for  $s = 1 \rightarrow r$  do
        if  $|N_s[p_i] \cap M| \leq (s + 1)/2$  then
            | continue;
        else
            | return 0;
        end
    end
end
return 1;
```

Algorithm 2: Maximum r -multipacking

Input: A point set P in \mathbb{R} and the value of r .
Output: A maximum multipacking M of P .

```

 $M = \phi$ ;
i = 1;
for  $p_i$  do
     $M = M \cup \{p_i\}$ ;
    if  $CMP_r(P, M) == 0$  then
        |  $M = M \setminus \{p_i\}$ ;
    end
    i++;
end
return  $M$ ;
```

Theorem 2. For any $1 \leq r \leq n - 1$, a maximum r -multipacking for a given point set $P \subset \mathbb{R}^1$ with $|P| = n$ can be computed in $O(n^2r)$ time.

It is worth mentioning that, computing the maximum multipacking set of a planar point set relates with the notion of independent set (which we show later in the paper) which becomes NP-hard in general. Thus it is important to study a bound for the multipacking number in \mathbb{R}^2 . Next, we address the bound on the size of P to achieve a multipacking of a desired size irrespective of the distribution of a set of points in \mathbb{R}^2 .

3 Minimum Number of Points to Achieve a Desired Multipacking Size in \mathbb{R}^2

In this section, we study the MP function. Recall the definition of the same. $\text{MP}_r(t)$ is the smallest number such that any point set $P \subset \mathbb{R}^2$ with $|P| = \text{MP}_r(t)$ admits an r -multipacking of size exactly t . As stated earlier $\text{MP}_{n-1}(t)$ is denoted by $\text{MP}(t)$ and $\text{MP}(1) = 1$. Now our goal is to find the value of $\text{MP}(2)$, which we state in the following theorem.

Theorem 3. $\text{MP}(2) = 6$.

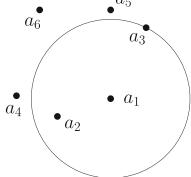


Fig. 1. 6 points with $\text{MP}(2) \leq 6$.

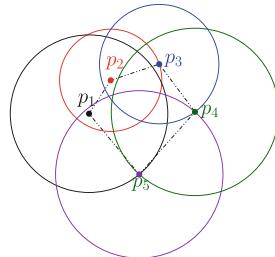


Fig. 2. 5 points with $\text{MP}(2) > 5$.

Proof. We prove this using contradiction. Suppose $\text{MP}(2) > 6$. That means there is a point set $P = \{a_1, a_2, \dots, a_6\} \subset \mathbb{R}^2$ such that $\text{mp}(P) = 1$. This implies, for each $i, j (i \neq j)$ there exists k such that $\{a_i, a_j\} \subset N_2[a_k]$, otherwise $\{a_i, a_j\}$ can form a multipacking of size 2 which is not allowed. If a_i and a_j are the 1st and 2nd neighbor of a_k respectively, we denote $C_2[a_k]$ to be the circle with center at a_k passing through a_j . Without loss of generality we can assume that $C_2[a_1]$ is the circle with the largest radius among all the circles $C_2[a_k]$, $k \in \{1, 2, \dots, 6\}$ and moreover a_2 and a_3 are the 1st and 2nd neighbor of a_1 (See Fig. 1). Observe that, $a_1 \notin N_2[a_k]$ for each $k \in \{4, 5, 6\}$, since $C_2[a_1]$ is the circle with largest radius. Therefore, each pair among the three pairs $\{a_1, a_4\}$, $\{a_1, a_5\}$ and $\{a_1, a_6\}$ is a subset of either $N_2[a_2]$ or $N_2[a_3]$. By Pigeonhole Principle, there are at least two pairs among the three pairs $\{a_1, a_4\}$, $\{a_1, a_5\}$ and $\{a_1, a_6\}$ which are subsets of either $N_2[a_2]$ or $N_2[a_3]$, but both $N_2[a_2]$ and $N_2[a_3]$ have size 3 which is a contradiction of $\text{mp}(P) = 1$. Therefore, $\text{MP}(2) \leq 6$.

Furthermore, we have point sets of sizes 3, 4 and 5 (See Fig. 2) that have multipacking number 1. We depict the point set P in Fig. 2 forming a convex pentagon. Every point $p_i \in P$ has p_{i-1} and $p_{i+1 \pmod 5}$ as its two nearest neighbors. As a result no two points can be present in the multipacking set of P . The correctness of the figure can be followed from the fact that each circle (marked with different colours) containing a point $p_i \in P$ as center and the second nearest neighbor of p_i on its boundary, has only one other point (namely the first nearest neighbor of p_i) in the interior. This example shows $\text{MP}(2) > 5$ implying, $\text{MP}(2) = 6$. \square

4 1-multipacking in \mathbb{R}^2

We note that Algorithm 2 in Subsect. 2.2 can not be adapted for a point set $P \subseteq \mathbb{R}^2$. Since we can not ensure the sorted order of the points with respect to their x or y coordinate, the maximum cardinality of the output can not be guaranteed. Thus it is interesting to address r -multipacking while $r < n - 1$.

The definition of r -multipacking for $r = 1$ coincides with the definition of independent set in case of graphs and it remains the same when we consider the maximum independent set of the NNG (Nearest-Neighbor-Graph) of P . The NNG of P is a forest when considered as an undirected graph with 2-cycles at the leaves. Then, a maximum 1-multipacking for P can be computed in polynomial time by computing the maximum independent set of NNG of P [15].

We write this observation as the following lemma.

Lemma 2. *Let P be a point set on the 2D plane and the NNG of P be G . The maximum independent sets of G are the maximum 1-multipackings of P .*

5 2-multipacking in \mathbb{R}^2

Next we study the 2-multipacking problem. We are interested in showing the hardness of finding a 2-multipacking of a given set of points P in the plane.

We construct a graph $G_P = (V, E)$ from the given point set P . Here the set of vertices $V = P$. For every vertex $v \in V$, we introduce three edges vu_1, vu_2 , and u_1u_2 where $u_1, u_2 \in V$ are the first and second neighbors of v respectively. The following lemma shows the relation between the independent set of G_P and a 2-multipacking of P .

Lemma 3. *A set M is a 2-multipacking of P if and only if the vertices in the graph G_P corresponding to the members of M form an independent set in G_P .*

Proof. It follows from the definition of 2-multipacking that for every tuple (v, u_1, u_2) , as described above, at most one can be chosen as a member of M . Since each edge of G_P is incident on a pair of vertices appearing in a tuple, there is no edge between two members of M appearing in G_P . Thus the vertices corresponding to the members of M form an independent set of G_P . The converse follows from a similar argument. \square

Thus, finding a maximum independent set of G_P is equivalent to finding a maximum 2-multipacking of P . In the rest of this section, we show that finding a maximum independent set of G_P is NP-complete, and we devise approximation and parameterized solutions to this problem.

5.1 NP-Completeness

We show that finding the maximum independent set in the graph G_P is NP-complete by reducing the well-known NP-complete problem *Planar Rectilinear Monotone (PRM) 3-SAT* [12] to this problem. This variant of the boolean formula contains three literals in each clause and all of them are either positive or negative. Moreover, a planar graph can be constructed for the formula such that a set of axis parallel rectangles represent the variables and the clauses with the following properties. The rectangles representing the variables lie on the x -axis. Whereas rectangles representing the clauses that contain the positive (negative) literals lie above (below) the x -axis. Additionally, the rectangles for the clauses can be connected with vertical lines to the rectangles for the variables that appear in the clauses.

Given a PRM 3-SAT formula ϕ with $t(> 1)$ variables and $m(> 1)$ clauses, we construct a planar point set P_ϕ such that ϕ is satisfiable if and only if G_{P_ϕ} has a maximum independent set of a given size. First, we construct a point set for the *variable gadget* for each variable that has only two subsets as the maximum independent set in G_{P_ϕ} . Then we construct another point set as clause gadgets for each clause in ϕ such that the clause gadget can incorporate only one vertex in the independent set only if at least one variable gadget has the desired set chosen as the maximum independent set. The clause and variable gadgets are connected via *connection gadgets* which are two point sets referred to as *translation* and *rotation* gadgets. We describe the construction of the point sets formally below.

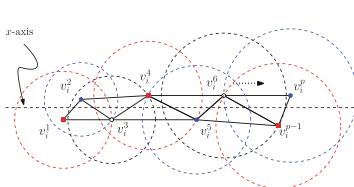


Fig. 3. A variable gadget in G_{P_ϕ} .

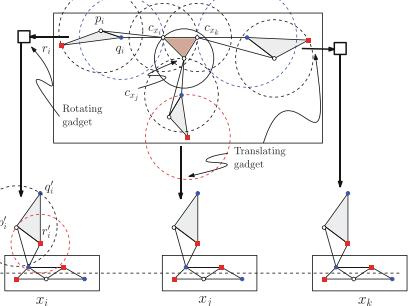


Fig. 4. A clause gadget and its connection with the variable gadgets.

Variable Gadget. We construct a point set for each variable x_i in ϕ consisting of $p = 6(m - 1) + 2$ points. A variable gadget is depicted in Fig. 3 with $p = 8$.

Each point is considered as the vertex in the graph G_{P_ϕ} . The leftmost point is v_i^1 having v_i^2 and v_i^3 as its first and second neighbor respectively. Thus v_i^1 introduces three edges $v_i^1v_i^2$, $v_i^1v_i^3$, and $v_i^2v_i^3$ in G_{P_ϕ} . The next point from the right is v_i^2 which has v_i^1 and v_i^3 as its first and second neighbor. Since the edges corresponding to the adjacency of v_i^2 are already drawn, it does not introduce any new edges. Then the third point v_i^3 introduces two new edges namely $v_i^3v_i^4$ and $v_i^2v_i^4$. Continuing in a similar fashion v_i^{p-1} introduces two edges $v_i^{p-1}v_i^p$ and $v_i^6v_i^p$. Finally, v_i^p does not introduce any new edges having v_i^{p-1} and v_i^6 as its first and second neighbors respectively. The second neighborhoods of the points are indicated with colored (referring to the online version in case of gray-scale documents) circles in the figure. The leftmost red circle contains the first and second neighbor of v_i^1 in Fig. 3. The following observation holds from the structure of the variable gadget in G_{P_ϕ} .

Observation 1. *The maximum independent set in a variable gadget with $6(m-1) + 2$ vertices, as described above, is of size $2m - 1$. There are exactly two options for selecting these two sets, either the sets consisting of the vertices $V_i^T = \{v_i^1, v_i^4, \dots, v_i^{p-1}\}$ or the sets consisting of the vertices $V_i^F = \{v_i^2, v_i^5, \dots, v_i^p\}$. Moreover, we can place the variable gadgets in such a way that the vertices $v_i^2, v_i^4, \dots, v_i^{p-1}$ lie above the x -axis and the other vertices lie below the x -axis.*

We interpret the assignments of the variable x_i to be true (false resp.) if V_i^T (V_i^F) is chosen as the maximum independent set in the variable gadget corresponding to x_i . Now we describe the structure of a clause gadget.

Clause Gadget. A clause gadget consists of a point set such that the corresponding graph contains 7 triangles. We describe the construction for a positive clause $C = (x_i \vee x_j \vee x_k)$ and the construction is analogous for the other clauses. The *central* triangle of the gadget in G_{P_ϕ} consists of three vertices c_{x_i}, c_{x_j} , and c_{x_k} as shown (with pink shade) in Fig. 4. The triangle is introduced by placing c_{x_j} and c_{x_k} as the first and second neighbor of c_{x_i} respectively. Then we draw three other vertices for each variable appearing in the clause. We describe the position of them for x_i and the others are analogous. p_i, q_i and r_i are the three vertices placed *near* c_{x_i} such that q_i has c_{x_i} as its first neighbor and p_i as its second neighbor. This introduces $\Delta q_i p_i c_{x_i}$ in G_{P_ϕ} . Then p_i has q_i and r_i as its first and second neighbor respectively introducing $\Delta p_i q_i r_i$ in G_{P_ϕ} . We adjust the neighborhood of the other vertices are adjusted in such a way that no other triangle is introduced in G_{P_ϕ} . For example, the first and second neighbors of r_i are q_i and p_i respectively and the first and second neighbors of c_{x_j} are c_{x_i} and c_{x_k} respectively. Considering the position of the points in the clause gadget following observation holds for the maximum independent set in the induced graph with the points in the clause gadget as the vertices.

Observation 2. *The maximum independent set in a clause gadget as described above has size 4 if and only if at least one of the three vertices r_i, r_j , and r_k is chosen in the independent set.*

We consider an ordering of the clauses from left to right in the planar embedding of ϕ . Let C be the ξ -th clause that is connected with the variable x_i from left to right. We *translate* each of the triangles $\Delta p_i q_i r_i$, $\Delta p_j q_j r_j$ and $\Delta p_k q_k r_k$ near to the corresponding variable gadgets to *connect* them. We place three points p'_i, q'_i , and r'_i near the point $v_i^{2+6(\xi-1)}$ such that r'_i has $v_i^{2+6(\xi-1)}$ and p'_i as its first and second neighbour. We call the vertex $v_i^{2+6(\xi-1)}$ as the connecting vertex of C and x_j . Then we construct a point set such that the maximum independent set induced by these points as vertices in G_{P_ϕ} has the following property. $p_i (q_i$ or r_i resp.) can be chosen in the maximum independent set if and only if $p'_i (q'_i$ or r'_i) is also chosen in the maximum independent set. We call this property as the *connection* of the clause to the variable gadgets. Once the connections to every clause and its variables are complete we have the following observation.

Observation 3. *The vertex c_{x_i} can be chosen in an independent set in G_{P_ϕ} only if the maximum independent set of the variable gadget corresponding to x_i is chosen to be V_i^T .*

The remaining work is to achieve the connection from C to x_i, x_j , and x_k . We use two point sets (gadgets) to perform this move. The first one is a *translating gadget* and the second one is a *rotating gadget* depicted in [8]. The following observation can be verified from the structure of the gadgets.

Observation 4. *Both the gadgets contain 6 points inside them and the size of the maximum independent set is 2.*

The translating gadget starts with a copy of a triangle and *propagates* the choice of the vertex chosen in an independent set in the underlying graph. The rotating gadget does the same thing but the final triangle in this gadget is a copy of the initial triangle in the gadget but rotated at a right angle. Thus we can use mirror-reflections of the rotating gadget depicted in the figure to achieve the rotating angle to be 180° and 270° . Together we call them the connecting gadgets.

Now the following observation is evident from the structure of the clause gadget and its connection with the variables.

Observation 5. *Let $C = (x_i \vee x_j \vee x_k)$ one clause that is connected with its variables in G_{P_ϕ} with ν connecting gadgets then the maximum independent set for the point set has size $3(2m + 1) + (\nu + 4)$ if and only if at least one of the three variables is assigned to be true.*

Now we adjust our drawing of the whole point set for G_{P_ϕ} such that the size of the independent set is a fixed number depending on ϕ . We place each clause gadget $C = (x_i \vee x_j \vee x_k)$ in such a way that the connecting gadgets for x_j do not contain a rotating gadget for the connection of c_{x_j} to its connecting vertex in x_j . We consider the fact that the nearest clause gadget from the x -axis can be connected with its variables by two rotating gadgets for the first and third variables and one translating gadget for the middle variable appearing in the

clause. Then we compute the maximum number of connecting gadgets required to connect a clause with the variables. Let l_v and w_v denote the length and width of the rectangles representing the variables in the planar drawing of ϕ . We place our variable gadgets inside the rectangles. So $l_v = O(m)$ as we have placed $p = 6(m-1) + 2$ points in each variable gadget. Since there are t variables and every clause rectangle can be connected with the variable rectangles using vertical lines, then the maximum length of a rectangle representing a clause could be $tl_v + \varepsilon$ for some small constant ε . On the other hand, if w_c and h_c^0 denote the width of the clause-rectangle and the height (distance from x -axis) of the nearest clause-rectangle from the x -axis, then the distance of the farthest clause rectangle from the x -axis is $mw_c + h_c^0$. Then the maximum number of connecting gadgets required for connecting a clause gadget to its variables is at most $3mw_c + 2tl_v$. Thus the clause gadget together with the translating and rotating gadget consists of $6(3mw_c + 2tl_v) + 12$ points. And thus contributing a $2(3mw_c + 2tl_v) + 4$ size maximum independent set. We draw our gadget in such a way that w_c contains 6 and l_v contains p points. Thus, every clause contains $6(18m + 2tp) + 12$ points together with its connecting gadgets.

To achieve this drawing, We expand the planar embedding of ϕ by a scalar $\alpha = (10tm)^2$. The size of the gadgets remains the same but the space between them increases such that there is a distance of at least α between any two connecting gadgets to accommodate the total number of points. The convolution to accommodate the extra points for the connection is depicted in [8]. A lower-level diagram for the embedding of such convolution with translating and rotating gadgets is depicted in [8]. Then we claim the following lemma.

Lemma 4. *Given a PRM-3SAT formula ϕ we can construct a planar point set P_ϕ such that G_{P_ϕ} has a maximum independent set of size $t(m+1) + m(2(18m + 2tp) + 4)$ if and only if ϕ is satisfiable.*

Since it is possible to check the correctness of a solution for an independent set we can conclude the following theorem.

Theorem 4. *Computing a maximum 2-multipacking of a given set of n points P in \mathbb{R}^2 is NP-complete.*

5.2 An Approximation and a Parameterised Algorithm

In this subsection, we proved that the maximum degree of G_P is bounded. The proof of the lemma stated below is given in the appendix.

Lemma 5. *The maximum degree of the graph G_P is at most 17.*

Using this we have shown that the 2-multipacking of a given set of points P in the plane can be approximated in polynomial time. We use existing techniques for computing the maximum independent set for bounded degree graphs.

The problem of finding the Maximum Independent Set in a bounded degree graph is denoted by *MAX IS-* Δ where the maximum node degree of the graph is bounded above by Δ . The following result states the approximation bound of the *MAX IS-* Δ problem.

Theorem 5. (Berman and Fujito [3]). *MAX IS- Δ can be approximated in polynomial time within a ratio arbitrarily close to $(\Delta + 3)/5$ for all $\Delta \geq 2$.*

By Theorem 5 and Lemma 5, we have the following result stated earlier.

Theorem 6. *A maximum 2-multipacking of a given set of points P in \mathbb{R}^2 can be approximated in polynomial time within a ratio arbitrarily close to 4.*

In 2019, Bonnet et al. have shown the following:

Theorem 7. (Bonnet et al. [4]). *For graphs of n vertices with degree bounded by Δ , we can compute an independent set of size k in time $(\Delta + 1)^k n^{O(1)}$, if it exists.*

Lemma 5 and Theorem 7 yield the following result stated earlier.

Theorem 8. *A 2-multipacking of size k of a given set of n points in \mathbb{R}^2 can be computed in time $18^k n^{O(1)}$, if it exists.*

6 Conclusion

The paper introduces a novel modeling of the maximum multipacking problem on the geometric domain concerning the Euclidean distance for a point set. An intriguing open topic in this area is addressing the difficulty of computing a maximum multipacking in general. We know neither the algorithmic difficulties nor the limitations as of yet for the $(n - 1)$ -multipacking problem.

The definition in this paper is motivated by problems in obnoxious facility location and multipacking in graph theory. In the definition given in this paper, one can introduce a fraction $k \in [0, 1]$ and generalise the given condition with respect to this fraction. For $k \in [0, 1]$, the condition on set M can be given as such: $|N_s[v] \cap M| \leq k(s + 1)$ for each point $v \in P$.

For different values of k , the bounds on the multipacking number as well as the algorithmic difficulties will differ significantly.

References

1. Abravaya, S., Segal, M.: Maximizing the number of obnoxious facilities to locate within a bounded region. *Comput. Oper. Res.* **37**(1), 163–171 (2010)
2. Beaudou, L., Brewster, R.C.: On the multipacking number of grid graphs. *Discrete Math. Theor. Comput. Sci.* **21**(Graph Theory) (2019)
3. Berman, P., Fujito, T.: On approximation properties of the independent set problem for low degree graphs. *Theor. Comput. Syst.* **32**, 115–132 (1999)
4. Bonnet, É., Bousquet, N., Thomassé, S., Watrigant, R.: When maximum stable set can be solved in fpt time. arXiv preprint [arXiv:1909.08426](https://arxiv.org/abs/1909.08426) (2019)
5. Brewster, R.C., MacGillivray, G., Yang, F.: Broadcast domination and multipacking in strongly chordal graphs. *Discret. Appl. Math.* **261**, 108–118 (2019)
6. Cappanera, P., Gallo, G., Maffioli, F.: Discrete facility location and routing of obnoxious activities. *Discret. Appl. Math.* **133**(1–3), 3–28 (2003)

7. Cornuéjols, G.: Combinatorial optimization: Packing and covering. SIAM (2001)
8. Das, A.K., Das, S., Islam, S.S., Mitra, R.M., Roy, B.: Multipacking in euclidean plane. arXiv preprint [arXiv:2411.12351](https://arxiv.org/abs/2411.12351) (2024)
9. Das, S., Foucaud, F., Islam, S.S., Mukherjee, J.: Relation between broadcast domination and multipacking numbers on chordal graphs. In: Conference on Algorithms and Discrete Applied Mathematics, pp. 297–308. Springer (2023)
10. Das, S., Islam, S.S.: Multipacking and broadcast domination on cactus graphs and its impact on hyperbolic graphs. arXiv preprint [arXiv:2308.04882](https://arxiv.org/abs/2308.04882) (2023)
11. Daskin, M.: Network and discrete location: models, algorithms and applications. *J. Oper. Res. Soc.* **48**(7), 763–764 (1997)
12. De Berg, M., Khosravi, A.: Optimal binary space partitions for segments in the plane. *Int. J. Comput. Geometry Appl.* **22**(03), 187–205 (2012)
13. Drezner, Z., Wesolowsky, G.O.: A maximin location problem with maximum distance constraints. *AIIE Trans.* **12**(3), 249–252 (1980)
14. Drezner, Z., Wesolowsky, G.O.: The location of an obnoxious facility with rectangular distances. *J. Regional Sci.* **23**(2) (1983)
15. Eppstein, D., Paterson, M.S., Yao, F.F.: On nearest-neighbor graphs. *Discrete Computat. Geom.* **17**, 263–282 (1997)
16. Foucaud, F., Gras, B., Perez, A., Sikora, F.: On the complexity of broadcast domination and multipacking in digraphs. *Algorithmica* **83**(9), 2651–2677 (2021)
17. Henzinger, M., Leniowski, D., Mathieu, C.: Dynamic clustering to minimize the sum of radii. *Algorithmica* **82**, 3183–3194 (2020)
18. Mehlhorn, K., Osbild, R., Sagraloff, M.: Reliable and efficient computational geometry via controlled perturbation. In: International Colloquium on Automata, Languages, and Programming, pp. 299–310. Springer (2006)
19. Meir, A., Moon, J.: Relations between packing and covering numbers of a tree. *Pac. J. Math.* **61**(1), 225–233 (1975)
20. Melachrinoudis, E.: Bicriteria location of a semi-obnoxious facility. *Comput. Ind. Eng.* **37**(3), 581–593 (1999)
21. Rakas, J., Teodorović, D., Kim, T.: Multi-objective modeling for determining location of undesirable facilities. *Transp. Res. Part D: Transp. Environ.* **9**(2), 125–138 (2004)
22. Rodríguez, J.S., García, C.G., Pérez, J.M., Casermeiro, E.M.: A general model for the undesirable single facility location problem. *Oper. Res. Lett.* **34**(4), 427–436 (2006)
23. Teshima, L.E.: Broadcasts and multipackings in graphs. Master's thesis, University of Victoria (2012)
24. Wolf, G.W.: Facility location: concepts, models, algorithms and case studies. series: Contributions to management science: edited by zanjirani farahani, reza and hekmatfar, masoud, heidelberg, germany, physica-verlag, 549 p. (2009). isbn 978-3-7908-2150-5 (hardprint), 978-3-7908-2151-2 (electronic) (2011)



Partial Domination in Some Geometric Intersection Graphs

Madhura Dutta^{1,4(✉)}, Anil Maheshwari², and Subhas C. Nandy³

¹ TCG CREST, Kolkata 700091, India

madhura1998kkg@gmail.com

² School of Computer Science, Carleton University, Ottawa, Canada

³ Ramkrishna Mission Vivekananda Centenary College, Kolkata 700118, India

⁴ Academy of Scientific and Innovative Research (AcSIR), Ghaziabad 201002, India

Abstract. *Partial domination problem* is a generalization of the *minimum dominating set problem* on graphs. Here, instead of dominating all the nodes, one asks to dominate at least a fraction of the nodes of the given graph by choosing a subset of nodes of minimum size. For any real number $\alpha \in (0, 1]$, α -partial domination problem can be proved to be NP-complete for general graphs. In this paper, we define the *maximum dominating k-set* of a graph which is polynomially transformable to the partial domination problem. We propose polynomial time algorithms for the maximum dominating k -set problem for some geometric intersection graphs, namely, interval graphs and unit square intersection graphs where the given squares are intersected by the straight line $L : y = -x$.

Keywords: Dominating set · partial domination · geometric intersection graphs

1 Introduction

The *minimum dominating set* problem for simple undirected graph $G = (V, E)$ is well-studied in graph theory. The objective is to choose a set $S \subseteq V$ of minimum cardinality such that for every node $v \in V \setminus S$, there exists at least one member $v' \in S$ satisfying $v \in N(v')$, where $N(v')$ is the set of nodes adjacent to v' . The *minimum dominating set* problem is a well-known NP-complete problem [15]. The hardness status remains unchanged for bipartite graphs and split graphs [3]. However, the problem is approximable to a $(1 + \log |V|)$ factor since it is a special instance of the set cover problem [21]. It admits a PTAS for planar graphs [2] and unit disk graphs [20]. The problem is linear time solvable for trees using dynamic programming [5, 9] and for interval graphs assuming sorted order of the end-points of the intervals corresponding to the nodes of the graph [6]. For the class of geometric intersection graphs of axis-parallel rectangles intersecting a straight line and some of its sub-classes, the problem has been studied in [25]. For a comprehensive survey on the dominating set problem, see [17, 18].

A. Maheshwari—Supported by NSERC.

Recently, in the context of communication networks, the problem of partial domination has become important. Usually, it is difficult to cover the entire network with a limited amount of resources. So, dominating (covering) a large subset using limited resources becomes important. In partial domination of a graph $G = (V, E)$, a real number $\alpha \in (0, 1]$ is given; the objective is to find a subset $S \subseteq V$ of minimum cardinality such that the size of the closed neighborhood of the set S , denoted by $N[S] = S \cup N(S)$, is greater than or equal to α times the size of the set V (i.e., $|N[S]| \geq \alpha|V|$) [4, 10], where $N(S) = \{u \mid u \in V \setminus S, \text{ and } (u, v) \in E \text{ for some } v \in S\}$. Theoretical study of partial domination is rarely observed in the literature. In [4], methods for evaluating the $\frac{1}{2}$ -partial dominating set for some graph classes, like cycles, paths, grids, etc., are suggested. Some practical applications of partial domination are also available in [4]. A parameterized algorithm for the partial dominating set problem of arbitrary graphs is given in [4], using results of [23]. Subexponential algorithms for this problem for planar graphs and apex-minor-free graphs are proposed in [1, 14], respectively.

Several variants of dominating set problems have been studied [11, 12, 19, 24]. We introduce the *maximum dominating k-set* problem for a graph. The objective is to dominate a maximum number of nodes in the graph with a suitably chosen subset of k nodes, where $k \geq 0$ is an integer. The maximum dominating k -set problem has many useful applications where a limited number of facilities are available, and the objective is to use them to cover most of the possible locations. This is similar to the maximum coverage problem, a variant of *set cover problem*; here, we must choose at most k sets whose union has maximum cardinality.

The study of *partial optimal solutions* to some optimization problems is an active area of research. In the context of social networks, the maximum k -cover problem is studied in [22], where the aim is to choose an influential subset of k individuals to reach (cover) as many individuals as possible. The problem is NP-hard, and an approximation algorithm is available [22]. In the context of geometric optimization, an $O(kn^2)$ time and $O(kn)$ space algorithm is available for the maximum hitting k -set problem for unit intervals where the objective is to place k points that hit the maximum number of unit intervals among n unit intervals that are arranged on a real line [8]. In [7], modified *local search* based PTASes have been proposed for computing the maximum hitting k -set problem for the intersection graphs of homothetic copies of convex objects, which includes disks, squares of arbitrary sizes, regular m -gons, etc.

Preliminaries: We first present some definitions to describe the problem.

Definition 1. Given a graph $G = (V, E)$ and a real number $\alpha \in (0, 1]$, an α -partial dominating set is a subset $S \subseteq V$ such that $|N[S]| \geq \alpha \cdot |V|$. The cardinality of an α -partial dominating set of minimum size is called the α -partial domination number of the graph, and it is denoted by γ_α .

Definition 2. Given a graph $G = (V, E)$ and an integer $k (\leq |V|)$, a maximum dominating k -set is a set $S \subseteq V$ with $|S| = k$ and $|N[S]|$ is maximized, i.e., for all $S' \subseteq V$ with $|S'| = k$, we have $|N[S]| \geq |N[S']|$. Here, $|N[S]|$ is referred to as the maximum dominated neighborhood size of a k -set.

Definition 3. A geometric intersection graph $G = (V, E)$ is a graph where each $v_i \in V$ corresponds to a geometric object, and $(v_i, v_j) \in E$ if and only if the objects corresponding to the vertices v_i and v_j have non-empty intersection.

For example, an interval graph is a geometric intersection graph of a set of intervals arranged on a straight line, and two vertices are adjacent if and only if the corresponding intervals overlap. Many hard problems in graph theory can be solved efficiently for the geometric intersection graph when the geometric objects corresponding to that graph satisfy some specific properties.

New Results: We study the maximum dominating k -set problem on graphs; the objective is to dominate a maximum number of nodes in the graph with a suitably chosen subset of k nodes. This problem and the partial domination problem are polynomially equivalent. The decision version of the partial dominating set problem (for a given $\alpha \in (0, 1]$) is NP-hard for arbitrary graphs using polynomial time reduction from the dominating set problem for arbitrary graphs [13]. Here, we study the maximum dominating k -set problem for different types of geometric intersection graphs. Our proposed algorithm runs in $O(nk \log n)$ time using $O(n)$ space for unit interval graphs. Using this, we construct an algorithm for arbitrary interval graphs. We also propose a polynomial time algorithm for the maximum dominating k -set problem for a unit square intersection graph, where the input squares are axis parallel and intersected by the straight line $y = -x$.

Our interest in studying the partial dominating set problem for some restricted classes of geometric intersection graphs stems from the fact that there exists polynomial time algorithms for the dominating set problem for these graph classes (see [6, 25]). In [1, 14], subexponential algorithms for partial domination problem are proposed for planar graphs and apex minor free graphs. For graphs with bounded local treewidth, these algorithms run in polynomial time. Note that interval graphs do not have bounded local treewidth, need not be planar, and can contain apex graphs as minor. The same applies to intersection graphs of d (≥ 2) dimensional geometric objects. This paper focuses on the partial domination problem for only interval graphs and intersection graphs of unit squares intersected by a straight line.

2 Max Dominating k -set problem for unit interval graphs

A graph $G = (V, E)$ is said to be an interval graph if there exists a layout of a set \mathcal{I} of n intervals such that each node $v_i \in V$ corresponds to an interval in \mathcal{I} ; each interval $v_i \in \mathcal{I}$ is specified by a pair of points (a_i, b_i) on a real line, termed as its start- and end-point. Each edge $e = (v_i, v_j) \in E$ implies that the intervals corresponding to v_i and v_j overlap in that layout. Testing whether a given graph $G = (V, E)$ is an interval graph needs $O(|V| + |E|)$ time [16]; in the same time, the corresponding interval layout can also be obtained. We can assume that the start- and end-points of the members in \mathcal{I} are distinct.

First, we solve the maximum dominating k -set problem for the unit interval graphs, where all intervals in \mathcal{I} have the same length. Here, for each pair of intervals $v_i, v_j \in V$, their corresponding intervals in \mathcal{I} are either disjoint or are properly overlapping (none of them is properly contained in the other interval).

2.1 Dynamic Programming Algorithm

Preprocessing: We sort the end-points of n intervals in \mathcal{I} , and name the intervals as $v_i = [a_i, b_i]$, $i = 1, \dots, n$, according to the increasing order of their right end-points. For each interval $v_i = [a_i, b_i]$, we store the following information:

1. $x(i)$ = Number of b_j 's ($j \neq i$) between a_i and b_i ,
2. $y(i)$ = Number of a_j 's ($j \neq i$) between a_i and b_i ,
3. $z(i)$ = Number of a_j 's ($j \neq i$) between b_{i-1} and b_i ,
4. $r_a(i) = \max\{a_j | a_i \leq a_j < b_i\}$, and
5. $r_b(i) = b_j$, where j is such that $r_a(i) = a_j$.

Algorithm: For a fixed b_i , $i \in \{1, \dots, n\}$, and a fixed integer ℓ ($\ell \leq k$), let us define three functions f , g and h as follows,

- $f(b_i, \ell) = \max\{|N[S_\ell]| : S_\ell \subseteq \{v_1, \dots, v_i\}, |S_\ell| = \ell, v_i \in S_\ell\}$ (* max dominated nbd size of an ℓ -set S_ℓ , where $S_\ell \subseteq \{v_1, \dots, v_i\}$ and $v_i \in S_\ell$ *),
- $g(b_i, \ell) = \max\{|N[S_\ell]| : S_\ell \subseteq \{v_1, \dots, v_i\}, |S_\ell| = \ell, v_i \notin S_\ell, v_i \in N[S_\ell]\}$ (* max dominated nbd size of an ℓ -set $S_\ell \subseteq \{v_1, \dots, v_i\}$, when $v_i \notin S_\ell, v_i \in N[S_\ell]$ *),
- $h(b_i, \ell) = \max\{|N[S_\ell]| : S_\ell \subseteq \{v_1, \dots, v_i\}, |S_\ell| = \ell, v_i \notin S_\ell, v_i \notin N[S_\ell]\}$ (* max dominated nbd size of an ℓ -set $S_\ell \subseteq \{v_1, \dots, v_i\}$, when $v_i \notin S_\ell, v_i \notin N[S_\ell]$ *).

Thus, the maximum dominated nbd size of a k -set is $\max\{f(b_n, k), g(b_n, k), h(b_n, k)\}$.

We find the values of the functions f , g , and h recursively using dynamic programming. We initialize these functions for b_1 as

- $f(b_1, 1) = 1 + y(1)$ and $f(b_1, j) = -\infty$ for $j = 0, 2, 3, \dots, k$.
- $g(b_1, j) = -\infty$ for all $j = 0, 1, 2, 3, \dots, k$.
- $h(b_1, 0) = 0$ and $h(b_1, j) = -\infty$ for $j = 1, 2, 3, \dots, k$.

We process b_i , $i = 1, 2, \dots, n$ in this order. For each b_i , we compute $f(b_i, l)$, $g(b_i, l)$ and $h(b_i, l)$, for $l = 0, 1, 2, \dots, k$, and proceed to compute the f, g, h functions for b_{i+1} . For all b_i , $f(b_i, 0) = g(b_i, 0) = -\infty$ (since these are undefined), and $h(b_i, 0) = 0$. Also, for all b_i , we set $f(b_i, 1) = 1 + y(i) + x(i)$. The recursive formulae for finding the values of the functions f , g , and h are stated below.

- $f(b_i, l) = \max\{T_1, T_2, T_3\}$, where

$$T_1 = 1 + \max_{\forall b_j < a_i, r_b(j) < a_i} \{(f(b_j, l-1) + y(i) + x(i))\}$$

$$T_2 = 1 + \max_{\forall b_j < a_i, r_b(j) = b_\lambda > a_i} \{f(b_j, l - 1) + y(i) + i - 1 - \lambda\}$$

$$T_3 = \max_{\forall b_j \in (a_i, b_i)} \{f(b_j, l - 1) + z(j + 1) + z(j + 2) + \dots + z(i - 1) + z(i)\}$$

- $g(b_i, l) = \max_{\forall b_j \in (a_i, b_i)} \{f(b_j, l)\}$
- $h(b_i, l) = \max_{\forall b_j < a_i} \{f(b_j, l)\}$

Note that, if there does not exist any $b_j \in (a_i, b_i)$ in the recursion formula of $g(b_i, l)$, then the value of $g(b_i, l) = -\infty$ is set, where $l \in \{1, 2, \dots, k\}$. Similarly, when there does not exist any b_j satisfying $b_j < a_i$ in the recursion formula of $h(b_i, l)$, then $h(b_i, l) = -\infty$ is set.

Justifications of the Formulae:

f(b_i, l): For the function f , the vertex corresponding to b_i , i.e., v_i is to be chosen, so we have the following three possibilities,

- **T₁:** In this case, v_i is not already dominated by the last chosen vertex v_j , so 1 is added. Here, additionally, because $r(b_j) < a_i$, we have to add $x(i)$ (number of intervals that ends in $[a_i, b_i]$) and $y(i)$ (number of intervals that starts in $[a_i, b_i]$), which are not dominated by any other previously chosen interval.
- **T₂:** Here also, v_i was not dominated by v_j , so 1 is added. Now, since $r(b_j) = b_\lambda \in (a_i, b_i)$, choosing v_i will newly dominate $y(i)$ (number of intervals that starts in $[a_i, b_i]$), and $(i - 1 - \lambda)$ (the number of intervals that ends in (b_λ, b_i)) nodes, which are not dominated by the previous chosen interval $v_j = [a_j, b_j]$.
- **T₃:** Here, the endpoint b_j of the last chosen interval v_j , lies in $[a_i, b_i]$; thus, v_i is already dominated by v_j . So unlike T_1 and T_2 , 1 is not added. As $a_j < a_i$, whatever end-points and start-points exist inside $[a_i, b_j]$, their corresponding intervals are already dominated and have been counted in $f(b_j, l - 1)$. So, we need to consider the part $[b_j, b_i]$. In between b_j and b_i , the only possible endpoints are $b_{j+1}, b_{j+2}, \dots, b_{i-1}, b_i$. But their start points are between a_j and a_i . So, they are already dominated by the interval $[a_j, b_j]$ and have already been counted before. So, we only need to add the number of intervals having their start point between b_j and b_i , which are $z(j + 1) + z(j + 2) + \dots + z(i - 1) + z(i)$. This is because they overlapped with v_i and were not yet counted.

g(b_i, l): In this case, the interval v_i is not chosen but was already dominated by a previously chosen interval v_j . It is indicated by b_j 's range in the max operation.

h(b_i, l): The interval v_i is neither chosen nor was dominated by any previously chosen interval v_j , and the range of b_j indicates this in the max operation of the recursion formula.

2.2 Processing, Correctness and Complexity Results

During the preprocessing, after sorting, we compute $x(i)$, $y(i)$, $z(i)$, $r_a(i)$ and $r_b(i)$ by processing the end-points of the intervals in \mathcal{I} in left to right order. We conceptually maintain a queue Q for storing the processed subset of left end-points of the intervals whose right end-points are not yet processed. Practically,

we maintain three scalar variables, Q_count (size of Q), Q_max (maximum element in Q), and Q^r_max (the right-end point of the interval corresponding to Q_max). While processing a left-end point, say a_j , we add 1 to Q_count , store a_j in Q_max , and its corresponding b_j in Q^r_max . When a right-end point b_i of an interval v_i is faced, its corresponding left-end point is the first element of Q (as all the intervals are of unit length); it is deleted, or equivalently, 1 is subtracted from Q_count . Note that, $y(i)$ is the size of Q at this instant and $r_a(i)$ is the maximum element in Q ; thus we set $y(i) = Q_count$, $r_a(i) = Q_max$, and $r_b(i) = Q^r_max$. If at this instant, Q is empty queue, then set $r_a(i) = a_i$ and $r_b(i) = b_i$. The $z(i)$ values are also computed in the same left-to-right scan by maintaining another scalar variable z_count . Another right-to-left scan is needed to compute $x(i)$ values for each interval $v_i \in \mathcal{I}$. Thus, the preprocessing time complexity is $O(n \log n)$ using $O(1)$ extra space. In dynamic programming, T_1 , T_2 and T_3 are computed as follows:

- While computing $f(b_i, l)$, note that $r_b(i) \geq b_i$ for all i . Thus, $T_1 = 1 + y(i) + x(i) + \max_{\forall r_b(j) < a_i} f(b_j, l-1) = 1 + y(i) + x(i) + T'_1$ (say). We maintain a leaf search height-balanced binary tree D_1 on $\{r_b(j), j = 1, 2, \dots, n\}$, where each leaf is associated with $f(b_j, l-1)$, and each internal node associated with the maximum of the $f(., l-1)$ values in the subtree rooted at that node. Thus, we can compute $T'_1 = \max_{\forall r_b(j) < a_i} f(b_j, l-1)$ in $O(\log n)$ time using D_1 .
- For computing T_2 , we need to compute $T'_2 = \max_{\forall b_j < a_i, a_i < r_b(j) = b_\lambda < b_i} (f(b_j, l-1) - \lambda)$, where $b_\lambda = r_b(j)$. Again, maintaining a similar leaf search height-balanced binary tree D_2 on $\{r_b(j), j = 1, 2, \dots, n\}$ with each leaf node associated with $(f(b_j, l-1) - \lambda)$, and each internal node associated with the maximum of the $(f(., l-1) - \lambda)$ values in the subtree rooted at that node, one can compute T'_2 in $O(\log n)$ time. Now, $T_2 = 1 + y(i) + i - 1 + T'_2$ is computed using T'_2 , obtained earlier.
- For computing T_3 , we use a D_3 data structure, which is a height-balanced leaf-search binary tree on $\{b_j, j = 1, 2, \dots, n\}$, whose each leaf b_α contains $val_\alpha = f(b_\alpha, l-1) + n_\alpha$, where n_α = number of start-points in the interval $[b_\alpha, b_i]$. Each internal node whose sub-tree contains leaves $\{b_\alpha, \dots, b_\beta\}$ represents a max-field containing the maximum of $\{val_\alpha, \dots, val_\beta\}$. Now, the T_3 value for b_i is obtained by searching the D_3 data structure for all b_j satisfying $a_i < b_j < b_i$. This also can be done in $O(\log n)$ time.

Thus, the computation of $f(b_i, l) = \max\{T_1, T_2, T_3\}$ needs $O(\log n)$ time using three data structures, each of size $O(n)$, and can be created in $O(n \log n)$ time. Similarly, $g(b_i, l)$ and $h(b_i, l)$ can also be computed in $O(\log n)$ time using a $O(n)$ size data structure which can be built in $O(n \log n)$ time. Thus,

Theorem 1. *The maximum dominating k-set problem algorithm for unit interval graphs runs in $O(nk \log n)$ time using data structures of size $O(n)$.*

3 Algorithm for General Interval Graph

Now, we relax the unit-interval assumption. Here, a pair of overlapping intervals in \mathcal{I} may either properly overlap or one interval properly contains the other one. We first execute a preprocessing phase based on the following observation.

Observation 1 *Let $v_i, v_j \in V$ be two vertices in G such that, in \mathcal{I} , the interval corresponding to v_i is completely contained in the interval corresponding to v_j . Here, if there exists an optimum solution S_{opt} of the maximum dominating k -set problem with $v_i \in S_{opt}$ then there exists another optimum solution S'_{opt} of that problem with $v_j \in S'_{opt}$ and $|S_{opt}| = |S'_{opt}|$.*

Thus, we execute a linear pass to identify the intervals of \mathcal{I} which are contained in at least one other interval of \mathcal{I} . These intervals are marked as deleted. Observe that the graph corresponding to the non-deleted intervals can be represented as a unit interval graph. Thus, we interchangeably use the terms unit interval layout and properly overlapping interval layout. During this processing, for each non-deleted interval $v_i, i \in \{1, 2, \dots, m\}$, we also compute the following information:

$$\begin{aligned} x_d(i) &= \text{Number of deleted intervals with only right-end point in } [a_i, b_i], \\ y_d(i) &= \text{Number of deleted intervals with only left-end point in } [a_i, b_i], \\ w_d(i) &= \text{Number of deleted intervals with both the end points in } [a_i, b_i]. \end{aligned}$$

Also, for each pair of intervals v_i and v_j , we find and store $c_d(i, j) = \text{number of deleted intervals dominated by both } v_i \text{ and } v_j$.

From now onwards, we use \mathcal{I}' to denote the non-deleted intervals. The end-points of the members in \mathcal{I} are stored in a sorted list \mathcal{L} , where each end-point of an interval $I \in \mathcal{I}$ points to its other end-point in \mathcal{L} . We execute a linear pass among the members in \mathcal{L} . During this execution, we maintain a balanced leaf-search binary tree \mathcal{T} . When the left-end point of a member of \mathcal{I} is processed, it is inserted in \mathcal{T} . When the right-end point b of an interval $I = [a, b]$ is processed, we search a in \mathcal{T} . If at least one entry to the left of a in \mathcal{T} exists, then I is fully contained in that interval; I is marked deleted and a is deleted from \mathcal{T} . Thus, in $O(n \log n)$ time we can compute the set \mathcal{I}' of properly overlapped intervals in the set \mathcal{I} . The set $\mathcal{I} \setminus \mathcal{I}'$ is the set of deleted intervals, which will not be considered in our dynamic programming algorithm for computing the maximum dominating k -set. To compute the values of the parameters defined earlier, we create a 2D range tree \mathcal{R} , whose each element is a 2D point (α, β) , which corresponds to the two end-points of a deleted interval $[\alpha, \beta] (\in \mathcal{I} \setminus \mathcal{I}')$. We consider each interval $I_i = [a_i, b_i] \in \mathcal{I}'$. Here, (i) $x_d(i) = \text{number of } \beta\text{-points lying in } I_i \text{ whose } \alpha\text{-point does not lie in } I_i$, (ii) $y_d(i) = \text{number of } \alpha\text{-points lying in } I_i \text{ whose } \beta\text{-point does not lie in } I_i$, and (iii) $w_d(i) = \text{number of points whose both } \alpha \text{ and } \beta \text{ values lie in } I_i$. All these three numbers can be computed in $O(\log^2 n)$ time.

To compute $c_d(i, j)$, for $v_i, v_j \in \mathcal{I}'$, we create a bipartite graph $B(\mathcal{I} \setminus \mathcal{I}', \mathcal{I}')$. While computing $x_d(i)$, $y_d(i)$ and $w_d(i)$, we have already identified the deleted intervals in $\mathcal{I} \setminus \mathcal{I}'$ that are dominated by each interval $I_i \in \mathcal{I}'$. Those edges are put in the graph B . At the end of this processing, we observe each node

$u_\theta \in \mathcal{I} \setminus \mathcal{I}'$, and add 1 to each cell $c_d(i, j)$ such that u_θ is adjacent to $v_i, v_j \in \mathcal{I}'$ (if any such pair exists). Thus, the worst-case time complexity of computing the graph B is $O(n^2)$, and computing $c_d(i, j), v_i, v_j \in \mathcal{I}'$ is also $O(n^2)$.

Note that the m intervals of \mathcal{I}' are properly overlapping intervals. So, this set of m intervals behaves as unit intervals, and computing the functions $x(i)$, $y(i)$, and $r_b(i)$ for $i = 1, 2, \dots, m$ is as in the previous section. Also, the definitions of the functions f , g , and h remain the same as in the previous section; their recursive formulae are given below. Finally, the maximum dominated neighborhood size of a k -set is computed as $\max\{f(b_m, k), g(b_m, k), h(b_m, k)\}$.

Initialization: The values of the three functions for $b_1 \in \mathcal{I}'$ are as follows,

- $f(b_1, 1) = 1 + y(1) + x_d(1) + y_d(1) + w_d(1)$ and $f(b_1, i) = -\infty$ for $i \neq 1$,
- $g(b_1, i) = -\infty$ for all $i = 0, 1, \dots, k$, and
- $h(b_1, 0) = 0$ and $h(b_1, i) = -\infty$ for $i = 1, 2, \dots, k$.

Also, we have $f(b_i, 0) = -\infty$, $g(b_i, 0) = -\infty$ and $h(b_i, 0) = 0$ for all $i \in \{1, 2, \dots, m\}$. For all b_i , $f(b_i, 1) = 1 + y(i) + x(i) + x_d(i) + y_d(i) + w_d(i)$ is also set in this initialization step.

Recursive Formulas: Recursive formulas for finding the values of the functions f , g , and h are as follows, which are minor tailoring of the formulae presented for unit intervals, and the justification of the tailoring is also mentioned.

- $f(b_i, l) = \max\{T_1, T_2, T_3\}$, where
 $T_1 = 1 + \max_{b_j < a_i, r_b(j) < a_i} \{f(b_j, l-1) + y(i) + x(i) + x_d(i) + y_d(i) + w_d(i) - c_d(i, j)\}$,
 $T_2 = 1 + \max_{b_j < a_i, r_b(j) = b_\lambda > a_i} \{f(b_j, l-1) + y(i) + i - 1 - \lambda + x_d(i) + y_d(i) + w_d(i) - c_d(i, j)\}$, and
 $T_3 = \max_{a_i < b_j < b_i} \{f(b_j, l-1) + z(j+1) + z(j+2) + \dots + z(i-1) + z(i) + x_d(i) + y_d(i) + w_d(i) - c_d(i, j)\}$,
• $g(b_i, l) = \max_{a_i < b_j < b_i} \{f(b_j, l)\}$,
• $h(b_i, l) = \max_{b_j < a_i} \{f(b_j, l)\}$.

If there does not exist any $b_j \in (a_i, b_i)$ in the recursive formula of $g(b_i, l)$, then $g(b_i, l) = -\infty$ is set, where $l \in \{1, 2, \dots, k\}$. Also, when no b_j with $b_j < a_i$ exists in the recursive formula of $h(b_i, l)$, then $h(b_i, l) = -\infty$, where $l \in \{1, 2, \dots, k\}$.

Justification of the Recursive Formulas:

When we consider the m non-deleted intervals, which are dominated by the chosen l -set, the recursive formulae will be the same as that of the unit interval graph on those m intervals. Below, we show that the count of the dominated deleted intervals by a chosen l -set is correct.

As the interval $v_i \in \mathcal{I}'$ is included in the chosen l -set, the number of deleted intervals dominated by v_i is $x_d(i) + y_d(i) + w_d(i)$; among them $c_d(i, j) =$ number of deleted intervals which are already dominated by the last included interval $v_j = [a_j, b_j] \in \mathcal{I}'$, was previously counted in $f(b_j, l-1)$. Hence, we added

$x_d(i) + y_d(i) + w_d(i) - c_d(i, j)$ in the recursion formula of $f(b_i, l)$, which is precisely the number of newly dominated deleted intervals by v_i .

For the functions g and h , the i -th interval is not included in the l -set while processing. Hence, no new deleted interval is dominated. Thus, the $O(n^2)$ time preprocessing leads to the following result.

Theorem 2. *The time complexity for computing the maximum dominating k -set problem for the intersection graph of a set of arbitrary intervals is $O(n^2k)$.*

4 Unit Square Intersection Graph Where a Straight Line Intersects Each Square

Let $S = \{s_1, s_2, \dots, s_n\}$ be a set of axis-parallel unit squares and $L : y = -x$ be a straight line intersecting all the members in S . The objective is to compute the maximum dominating k -set for the intersection graph of the squares in S .

Let T be a strip of width $\sqrt{2}$ divided into two strips of equal width by the given line L . We divide the strip into square-sized boxes of side-length $\sqrt{2}$ (Fig. 1). As the members in S are unit squares, each $s_i \in S$ may have a non-empty intersection with at most two consecutive boxes in the strip T . So, at most, $2n$ boxes may contain some portion of the squares in S . Let these boxes be denoted as T_1, T_2, \dots, T_m ($m \leq 2n$). Note that the centers of the squares in S also lie in these $2n$ boxes. We use $S_i (\subseteq S)$ to denote the squares whose centers lie in T_i .

Result 1. [25] *For each $i \in \{1, 2, \dots, m\}$, (a) there exists at most 4 squares of S which can dominate all the squares whose centers lie in T_i , and (b) the minimum number of members in S that are needed to dominate all the squares intersecting the box T_i is at most 12.*

Result 1(a) leads to the following lemma, which determines the time complexity of our algorithm for the maximum dominating k -set problem for S .

Lemma 1. *There exists a maximum dominating k -set, say OPT , where each box T_i contains the centers of at most 11 members of OPT .*

Proof (By contradiction). Let OPT be an optimum solution to the said problem with at least one box in T having the centers of more than 11 squares of OPT . Let T_j ($j \geq 1$) be the leftmost such box containing the centers of the squares $O_j \subseteq OPT$, $|O_j| \geq 12$ (here, $O_j = S_j \cap OPT$). By Result 1(a), we can replace/choose at most 4 squares which can dominate all the squares with center inside T_j . Thus, the remaining at least 8 squares in O_j are used to dominate squares centered inside $T_{j-1} \cup T_{j+1}$. By Result 1(a), instead of these remaining squares of O_j , we can take 4 squares from S_{j-1} and 4 from S_{j+1} to dominate all the squares centered in $T_{j-1} \cup T_{j+1}$. Also, by pigeonhole principle, among the remaining at least 8 squares in O_j , at least 4 will contribute in dominating elements of S_{j-1} (or S_{j+1}). Without loss of generality, assume that $O'_j \subseteq O_j$ ($|O'_j| \geq 4$)

is used to dominate the squares in S_{j-1} . We delete O'_j from OPT , and add at most 4 squares, say O''_j from S_{j-1} , centered at four sub-boxes of T_{j-1} , in OPT to dominate all the squares in S_{j-1} . These, in addition, may dominate some squares in S_{j-2} . Thus, we now have $\tilde{O}_{j-1} = O_{j-1} \cup O''_j$, and the total size of OPT is not increased. Also, $|O_j|$ became < 12 . If $|\tilde{O}_{j-1}| \geq 12$, then again, we apply the same technique to delete some squares from \tilde{O}_{j-1} and add the required number (≤ 4) squares from S_{j-2} without increasing the initial size of OPT . The process continues up to the box T_1 . If the size of \tilde{O}_1 (after adding some new squares) is ≥ 11 , we can replace it by at most 4 squares. Thus, the size of OPT is reduced maintaining the size of the dominated squares in $S_1 \cup \dots \cup S_j$ unchanged or increased, while the number of squares in $O_{j+1} \cup \dots \cup O_m$ remains unchanged. Thus, we have a contradiction that the chosen solution OPT is not optimum. \square

Using Lemma 1, we construct a dynamic programming based algorithm for the maximum dominating k -set problem.

- S_i^l : a subset of S_i of size l .
- $d(S_i^l)$: set of all squares dominated by S_i^l .
- $d_c(S_i^{l_1} \cup S_{i+1}^{l_2}, S_{i+2}^{l_3})$: set of all squares dominated by both $S_i^{l_1} \cup S_{i+1}^{l_2}$ and $S_{i+2}^{l_3}$.

For a specific choice of the tuple $(l_1, l_2, S_{i-1}^{l_3}, S_i^{l_4})$, where l_1, l_2, l_3 and l_4 are non-negative integers, $S_{i-1}^{l_3}$ is a subset of size l_3 of S_{i-1} and $S_i^{l_4}$ is a subset of size l_4 of S_i , we denote by $N(l_1, l_2, S_{i-1}^{l_3}, S_i^{l_4})$ the maximum dominated neighborhood size of a $(l_1 + l_2 + l_3 + l_4)$ -set, where $S_{i-1}^{l_3}$ is a subset of l_3 elements chosen from S_{i-1} , $S_i^{l_4}$ is a subset of l_4 elements chosen from S_i , a subset of l_2 elements is chosen from S_{i-2} and a subset of l_1 elements is chosen from $\cup_{j=1}^{i-3} S_j$ in the $(l_1 + l_2 + l_3 + l_4)$ -set.

With these notations, the recursion formula for $N(l_1, l_2, S_{i-1}^{l_3}, S_i^{l_4})$ is

$$N(l_1, l_2, S_{i-1}^{l_3}, S_i^{l_4}) = \max\{N(l_1 - a, a, S_{i-2}^{l_2}, S_{i-1}^{l_3}) + d(S_i^{l_4}) - d_c(S_{i-2}^{l_2} \cup S_{i-1}^{l_3}, S_i^{l_4}) : S_{i-2}^{l_2} \subseteq S_{i-2} \text{ and } |S_{i-2}^{l_2}| = l_2, \text{ and } 0 \leq a \leq \min\{l_1, 11\} \text{ is an integer}\}.$$

Lemma 1 says that $l_2, l_3, l_4 \leq 11$ for any choice of the tuple $(l_1, l_2, S_{i-1}^{l_3}, S_i^{l_4})$. The choice for l_1 can be any integer in $\{0, 1, \dots, k\}$.

In the preprocessing step of the dynamic programming algorithm, the values of the functions d and d_c are calculated for all the required subsets of S . Then, each box T_i is processed for $i = 1, 2, \dots, m$, in this order. We calculate and store all possible values of $N(l_1, l_2, S_{i-1}^{l_3}, S_i^{l_4})$ using the recursion formula. Finally, an optimal maximum dominated neighborhood size OPT of a k -set is obtained as

$$OPT = \max\{N(l_1, l_2, S_{m-1}^{l_3}, S_m^{l_4}) : S_{m-1}^{l_3} \subseteq S_{m-1} \text{ with } |S_{m-1}^{l_3}| = l_3, S_m^{l_4} \subseteq S_m \text{ with } |S_m^{l_4}| = l_4 \text{ and } l_1 + l_2 + l_3 + l_4 = k, 0 \leq l_2, l_3, l_4 \leq 11\}.$$

Note that for the initialization of the dynamic programming, we add four dummy boxes T_{-3}, T_{-2}, T_{-1} and T_0 respectively just before the box T_1 and initialize $N(0, 0, \emptyset, \emptyset) = 0$ and all other entries which are undefined are initialized to $-\infty$.

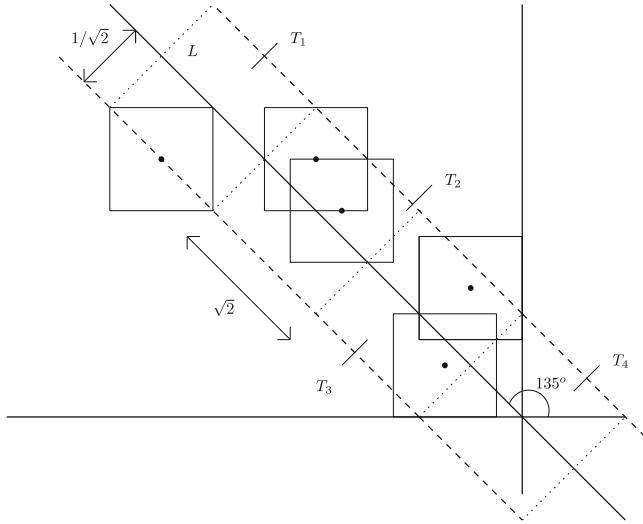


Fig. 1. Demonstration of the problem

Justification of the Recursion Formula: While processing T_i , when we choose l_4 elements of the set $S_i^{l_4}$ from i -th box T_i , it dominates $d(S_i^{l_4})$ elements. Now, there are only three possibilities,

1. None of these $d(S_i^{l_4})$ elements were previously dominated. In this case, the last chosen squares can have centers in at most T_{i-3} ; not in any later box.
2. Some of these $d(S_i^{l_4})$ elements were already dominated, but none of the l_4 elements of $S_i^{l_4}$ were dominated. In this case, the last chosen squares have centers in the box T_{i-2} .
3. Some of the l_4 elements of $S_i^{l_4}$ were already dominated. Here, the last chosen squares have centers in the box T_{i-1} .

Correctness follows from the fact that the above three cases have been taken care of in the recursion formula.

Time Complexity: Calculating $N(l_1, l_2, S_{i-1}^{l_3}, S_i^{l_4})$ for a given $(l_1, l_2, S_{i-1}^{l_3}, S_i^{l_4})$ requires $O(n^{11})$ time. For each i , number of tuples of the form $(l_1, l_2, S_{i-1}^{l_3}, S_i^{l_4})$ is $O(kn^{22})$ and there are at most $2n$ possible values of i . Hence, the time needed to calculate the function N for all possible arguments is $O(kn^{34})$. Finally, calculating OPT requires $O(n^{11} \cdot n^{11}) = O(n^{22})$ time. Hence, the time complexity of the algorithm is $O(kn^{34}) + O(n^{22}) = O(kn^{34})$, a polynomial in n (since $k \leq n$).

References

1. Amini, O., Fomin, F.V., Saurabh, S.: Implicit branching and parameterized partial cover problems. *J. Comput. Syst. Sci.* **77**(6), 1159–1171 (2011)
2. Baker, B.S.: Approximation algorithms for NP-complete problems on planar graphs. *J. ACM* **41**, 153–180 (1994)
3. Bertossi, A.A.: Dominating sets for split and bipartite graphs. *Inf. Process. Lett.* **19**, 37–40 (1984)
4. Case, B.M., Hedetniemi, S.T., Laskar, R.C., Lipman, D.J.: Partial domination in graphs. In: Southeastern International Conference on Combinatorics, Graph Theory and Computing, 6–10 March 2017
5. Chang, G.J.: Algorithmic Aspects of Domination in Graphs. Springer (2013)
6. Chang, M.S.: Efficient algorithms for the domination problems on interval and circular arc graphs. *SIAM J. Comput.* **27**, 1671–1694 (1998)
7. Chaplick, S., De, M., Ravsky, A., Spoerhase, J.: Approximation schemes for geometric coverage problems. *ESA* **17**(1–17), 15 (2018)
8. Chung, C., Vignaron, A., Ahn, H.K.: Maximum coverage by k lines, Unpublished manuscript (2023)
9. Cockayne, E., Goodman, S., Hedetniemi, S.: A linear algorithm for the domination number of a tree. *Inf. Process. Lett.* **4**, 41–44 (1975)
10. Das, A.: Partial domination in graphs. *Iranian J. Sci. Technol. Trans. A: Sci.* **43**, 1713–1718 (2019)
11. Das, A., Laskar, R.C., N., Rad, J.: On α -domination in graphs. *Graphs Combinatorics* **34**, 193–205 (2018)
12. Domke, B.S., Fricke, G.H., Laskar, R.R., Majumdar, A.: Fractional domination and related parameters. In: Haynes, T.W., Hedetniemi, S.T., Slator, P.J. (eds.) Domination in Graphs, Chapter 3, Taylor & Francis (1998)
13. Dutta, M., Maheshwari, A., Nandy, S.C.: Partial domination in some geometric intersection graphs (2024)
14. Fomin, F.V., Lokshtanov, D., Raman, V., Saurabh, S.: Subexponential algorithms for partial cover problems. *Inf. Process. Lett.* **111**(16), 814–818 (2011)
15. Garey, M.R., Johnson, D.S., Klee, V. (ed.). Computers and intractability: a guide to the theory of NP-completeness. A Series of Books in the Mathematical Sciences. San Francisco, Calif.: W. H. Freeman and Co. (1979)
16. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Academic Press (1980)
17. Haynes, T.W., Hedetniemi, S.T., Slater, P.J. (eds.). Domination in Graphs: Advanced Topics, Marcel Dekker, New York (1998)
18. Haynes, T.W., Hedetniemi, S.T., Slater, P.J.: Fundamentals of Domination in Graphs. Marcel Dekker, New York (1998)
19. Hedetniemi, S.M., Hedetniemi, S.T., Wimer, T.: Linear time resource allocation algorithms for trees. Technical Report, (1987). URL-014
20. Hunt, H.B., III., Marathe, M.V., Radhakrishnan, V., Ravi, S.S., Rosenkrantz, D.J., Stearns, R.E.: NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *J. Algorithms* **26**, 238–274 (1998)
21. Johnson, D.S.: Approximation algorithms for combinatorial problems. *J. Comput. System Sci.* **9**, 256–278 (1974)
22. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the Spread of Influence through a Social Network. *Theory of Computing - An Open Access Journal*. vol. 11, Article 4, pp. 105–147 (2015)

23. Kneis, J., Mölle, D., Rossmanith, P.: Partial vs. complete domination: t -dominating set. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 367–376. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-69507-3_31
24. Lan, J.K., Chang, G.J.: Algorithmic aspects of the k -domination problem in graphs. Discrete Appl. Math. **161**, 1513–1520 (2013)
25. Pandit, S.: Dominating set of rectangles intersecting a straight line. J. Comb. Optim. **41**(2), 414–432 (2021). <https://doi.org/10.1007/s10878-020-00685-y>



Generalized Lettericity of Graphs

Zhidan Feng¹, Henning Fernau¹, Kevin Mann¹, Indhumathi Raman^{2(✉)},
and Silas Cato Sacher¹

¹ Abteilung Informatikwissenschaften, Fachbereich 4, Universität Trier, 54286 Trier,
Germany

{fernau,mann,sacher}@uni-trier.de

² Department of Computing Technologies, SRM Institute of Science and Technology,
Kattankulathur, Chennai 603203, India
indhumar2@srmist.edu.in

Abstract. We introduce a new graph parameter called *generalized lettericity* as a generalization of the *lettericity* parameter that was introduced two decades ago. Given a word $w = w_1 w_2 \cdots w_n$ of length $|w| = n$ over a finite alphabet Σ and a decoder set \mathcal{D} of words over Σ with length at least 2, we define a *generalized letter graph* $G(\mathcal{D}, w)$ with respect to \mathcal{D} and w . $G(\mathcal{D}, w)$ has vertex set $[n]$ and there is an edge between $i < j$ if there is a subsequence s of $w_i \cdots w_j$, with $s_1 = w_i$ and $s_{|s|} = w_j$, such that $s \in \mathcal{D}$. We define the generalized lettericity $gl(G)$ of a graph G as the least size of the alphabet Σ such that there exists a decoder \mathcal{D} and a word w over Σ such that G is isomorphic to $G(\mathcal{D}, w)$. In this paper, we analyze some properties of $gl(G)$ for specific classes of G and also precisely determine $gl(G)$ when the graph G is (i) a complete k -partite graph or its complement, (ii) a cycle, (iii) a path, and (iv) a comb.

Keywords: Graph parameters · Letter graphs · Lettericity

1 Introduction

Lettericity is a graph parameter that has been introduced in [10] and has recently received renewed attention, see [1–9]. For instance, Alecu et al. [3] showed that the problem of determining if a given graph has lettericity at most k is fixed-parameter tractable for the parameter k using MSO-formulas. They conjectured this problem to be NP-complete. Otherwise, most papers focus on purely graph-theoretic considerations, and this is also the focus of our paper concerning a new notion called *generalized lettericity* that we will introduce.

For a set A , we denote by $|A|$ its cardinality. The set of positive integers is denoted as \mathbb{N}^+ . For $k \in \mathbb{N}^+$, we define $[k] = \{1, \dots, k\}$. All graphs considered in this paper are finite, simple and undirected. Given a graph G , we denote by $V(G)$ its vertex set and by $E(G)$ its edge set. For a vertex $v \in V$, we denote by $N(v)$ and $N[v]$ the open and closed neighborhood of v and its number of neighbors (called its degree) by $\deg(v)$. We denote by $G_1 \cong G_2$ that G_1 is isomorphic to G_2 and by $G[M]$ the induced subgraph of G induced by $M \subseteq V(G)$, i.e., $V(G[M]) = M$ and $E(G[M]) = \{\{u, v\} \in E(G) \mid u, v \in M\}$. The complement of G , written

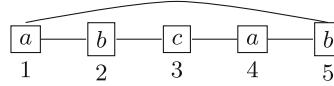


Fig. 1. $l(C_5) \leq 3$ since $C_5 \cong \Gamma_{\mathcal{D}}(w)$ for $w = abcab$ and $\mathcal{D} = \{ab, bc, ca\}$.

as \overline{G} , is a graph with $V(\overline{G}) = V(G)$ and $E(\overline{G})$ containing an unordered pair $\{u, v\} \subseteq V(G)$ if and only if $\{u, v\} \notin E(G)$. We write the cycle with n vertices as C_n and the path with n vertices as P_n . We denote the complete graph or clique on n vertices by K_n , so $\overline{K_n}$ is an independent set or null graph on n vertices. For positive integers n_1, \dots, n_k , K_{n_1, \dots, n_k} denotes the complete k -partite graph, with vertex set $V = V_1 \cup \dots \cup V_k$ with $|V_i| = n_i$ such that $\overline{K_{n_1, \dots, n_k}}$ is a cluster graph with clique components V_1, \dots, V_k , i.e., $\overline{K_{n_1, \dots, n_k}} = K_{n_1} \cup \dots \cup K_{n_k}$. Here, \cup denotes graph union, i.e., union is applied to vertex and edge sets, implicitly assuming that the vertex sets of the participating graphs are disjoint. Then nK_2 can be seen as a shorthand of the union of n (copies of) K_2 .

For an alphabet Σ , we denote by $\Sigma^{\geq i}$ (or by Σ^i) the set of all words over Σ with length at least (or exactly) i . As usual, $\Sigma^{\geq 0}$ is denoted by Σ^* . For a word $w \in \Sigma^*$, $|w|$ is the length of w . For a word $w = w_1 w_2 \dots w_{|w|}$ and $i \in [|w|]$, w_i denotes the i -th letter of w . If $i_1 < i_2 < \dots < i_m \leq |w|$, then $w_{i_1} w_{i_2} \dots w_{i_m}$ is a subsequence of w . Letter graphs, introduced by Petkovšek [10], describe a way to build graphs from words.

Definition 1 ([10]). *Let Σ be an alphabet and let $\mathcal{D} \subseteq \Sigma^2$ be a language called a decoder. For a word $w \in \Sigma^*$, the letter graph $\Gamma_{\mathcal{D}}(w)$ is defined to be a graph with vertex set $[|w|]$ and edge set $\{(i, j) \mid i < j \text{ and } w_i w_j \in \mathcal{D}\}$. More precisely, $\Gamma_{\mathcal{D}}(w)$ is said to be a $|\Sigma|$ -letter graph.*

Trivially, every graph on n vertices is isomorphic to an n -letter graph. This motivates the following definition.

Definition 2 ([10]). *The lettericity $l(G)$ of a graph G is defined as the minimum integer k such that there exists a word w over some Σ with $|\Sigma| = k$ and a decoder set $\mathcal{D} \subseteq \Sigma^2$ such that G is isomorphic to the letter graph $\Gamma_{\mathcal{D}}(w)$.*

Example 1. Consider $w = abcab$ over the alphabet $\Sigma = \{a, b, c\}$ and the decoder $\mathcal{D} = \{ab, bc, ca\}$. The letter graph $\Gamma_{\mathcal{D}}(w)$ has vertex set $[5]$ and the edges $\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{1, 5\}$ which is a C_5 , showing that $l(C_5) \leq 3$; see Fig. 1. The upper bound is the exact value due to Proposition 1.

We state the following results on lettericity of graphs proved in [7, 10]. In [10], Petkovšek showed that lettericity of a path $l(P_n) \leq \lfloor \frac{n+4}{3} \rfloor$ and recently Ferguson [7] showed that the lower bound matches this upper bound.

Proposition 1. *Let G denote any graph on $n = |V(G)|$ vertices. Then*

1. [10] $l(G) \leq n - 1$ if $n \geq 2$.

2. [10] $l(G) = l(\overline{G})$.
3. [10] $l(G) = 1$ if and only if $G \cong K_n$ or $G \cong \overline{K_n}$.
4. [10] $l(C_n) = \lfloor \frac{n+4}{3} \rfloor$ for $n \geq 4$.
5. [7] $l(P_n) = \lfloor \frac{n+4}{3} \rfloor$ for $n \geq 3$.

In this paper, we define *generalized lettericity* of a graph in Sect. 2 and study some of its properties in Sect. 3. We then provide lower bounds for the generalized lettericity of certain graphs like cycles, paths, regular graph, etc. in Sect. 4. In Sects. 5 and 6, we provide exact values for generalized lettericity of cycles/paths and comb graphs, respectively.

2 Generalized Lettericity

In this section, we introduce a new graph parameter called *generalized lettericity* of a graph by considering longer words in the decoder, i.e., we allow the decoder of a generalized letter graph to have words of length at least 2 which is in contrast to the decoder containing words of length exactly 2 in case of lettericity.

Definition 3. Let Σ be an alphabet and let $\mathcal{D} \subseteq \Sigma^{\geq 2}$ called a decoder. For any $w = w_1 w_2 \dots w_{|w|} \in \Sigma^*$, the generalized letter graph of w w.r.t. \mathcal{D} is $G(\mathcal{D}, w) = (V(\mathcal{D}, w), E(\mathcal{D}, w))$ where $V(\mathcal{D}, w) = [|w|]$ and $E(\mathcal{D}, w) = \bigcup_{s \in \mathcal{D}} E(s, w)$ with

$$E(s, w) = \{\{i_1, i_m\} \mid 1 \leq i_1 < i_2 < \dots < i_m \leq |w| \text{ and } w_{i_1} w_{i_2} \dots w_{i_m} = s\}$$

The generalized lettericity $gl(G)$ of G is the smallest $k \in \mathbb{N}^+$ such that there is a Σ with $|\Sigma| = k$, some $\mathcal{D} \subseteq \Sigma^{\geq 2}$ and some $w \in \Sigma^*$ such that $G \cong G(\mathcal{D}, w)$.

From the definition, it is clear that $1 \leq gl(G) \leq l(G)$ for all graphs G . Using this inequality and a result of [10] from Proposition 1, we observe the following.

Proposition 2. For any $n \in \mathbb{N}^+$, $gl(K_n) = gl(\overline{K_n}) = 1$.

In contrast to lettericity, where no other graphs but complete and null graphs have lettericity one, there are more graphs with generalized lettericity one:

Example 2. Consider $w = aaaa$ over the alphabet $\Sigma = \{a\}$ and $\mathcal{D} = \{aaa\}$. The generalized letter graph $G(\mathcal{D}, w)$ has edges $\{1, 3\}$, $\{1, 4\}$, $\{2, 4\}$, i.e., $G(\mathcal{D}, w) \cong P_4$; see Fig. 2. Hence, $gl(P_4) = 1 < 2 = l(P_4) = \lfloor \frac{4+4}{3} \rfloor$ by Proposition 1.

This shows that $gl(G) < l(G)$ is indeed possible. This example clearly generalizes to all $G(\{aaa\}, a^n)$ for $n > 3$, describing graphs with $1 = gl(G) < l(G)$. We now look at one example with $2 = gl(G) < l(G)$ that should motivate us through the rest of the paper where we study the question when $gl(G) = l(G)$ and when $gl(G) < l(G)$ and also how large the difference $l(G) - gl(G)$ could get.

Example 3. Consider $w' = abababa$ over $\Sigma = \{a, b\}$ and $\mathcal{D} = \{abab, baba\}$. Since $E(abab, w_3) = \{\{1, 6\}, \{1, 4\}, \{3, 6\}\}$ and $E(baba, w_3) = \{\{2, 5\}, \{2, 7\}, \{4, 7\}\}$, $G(\mathcal{D}, w_3) \cong P_7$; see Fig. 2. So, $gl(P_7) \leq 2$.

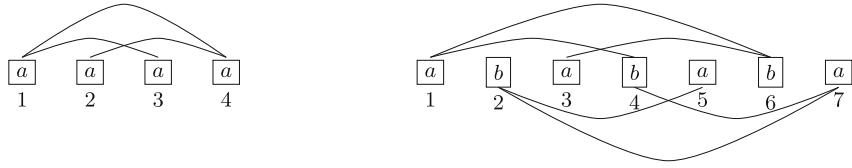


Fig. 2. Examples of generalized lettericity: $gl(P_4) = 1$ and $gl(P_7) = 2$.

Remark 1. Observe that $gl(P_7) \neq 1$. Suppose, to the contrary, that $gl(P_7) = 1$. Then a decoder $\mathcal{D}' \subseteq \Sigma^{\geq 2}$ with $\Sigma = \{a\}$ exists such that $G(\mathcal{D}', a^7) \cong P_7$. If \mathcal{D}' contains a word $v = a^k$ with $2 \leq k \leq 5$, then $\deg(1) > 2$. However, if $\mathcal{D}' \subseteq \{a^6, a^7\}$, then $\deg(3) = 0$. Both cases yield a contradiction to our assumption. Hence, we conclude that $gl(P_7) = 2 < 3 = l(P_7) = \lfloor \frac{7+4}{3} \rfloor$ by Proposition 1. \square

3 Properties of Generalized Lettericity of Graphs

In this section, we collect a number of combinatorial observations on properties of our new graph parameter. They will be very useful when we study several special graph classes. We illustrate this already in this section by various examples where we apply our observations. The first observation is immediate from the definition.

Lemma 1. *Let $\alpha, \beta \in \Sigma$ and $u, v, w \in \Sigma^*$. If $u = \alpha u' \beta$ is a subsequence of $v = \alpha v' \beta$, then $E(v, w) \subseteq E(u, w)$.*

Proposition 3. *Let G be a graph of order n . If $gl(G) = 1$, then there exists a decoder $\mathcal{D} = \{a^{k+1}\}$ of size one such that $G = G(\mathcal{D}, a^n)$ for some $k \in [n]$.*

Proof. The complete graph K_n and its complement $\overline{K_n}$ on n vertices has generalized lettericity 1 and there exists a decoder $\mathcal{D} = \{a^{k+1}\}$ for $k = 1$ and $k = n$, respectively, such that $G(\{a^2\}, a^n) \cong K_n$ and $G(\{a^{n+1}\}, a^n) \cong \overline{K_n}$. For $2 \leq k \leq n - 1$, assume $gl(G) = 1$ is certified by a decoder \mathcal{D}' of size more than one. Let a^k be the word of smallest length in \mathcal{D}' . Clearly, a^k is a subsequence of any element $u \in \mathcal{D}'$. By Lemma 1, $E(\{u\}, a^n) \subseteq E(\{a^k\}, a^n)$. Hence, $G(\{a^k\}, a^n) = G(\mathcal{D}', a^n)$. \square

By Proposition 3, we know that for each $n \in \mathbb{N}$ there is surjection from $[n]$ to the graphs of order n and generalized lettericity one. Hence, we have:

Proposition 4. *For $n \geq 1$, there are at most n non-isomorphic graphs G_i , $1 \leq i \leq n$ such that $|V(G_i)| = n$ and $gl(G_i) = 1$ for all i , $1 \leq i \leq n$.* \square

Quite interesting are structural properties of graphs of certain values of gl .

Proposition 5. *If $gl(G) = 1$, then G has zero or two vertices of degree one.*

Proof. Let G be a graph of order n . By Proposition 3, we can assume $G = G(\mathcal{D}, a^n)$, $\mathcal{D} = \{a^{k+1}\}$ for some $k \in [n]$. If $2(k+1) \leq n$, it is clear that every vertex has degree at least two. Hence, we assume $2k+2 > n$ now. Hence, in the linear ordering of $[n]$, vertices will have either only neighbors to their left or only neighbors to their right. More precisely, vertices j with $j \in [k]$ will only have neighbors to their right, so that then $\deg(j) = \max(n - k - j + 1, 0)$. For this range, $n - k - j + 1 = 1$ means $j_1 = n - k$ as a possible vertex of degree one. Vertices left to j_1 (i.e., in $[j_1 - 1]$) will have degree more than one. If $j_1 \in [n]$, then we have found one vertex of degree one. A symmetric argument applies to vertices j with $n - k - 1 < j \leq n$. Hence, there might be another vertex $j_2 := n - j_1 = k$ of degree one. As j_1 exists if and only if j_2 exists, we actually find these two vertices of degree one if $2k+2 > n$ as then $j_1 < j_2$. In other words, $G = G(\mathcal{D}, a^n)$ has zero or exactly two vertices of degree one if and only if $2k+2 \leq n$ or $2k+2 > n$, respectively. \square

Remark 2. The converse of Proposition 5 is not true. The path P_3 has exactly two vertices of degree one. We claim that $gl(P_3) \neq 1$. Suppose, if $gl(P_3) = 1$, then we have to consider the word $w = aaa$ and a non-empty decoder $\mathcal{D} \subseteq \{aa, aaa\}$. By Proposition 3, this leaves us with two cases. If $\{aa\} = \mathcal{D}'$, then $G(\mathcal{D}', w) \cong C_3$. If $\{aaa\} = \mathcal{D}'$, then $G(\mathcal{D}', w)$ is isomorphic to the disjoint union of K_2 and K_1 . In either case, $G(\mathcal{D}', w)$ is not isomorphic to P_3 . Hence $gl(P_3) \neq 1$. \square

Corollary 1. *If P_3 denotes the path on 3 vertices, then $gl(P_3) = 2$.*

Proof. By Proposition 1 and Remark 2, it is clear that $gl(P_3) \leq l(P_3) \leq \lfloor \frac{3+4}{2} \rfloor = 2$ and $gl(P_3) \neq 1$, respectively. Hence, $gl(P_3) = 2$. \square

It is also interesting to compare the structural properties of graphs of bounded lettericity with those of graphs of bounded generalized lettericity. From the examples that we saw so far, we can already deduce two important differences.

Proposition 6. *Let $k \in \mathbb{N}^+$. While induced subgraphs of graphs of lettericity k have lettericity at most k , this closure property is in general not true for graphs of bounded generalized lettericity.*

Proof. It is known that classes of graphs of bounded lettericity are hereditary, see [10]. Remark 2 shows that the class of graphs of generalized lettericity k need not be closed under induced subgraphs. Consider $k = 1$: We have $gl(P_4) = 1$, see Example 2, but $gl(P_3) = 2$, cf. Corollary 1. \square

Another structural property, also mentioned in Proposition 1, is that graphs of bounded lettericity are closed under graph complement. Again, this nice property is not true for bounded generalized lettericity.

Proposition 7. *$gl(G) = gl(\overline{G})$ is not true for all graphs G .*

Proof. Observe that $\overline{P_3}$ is the disjoint union of a K_1 and a K_2 . By Remark 2, $gl(\overline{P_3}) = 1$, while by Corollary 1, $gl(P_3) = 2$. \square

It is of course interesting to see if we can make quantitative statements: How different can the generalized lettericity of a graph G , namely $gl(G)$, be from $gl(\overline{G})$? We attempt to answer this question in the next section.

4 Lower Bound Techniques

In order to determine lower bounds on any graph parameter related to a minimization problem, general techniques should be developed. This will avoid repeating the same type of arguments over and over again. The following result will be our main tool in the context of generalized lettericity.

Theorem 1. *If G has an induced subgraph $G' \cong nK_2$, then $gl(G) \geq n$.*

Proof. Let $G = (V, E)$ be a graph with an induced subgraph $G[M] \cong nK_2$, $M \subseteq V$. Assume that $gl(G) < n$, and $G = (\mathcal{D}, w)$, where $w \in \Sigma^{|V|}$, $\mathcal{D} \subseteq \Sigma^{\geq 2}$, then by pigeon hole, there must be a letter $a \in \Sigma$ representing at least 3 of the $2n$ vertices in $G[M]$. Without loss of generality, let $w_i = w_j = w_k = a$, $i < j < k$, where $w_i, w_j, w_k \in M$, then for the middle vertex w_j , if its neighbor w_x on $G[M]$ appears before w_j , then w_x must be also adjacent to w_k . If w_x appears after w_j , analogously, it must be also adjacent to w_i . In both cases, w_x has at least two neighbors on $G[M]$, which contradicts to $G[M] \cong nK_2$. Hence, $gl(G) \geq n$. \square

We will now quantify and sharpen Proposition 7 towards the following result.

Theorem 2. *There is a family of graphs (G_ℓ) such that $gl(G_\ell)$ is constant, while with growing ℓ , $gl(\overline{G}_\ell)$ grows beyond all bounds.*

This theorem is proved by considering, more concretely, the complete multipartite graphs. This result is presented next.

Theorem 3. *For $k \geq 2$ and $n_1, n_2, \dots, n_k \geq 2$, the complete k -partite graph K_{n_1, n_2, \dots, n_k} has generalized lettericity 2, but its complement graph $\overline{K}_{n_1, n_2, \dots, n_k}$ has generalized lettericity k .*

Proof. By choosing $w = x_1^{n_1} x_2^{n_2} \cdots x_k^{n_k}$ over $\Sigma = \{a, b\}$, where $x_i = b$ if i is even and $x_i = a$ if i is odd, we note that $G(\{ab, ba, aba, bab\}, w) \cong K_{n_1, n_2, \dots, n_k}$. Hence $gl(K_{n_1, n_2, \dots, n_k}) \leq 2$. The equality follows from Proposition 5.

Clearly, $\overline{K}_{n_1, n_2, \dots, n_k}$ is a union of k cliques of size at least 2, then we can arbitrarily select one edge from each clique, forming an induced matching kK_2 , so $gl(\overline{K}_{n_1, n_2, \dots, n_k}) \geq k$ from Theorem 1. Then choose $w = a_1^{n_1} a_2^{n_2} a_3^{n_3} \cdots a_k^{n_k}$ over $\Sigma = \{a_1, a_2, \dots, a_k\}$. The (generalized) letter graph $G(\{a_i a_i \mid 1 \leq i \leq k\}, w)$ is isomorphic to $\overline{K}_{n_1, n_2, \dots, n_k}$. Hence $gl(\overline{K}_{n_1, n_2, \dots, n_k}) \leq k$. \square

Next, we quantify the statement that the parameters lettericity and generalized lettericity may be different, by proving the next result. Notice that this will largely sharpen what we understood from Example 2.

Theorem 4. *There is a family of graphs (G_ℓ) such that $gl(G_\ell)$ is constant, while with growing ℓ , $l(G_\ell)$ grows beyond all bounds.*

Interestingly, we can now take the same example as in Theorem 3.

Theorem 5. For $k \geq 2$ and $n_1, n_2, \dots, n_k \geq 2$, the complete k -partite graph K_{n_1, n_2, \dots, n_k} has generalized lettericity 2, but lettericity k .

Proof. Based on Proposition 1 and Theorem 3, we know $k \leq gl(\overline{K_{n_1, n_2, \dots, n_k}}) \leq l(\overline{K_{n_1, n_2, \dots, n_k}}) = l(K_{n_1, n_2, \dots, n_k})$. Then choosing $v = a_1^{n_1} a_2^{n_2} \cdots a_k^{n_k}$ for distinct letters a_1, a_2, \dots, a_k , the letter graph $\Gamma_{\mathcal{D}}(v)$ where $\mathcal{D} = \{a_i a_j \mid 1 \leq i < j \leq k\}$ is isomorphic to K_{n_1, n_2, \dots, n_k} . Hence, $l(K_{n_1, n_2, \dots, n_k}) \leq k$. \square

Next, we will collect a number of combinatorial lower bound results on graphs with specific properties, as having a certain maximum degree.

Proposition 8. Let $G = (V, E)$ be a graph with n vertices and $r \in [n - 1]$.

1. If G has maximum degree r and no isolates, then $\frac{n}{2r} \leq gl(G)$.
2. If G is r -regular and does not contain a $K_{r,r}$ as an induced subgraph, then $\frac{n}{2r-1} \leq gl(G)$.
3. If G is a path and $n > 4$, then $\frac{n-1}{3} \leq gl(G)$.
4. If G is a cycle and $n \geq 3$, then $\frac{n}{3} \leq gl(G)$.

Proof. 1. Assume there is a $w \in \Sigma^n$, a $\mathcal{D} \subseteq \Sigma^{\geq 2}$ and $a \in \Sigma$ such that $E = E(\mathcal{D}, w)$ and $|w|_a > 2r$. Let $i \in [|w|]$ such that $w_i = a$ and $|w_1 \cdots w_i|_a = r + 1$. Let z_1, \dots, z_r denote the vertices $w_{z_p} = a$ and $z_p < i$ for $p \in [r]$. Hence, $|w_i \cdots w_n|_a \geq r + 1$. As i is not isolated, there exists a $j \in [n] \cap N(i)$. Assume $i < j$ (for $j < i$, the proof works analogously). Hence, there exists a $u = u_1 \dots u_l \in \mathcal{D}$ and a sequence $k_1, \dots, k_l \in [n]$ such that $k_1 = i$, $k_l = j$, $w_{k_t} = u_t$ and $k_s < k_t$ for all $s, t \in [l]$ with $s < t$. Then $w_{z_p} u_2 \cdots u_l = u \in D$ for each $p \in [r]$. Hence, $\deg(w_j) \geq r + 1$ which is a contradiction to $\deg(v) \leq r$ for all $v \in [n]$. Therefore, there is no symbol in Σ which appears more than $2r$ times and hence $\frac{n}{2r} \leq gl(G)$.

2. Let G be an r -regular graph without an induced $K_{r,r}$. Assume there is a $w \in \Sigma^n$, a $\mathcal{D} \subseteq \Sigma^*$ and $a \in \Sigma$ such that $E = E(\mathcal{D}, w)$ and $|w|_a \geq 2r$. Let $i \in [|w|]$ such that $w_i = a$ and $|w_1 \cdots w_i|_a = r$. Let z_1, \dots, z_{r-1} denote the vertices $w_{z_p} = a$ and $z_p < i$ for $p \in [r-1]$. Furthermore, $|w_i \cdots w_n|_a \geq r + 1$. If i has a neighbor j with $j < i$, analogously to above, $\deg(j) \geq r + 1$. Thus, for all $j \in N(i)$, we find $i < j$. As above, j is neighbor of z_p for $p \in [r-1]$. Therefore, $\deg(j) \geq r + 1$ or $N(j) = \{z_1, \dots, z_{r-1}, i\}$. Since this holds for all $j \in N(i)$, $G[N(i) \cup \{z_1, \dots, z_{r-1}, i\}] = K_{r,r}$. This contradicts that G is $K_{r,r}$ -free. Hence, for all $a \in \Sigma$, $|w|_a \leq 2r - 1$ and $\frac{n}{2r-1} \leq gl(G)$.

3. For $n \in \{5, 6, 7\}$, by the proof of Proposition 3 it is easy to see that $gl(G) \neq 1$. By the proof of item 1, we only need to consider $w \in \Sigma^n$, a $\mathcal{D} \subseteq \Sigma^*$ and $a \in \Sigma$ such that $E = E(\mathcal{D}, w)$ and $|w|_a \leq 4$ for all $a \in \Sigma$. Assume there are $a, b \in \Sigma$ with $|w|_a = |w|_b = 4$. Define i_p and j_p such that $w_{i_p} = a$ and $w_{j_p} = b$ for all $p \in [4]$ as well as $i_p < i_q$ and $j_p < j_q$ for $p, q \in [4]$ with $p < q$. Since a path has at most two vertices with degree one, there exists $t \in \{i_2, i_3, j_2, j_3\}$ with $\deg(t) = 2$. Without loss of generality, $t = i_2$ (if $t \in \{j_2, j_3\}$ then swap a and b , if $t = i_3$ consider w^R and \mathcal{D}^R). Let k, l be the neighbors of i_2 . If

$k < i_2$ (analogously for $l < i_2$), then $i_2, i_3, i_4 \in N(k)$. Hence, $i_2 < k, l$. Then $k, l \in N(i_1)$ and (i_1, k, i_2, l, i_1) is a cycle on G . Therefore, there exists at most one $a \in \Sigma$ with $|w|_a = 4$. Thus, the statement holds.

4. Let $G = C_n$. If $n \neq 4$ then C_n is a $K_{2,2}$ -free 2-regular graph and the result follows by above. For $n = 4$, $C_n = K_{2,2}$. Assume $gl(C_4) \neq 1$. By Proposition 3, there is a $k \in [4]$ with $C_4 \cong G(\{a^k\}, aaaa)$. For $k = 1$, G' does not contain an edge while $G' \cong K_4$ for $k = 2$. If $k = 3$, then $G' \cong P_4$. Since G' only has one edge for $k = 4$, $gl(C_4) \neq 1$. \square

5 Generalized Lettericity of Cycles and Paths

In this section, we determine the exact value of $gl(C_n)$ for all $n \geq 3$ and of $gl(P_n)$ for all $n \in \mathbb{N}^+$. First, let us focus on cycles. By Proposition 8, $gl(C_n) \geq \frac{n}{3}$ for $n \geq 3$, and similarly, for paths. We establish a matching upper bound, namely $gl(C_n) \leq \frac{n}{3}$ for $n \geq 4$ in three cases depending on the whether $n \equiv 0, 1, 2 \pmod{3}$ in Lemmas 2, 3 and 4, respectively.

Lemma 2. *For each $m \in \mathbb{N}^+$, $gl(C_{3m}) = m$.*

Proof. By Proposition 8, $m \leq gl(C_{3m})$. To establish an upper bound, for $m = 1$, choose $w = \{aaa\}$ and $\mathcal{D} = \{aa\}$. Then $G(\mathcal{D}, w) \cong C_3$. Hence $gl(C_3) = 1$. For $m \geq 2$, we choose $w := a_1a_2 \cdots a_{m-1}a_ma_1a_2 \cdots a_{m-1}a_ma_1a_2 \cdots a_{m-1}a_m$ over the alphabet $\Sigma = \{a_1, a_2, \dots, a_m\}$ with the decoder

$$\mathcal{D}_o := \{a_ma_{m-1}, a_{m-1}a_{m-2}, \dots, a_2a_1\} \cup \{a_1a_ma_m\}.$$

For all $1 < k \leq m$,

$$E(a_ka_{k-1}, w) = \{\{k, m+k-1\}, \{m+k, 2m+k-1\}, \{k, 2m+k-1\}\}.$$

Furthermore, $E(a_1a_ma_m, w) = \{\{1, 2m\}, \{1, 3m\}, \{m+1, 3m\}\}$; refer to Fig. 3a. Hence, the graph $G(\mathcal{D}_o, w)$ contains the cycle $(2m, 3m-1, m, 2m-1, 3m-2, m-1, 2m-2, \dots, m+2, 2m+1, 2, m+1, 3m, 1, 2m)$, and no further edges, implying $G(\mathcal{D}_o, w) \cong C_{3m}$. This shows $gl(C_{3m}) \leq m$. \square

The next lemmas are quite analogous, so that the proofs are shortened.

Lemma 3. *For each $m \in \mathbb{N}^+$, $gl(C_{3m+1}) = m+1$.*

Proof. By Proposition 8, we already know that $m+1 \leq gl(C_{3m+1})$. To establish an upper bound, for $m = 1$, observe $G(\{ab, ba\}, abab) \cong C_4$. For $m \geq 2$, choose $w' = a_{m+1}a_2a_3 \dots a_{m-1}a_ma_1a_ma_{m+1}a_2a_3 \dots a_{m-1}a_ma_1a_2 \dots a_{m-1}a_m$ over the alphabet $\Sigma' = \{a_1, a_2, \dots, a_m, a_{m+1}\}$ with the decoder

$$\mathcal{D}' = \{a_ma_{m-1}, a_{m-1}a_{m-2}, \dots, a_2a_1\} \cup \{a_1a_{m+1}, a_{m+1}a_ma_m\}.$$

The edges of $E(\{a_1a_{m+1}\}, w') \cup E(\{a_{m+1}a_ma_m\}, w')$ are shown in Fig. 3b. \square

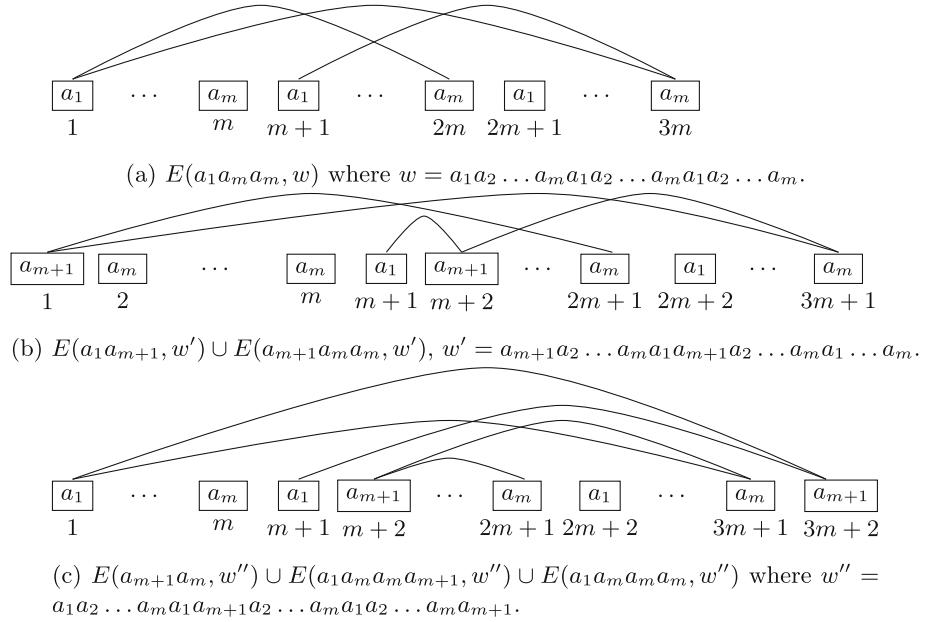


Fig. 3. Some edges of C_{3m} , C_{3m+1} , and C_{3m+2} , respectively.

Lemma 4. For each $m \in \mathbb{N}^+$, $gl(C_{3m+2}) = m + 1$.

Proof. By Proposition 8, we already know that $m + 1 \leq gl(C_{3m+2})$. To establish an upper bound for $m = 1$, observe $G(\{baa, aab, aaa\}, baaab) \cong C_5$. For $m \geq 2$, choose the word $w'' = a_1 \dots a_m a_1 a_{m+1} a_2 \dots a_m a_1 \dots a_m a_{m+1}$ over the alphabet $\Sigma'' = \{a_1, a_2, \dots, a_m, a_{m+1}\}$ with the decoder

$$\mathcal{D}'' = \{a_m a_{m-1}, \dots, a_3 a_2, a_2 a_1\} \cup \{a_{m+1} a_m, a_1 a_m a_m a_{m+1}, a_1 a_m a_m a_m\}.$$

One may refer to Fig. 3c that depicts the edges of $E(\{a_{m+1} a_m\}, w'') \cup E(\{a_1 a_m a_m a_{m+1}\}, w'') \cup E(\{a_1 a_m a_m a_m\}, w'')$. \square

Combining Lemma 2, Lemma 3, Lemma 4, we have the following theorem.

Theorem 6. For each cycle C_n , $n \geq 3$, $gl(C_n) = \lceil \frac{n}{3} \rceil$. \square

Comparing the result of Theorem 6 to the known result from [10] which states $l(C_n) = \lfloor \frac{n+4}{3} \rfloor$, we can hence conclude the following.

Corollary 2. For each cycle C_n for $n \geq 3$,

$$gl(C_n) = \begin{cases} l(C_n) - 1 & \text{if } n \not\equiv 1 \pmod{3}, \\ l(C_n) & \text{if } n \equiv 1 \pmod{3} \end{cases}$$

\square

Next, we determine the exact value of $gl(P_n)$ for all $n \geq 4$. By Proposition 8, $gl(P_n) \geq \frac{n-1}{3}$ for $n > 4$. We establish the upper bound, namely $gl(P_n) \leq \frac{n-1}{3}$ for $n > 4$, in three cases, depending on whether $n \equiv 0, 1, 2 \pmod{3}$ in Lemmas 5, 6 and 7, respectively. Due to the similarity to the cycle case, we suppress most correctness arguments for the constructions.

Lemma 5. *For each $m \in \mathbb{N}$ with $m \geq 2$, $gl(P_{3m}) = m$.*

Proof. The lower bound $m \leq gl(P_{3m})$ is given by Proposition 8. For the upper bound, define $w := (a_1 \cdots a_m)^2 a_m \cdots a_1$ over the alphabet $\Sigma = \{a_1, a_2, \dots, a_m\}$, as well as $\mathcal{D} := \{a_m a_m a_1\} \cup \{a_i a_{i+1} a_{i+1} \mid i \in [m-1]\}$. Then $G = ([m], E) = G(\mathcal{D}, w)$ with edge set E being equal to

$$\{\{k, m+k+1\}, \{k, 3m-k\}, \{m+k, 3m-k\} \mid k \in [m-1]\} \cup \{\{m, 3m\}, \{2m, 3m\}\}.$$

$(m+1, 3m-1, 1, m+2, 3m-2, 2, m+3, \dots, 2m-1, 2m+1, m-1, 2m, 3m, m)$ is a path. As $|E| = 3m-1$ and the path uses all edges, G is indeed a path. \square

Lemma 6. *For each $m \in \mathbb{N}^+$, $gl(P_{3m+1}) = m$.*

Proof. The lower bound $m \leq gl(P_{3m+1})$ is given by Proposition 8. For $m = 1$, refer to Example 2. Otherwise, consider $w := (a_1 \cdots a_m)^3 a_m$ over $\Sigma = \{a_1, \dots, a_m\}$ and the decoder $\mathcal{D} = \{a_m a_{m-1}, a_{m-1} a_{m-2}, \dots, a_2 a_1\} \cup \{a_1 a_m a_m a_m\}$. This leads to the graph $G(\mathcal{D}, w)$ exactly representing the following path as required: $(2m, 3m+1, m, 2m-1, 3m-2, \dots, 2m+1, 2, m+1, 3m+1, 1, 3m)$. \square

Lemma 7. *For each $m \in \mathbb{N}^+$, $gl(P_{3m-1}) = m$.*

Proof. The lower bound $m \leq gl(P_{3m-1})$ is given by Proposition 8. For $m = 1$, consider $G(\{aa\}, aa) \cong P_2$. Otherwise, we again use the idea of Lemma 2. By deleting the last symbol a_m in the word w of Lemma 2, we consider the resultant word $w = (a_1 a_2 \dots a_m)^2 a_1 \dots a_{m-1}$ over the alphabet $\Sigma = \{a_1, a_2, \dots, a_m\}$ and the (same) decoder \mathcal{D}_o . Then the graph $G(\mathcal{D}_o, w)$ describes the following path: $(1, 2m, 3m-1, m, 2m-1, 3m-2, \dots, m+2, 2m+1, 2, m+1)$. \square

Combining the results of Lemmas 5, 6, and 7, we have:

Theorem 7. *For each path P_n , $n \geq 4$, $gl(P_n) = \lceil \frac{n-1}{3} \rceil$.* \square

Theorem 7 does not hold for $n = 3$; see Corollary 1. Moreover, from Theorem 7 and Proposition 1, we get:

Corollary 3. *For each path P_n , $n \geq 4$, $gl(P_n) = l(P_n) - 1$.* \square

6 Generalized Lettericity of Comb Graphs

In this section, we consider the *comb graph with m teeth* E_{2m} , with vertex set $\{v_1, \dots, v_m, u_1, \dots, u_m\}$ and edges $\{\{u_i, v_i\} \mid i \in [m]\} \cup \{\{v_i, v_{i+1}\} \mid i \in [m-1]\}$.

Proposition 9. *Let $G = (V, E)$ be a tree such that every vertex has exactly one leaf in its closed neighborhood. If $G = G(\mathcal{D}, w)$ for some Σ , $w \in \Sigma^{|V|}$ and $\mathcal{D} \subseteq \Sigma^{\geq 2}$, then for all $a \in \Sigma$, $|w|_a \leq 4$.*

Proof. Assume there is a $a \in \Sigma$, a $\mathcal{D} \subseteq \Sigma^{\geq 2}$ and a $w \in \Sigma^{|V|}$ such that $|w|_a = k \geq 5$ and $G = G(\mathcal{D}, w)$ is a tree with $|\{u \in N[v] \mid \deg(u) = 1\}| = 1$, for all $v \in V$. Let $1 \leq i_1 < i_2 < \dots < i_k \leq |V|$ be with $w_{i_j} = a$ for all $j \in [k]$. We will consider i_3 . If i_3 has two neighbors $j_1, j_2 > i_3$, then these are also neighbors of i_1, i_2 . Hence $(i_1, j_1, i_2, j_2, i_1)$ is a cycle. This is a contradiction as G is a tree. Analogously, i_3 has two neighbors $j_1, j_2 < i_3$. Therefore, $\deg(i_3) \leq 2$.

Furthermore, assume that $\deg(i_3) = 2$, and there are $j_1, j_2 \in N(i_3)$ with $j_1 < i_3 < j_2$. Then $|N(j_1)| \geq 3, |N(j_2)| \geq 3$. This is not possible, since for any $v \in V$, the closed neighborhood $N[v] \cap L \neq \emptyset$, where L is the set of leaves.

So assume $\deg(i_3) = 1$, and $N(i_3) = \{j\}$ with $i_3 < j$. Since each vertex in G has exactly one leaf, there exists an $l \in V$ such that $\{j, l\} \subseteq N(i_2)$. If $l < i_2$, then $\{i_3, l\} \in E$ and (i_3, j, i_2, l, i_3) is a cycle. If $i_2 < l$, then $\{i_1, l\} \in E$ and (i_1, j, i_2, l, i_1) is a cycle. Thus, $\deg(i_2) = 1$ and j has two neighbors of degree 1. Hence, for each $a \in \Sigma$, $|w|_a \leq 4$. \square

Both Theorem 1 and Proposition 9 (independently) imply the following.

Corollary 4. *If E_n is a comb of order n , then, $\lceil \frac{n}{4} \rceil \leq gl(E_n) \leq l(E_n)$.* \square

Theorem 8. *For all $m \in \mathbb{N}^+$, $gl(E_{2m}) = \lceil \frac{m}{2} \rceil$.*

Proof. From Corollary 4, for a $2m$ -vertex comb graph E_{2m} , we have $gl(E_{2m}) \geq \lceil \frac{2m}{4} \rceil = \lceil \frac{m}{2} \rceil$. To establish the upper bound, we choose

$$w_m := \begin{cases} a_1 a_1, & m = 1, \\ a_1 a_1 a_1 a_1, & m = 2, \\ a_k w_{m-1} a_k, & m = 2k - 1, k \geq 2, \\ w_{m-1} a_k a_k, & m = 2k, k \geq 2. \end{cases}$$

Clearly, for $m = 1$, $E_2 \cong P_2 \cong G(\{a_1 a_1\}, w_1)$, and for $m = 2$, $E_4 \cong P_4 \cong G(\{a_1 a_1 a_1 a_1\}, w_2)$ by Example 2. Hence $gl(E_{2m}) \leq 1 = \lceil \frac{m}{2} \rceil$ for $m = 1, 2$.

If $m = 2k - 1, k \geq 2$, we choose the decoder $\mathcal{D}^o = \{a_j a_j a_j, a_{j+1} a_j a_j a_j \mid j \in [k-1]\} \cup \{a_k a_k\}$, then for all $j \in [k-1]$, $E(\{a_j a_j a_j\}, w_j) = \{\{k+1-j, k+3j-1\}, \{k+1-j, k+3j\}, \{k+3j-2, k+3j\}\}$, which are the edges between the vertices with the same letter a_j ; and $E(\{a_{j+1} a_j a_j a_j\}, w_j) = \{k-j, k+3j\}$, which connects two vertices with two consecutive letters a_{j+1} and a_j . Furthermore, $E(\{a_k a_k\}, w_m) = \{1, 4k-2\}$, and we see no further edges, giving the comb graph E_{2m} of order $2m = 4k-2$. Therefore, $gl(E_{2m}) \leq k = \lceil \frac{m}{2} \rceil$.

If $m = 2k, k \geq 2$, we choose the decoder $\mathcal{D}^e = \{a_j a_j a_j, a_{j+1} a_j a_j a_j \mid j \in [k]\}$, and $G(\mathcal{D}^e, w_j) \cong E_{2m}$ of order $2m = 4k$. So, $gl(E_{2m}) \leq k = \lceil \frac{m}{2} \rceil$. \square

Table 1. Contrasting properties of lettericity and generalized lettericity of graphs

Lettericity	Generalized Lettericity
1. For all graphs G , $l(G) = l(\overline{G})$.	$gl(P_3) \neq gl(\overline{P_3})$.
2. $l(K_{n_1, n_2, \dots, n_k}) = l(\overline{K_{n_1, \dots, n_k}}) = k$.	$gl(K_{n_1, n_2, \dots, n_k}) = 2$ and $gl(\overline{K_{n_1, \dots, n_k}}) = k$.
3. $l(G) = 1$ if and only $G \cong K_n$ or $G \cong \overline{K_n}$.	There are n non-isomorphic graphs G_i such that $ V(G_i) = n$ and $gl(G_i) = 1$.
4. The class of graph with lettericity at most k is hereditary.	Does not obey this closure property, $gl(P_4) = 1$ but for its induced subgraph P_3 , $gl(P_3) = 2$.

7 Conclusion and Open Problems

In this paper, we introduced a new graph parameter called generalized lettericity $gl(G)$ for a graph G . We showed some contrasting properties of generalized lettericity in comparison to the classical lettericity and also determined the generalized lettericity for cycles, paths and comb graphs; refer to Table 1.

It is clearly interesting to explore generalized lettericity for more classes of graphs. Apart from this, one can also consider to bound the length of the words in the decoder. In this paper, while calculating the generalized lettericity of cycles or paths or combs, we used decoder words of maximum length 4. It would be interesting (from a descriptive complexity point of view) to explore similar properties for decoder words of length at most three.

References

- Alecu, B., Alekseev, V.E., Atminas, A., Lozin, V.V., Zamaraev, V.: Graph parameters, implicit representations and factorial properties. *Discret. Math.* **346**(10), 113573 (2023)
- Alecu, B., Ferguson, R., Kanté, M.M., Lozin, V.V., Vatter, V., Zamaraev, V.: Letter graphs and geometric grid classes of permutations. *SIAM J. Discrete Math.* **36**(4), 2774–2797 (2022)
- Alecu, B., Kanté, M.M., Lozin, V.V., Zamaraev, V.: Lettericity of graphs: an FPT algorithm and a bound on the size of obstructions. Technical Report [arXiv: 2402.12559](https://arxiv.org/abs/2402.12559), ArXiv, Cornell University (2024)
- Alecu, B., Lozin, V.V.: Understanding lettericity I: a structural hierarchy. Technical Report [arXiv: 2106.03267](https://arxiv.org/abs/2106.03267), ArXiv, Cornell University (2021)
- Alecu, B., Lozin, V.V., de Werra, D., Zamaraev, V.: Letter graphs and geometric grid classes of permutations: characterization and recognition. *Discret. Appl. Math.* **283**, 482–494 (2020)
- Bera, S., Mahalingam, K.: Structural properties of word representable graphs. *Math. Comput. Sci.* **10**(2), 209–222 (2016)
- Ferguson, R.: On the lettericity of paths. *Australas. J. Comb.* **78**, 348–351 (2020)
- Ferguson, R., Vatter, V.: Letter graphs and modular decomposition. *Discret. Appl. Math.* **309**, 215–220 (2022)

9. Mandrick, S., Vatter, V.: Bounds on the lettericity of graphs. Technical Report [arXiv:2305.02920](https://arxiv.org/abs/2305.02920), ArXiv, Cornell University (2023)
10. Petkovsek, M.: Letter graphs and well-quasi-order by induced subgraphs. *Discret. Math.* **244**(1), 375–388 (2002)



Polynomial-Time Algorithms for PATH COVER on Trees and Graphs of Bounded Treewidth

Florent Foucaud¹ , Atrayee Majumder² , Tobias Mömke² ,
and Aida Roshany-Tabrizi²

¹ CNRS, Clermont Auvergne INP, Mines Saint-Étienne, LIMOS,
63000 Clermont-Ferrand, France

² University of Augsburg, Augsburg, Germany
aida.roshanytabrizi@uni-a.de

Abstract. In the PATH COVER problem, one asks to cover the vertices of a graph using the smallest possible number of (not necessarily disjoint) paths. While the variant where the paths need to be pairwise vertex-disjoint, which we call PATH PARTITION, is extensively studied, surprisingly little is known about PATH COVER. We start filling this gap by designing a linear-time algorithm for PATH COVER on trees. Let t be the treewidth of a given graph. We then show that PATH COVER can be solved in polynomial time on graphs of bounded treewidth, in XP time $n^{t^{O(t)}}$, using a dynamic programming scheme. Our algorithm gives an FPT $2^{O(t \log t)} n$ algorithm for PATH PARTITION as a corollary. These results also apply to the variants where the paths are required to be induced (i.e. chordless) and/or edge-disjoint.

Keywords: Path Cover · Trees · Treewidth

1 Introduction

Path problems in graphs are fundamental problems in algorithmic graph theory, consider for example the problem of computing shortest paths in a graph, which has been one of the first studied graph problems from which efficient algorithms were obtained [6, Chapter 24]. Finding *disjoint paths* is also a problem of utmost importance, both in algorithmic and structural graph theory [25]. When it comes to *covering* the graph, the HAMILTONIAN PATH problem is another classic path-type problem studied both in combinatorics and computer science. We study its

F. Foucaud—Funded by French government IDEX-ISITE initiative 16-IDEX-0001 (CAP 20–25), International Research Center “Innovation Transportation and Production Systems” of the I-SITE CAP 20–25, and ANR project GRALMEO (ANR-21-CE48-0004)

A. Majumder—Supported by DFG Grant 439637648 (Sachbeihilfe)

T. Mömke—Partially supported by DFG Grant 439522729 (Heisenberg-Grant) and DFG Grant 439637648 (Sachbeihilfe).

generalizations, PATH COVER and PATH PARTITION, which are about covering the vertices of a graph using a minimum number of paths (unrestricted and pairwise vertex-disjoint, respectively). They are formally defined as follows.

PATH COVER

Input: A graph G .

Problem: Compute a minimum-size *path cover*, that is, a set of paths of G such that every vertex of G belongs to at least one of the paths.

Its variant that requires the solution paths to be pairwise vertex-disjoint is very well-studied and defined as follows.

PATH PARTITION

Input: A graph G .

Problem: Compute a minimum-size *path partition*, that is, a set of pairwise vertex-disjoint paths of G such that every vertex of G belongs to at least one of the paths.

Our goal is to study the algorithmic complexity of PATH COVER and PATH PARTITION on trees and graphs of bounded treewidth. Treewidth is an important graph parameter and the associated tree-decompositions enable to solve various problems efficiently when this parameter is bounded. We refer to the book [8] for more on the topic of algorithms for graphs of bounded treewidth.

Related Work. As both problems generalize HAMILTONIAN PATH (which amounts to decide whether a graph can be covered by a single path), they are both NP-hard, and this holds even, for example, for 2-connected cubic bipartite planar graphs [1].

Unlike us, most work in the literature refers to PATH PARTITION as “PATH COVER”; indeed, the former is much more studied than the latter: see [7, 15] for such works on PATH PARTITION. In some cases, PATH PARTITION is also called HAMILTONIAN COMPLETION [12, 13, 18]. To avoid any confusion between the two problems, we use the terminology of PATH COVER and PATH PARTITION as defined above, a choice taken from the survey [21] on path-type problems.

Several papers from the 1970s studied PATH PARTITION on trees [4, 13, 18]. However, they did not explicitly analyze the running times. A properly analyzed linear-time algorithm was given in 2002 [12]. PATH PARTITION was also shown to be solvable in polynomial time on many other graph classes such as cographs [20], distance-hereditary graphs [15], cocomparability graphs [7] (which contain all interval graphs), or block graphs/cactii [23]. We do not know of any explicit algorithm for PATH PARTITION on graphs of bounded treewidth, but algorithms exist for the closely related CYCLE PARTITION problem [9].

Both PATH COVER and PATH PARTITION have numerous applications, in particular in program testing [22], circuit testing [2], or machine translation [19], to name a few. Although PATH PARTITION is more widely studied, PATH COVER is also a natural problem, with specific applications in bio-informatics when

restricted to directed acyclic graphs [5, 24]. We refer to [5, 10, 22, 24] for the few references about PATH COVER that we are aware of.

Our Results. Although PATH PARTITION is well-studied on trees and other graph classes, surprisingly, this is not the case of PATH COVER. Note that despite the two problems having similar statements, they typically have very different optimal solutions. For example, on a star with k leaves, an optimal path partition has size $k - 1$, but an optimal path cover has size $\lceil k/2 \rceil$.

We first focus (in Sect. 2) on PATH COVER on trees, for which no efficient algorithm has been given in the literature. For trees, the size of an optimal solution is given by the ceiling of half of the number of leaves. The proof of this fact was given by Harary and Schwenk in 1972 [14] for the problem of covering the *edges* of the tree. An analysis of their proof yields a quadratic-time algorithm. We show how PATH COVER can in fact be solved in linear time on trees, by giving an improved algorithm based on depth-first-search (DFS).

We then study graphs of bounded treewidth in Sect. 3. We design an explicit dynamic programming algorithm that runs in time $n^{t^{O(t)}}$ for graphs of treewidth at most t and order n . With a slight simplification, the same algorithm also solves PATH PARTITION and runs in improved FPT running time $2^{O(t \log t)} n$.

It is not clear whether PATH COVER can be solved in FPT time as well or not; however, we give some indications of why that might not be the case.

Moreover, we argue that our algorithms also apply to the versions of both PATH COVER and PATH PARTITION where the paths in the solution are required to be *induced* (i.e. chordless) or pairwise edge-disjoint. These variants have been studied in the literature (see [11, 21] and references therein).

We finally conclude in Sect. 4.

2 PATH COVER on Trees

We first study PATH COVER on trees. In [14, Theorem 7], it is proved that the minimum number of paths needed to cover the *edges* of a tree is equal to $\lceil \ell/2 \rceil$, where ℓ is the number of leaves of a tree. This is an obvious lower bound, since for any leaf of a tree, there must be a solution path starting at that leaf. This also holds for covering the *vertices*. The argument of [14], based on pairing the leaves arbitrarily and switching the pairing to increase the number of covered vertices at each step, leads to a quadratic-time algorithm for PATH COVER on trees. We next present an algorithm for solving PATH COVER of a tree with a runtime that is linear in the number of vertices.

Theorem 1. PATH COVER can be solved in linear time on trees, and the optimal size of a solution for a tree with ℓ leaves is $\lceil \ell/2 \rceil$.

Consider the input tree T to be rooted at an arbitrary internal vertex r of T . The intuition here is to cover the vertices of the tree by simulating the Depth-First-Search (DFS) algorithm with some modifications in the steps, thus the running time would become the same as the running time of DFS.

First, we recall the recursive DFS algorithm, see e.g. [6, Chapter 22.3]. In this algorithm, there are two timestamps assigned to each vertex: the first timestamp is given when we discover the vertex for the first time while traversing the tree, and the second timestamp is given when we finish traversing all the vertices of the sub-tree rooted at that particular vertex. We use these timestamps in our algorithm. We use the first timestamp to mark the vertex v as visited and the second timestamp to consider a valid solution of PATH COVER to cover the vertices rooted at v , including v . The algorithm starts at the root and marks the vertices as visited in DFS order until it reaches a leaf. At the leaf on the way back, it starts a path with endpoints at the leaf and continues toward the parent, and the paths are recorded. Now we need some definitions to specify the steps of our algorithm.

We use $p = (x_1, x_2, \dots, x_k)$ as a notation for paths where x_i are all distinct vertices of T connected by the path p . The vertices x_2, \dots, x_{k-1} are internal vertices; the vertices x_1 and x_k are endpoints of the path p .

Definition 1. A path is closed if both of its endpoints are leaves, and a path is open if one of its endpoints is not a leaf. We assume a singleton path at a leaf of the tree to be an open path.

Definition 2. Let v be a vertex in the tree T and \mathcal{P} be a set of paths in the subtree rooted at v . For instance $p_1 = (a_1, \dots, a_j)$ and $p_2 = (b_1, \dots, b_k)$ are two paths in $\mathcal{P} = \{p_1, p_2, \dots\}$ where a_1 and b_1 are leaves of T . We define the following operations:

- We use the notation $\mathcal{P} \cdot v$ to concatenate the paths in \mathcal{P} with vertex v , that is, For each path $p \in \mathcal{P}$ if there exists an edge between v and the endpoint of $p \in \mathcal{P}$, then we add vertex v to the path p . As an example $\mathcal{P} \cdot v = \{(a_1, \dots, a_j, v), (b_1, \dots, b_k, v), \dots\}$. If $\mathcal{P} = \{p\}$ for a single path p , we write $p \cdot v$ as shorthand for $\mathcal{P} \cdot v$.
- Two vertex-disjoint open paths $p_1 = (a_1, \dots, a_j)$ and $p_2 = (b_1, \dots, b_k)$ are combined at the vertex v if v is connected to one of the endpoints of each path. By combining p_1 and p_2 at vertex v , we obtain a new path, formally, $\text{comb}(p_1, p_2) = (a_1, \dots, a_j, v, b_k, \dots, b_1)$.

Let T_v be the subtree of T rooted at a given vertex v . Let $\{T_v^1, T_v^2, \dots\}$ be the set of connected components of $T_v \setminus \{v\}$. Two open paths coming from these connected components to v can only be combined at v if they are coming from two different components from $\{T_v^1, T_v^2, \dots\}$. Since, for the combining operation, the paths need to be vertex-disjoint, the open paths coming from the same component to v can not be combined at v . Two open paths are called unrelated if they come from two different components from $\{T_v^1, T_v^2, \dots\}$ at v .

In the algorithm, for each vertex $v \in T$, we define three sets of paths:

- $\mathcal{P}_v^{\text{close}}$ is a set of paths that consists of closed paths combined at vertex v ,
- $\mathcal{P}_v^{\text{open}}$ is a set of paths that consists of open paths that will be extended further from vertex v ,

Algorithm 1. $PC(T, v, S)$

```

1: Mark  $v$  as visited
2: for Each child  $u$  of  $v$  do
3:   if  $u$  is unvisited then
4:     recursively call  $PC(T, u, S)$ 
5:   end if
6: end for
7: if  $v$  is a leaf then
8:    $\mathcal{P}_v^{open} \leftarrow \{(v, v)\}$       and       $\mathcal{P}_v^{close} \leftarrow \emptyset$ 
9: else
10:   $\mathcal{P}_v \leftarrow \bigcup_c \mathcal{P}_c^{open}$      $\triangleright$  union over all the paths coming from all children  $c$  of  $v$ 
11:   $\mathcal{P}_v^{close} \leftarrow \emptyset$ 
12:  while  $|\mathcal{P}_v| > 2$  do
13:    Find two unrelated paths  $p_i$  and  $p_j$ 
14:     $\mathcal{P}_v^{close} \leftarrow \mathcal{P}_v^{close} \cup \text{comb}(p_i, p_j)$ 
15:     $\mathcal{P}_v \leftarrow \mathcal{P}_v \setminus \{p_i, p_j\}$ 
16:  end while
17:  if  $v$  is the root of  $T$  and  $|\mathcal{P}_v| = 2$  then
18:     $\mathcal{P}_v^{close} \leftarrow \mathcal{P}_v^{close} \cup \text{comb}(p_1, p_2)$ 
19:  else if  $v$  is the root of  $T$  and  $|\mathcal{P}_v| = 1$  then
20:     $\mathcal{P}_v^{close} \leftarrow \mathcal{P}_v^{close} \cup p_1 \cdot v$ 
21:  else
22:     $\mathcal{P}_v^{open} \leftarrow \mathcal{P}_v \cdot v$            $\triangleright$  when  $v$  is not the root of  $T$  and  $|\mathcal{P}_v| \leq 2$ 
23:  end if
24: end if
25: Mark  $v$  as covered
26:  $S \leftarrow S \cup \mathcal{P}_v^{close}$ 
27: return  $\mathcal{P}_v^{open}$ 

```

– \mathcal{P}_v is the set of all paths extended as open paths from the children of v .

Among these sets, we need to store the set \mathcal{P}_v^{open} corresponding to each node for further usage in the parent node of v .

The aim of our algorithm is to have all paths closed, and when a path is open it means we extend the path until it gets combined and closed. Note that we mark a vertex covered when all the vertices of the subtree rooted at that vertex are visited and covered. Now, we present Algorithm 1 to compute a path cover of T . The input to our algorithm is a tree T , a designated root vertex r , and a solution set S , which is initially \emptyset . After the execution of Algorithm 1, the solution set S provides an optimal path cover of T . We store a path $p = (a_1, a_2, \dots, a_j)$ in terms of its endpoints i.e. $p = (a_1, a_j)$.

Proof (Proof of Theorem 1). First, we have to prove that the size of S is $\lceil \frac{\ell}{2} \rceil$ where the number of leaves of T is ℓ . We start a path in the leaf of T and keep it as an open path as long as it is not combined with another path and closed. When a path is closed, both of its endpoints become leaves of T . Additionally, in the root, all the paths get closed except for possibly one path. Hence, for an

even number of leaves, the size of S is $\frac{\ell}{2}$ and for an odd number of leaves, the size of S is $\frac{\ell-1}{2} + 1$, which proves the first part of the theorem.

Now, we show that the running time of Algorithm 1 is $O(n)$ where n is the number of vertices of T . Algorithm 1 closely resembles the DFS algorithm, with the addition of some constant time operations to store the paths covering the vertices. Two operations, $\text{comb}(p, q)$ and $\mathcal{P}_v \cdot v$ take $O(1)$ time as we store the paths by their endpoints, and after each of these operations, only the endpoints are changed while the new paths are created. Finding a pair of unrelated paths can also be done in $O(1)$ time, as the children of a particular vertex can be ordered from left to right, and the paths coming from the children can also be ordered accordingly. For each path p , we just need to check at most two consecutive paths in this ordering to find a path q which is unrelated to p . Therefore, in each of the iterations of the while loop (line 12–16) we do the operations in $O(1)$ time. All the iterations of the while loop together traverse each of the edges between v and its children $O(1)$ time. These edges are only traversed in v 's recursive call. The recursive $PC(T, v, S)$ call is made only once for each vertex. Therefore, the recursive calls altogether make the running time of the algorithm $O(n+m)$, where m is the number of edges of T (since each edge of T is traversed $O(1)$ times), which is same as the running time of DFS. As m is $O(n)$ for T , the total running time becomes $O(n)$, which proves the second part of the theorem. \square

3 PATH COVER on Graphs with Bounded Treewidth

In this section, we present an algorithm that solves PATH COVER on general graphs in XP time when parameterized by treewidth. The algorithm is a classic dynamic programming scheme over a tree decomposition.

Theorem 2. PATH COVER can be solved in time $n^{t^{O(t)}}$, where n is the number of vertices, and t is the treewidth of the input graph.

See [3, 8, 16] for basic definitions of tree decompositions and treewidth. We distinguish between vertices of G and vertices of the tree decomposition by referring to the latter as *nodes*. Each node v is associated with a *bag* X_v : a subset of vertices of graph G . We use the classic *nice tree decompositions*:

Definition 3 ([16]). A tree decomposition \mathcal{T} of graph G is nice if:

- \mathcal{T} is rooted at node r with $X_r = \emptyset$.
- Each leaf node v of \mathcal{T} has an empty bag $X_v = \emptyset$.
- Each non-leaf node is of one of the types below:
 1. An **Introduce node** v has exactly one child u such that $X_v = X_u \dot{\cup} \{x\}$ for some vertex x in V .
 2. A **Forget node** v has exactly one child u such that $X_v = X_u \setminus \{x\}$ for some vertex x in V .
 3. A **Join node** v has two children u_1, u_2 such that $X_v = X_{u_1} = X_{u_2}$.

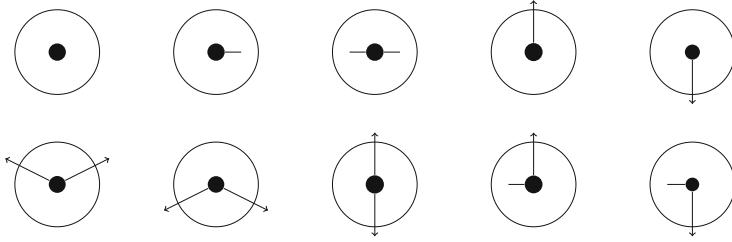


Fig. 1. Illustration of the different types of a vertex with respect to some node v and partial path. The set of types is $\mathcal{N} = \{\emptyset, \{-\}, \{-,-\}, \{\uparrow\}, \{\downarrow\}, \{\uparrow,\uparrow\}, \{\downarrow,\downarrow\}, \{\uparrow,\downarrow\}, \{\downarrow,\uparrow\}, \{\uparrow,-\}, \{\downarrow,-\}\}$. “ $-$ ” shows a vertex with a neighbour inside the bag X_v , “ \uparrow ” shows a vertex with a neighbour in $G - G_v$, and “ \downarrow ” shows a vertex with a neighbour in $G_v - X_v$.

Given a graph G on n vertices and an integer t , there is an algorithm that either outputs a tree decomposition of G of width at most $2t+1$, or determines that the treewidth of G is larger than t , in time $2^{O(t)}n$ [17]. Given a tree decomposition of G of width t and $O(n)$ nodes, a nice tree decomposition of width t with at most $4n$ nodes can be constructed in time $O(t \cdot n)$ [3].

Let $G = (V, E)$ be a graph and (\mathcal{T}, X) be a nice tree decomposition of G with width at most t . Let \mathcal{T}_v be the subtree of \mathcal{T} rooted at node v and V_v be the union of all bags in this subtree including X_v . We define G_v as the subgraph of G induced by V_v . We will use the following lemma from [8].

Lemma 1. *Let (\mathcal{T}, X) be a tree decomposition of a graph G and let uv be an edge of \mathcal{T} . The forest $\mathcal{T} - uv$ obtained from \mathcal{T} by deleting edge uv consists of two connected components \mathcal{T}_u and \mathcal{T}_v . Then (V_u, V_v) is a separation of G with separator $X_u \cap X_v$.*

We implement our dynamic programming scheme for PATH COVER in a bottom-up manner, starting at the leaf nodes of \mathcal{T} . At each node v , we deal with a potential solution for G that covers the vertices of X_v . We define a path as a sequence of vertices where any vertex in a path p is incident to at most two vertices from p , giving at most two neighbours in the subgraph induced by p . Each neighbour of a vertex in p can either belong to $G_v - X_v$ (and thus have been forgotten in the bag of some descendant node of v), which we represent with “ \downarrow ”, or belong to $G - G_v$ (it will appear in the bag of some ancestor node of v), which we represent by “ \uparrow ”, or belong to X_v , which we represent by “ $-$ ”. Therefore, we characterize each vertex by the types of its neighbour(s) inside a potential solution path. To this end, let \mathcal{N} be the set of multisets of size at most two whose elements are taken from the set $\{-, \uparrow, \downarrow\}$.

At each node v , we define a *partial path* p (representing the intersection of a path of G with the bag X_v) by (Π_p, φ_p) where Π_p is an ordered subset of X_v and $\varphi_p : \Pi_p \rightarrow \mathcal{N}$ which describes, for each vertex of p , the types of its neighbour(s) inside the path of G represented by p . These paths are called “partial” since they consist of several paths, and might have neighbour(s)

in the path that lie outside the node. Each vertex x of a partial path p in X_v has the possibility of being in either of the *types* illustrated in Fig. 1 with respect to the path p' of G represented by p .

We say that a partial path for node v is *consistent* if it corresponds to the intersection with X_v of some path of G .

Now, we extend the notion of a partial path p of a node v to that of a partial path p of a subgraph G_v in a natural way as follows. A partial path p of G_v is an ordered subset Π_p of V_v and a function $\varphi_p : \Pi_p \rightarrow \mathcal{N}$ which describes, for each vertex of a path, the types of its neighbour(s) inside the path, where “ \uparrow ” now refers to a neighbour in $V(G) \setminus V_v$, “ $-$ ” refers to a neighbour in V_v , and there is no vertex of p whose type contains “ \downarrow ”.

We say that a partial path p of v *agrees* with a partial path p' of G_v if the following conditions hold:

- $\Pi_p \subseteq \Pi_{p'}$;
- the order of the vertices in $\Pi_p \cap \Pi_{p'}$ is the same in Π_p and $\Pi_{p'}$;
- all vertices in $\Pi_{p'} \setminus \Pi_p$, have a type in $\{-\}, \{-,-\}, \emptyset$;
- $\varphi_p(x) = \varphi_{p'}(x)$, for each vertex x with types $\emptyset, \{-\}, \{-,-\}, \{\uparrow\}$, and $\{\uparrow, -\}$;
- If $\varphi_p(x) = \{\downarrow\}$ then $\varphi_{p'}(x) = \{-\}$;
- If $\varphi_p(x) = \{\uparrow, \uparrow\}$ then $\varphi_{p'}(x) = \{\uparrow, \uparrow\}$;
- If $\varphi_p(x) = \{\downarrow, \downarrow\}$ then $\varphi_{p'}(x) = \{-, -\}$;
- If $\varphi_p(x) = \{\uparrow, \downarrow\}$ then $\varphi_{p'}(x) = \{\uparrow, -\}$;
- If $\varphi_p(x) = \{\downarrow, -\}$ then $\varphi_{p'}(x) = \{-, -\}$.

For a node v , we define a *partial solution* S of G_v as a collection P of partial paths of G_v whose vertices cover all the vertices in V_v . We require that every partial path is consistent, that is, it may correspond to the intersection of a path of G with G_v . We also require that any partial path that has no vertex whose type contains “ \uparrow ”, is unique (i.e. appears only once in S). However, partial paths with vertices whose type contains “ \uparrow ” may appear multiple times (at most n times). Indeed, they might correspond to different paths in a path cover of G , that all happen to have the same intersection with G_v . For notational convenience, a partial solution S is stored as a set P of partial paths, and a function f where, for every partial path p in S , we let $f(p) \in \{1, \dots, n\}$ be the number of times the partial path p appears in the partial solution S .

Let P be a set of partial paths of a node v together with a function $f : P \rightarrow \{1, \dots, n\}$, and $S = (P', f')$ be a partial solution of G_v . Let M be the multiset obtained from P by adding, for every partial path p of P , $f(p)$ copies of p to M . Similarly, let M' be the multiset obtained from P' obtained from the subset of partial paths of P' that intersect X_v , by adding to M' , $f'(p)$ copies of each such path p of P' . We say that P *corresponds* to S if there is a bijection ψ from the M to M' , such that p agrees with $\psi(p)$ for every partial path p of P .

Dynamic Programming. We define a DP-state $[v, P, f]$ as follows:

- v is a node in a nice tree decomposition (\mathcal{T}, X) ;
- P is a set of partial paths that covers the vertices of X_v ;

- f is a function $f : P \rightarrow \{1, \dots, n\}$ that maps each partial path p in P to a value that shows how many times p is used.

We say that a DP-state $[v, P, f]$ is *valid* if every partial path in P is consistent, and (P, f) corresponds to a partial solution S of G_v . For a valid DP-state, we define $\text{opt}[v, P, f]$ as the minimum number of partial paths in a partial solution of G_v corresponding to (P, f) . If there exists no such partial solution, then the DP-state is invalid, and $\text{opt}[v, P, f] = \infty$.

Now, we explain how to compute the value of a DP-state from the values of the children's DP-states. We need to check the compatibility condition between DP-states of a node and the children's DP-states; this is specific for each node type. Therefore, we consider each case individually.

Let v be an introduce node with a child node u . Two valid DP-states $[v, P, f]$ and $[u, P', f']$ are *compatible* if there exist partial solutions S_v of G_v and S_u of G_u corresponding to (P, f) and (P', f') respectively, such that the intersection of S_v with G_u is S_u . Let v be a forget node with a child u . Two valid DP-states $[v, P, f]$ and $[u, P', f']$ are *compatible* if there exist partial solutions S_v of G_v and S_u of G_u corresponding to (P, f) and (P', f') respectively, such that the intersection of S_u with G_v is S_v .

Let v be a join node with two children u_1 and u_2 . Three valid DP-states $[v, P, f]$, $[u_1, P', f']$, and $[u_2, P'', f'']$ are *compatible* if there exist partial solutions S_v, S_{u_1}, S_{u_2} of G_v, G_{u_1} and G_{u_2} corresponding to (P, f) , (P', f') and (P'', f'') respectively, such that the intersection of S_v with G_{u_1} is S_{u_1} and the intersection of S_v with G_{u_2} is S_{u_2} .

The algorithm starts at the leaf nodes and approaches the root. At each node, the algorithm computes a value for each possible DP-state using the values of the children's compatible DP-states, and chooses the minimum possibility. We next describe the computations done at each node, depending on their type. (We omit the case of forget nodes and join nodes due to space constraints.)

Leaf Node. For each leaf node v , $X_v = \emptyset$, so it is trivial that $\text{opt}[v, \emptyset, f] = 0$.

Introduce Node. Let v be an introduce node with child node u such that $X_v = X_u \dot{\cup} \{x\}$ for some $x \in V(G)$. Let $[v, P, f]$ be a valid DP-state for node v .

Note that if there is a partial path $p \in P$ and x with one of the values $\varphi_p(x) = \{\downarrow\}$, $\varphi_p(x) = \{\uparrow, \downarrow\}$, $\varphi_p(x) = \{-, \downarrow\}$, and $\varphi_p(x) = \{\downarrow, \downarrow\}$, then the DP-state is not valid and $\text{opt}[v, P, f] = \infty$. Indeed, by Lemma 1, there is no edge between x and a vertex in $V_v \setminus X_v$.

Otherwise, we will look for all DP-states $[u, P', f']$ for u compatible with $[v, P, f]$. To check whether $[u, P', f']$ and $[v, P, f]$ are compatible, we only need to check the type of x in every partial path p (and the type of its neighbour(s) in p), since the remaining state in X_u must be the same as X_v .

Let $P_{\bar{x}}$ be the set of partial paths of P that contains at least one vertex other than x . We must find a bijection between the partial paths of $P_{\bar{x}}$ and those of P' (taking into account their multiplicities), in the following way. For every partial path p of P not containing x at all, we require that p also belongs to P' , with

$f(p) = f'(p)$. Moreover, for every partial path p of P containing x and at least one other vertex of X_v , there must be a partial path p' of P' such that p' agrees with p after removing x .

Let C be the collection of all DP-states $[u, P', f']$ that are compatible with DP-state $[v, P, f]$ and let k be the number of partial paths of P (accounting for their multiplicity via function f) containing only x . We have:

$$opt[v, P, f] = \min_{[u, P', f'] \in C} \{opt[u, P', f']\} + k$$

To justify the validity of this formula, note that (by induction hypothesis) our process constructs only valid partial solutions. Indeed, consider a partial solution S' corresponding to (P', f') of size $opt[u, P', f']$, where $[u, P', f'] \in C$ is such that $opt[u, P', f']$ is minimum. We obtain a partial solution S of size $|S'| + k$ corresponding to (P, f) by extending S' . To do so, include vertex x into the partial paths of P' that correspond to partial paths of P containing x , as explained in the compatibility check. Moreover, add as many singleton partial paths x as needed. This shows that $opt[v, P, f] \leq |S| \leq |S'| + k = \min_{[u, P', f'] \in C} \{opt[u, P', f']\} + k$.

Conversely, note that an optimal partial solution S of size $opt[v, P, f]$ corresponding to (P, f) can be transformed into another one, S' , by deleting x from it, with $|S'| = |S| - k$ and S' corresponds to some (P', f') where $[u, P', f']$ is compatible with $[v, P, f]$. This shows that $\min_{[u, P', f'] \in C} \{opt[u, P', f']\} \leq |S'| \leq opt[v, P, f] - k$ and thus, $opt[v, P, f] \geq \min_{[u, P', f'] \in C} \{opt[u, P', f']\} + k$.

Proof (Proof of Theorem 2). The algorithm goes through the tree in a bottom-up fashion and computes, for each possible DP-state of the current node, the optimal value for this state using the children's optimal values. The correctness follows from the above discussion and the correctness of the inductive formulas. The optimal value of a solution is found at the root. To obtain the actual path cover, one may use a standard backtracking procedure to build it inductively.

The running time is dominated by the generation, at each node, of all possible DP-states. By the preliminary discussions, we have $|X_v| \leq 2t + 2$ for each node v of the tree decomposition, if G is of treewidth t . Let us count the number of possible DP-states $[v, P, f]$ for a node v . P is a collection of partial paths that covers the vertices of X_v . Each partial path in P is an ordered subset of X_v , where each vertex has 10 possible types inside the partial path. Therefore, there are at most $(10t)! = t^{O(t)}$ possible partial paths. To each of them, we associate a number between 1 and n using the function f . There are at most $n^{(10t)!}$ possible functions f , thus, the total number of partial solutions inside each node is at most $n^{t^{O(t)}}$, and we can compute them in this time. We can also check the validity, compatibility, etc. of the DP-states in time that is polynomial in the size of a DP-state. In a forget/introduce node, we process all pairs of DP-states coming from the parent and child node, which takes k^2 time (if there are at most k possible DP-states per node), and in a join node, we go through all triples of DP-states coming from the parent and the two children nodes, which

takes k^3 time. In each case, we have $k = n^{t^{O(t)}}$ and thus this is still $n^{t^{O(t)}}$. We have at most $4n$ nodes in the tree decomposition; hence, the algorithm solves PATH COVER in time $n^{t^{O(t)}}$. \square

The Case of PATH PARTITION. In the PATH PARTITION version of the problem, we are dealing with vertex-disjoint paths. Therefore, the number of paths that pass through each bag of the tree decomposition is at most the size of the bag. Hence, a DP-state simply needs a collection of partial paths that forms a partition of the bag into ordered sets, and thus, there are at most $t^{O(t)} = 2^{O(t \cdot \log t)}$ possible DP-states for every node, since there are at most $(t+1)! = t^{O(t)}$ orderings and $t^{O(t)}$ possible partitions of a bag. Thus, we obtain an FPT algorithm for PATH PARTITION in time $2^{O(t \cdot \log t)}n$.

The Case of Induced Paths and Edge-Disjoint Paths. The case where the solution paths are required to be induced can be handled easily, indeed, a solution path is induced if and only if its intersection with every bag is induced as well. Thus, it suffices to only consider the DP-states where the partial paths have no unwanted chord inside the bag. Otherwise, the algorithm remains the same.

For the edge-disjoint variants, it suffices to check that the partial paths in every considered DP-state are edge-disjoint; otherwise the algorithm is the same.

4 Conclusion

We have re-initiated the study of PATH COVER, which surprisingly is not extensively studied. We settled its complexity for trees by giving a linear-time algorithm on this class, and we gave an explicit $n^{t^{O(t)}}$ XP-time dynamic programming scheme for graphs of treewidth t . The same algorithm modified for PATH PARTITION gives a $2^{O(t \cdot \log t)}n$ FPT running time. These running times also hold for the variants of PATH COVER and PATH PARTITION where the solution paths are required to be induced and/or edge-disjoint.

It would be nice to improve the running times of our algorithms. Probably, one can design a $2^{O(t)}n$ algorithm for PATH PARTITION using the ideas of [9], where they solved CYCLE PARTITION in this running time. But can one get an $n^{O(t)}$ algorithm for PATH COVER? Can PATH COVER even be solved in FPT time for parameter treewidth? It is possible that this is not the case, as the number of solution paths going through one bag of the tree-decomposition can be arbitrarily large. In case of a negative answer, this would show a striking contrast between the algorithmic complexities of PATH COVER and PATH PARTITION.

References

1. Akiyama, T., Nishizeki, T., Saito, N.: NP-completeness of the Hamiltonian cycle problem for bipartite graphs. *J. Inform. Process.* **3**(2), 73–76 (1980/81)
2. Andreatta, G., Mason, F.: Path covering problems and testing of printed circuits. *Discret. Appl. Math.* **62**(1–3), 5–13 (1995)
3. Bodlaender, H.L., Bonsma, P., Lokshtanov, D.: The fine details of fast dynamic programming over tree decompositions. In: Gutin, G., Szeider, S. (eds.) IPEC 2013. LNCS, vol. 8246, pp. 41–53. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03898-8_5
4. Boesch, F., Chen, S., McHugh, J.: On covering the points of a graph with point disjoint paths. In: Graphs and Combinatorics, pp. 201–212. Springer (1974)
5. Cáceres, M., Cairo, M., Mumey, B., Rizzi, R., Tomescu, A.I.: Sparsifying, shrinking and splicing for minimum path cover in parameterized linear time. In: Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, pp. 359–376 (2022)
6. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. The MIT Press, 3rd edition (2009)
7. 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)
8. Cygan, M., et al.: Parameterized Algorithms. Springer, Cham (2015)
9. Cygan, M., et al.: Solving connectivity problems parameterized by treewidth in single exponential time. *ACM Trans. Algorithms* **18**(2), 17:1–17:31 (2022)
10. Cáceres, M., et al.: Safety in multi-assembly via paths appearing in all path covers of a dag. *IEEE/ACM Trans. Comput. Biol. Bioinf.* **19**(6), 3673–3684 (2022)
11. Fernau, H., Foucaud, F., Mann, K., Padariya, U., Rao, K.N.R.: Parameterizing path partitions. In: Mavronicolas, M. (ed.) CIAC 2023. LNCS, vol. 13898, pp. 187–201. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-30448-4_14
12. Franzblau, D.S., Raychaudhuri, A.: Optimal hamiltonian completions and path covers for trees, and a reduction to maximum flow. *ANZIAM J.* **44**(2), 193–204 (2002)
13. Goodman, S., Hedetniemi, S.: On the hamiltonian completion problem. In: Bari, R.A., Harary, F. (eds.) Graphs and Combinatorics, pp. 262–272. Springer, Berlin Heidelberg (1974)
14. Harary, F., Schwenk, A.: Evolution of the path number of a graph: covering and packing in graphs, II. *Graph Theory and Computing*, pp. 39–45 (1972)
15. Hung, R., Chang, M.: Finding a minimum path cover of a distance-hereditary graph in polynomial time. *Discret. Appl. Math.* **155**(17), 2242–2256 (2007)
16. Kloks, T.: Treewidth: computations and approximations. Springer (1994)
17. Korhonen, T.: A single-exponential time 2-approximation algorithm for treewidth. In: 62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, pp. 184–192 (2021)
18. Kundu, S.: A linear algorithm for the hamiltonian completion number of a tree. *Inf. Process. Lett.* **5**(2), 55–57 (1976)
19. Lin, G., Cai, Z., Lin, D.: Vertex covering by paths on trees with its applications in machine translation. *Inf. Process. Lett.* **97**(2), 73–81 (2006)
20. Lin, R., Olariu, S., Pruesse, G.: An optimal path cover algorithm for cographs. *Comput. Math. Appl.* **30**(8), 75–83 (1995)

21. Manuel, P.: Revisiting path-type covering and partitioning problems. arXiv preprint [arXiv:1807.10613](https://arxiv.org/abs/1807.10613) (2018)
22. Ntafos, S., Hakimi, S.: On path cover problems in digraphs and applications to program testing. IEEE Trans. Softw. Eng. **SE-5**(5), 520–529 (1979)
23. Pan, J., Chang, G.J.: Path partition for graphs with special blocks. Discret. Appl. Math. **145**(3), 429–436 (2005)
24. Rizzi, R., Tomescu, A.I., Mäkinen, V.: On the complexity of minimum path cover with subpath constraints for multi-assembly. BMC Bioinform. **15**(S-9), S5 (2014)
25. Robertson, N., Seymour, P.D.: Graph minors. XIII. The disjoint paths problem. J. Combinatorial Theor. Ser. B **63**(1), 65–110 (1995)



Fast FPT Algorithms for Grundy Number on Dense Graphs

Sina Ghasemi Nezhad¹ , Maryam Moghaddas¹ , and Fahad Panolan²

¹ Department of Mathematical Sciences, Sharif University of Technology, Tehran, Iran

{sina.ghaseminejad,maryam.moghaddas}@sharif.edu

² School of Computer Science, University of Leeds, Leeds, UK

f.panolan@leeds.ac.uk

Abstract. In this paper, we investigate the GRUNDY COLORING problem for graphs with a cluster modulator, a structure commonly found in dense graphs. The Grundy chromatic number, representing the maximum number of colors needed for the first-fit coloring of a graph in the worst-case vertex ordering, is known to be $W[1]$ -hard when parameterized by the number of colors required by the most adversarial ordering. We focus on fixed-parameter tractable (FPT) algorithms for solving this problem on graph classes characterized by dense substructures, specifically those with a cluster modulator. A cluster modulator is a vertex subset whose removal results in a cluster graph (a disjoint union of cliques). We present FPT algorithms for graphs where the cluster graph consists of one, two, or k cliques, leveraging the cluster modulator's properties to achieve the best-known FPT runtimes, parameterized by both the modulator's size and the number of cliques.

Keywords: Grundy Coloring · Cluster Graphs · Max Flow · Fixed-Parameter Tractable (FPT) Algorithms · Integer Program

1 Introduction

The *first-fit* coloring is a heuristic that assigns to each vertex, arriving in a specified order σ , the smallest available color such that the coloring remains proper. Here, we denote the colors as natural numbers $\{1, 2, 3, \dots\}$. The *Grundy chromatic number* (or simply *Grundy number*) is the number of colors that are needed for the most adversarial vertex ordering σ , i.e., the maximum number of colors that the first-fit coloring requires over all possible vertex orderings. In other words, if the Grundy number of a graph G is ℓ , then ℓ is the largest integer such that the following holds. There is an ordered partition (V_1, \dots, V_ℓ) of the vertex set $V(G)$ of G such that for all $i \in [\ell]$ and $v \in V_i$, $N_G(v) \cap V_j \neq \emptyset$ for all $j \in \{1, \dots, i-1\}$ and V_i is an independent set. Here, we use $N_G(v)$ to denote the set of neighbors of v and $[\ell]$ to denote the set $\{1, 2, \dots, \ell\}$. Also, we say that any ordered partition (U_1, \dots, U_ℓ) of $V(G)$ is a *Grundy coloring* of G if it satisfies

the above-mentioned property. That is, for all $i \in [\ell']$ and $v \in U_i$, $N_G(v) \cap U_j \neq \emptyset$ for all $j \in \{1, \dots, i-1\}$ and U_i is an independent set. The Grundy number was first introduced by Patrick Michael Grundy but was formally defined by Christen and Selkow [4].

In the GRUNDY COLORING problem, we are given a graph G and an integer ℓ , and we want to test whether the Grundy number of G is at least ℓ or not. The GRUNDY COLORING problem has been shown to be NP-hard even on special graph classes like bipartite graphs, their complements, chordal graphs, and line graphs, as established by known results in the literature [7, 8, 10, 12]. While this makes solving the problem in general infeasible in polynomial time, efficient algorithms have been designed for special graph classes. The GRUNDY COLORING for a tree can be solved in linear time [9], and this result has been generalized with a polynomial time algorithm for the bounded-treewidth graphs [11].

GRUNDY COLORING is well studied in the realm of parameterized complexity. Aboulker et al. [1] proved that GRUNDY COLORING is $W[1]$ -hard when parameterized by the number of colors required by the most adversarial vertex ordering, leveraging the structure of half-graphs. Furthermore, their work also establishes that the brute-force algorithm, with a running time of $f(\ell)n^{2^{\ell-1}}$, is essentially optimal under the Exponential Time Hypothesis (ETH). In the context of parameterized complexity, the problem has been studied with various parameters. For instance, Bonnet et al. [3] explored the complexity of the GRUNDY COLORING problem, identifying a range of parameterizations for which the problem remains computationally challenging. Similarly, Belmonte et al. [2] considered the parameters treewidth and pathwidth, and proved the following surprising result. GRUNDY COLORING is $W[1]$ -hard when parameterized by treewidth, but FPT when parameterized by pathwidth. Their work also demonstrates that GRUNDY COLORING is FPT when parameterized by the neighborhood diversity (and more generally modular width). The running time of their algorithm is $2^{O(w^2w)}n^{O(1)}$, where w is the modular width or neighborhood diversity of the input graph.

We study GRUNDY COLORING problem on dense graphs from the perspective of parameterized complexity. We consider the problem when parameterized by k -cluster modulator. That is, the input consists of a graph G and a vertex subset $R \subseteq V(G)$ of size r such that $G - R$ is a k -cluster graph (i.e., $G - R$ has k connected components and each of them is a complete graph). Here, the parameter is $k + r$. Observe that in this case, the neighborhood diversity of G is at most $k2^r + r$. Thus, the result of Belmonte et al. [2] implies that there is an algorithm with a running time of $2^{O(2^{k2^r})}n^{O(1)}$. Notice that even when $k = 1$, this running time is at least $2^{\Omega(2^r)}$.

Our main results are when $k = 1$ and $k = 2$. When $k = 1$, the parameter is the clique modulator. In this case, we design an algorithm with a running time of $O(2^{O(r^2)} + n + m)$, where m is the number of edges in the graph, and this result can be found in Sect. 3.

As a byproduct of our proof, we also get that the problem admits a kernel of size $O(r2^r)$. When $k = 2$, we design an algorithm with a running time of

$O(2^{O(r^2)}n^6)$. To prove this result we model the problem as many instances of the classic max flow problem. This result can be found in Sect. 4.

For general k we give faster algorithm with a running time of $2^{O(2^{kr})}n^{O(1)}$ in Sect. 5. To prove this result we first observe some structural results on Grundy coloring and then model the problem as an Integer Linear Program with few variables, like the method used by Belmonte et al. [2].

2 Preliminaries

For a natural number $q \in \{1, 2, \dots\}$, we use $[q]$ to denote the set $\{1, 2, \dots, q\}$. Let $a = (a_1, \dots, a_n)$ and $b = (b_1, \dots, b_m)$ be two sequences. The *concatenation* of a and b , denoted ab , is the sequence formed by appending all elements of b to the end of a , resulting in $(a_1, \dots, a_n, b_1, \dots, b_m)$. A sequence $c = (c_1, \dots, c_s)$ is called a *subsequence* of a sequence $d = (d_1, \dots, d_t)$ if there exists a strictly increasing sequence of indices $1 \leq i_1 < \dots < i_s \leq t$ such that $c_j = d_{i_j}$ for all $j \in \{1, \dots, s\}$.

We use standard graph notations in this paper: for a graph G , the set of vertices is denoted by $V(G)$ (with a size of $|V(G)| = n$), and the set of edges is denoted by $E(G)$ (with a size of $|E(G)| = m$). For a vertex $v \in V(G)$, we use $N_G(v)$ to denote the open neighborhood set of v in G . That is, $N_G(v) = \{u \in V(G) : \{u, v\} \in E(G)\}$. We also use $N_G[v]$ for the closed neighborhood set of v in G , i.e., $N_G(v) \cup \{v\}$. For a vertex subset $X \subseteq V$, $G[X]$ represents the induced subgraph of G over X . That is, $V(G[X]) = X$ and $E(G[X]) = \{\{u, v\} \in E(G) : u, v \in X\}$. For $X \subseteq V(G)$, $G - X$ denotes the induced subgraph $G[V(G) \setminus X]$.

A vertex subset $I \subseteq V(G)$ is called an *independent set* if for any pair of vertices $u, v \in I$, $\{u, v\} \notin E(G)$. A vertex subset $K \subseteq V(G)$ is called an *clique* if for any pair of vertices $u, v \in K$, $\{u, v\} \in E(G)$ (i.e., $G[K]$ is a complete graph). A graph G is a *cluster graph* if each connected component of G is a complete graph. We say that G is a k -*cluster graph* if G is a cluster graph and the number of connected components in G is k . For a graph G , a vertex subset $R \subseteq V(G)$ is a *clique modulator* if $V(G) \setminus R$ is a clique in G . For a graph G , a vertex subset $R \subseteq V(G)$ is a *cluster modulator* (respectively, k -*cluster modulator*) if $G - R$ is a cluster graph (respectively, k -cluster graph).

A graph coloring is called *proper* if no two adjacent vertices share the same color; otherwise, it is referred to as *improper*. In the context of proper colorings, we define *color classes* as subsets $C \subseteq V(G)$, where all vertices within C are assigned the same color in the coloring of graph G . We denote the color classes of G as (C_1, \dots, C_γ) , where γ represents the total number of colors used.

Now we prove some simple results about Grundy coloring which we use in the later sections. The proofs of results marked (\star) are omitted due to paucity of space.

Lemma 1 (\star). *Let G be a graph and $u, v \in V(G)$ such that $N_G[u] = N_G[v]$. Let σ be a permutation of $V(G)$ and $\hat{\sigma}$ is the permutation obtained from σ by swapping the vertices u and v in σ . Then, the first-fit colorings of G with respect to σ and $\hat{\sigma}$ use the same number of colors.*

Lemma 2 (★). Consider a permutation σ of the vertices and perform the first-fit coloring to it to reach the color classes (C_1, \dots, C_γ) . For each $1 \leq i \leq \gamma$, let the vertices in C_i be $\{v_{i1}, \dots, v_{in_i}\}$ where n_i is the number of vertices in the i^{th} color class. Then the following permutation is called the sorted permutation of σ and will have the exact same color classes as σ under the first-fit coloring:

$$\sigma_s = v_{11}, v_{12}, \dots, v_{1n_1}, v_{21}, v_{22}, \dots, v_{2n_2}, \dots, v_{\gamma 1}, v_{\gamma 2}, \dots, v_{\gamma n_\gamma}$$

Due to the above lemma, throughout this paper, we will present a color classification instead of a permutation of vertices to represent a Grundy coloring.

Lemma 3. Let G be a graph, and γ be the Grundy number of G . There is a Grundy coloring (C_1, \dots, C_γ) of G with the following property. There is an index $i \in \{1, \dots, \gamma\}$ such that for each $j \leq i$, we have $|C_j| > 1$, and for each $j > i$, we have $|C_j| = 1$.

Proof. The lemma states that we can rearrange the singleton color classes in a Grundy coloring of G such that all singleton color classes appear at the end, resulting in a proper coloring that still uses the maximum number of colors.

Consider a permutation σ such that the first-fit coloring of G with respect to σ uses γ colors. Let (C_1, \dots, C_γ) be the corresponding color classes. If this coloring satisfies the property stated in the lemma, then we are done. Suppose, that this is not the case. Then, there must exist indices x and w such that $w > x$, $|C_x| = 1$, and $|C_w| \geq 2$. Since C_x is a singleton, the vertex $u \in C_x$ must be adjacent to at least one vertex in each of the color classes C_y for all $y < x$. Additionally, the vertices in C_z for all $z > x$ must also be adjacent to u . This implies that u is adjacent to a vertex in each color class except C_x . Therefore, $(C_1, \dots, C_{x-1}, C_{x+1}, \dots, C_\gamma, C_x)$ is a Grundy coloring of G .

By repeating this process for all singleton classes, we obtain a proper color classification in which all singleton classes appear at the end, which is the desired result and completes the proof. \square

3 Grundy Coloring with a Clique Modulator

In this section, we design a faster FPT algorithm for GRUNDY COLORING when parameterized by the size of the given clique modulator. Here, the input is a graph G and a clique modulator R of G of size r . Our main idea is to construct a permutation of vertices in G , which results in a maximum number of colors using first-fit coloring. We say that a Grundy coloring of G is an optimal solution if the number of colors used is the Grundy number of G . We start describing the properties of an optimal solution with the following lemmas.

Lemma 4. Let G be a graph, R be a clique modulator of G , and γ be the Grundy number of G . There is a Grundy coloring (C_1, \dots, C_γ) of G with the following property. There is an index $i \in \{1, \dots, \gamma\}$ such that for each $j \leq i$, $C_j \cap R \neq \emptyset$, and for each $j > i$, $C_j \cap R = \emptyset$ and $|C_j| = 1$.

Proof. The proof follows a similar approach to that of Lemma 3. Note that at most one vertex from the clique can be in any color class. By applying the same method, but focusing only on the color classes C_j of size one that satisfies $C_j \cap R = \emptyset$, instead of considering all color classes C_j of size one, we arrive at the desired result. \square

Lemma 5. *Let G be a graph, R be a clique modulator of size at most r , and γ be the Grundy number of G . There is a vertex subset $Q \subseteq V(G) \setminus R$ of size r with the following property. Let σ_2 be an arbitrary ordering of $V(G) \setminus (R \cup Q)$. Then, there exists an ordering σ_1 of $Q \cup R$ such that the first-fit coloring of G with respect to $\sigma_1\sigma_2$ uses γ number of colors.*

Proof. We begin by applying Lemma 4, which guarantees the existence of a Grundy coloring (C_1, \dots, C_γ) of G such that there exists an index $i \in \{1, \dots, \gamma\}$ with the following properties: For each $j \leq i$, $C_j \cap R \neq \emptyset$, and for each $j > i$, $C_j \cap R = \emptyset$ and $|C_j| = 1$. This implies that all singleton color classes, consisting of vertices from $V(G) \setminus R$, appear at the end of the coloring. Since each color class C_j for $j \leq i$ contains at most one vertex from $V(G) \setminus R$, we define $Q = \bigcup_{j=1}^r (C_j \setminus R)$. By construction, Q contains at most r vertices from $V(G) \setminus R$ within the first r color classes, ensuring that $|Q| \leq r$.

Observe that we have $i \leq r$. All remaining color classes $(C_{r+1}, \dots, C_\gamma)$ are singletons where $C_j \cap R = \emptyset$ for each $j \in \{r+1, \dots, \gamma\}$. Therefore, we can define an arbitrary ordering σ_2 for the vertices in $V(G) \setminus (R \cup Q)$, corresponding to the color classes $(C_{r+1}, \dots, C_\gamma)$. We then construct the ordering σ_1 for the vertices in $Q \cup R$ based on the color classes (C_1, \dots, C_r) , ensuring that it respects the first-fit coloring that resulted in the color classes (C_1, \dots, C_γ) . Thus, by concatenating σ_1 and σ_2 , the first-fit coloring of G with respect to this ordering uses exactly γ colors, completing the proof. \square

Let $S = V(G) \setminus R$. Notice that S is a clique in G . Now we define an equivalence relation \sim_R on S as follows. For any two vertices $u, v \in S$, $u \sim_R v$ if and only if $N_G[u] = N_G[v]$. Let E_1, \dots, E_q be the equivalence classes of \sim_R . It is easy to see that $q \leq 2^r$. Now from each equivalence class E_i arbitrarily select a subset $F_i \subseteq E_i$ of size $\min\{r, |E_i|\}$. Let $F = \bigcup_{i=1}^q F_i$. Constructing F is efficient. For each vertex v , we determine its equivalence class by examining its neighbors in R . The vertex v is included in F only if its equivalence class within F contains fewer than r vertices. This procedure involves iterating over all vertices and the edges between R and S , resulting in a time complexity of $O(n + m)$.

Now we prove the following lemma.

Lemma 6. *The Grundy number of G is equal to the sum of the Grundy number of $G[R \cup F]$ and $|V(G) \setminus (R \cup F)|$.*

Proof. The proof is based on the following observation. If σ_2 is an arbitrary ordering of $V(G) \setminus (R \cup F)$. Then for any ordering σ_1 of $G[R \cup F]$, the number of colors in the first-fit coloring of G with respect to $\sigma_1\sigma_2$ is equal to the sum of the number of colors in the first-fit coloring of G with respect to σ_1 , and $|V(G) \setminus (R \cup F)|$.

According to Lemma 4, there exists a color classification (C_1, \dots, C_γ) using Grundy number of colors such that there is an index $i \in \{1, \dots, \gamma\}$ where, for each $j \leq i$, we have $C_j \cap R \neq \emptyset$, and for each $j > i$, we have $C_j \cap R = \emptyset$ and $|C_j| = 1$. Since $C_j \cap R = \emptyset$ for all $j > i$, and R contains r vertices, we have $i \leq r$.

Consider the first r' color classes, $(C_1, \dots, C_{r'})$, containing exactly r vertices from the clique S . Since there are at least r vertices in S , we have $r' \geq r$. Using Lemma 1, we can swap any vertex from S in these color classes with a vertex from the set F . Given that F includes at least $\min\{r, |E_i|\}$ vertices from each equivalence class of \sim_R , this swap is feasible. After the swap, the vertices from S in the first r' color classes are exactly those from F , and their number is r . Let $Q \subseteq F$ denote these r vertices from F .

The modified color classification $(C_1, \dots, C_{r'})$ now includes vertices from $R \cup Q$ in the first r' color classes, with the remaining color classes being singletons that can be ordered at the end. The vertices in $F \setminus Q$ can be placed as singleton color classes before the singleton color classes from $V(G) \setminus (R \cup F)$. Finally, with $|V(G) \setminus (R \cup F)|$ singleton vertices left, these can be assigned in any order at the end. Thus, the Grundy number of G equals the sum of the Grundy number of $G[R \cup F]$, and $|V(G) \setminus (R \cup F)|$, as the first r' color classes include all non-singleton vertices from $R \cup Q \subseteq R \cup F$, and the remaining vertices are assigned their own colors.

The set Q satisfies the necessary conditions of Lemma 5. Therefore, if σ'_2 is an arbitrary ordering of $V(G) \setminus (R \cup Q)$, then there exists an ordering σ'_1 of $Q \cup R$ such that the first-fit coloring of G with respect to the concatenated order $\sigma'_1 \sigma'_2$ uses exactly γ colors. To leverage this result, we choose σ'_2 as an ordering in which the vertices of $F \setminus Q$ appear first. We now define σ_1 to be the concatenation of σ'_1 with the first $|F \setminus Q|$ vertices from σ'_2 , effectively placing these vertices immediately after the vertices in $R \cup Q$. Finally, we define σ_2 as the ordering of the remaining vertices in σ'_2 . This construction ensures that the orderings σ_1 and σ_2 satisfy the desired conditions, and the proof is complete. \square

The following theorem follows from Lemma 6.

Theorem 1. *The GRUNDY COLORING problem parameterized by the size r of a clique modulator admits a kernel of size $O(r2^r)$.*

Theorem 2. *There is an algorithm that given a graph G and a clique modulator R of size r , runs in time $O(2^{O(r^2)} + n + m)$ and outputs a Grundy coloring of G using the maximum number of colors.*

Proof. Let $F \subseteq V(G)$ be the subset mentioned in Lemma 6. Recall that $|F| \leq r2^r$. Let $G' = G[R \cup F]$. Moreover, by Lemma 6, to prove the theorem it is enough to get an ordering σ_1 of G' such that first-fit coloring with respect to σ_1 uses the Grundy number of G' many colors. To get such an ordering σ_1 , we use Lemma 5. By Lemma 5, there is a vertex subset $Q \subseteq V(G') \setminus R$ of size r with the following property. Let σ'_2 be an arbitrary ordering of $V(G') \setminus (R \cup Q)$. Then, there exists an ordering σ'_1 of $Q \cup R$ such that the first-fit coloring of G' with respect to $\sigma'_1 \sigma'_2$ uses Grundy number of G' many colors.

Since $V(G') \setminus R = F$ and $|F| \leq r2^r$, the number of choices for Q is bounded by $\binom{r2^r}{r}$, which is upper bounded by $r^r 2^{r^2}$. For each Q , the number of choices for σ'_1 is upper bounded by r^{2r} . Thus trying all choices for Q and σ'_1 is upper bounded by $2^{O(r^2)}$. Thus in time $2^{O(r^2)}$, we can compute an ordering σ_1 such that the first fit coloring of G' with respect to σ_1 uses Grundy number of G' many colors. Since the construction of the set F takes $O(n + m)$ time, the total running time is $O(2^{O(r^2)} + n + m)$. \square

4 Grundy Coloring with a 2-Cluster Modulator

Next, we examine the parameter of a 2-cluster modulator. After removing the 2-cluster modulator R from G , the remaining graph $G - R$ forms a 2-cluster graph, meaning that $G - R$ consists of two connected components, K_1 and K_2 , which are complete graphs. Let $S = V(K_1)$ and $S' = V(K_2)$, where S and S' are cliques in G . The key idea for this case is to reduce it to a max-flow problem. Let the size of the modulator be $|R| = r$. We begin with the following simple lemma.

Lemma 7. *Let (C_1, \dots, C_γ) be a Grundy coloring of G . Then, there do not exist two distinct color classes C_i and C_j such that $|C_i| = |C_j| = 1$ and $C_i \subseteq S$ and $C_j \subseteq S'$*

Proof. Assume that there exist color classes C_i and C_j such that $|C_i| = |C_j| = 1$ such that the element $x \in C_i$ is a vertex of S , and the element $y \in C_j$ is a vertex of S' . Without loss of generality, assume that $1 \leq i < j \leq \gamma$. Therefore, since y is colored with a number greater than i , due to the definition of first-fit coloring, it must have an edge to a vertex with color i , which can only be x . But since x and y are from two disjoint cliques with no edges between them, this is not possible. This contradiction completes the proof. \square

We now define an equivalence relation \sim_R on the set $S \cup S'$. Specifically, for any two vertices $u, v \in S \cup S'$, we say $u \sim_R v$ if and only if they have the same closed neighborhood, i.e., $N_G[u] = N_G[v]$. The equivalence classes of \sim_R are denoted E_1, \dots, E_q and $E'_1, \dots, E'_{q'}$, where E_i 's come from the clique S and E'_j 's come from the clique S' . Similar to the case of a clique modulator, we have $q + q' \leq 2^{r+1}$. Now from each equivalence class E_i arbitrarily select a subset $F_i \subseteq E_i$ of size $\min\{r, |E_i|\}$. Also, for each equivalence class E'_j arbitrarily select a subset $F'_j \subseteq E'_j$ of size $\min\{r, |E'_j|\}$. Let $F = (\bigcup_j F_j) \cup (\bigcup_i F'_i)$. Notice that $|F| \leq r2^{r+1}$.

Lemma 8. *There is a vertex subset $Q \subseteq F$ of size at most $2r$ and a (not necessarily optimal) Grundy coloring $(C'_1, \dots, C'_{\gamma'})$ of $G[Q \cup R]$ with the following property. There is an optimal Grundy coloring (C_1, \dots, C_γ) of G such that $(C'_1, \dots, C'_{\gamma'})$ is a subsequence of (C_1, \dots, C_γ) .*

Proof. Consider an optimal Grundy coloring $(\widehat{C}_1, \dots, \widehat{C}_\gamma)$ of G . Let $(\widehat{C}_{\alpha_1}, \dots, \widehat{C}_{\alpha_{r'}})$ be the color classes that contain at least one vertex from R . Clearly, $r' \leq r$, as there are at most r such color classes. Furthermore, note that each of the color classes $\widehat{C}_{\alpha_1}, \dots, \widehat{C}_{\alpha_{r'}}$ can contain at most two vertices from $V(G) \setminus R$, since each class contains at most one vertex from each clique in $G - R$.

By Lemma 1, we can swap the vertices of $(\widehat{C}_{\alpha_1}, \dots, \widehat{C}_{\alpha_{r'}})$ that belong to $V(G) \setminus (R \cup F)$ with vertices from F that are in the same equivalence classes under \sim_R . This is feasible because F contains $\min\{r, |E_i|\}$ and $\min\{r, |E'_j|\}$ vertices from each equivalence class E_i and E'_j , respectively, and each color class contains at most one vertex from the clique S or S' , ensuring that the swap maintains the structure of the coloring.

After performing these swaps, we obtain a new Grundy coloring (C_1, \dots, C_γ) , and $(C_{\alpha_1}, \dots, C_{\alpha_{r'}})$ denote the color classes that contain the vertices from R . Moreover, for each $i \in [r']$, $C_{\alpha_i} \subseteq R \cup F$. Let $Q = (C_{\alpha_1} \cup \dots \cup C_{\alpha_{r'}}) \cap F$. Notice that $|Q| \leq 2r$. Thus, (C_1, \dots, C_γ) is the desired optimal Grundy coloring. \square

Definition 1. A Grundy coloring $(C'_1, \dots, C'_{\gamma'})$ of $G[Q \cup R]$ is called extendable if there is an optimal Grundy coloring (C_1, \dots, C_γ) of G such that $(C'_1, \dots, C'_{\gamma'})$ is a subsequence of (C_1, \dots, C_γ) .

Lemma 9. Let $(C'_1, \dots, C'_{\gamma'})$ be a (not necessarily optimal) Grundy coloring of $G[Q \cup R]$, that is extendable. Let $\beta = \max\{|S \setminus Q|, |S' \setminus Q|\}$. Then, Grundy number of G is $\gamma' + \beta$.

Proof. Since the sequence of color classes $(C'_1, \dots, C'_{\gamma'})$ is extendable, there exists an optimal Grundy coloring (C_1, \dots, C_γ) of G such that $(C'_1, \dots, C'_{\gamma'})$ is a subsequence of (C_1, \dots, C_γ) . Furthermore, note that the color classes which appear in (C_1, \dots, C_γ) but not in $(C'_1, \dots, C'_{\gamma'})$ consist exclusively of vertices from $V(G) \setminus R$.

By Lemma 7, these remaining color classes are either composed of two vertices (one from each clique) or a single vertex (from the clique with the greater number of remaining vertices). Hence, the total number of these additional color classes corresponds to $\beta = \max\{|S \setminus Q|, |S' \setminus Q|\}$, where S and S' are the vertex sets of the cliques in $G - R$. Consequently, the Grundy number of G is given by $\gamma = \gamma' + \beta$, where γ' represents the number of color classes in the subsequence $(C'_1, \dots, C'_{\gamma'})$. \square

Because of Lemma 8, the number of choices for Q is at most $2^{O(r^2)}$. For each such choice the number of Grundy colorings of $G[Q \cup R]$ is $2^{O(r \log r)}$ because $|Q \cup R| \leq 3r$. Thus, we can guess the correct choice for Q and a Grundy coloring $(C'_1, \dots, C'_{\gamma'})$ of $G[Q \cup R]$, that is extendable in time $2^{O(r^2)}$. Thus our next job is to test whether $(C'_1, \dots, C'_{\gamma'})$ is indeed extendable. Towards that, we construct a flow network and prove that the network has a *large* flow if and only if $(C'_1, \dots, C'_{\gamma'})$ is extendable.

Theorem 3. *There is an algorithm that given a graph G and a Grundy coloring $(C'_1, \dots, C'_{\gamma'})$ of $G[Q \cup R]$, runs in time $O(n^6)$ and decides if $(C'_1, \dots, C'_{\gamma'})$ is extendable.*

Proof. The algorithm we propose is based on a flow network. Without loss of generality, assume that $|S \setminus Q| \geq |S' \setminus Q|$. Let the vertices in $S \setminus Q$ be u_1, \dots, u_s , and the vertices in $S' \setminus Q$ be $u'_1, \dots, u'_{s'}$. Our flow network consists of $s+1$ main sections that form a bipartite subgraph, s auxiliary vertices, a source v_{source} that is adjacent to one part, and a sink v_{sink} that is adjacent to the second part.

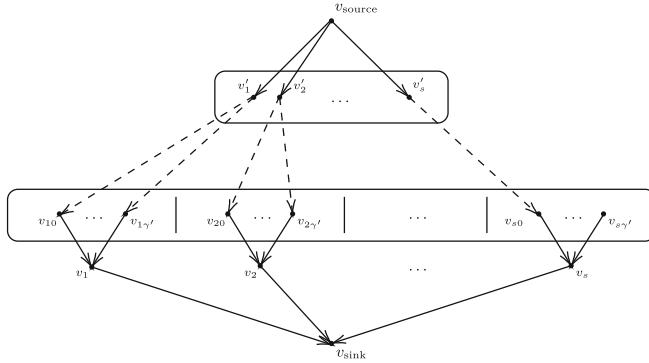


Fig. 1. An overview of the flow network for extendable checking algorithm.

For each vertex u_i of $S \setminus Q$, consider $\gamma' + 1$ vertices $v_{i0}, \dots, v_{i\gamma'}$ that each represents the location of the paired or singleton color class containing u_i between the gaps of color classes $(C'_1, \dots, C'_{\gamma'})$. The vertices $v_{i\lambda}$ for $1 \leq i \leq s$ and $0 \leq \lambda \leq \gamma'$, form a part of our bipartite subgraph. After considering these main vertices, we create an extra auxiliary vertex v_i for each of the vertices u_i of $S \setminus Q$ to handle our constraints. For each vertex u'_j of $S' \setminus Q$, consider a vertex v'_j that represents u'_j in our bipartite subgraph. In addition, let us add $s - s'$ dummy vertices $v'_{s'+1}, \dots, v'_s$ that represent null nodes that will be used to determine singleton color classes. These dummy vertices, along with the vertices v'_j for $1 \leq j \leq s'$, form the second part of our bipartite subgraph. Now it is time to complete the flow network. Figure 1 shows an example of such a flow network.

First, we add $(v_{\text{source}}, v'_j)$ edges for all $1 \leq j \leq s$ to complete the first half of our network. Then, we add $(v_{i\lambda}, v_i)$ edges for all $1 \leq i \leq s$ and $0 \leq \lambda \leq \gamma'$. We also add (v_i, v_{sink}) edges for all $1 \leq i \leq s$ to complete the second half of the network. Now, it remains to add the edges between these two halves. For each $1 \leq i, j \leq s$ and $0 \leq \lambda \leq \gamma'$, we add the $(v'_j, v_{i\lambda})$ edge to the network, if and only if creating a paired color class with u'_j and u_i and placing it immediately after the λ^{th} color class of $(C'_1, \dots, C'_{\gamma'})$ does not violate any constraints. That is, for each $p \in \{1, \dots, \lambda\}$, u'_j and u_i must have an edge to at least one vertex in C'_p . Also, for each $q \in \{\lambda + 1, \dots, \gamma'\}$, and $y \in C'_q$, y is adjacent to a vertex

in $\{u'_j, u_i\}$. When v'_j represents a null node, we only add the edge $(v'_j, v_{i\gamma'})$, if and only if the singleton color class consisting of u_i can be placed immediately after the last color class of $(C'_1, \dots, C'_{\gamma'})$. This holds when u_i has at least one neighbor in each of the color classes in $(C'_1, \dots, C'_{\gamma'})$.

Finally, the capacity of all edges is set to 1, and we are ready to use this flow network in our algorithm. Since the capacities are integers, the maximum flow in this network will also be integral. By running a MAXIMUM FLOW solver on this network, we obtain a maximum matching in the bipartite subgraph formed by the $s + 1$ main sections, with the condition that for each $1 \leq i \leq s$, only one of the vertices $v_{i0}, \dots, v_{i\gamma'}$ can be a part of the matching. This matching represents the paired and singleton color classes placed in their respective gaps. Note that singletons can appear in the final gap, according to the constraints on adding edges in the flow. The order of the pairs inside each gap can be arbitrary, as each pair includes a vertex from each clique. However, in the final gap, where singletons may appear, we arrange the pairs in an arbitrary order before placing the singletons (the order of the singletons can also be arbitrary). As outlined in Lemma 3, in any arbitrary Grundy coloring, a singleton can be placed at the end without affecting any properties of the Grundy coloring. This is why we only consider singletons to appear at the end. The size of this matching determines the extendability of $(C'_1, \dots, C'_{\gamma'})$: if it is equal to s , the answer is positive; otherwise, it is negative. The total running time of this algorithm is

$$\underbrace{s(\gamma' + 2) + 2}_{\text{Vertices}} + \underbrace{s(\gamma' + 2)}_{\text{Fixed edges}} + \underbrace{s^2\gamma'}_{\text{Other edges}} \times \underbrace{2(n - r - s - s')}_\text{Needed operations for checking an edge} + \underbrace{O((s(\gamma' + 2) + 2)^3)}_\text{Solving for MAXIMUM FLOW[6]}.$$

This running time can be upper-bounded, yielding a total time complexity of $O(n^2) + O(n^2) + O(n^3) \times O(n) + O((n^2)^3) = O(n^6)$. Thus, we provided a $O(n^6)$ algorithm to check the extendability of the Grundy coloring $(C'_1, \dots, C'_{\gamma'})$ of $G[Q \cup R]$, concluding the proof. \square

Theorem 4. *There is an algorithm that given a graph G and a 2-cluster modulator R of size r , runs in time $O(2^{O(r^2)}n^6)$ and outputs a Grundy coloring of G using the maximum number of colors.*

Proof. Let $F \subseteq V(G)$ be the subset from Lemma 8, with $|F| \leq r2^{r+1}$. Let $G' = G[R \cup F]$. By Lemma 8, there exists a subset $Q \subseteq V(G) \setminus R$ of size at most $2r$ such that $G[Q \cup R]$ has the following property: an optimal Grundy coloring (C_1, \dots, C_γ) of G contains a subsequence $(C'_1, \dots, C'_{\gamma'})$.

Therefore, we can guess the correct Q and a Grundy coloring $(C'_1, \dots, C'_{\gamma'})$ of $G[Q \cup R]$, extendable in $2^{O(r^2)}$ time. Testing if $(C'_1, \dots, C'_{\gamma'})$ is extendable can be done in $O(n^6)$, as per Theorem 3. After confirming extendability, Lemma 9 allows us to calculate the Grundy number and Grundy coloring of G . Using a similar approach explained in Sect. 3, constructing F takes $O(n+m)$ time. Thus, the total running time would be of $O(2^{O(r^2)}n^6)$, as desired. \square

5 Solving the k -Cluster Modulator Case

Finally, we consider the more general scenario involving a k -cluster modulator. In this setting, after removing the k -cluster modulator R from G , the remaining graph $G - R$ forms a k -cluster graph.

Let K_1, \dots, K_k denote the connected components of $G - R$, each of which is a complete graph. Define $S_1 = V(K_1), \dots, S_k = V(K_k)$ to be the cliques in G . Throughout this section, we denote $r = |R|$. To address this case, we encode the problem as an Integer Linear Program (ILP) with a number of variables bounded by a function of $r + k$. First, we define some notations and prove some auxiliary lemmas.

Definition 2. Let (C_1, \dots, C_γ) be a Grundy coloring of G . Define $CL(C_i)$ to be the set of cliques with a vertex in the color class C_i .

Lemma 10. Let (C_1, \dots, C_γ) be a Grundy coloring of G and let $(C_{\alpha_1}, \dots, C_{\alpha_t})$ be the color classes that do not contain a vertex from the modulator R , with $\alpha_1 \leq \dots \leq \alpha_t$. Then we have $CL(C_{\alpha_t}) \subseteq \dots \subseteq CL(C_{\alpha_1})$.

Proof. Suppose, for contradiction, that there exist two color classes C_{α_i} and C_{α_j} with $i > j$ such that $CL(C_{\alpha_i}) \setminus CL(C_{\alpha_j}) \neq \emptyset$. Let S_x be a clique in $CL(C_{\alpha_i}) \setminus CL(C_{\alpha_j})$. By definition, there exists a vertex $v \in S_x$ that appears in C_{α_i} . Since $S_x \notin CL(C_{\alpha_j})$, this vertex v is not adjacent to any vertex in C_{α_j} . However, this contradicts the properties of a Grundy coloring, which requires that every vertex in a later color class (here, C_{α_i}) must be adjacent to at least one vertex in every earlier color class (here, C_{α_j}). This contradiction proves the containment relationship, establishing the desired result. \square

Similar to the previous sections, let us define an equivalence relation \sim_R on the set $\bigcup_{i=1}^k S_i$. For any two vertices $u, v \in \bigcup_{i=1}^k S_i$, we say $u \sim_R v$ if and only if they have the same closed neighborhood, i.e., $N_G[u] = N_G[v]$. Notice that each equivalence class is a subset of a clique. The equivalence classes of \sim_R , that are subsets of the clique S_i , are denoted by $E_{i,1}, \dots, E_{i,q(i)}$, where $q(i)$ is the number of equivalent classes in S_i . The total number of equivalence classes is $\sum_{i=1}^k q(i) \leq k2^r$. Now from each equivalence class $E_{i,j}$ arbitrarily select a subset $F_{i,j} \subseteq E_{i,j}$ of size $\min\{r, |E_{i,j}|\}$. Let $F = \bigcup_{i=1}^k \left(\bigcup_{j=1}^{q(i)} F_{ij} \right)$. Note that we have $|F| \leq rk2^r$.

Lemma 11. There is vertex subset $Q \subseteq F$ of size at most kr and a (not necessarily optimal) Grundy coloring $(C'_1, \dots, C'_{\gamma'})$ of $G[Q \cup R]$ with the following property. There is an optimal Grundy coloring (C_1, \dots, C_γ) of G such that $(C'_1, \dots, C'_{\gamma'})$ is a subsequence of (C_1, \dots, C_γ) .

Proof. The proof is almost identical to the proof of Lemma 8. \square

Recall the definition of extendable Grundy coloring of $G[R \cup Q]$ (Definition 1).

Lemma 12. Let $(C'_1, \dots, C'_{\gamma'})$ be a (not necessarily optimal) Grundy coloring of $G[Q \cup R]$, that is extendable. Let $\beta = \max_{i=1}^k \{|S_i \setminus Q|\}$. Then, Grundy number of G is $\gamma' + \beta$.

Proof. By applying Lemma 10 instead of Lemma 7 within the detailed steps of the proof of Lemma 9, it is clear that the proof is derived similarly. \square

Using Theorem 6.4 from [5], we present an Integer Linear Programming (ILP) algorithm to achieve the following result:

Theorem 5 (\star). There is an algorithm that given a graph G and a Grundy coloring $(C'_1, \dots, C'_{\gamma'})$ of $G[Q \cup R]$, decides if $(C'_1, \dots, C'_{\gamma'})$ is extendable. This algorithm runs in time $O(p^{2.5p+o(p)})$, where $p = O(2^{kr}r)$.

Theorem 6. There is an algorithm that given a graph G and a k -cluster modulator R of size r , outputs a Grundy coloring of G using the maximum number of colors. This algorithm runs in time $O(2^{O(kr^2)}p^{2.5p+o(p)})$, where $p = O(2^{kr}r)$.

Proof. Let $F \subseteq V(G)$ be the subset from Lemma 11. Following a similar approach as in the proof of Theorem 4, we find that $|F| \leq r2^{r+k}$. We can guess the correct Q and a Grundy coloring $(C'_1, \dots, C'_{\gamma'})$ of $G[Q \cup R]$, extendable in $2^{O(kr^2)}$ time. To check whether $(C'_1, \dots, C'_{\gamma'})$ we can use the algorithm of Theorem 5 in $O(p^{2.5p+o(p)})$ time. By applying the same method as before, we obtain a total running time of $O(2^{O(kr^2)}p^{2.5p+o(p)})$, as desired. \square

Acknowledgments. We would like to sincerely thank the anonymous reviewers at CALDAM 2025 for their insightful and constructive comments. Their feedback significantly contributed to improving the clarity and quality of this work.

References

1. Aboulker, P., Bonnet, É., Kim, E.J., Sikora, F.: Grundy coloring and friends, half-graphs, bicliques. *Algorithmica* **85**(1), 1–28 (2023)
2. Belmonte, R., Kim, E.J., Lampis, M., Mitsou, V., Otachi, Y.: Grundy distinguishes treewidth from pathwidth. *SIAM J. Discret. Math.* **36**(3), 1761–1787 (2022)
3. Bonnet, É., Foucaud, F., Kim, E.J., Sikora, F.: Complexity of grundy coloring and its variants. *Discret. Appl. Math.* **243**, 99–114 (2018)
4. Christen, C.A., Selkow, S.M.: Some perfect coloring properties of graphs. *J. Comb. Theory Ser. B* **27**(1), 49–59 (1979)
5. Cygan, M., et al.: Parameterized Algorithms, vol. 5. Springer (2015)
6. Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum-flow problem. *J. ACM* **35**(4), 921–940 (1988). <https://doi.org/10.1145/48014.61051>
7. Havet, F., Maia, A.K., Yu, M.-L.: Complexity of greedy edge-colouring. *J. Braz. Comput. Soc.* **21**(1), 1–7 (2015). <https://doi.org/10.1186/s13173-015-0036-x>

8. Havet, F., Sampaio, L.: On the grundy and b-chromatic numbers of a graph. *Algorithmica* **65**, 885–899 (2013)
9. Hedetniemi, S.M., Hedetniemi, S.T., Beyer, T.: A linear algorithm for the grundy (coloring) number of a tree. *Congr. Numer.* **36**, 351–363 (1982)
10. Sampaio, L.: Algorithmic aspects of graph colourings heuristics. Ph.D. thesis, Université Nice Sophia Antipolis (2012)
11. Telle, J.A., Proskurowski, A.: Algorithms for vertex partitioning problems on partial k-trees. *SIAM J. Discret. Math.* **10**(4), 529–550 (1997)
12. Zaker, M.: Grundy chromatic number of the complement of bipartite graphs. *Australas. J Comb.* **31**, 325–330 (2005)



On a Tight Bound for the Maximum Number of Vertices that Belong to Every Metric Basis

Anni Hakanen¹, Ville Junnila¹, Tero Laihonen¹, Havu Miikonen^{1(\bowtie)},
and Ismael G. Yero²

¹ Department of Mathematics and Statistics, University of Turku, Turku, Finland
havu.e.miikonen@utu.fi

² Department of Mathematics, Universidad de Cádiz (Algeciras Campus),
Algeciras, Spain

Abstract. Metric bases of graphs have been widely studied since their introduction in the 1970’s by Slater and, independently, by Harary and Melter. In this paper, we concentrate on the existence of vertices in a graph G that belong to all metric bases of G . We call these basis forced vertices, and denote the number of them by $\text{bf}(G)$. We show that $\text{bf}(G) \leq 2/3(n - k - 1)$ for any connected nontrivial graph G of order n having k vertices in each metric basis. In addition, we show that this bound can be attained. Furthermore, the previous result implies the bound $\text{bf}(G) \leq 2/5(n - 1)$ formulated in terms of the order n of the graph for any nontrivial connected graph G . This result answers a question posed by Bagheri *et al.* in 2016. Moreover, we provide some realization results and consider some extremal cases related to basis forced vertices in a graph.

Keywords: Metric dimension · Metric basis · Basis forced vertices · Realization · Extremal graphs

AMS Subj. Class. (2020): 05C12

1 Introduction

A resolving set $R \subseteq V(G)$ in a graph $G = (V(G), E(G))$ represents a structure with the capability of uniquely recognizing all the vertices in $V(G)$ through a vector of distances to the vertices of R . The vertices of resolving sets are usually called “landmarks”, and the optimization of the quantity of vertices in any resolving set has led to the notion of a metric basis, which is a resolving set of the smallest possible cardinality in G . These notions were first and separately introduced in [17] by Slater, and in [11] by Harary and Melter, but they remained almost without any attention until the work [3], which created a breaking point in the interest on them. Nowadays, the metric dimension of graphs is a very well-known topic and there is a huge amount of information about it. Some recent

and interesting works on this parameter are for instance [4–6, 8, 13–16, 20, 21]. Moreover, for more information on this area we suggest the very interesting survey [19].

Several applications of resolving sets have been proposed in literature. Generally, the idea is to place sonar stations to some nodes of a network so that any object in a node can be located based on the distances given by the sonar stations [17]. An application of resolving sets to robot navigation already appeared in [11], and have been further developed in several other works. A recent and novel location property was presented in [18], where (certain kind of) resolving sets were used while identifying biological sequence data. In order to avoid presenting a large list of such works, any interested reader might simply check the recent survey [19], which contains a fairly complete compendium of applications, and combinatorial or computational properties of resolving sets and metric bases.

An interesting fact, regarding the metric bases of a given graph G , relates to the possible existence of vertices of G such that they are required to be landmarks in every metric basis, in order to locate or resolve the vertices of G . This fact means that a vertex satisfying this property plays a crucial role as a landmark, and thus, identifying such vertices is worthwhile. However, one cannot efficiently decide that a given vertex of a graph possesses such a property since, as proved in [9], it is a co-NP-hard problem to check whether a given vertex of a graph belongs to every metric basis of the graph. The vertices of a graph satisfying the previously mentioned property were called *basis forced vertices* in [9].

This suggests that finding tight bounds on the number of basis forced vertices in a given graph deserves attention. Hence, the aim of this work is to make significant contributions to this direction. Notice that in [1] Bagheri et al. studied graphs with unique metric bases, that is, graphs with a metric basis where all the vertices are basis forced. They provided bounds and posed a question on the maximum cardinality of a unique metric basis in any graph. In this paper, as a byproduct of Corollary 1, we are able to fully answer that open question. Some previous bounds for the number of basis forced vertices were already given in [9], and some other ones for specific families of graphs recently appeared in [10], but the former ones (and more general), were indeed not the best possible. We significantly improve them in this work.

Formally, given a connected graph $G = (V(G), E(G))$, it is said that a vertex $x \in V(G)$ *resolves* two vertices $u, v \in V(G)$ (or that u, v are *resolved* by x), if $d_G(v, x) \neq d_G(u, x)$, where $d_G(y, z)$ represents the distance between y and z , which is the number of edges in a shortest $y - z$ path in G . A set $R \subseteq V(G)$ is a *resolving set* for G if all the vertices of G are pairwise resolved by a vertex of R . A resolving set having the smallest possible cardinality in G is called a *metric basis*. The cardinality of a metric basis is the *metric dimension* of G , and denoted by $\dim(G)$.

Now, a *basis forced vertex* of a graph G is understood as a vertex $v \in V(G)$ such that it belongs to every metric basis of G . The term “basis forced vertex” was first used in [9], although it has some antecedents in the articles [1, 2], where the graphs that have a unique metric basis were studied. From now on, we represent the

number of basis forced vertices of G by $\text{bf}(G)$. Notice that we can have $\text{bf}(G) = 0$ for many graphs. For example, a nontrivial path P has $\text{bf}(P) = 0$.

1.1 Terminology and Notation

Throughout our exposition, the graphs G , for which $\text{bf}(G)$ is studied, are nontrivial (that is, they have at least two vertices), finite, undirected and connected. The complete graph on m vertices is denoted by K_m and a path on n vertices by P_n . The complement of a graph G is denoted by \bar{G} .

Given a graph G and a set $R \subseteq V(G)$, in [9], the *colour graph* G_R of G (with respect to R) is defined as follows. Let $r \in R$. We denote

$$\mathcal{U}_R(r) = \{\{x, y\} \in V(G)^2 \mid d_G(r, x) \neq d_G(r, y) \text{ and } \forall t \in R \setminus \{r\}: d_G(t, x) = d_G(t, y)\}.$$

In other words, the set $\mathcal{U}_R(r)$ consists of the pairs of vertices for which r is the unique element in R that resolves the pairs.

With this in mind, the graph G_R has the vertex set $V(G_R) = V(G)$ and the edge set

$$\bigcup_{r \in R} \mathcal{U}_R(r).$$

Each $r \in R$ is assigned a colour (or a label), and we colour the edges in G_R given by $\mathcal{U}_R(r)$ with the colour associated with r . Observe that the graph G_R can be disconnected while G is connected. Moreover, note that this graph G_R might be constructed for any set of vertices $R \subseteq V(G)$, although for our purposes, we will require that such R is a resolving set (or a metric basis). If there is no edge between x and y in G_R , then there are at least two elements in a resolving set R that resolve x and y . If R is a metric basis of G , then the graph G_R has at least one edge of the colour associated with each $r \in R$. In other words, the set $\mathcal{U}_R(r)$ is nonempty for all $r \in R$.

For an example (see [9]), consider the graph G illustrated in Fig. 1(a) and its resolving set $R = \{r_1, r_2\}$. For the colour graph G_R , we first form the following sets: $\mathcal{U}_R(r_1) = \{\{r_1, v_1\}, \{r_1, v_3\}, \{v_1, v_3\}, \{v_2, v_4\}\}$ and $\mathcal{U}_R(r_2) = \{\{r_2, v_1\}, \{r_2, v_2\}, \{v_1, v_2\}, \{v_3, v_4\}\}$. Then we obtain the colour graph $G_R = (V(G_R), E(G_R))$, where $V(G_R) = V(G)$ and $E(G_R) = \mathcal{U}_R(r_1) \cup \mathcal{U}_R(r_2)$. In Fig. 1(b) illustrating the colour graph G_R , the edges corresponding to r_1 and r_2 are associated with black and “dashed” edges, respectively.

2 Properties of the Colour Graph and Bounds for the Metric Dimension

The colour graph G_R regarding a resolving set R in a graph G plays an important role in the proofs of this paper. Therefore, we begin by giving some properties of the graph G_R stated in the next technical lemmas. The results of the Lemma 1 are already known from [9], whereas the claims in Lemma 2 are new. In this



Fig. 1. (a) The graph G with the resolving set $R = \{r_1, r_2\}$ and (b) the colour graph G_R .

section, we also provide (in Proposition 1) bounds on the metric dimension $\dim(G)$ when there are basis forced vertices in G . We will need these bounds for our main results in Sects. 3 and 4.

Lemma 1 ([9]). *Let G be a graph and let R be a resolving set of G . Then the following properties hold for G_R .*

- (i) *A colour that appears in a cycle of G_R appears at least twice in that cycle.*
- (ii) *Let $x, y, z \in V(G)$. If the edges $\{x, y\}$ and $\{x, z\}$ have the same colour in G_R , then the edge $\{y, z\}$ also has the same colour in G_R .*
- (iii) *If $b \in V(G)$ is a basis forced vertex of G and R is a metric basis of G , then the graph G_R has at least two edges of the colour associated with b .*
- (iv) *If $b \in V(G)$ is a basis forced vertex of G and R is a metric basis of G , then the graph G_R has at least one edge $\{x, y\}$, $x, y \in V(G) \setminus R$, of the colour associated with b .*
- (v) *The set of vertices R forms an independent set in G_R .*
- (vi) *If there is an edge in G_R incident to $r \in R$, then the edge has the colour associated with r .*

We next show some new properties of the colour graph G_R of a graph $G = (V(G), E(G))$ with respect to some resolving set or metric basis R . If we replace a vertex u in R by a vertex $v \in V(G)$ (while keeping R otherwise intact), then we denote the new set by $R[u \leftarrow v]$. In other words,

$$R[u \leftarrow v] = (R \setminus \{u\}) \cup \{v\}.$$

Lemma 2. *Let G be a graph and let R be a resolving set of G . Then the following properties hold for G_R .*

- (i) *If $b \in V(G)$ is a basis forced vertex of G and R is a metric basis of G , then the graph G_R has at least two edges $\{x, y\}$, $x, y \in V(G) \setminus R$, of the colour associated with b .*
- (ii) *If R is a metric basis of G and the only edge of the colour associated with $r \in R$ in G_R is of type $\{r, x\}$, where $x \in V(G) \setminus R$, then $R[r \leftarrow x]$ is a metric basis of G .*

Proof. (i) Let us assume that $b \in V(G)$ is a basis forced vertex. By Lemma 1(iv) we know that there is at least one edge, say $\{x, y\} \subseteq V(G) \setminus R$ associated with the colour b and, by Lemma 1(iii), there is also at least one more edge, say $\{w, z\}$, of the same colour. If $\{w, z\} \subseteq V(G) \setminus R$, we are done. Clearly, by Lemma 1(v), we cannot have $\{w, z\} \subseteq R$, so it suffices to assume that $w \in R$. Moreover, $w = b$ by Lemma 1(vi). Next we consider separately two possible cases, namely, $z \in \{x, y\}$ or $z \notin \{x, y\}$.

Case 1. Assume that $z \notin \{x, y\}$. If there is a third edge of colour b , then it is enough to assume that it is $\{b, z'\}$ where $z' \in V(G) \setminus R$. However, in that case, by Lemma 1(ii) there exists an edge $\{z, z'\}$ with $z, z' \in V(G) \setminus R$ and the claim follows. Hence, we may assume that $\{x, y\}$ and $\{b, z\} = \{w, z\}$ are the only edges of colour b . The set $R[b \leftarrow x]$ cannot be a metric basis, since b is a basis forced vertex which must be in every metric basis. Therefore, $d_G(x, z) = d_G(x, b)$ as the pair z and b is the only one that cannot be resolved with respect to $R[b \leftarrow x]$ (indeed, all the other pairs of vertices in $V(G)$ apart from the pair b and z and the pair x and y were resolved by other elements of R than b). Similarly, as $R[b \leftarrow y]$ (resp. $R[b \leftarrow z]$) cannot be a metric basis, we have $d_G(y, z) = d_G(y, b)$ (resp. $d_G(z, x) = d_G(z, y)$). These three distance equations imply together that $d_G(b, y) = d_G(z, y) = d_G(z, x) = d_G(b, x)$. However, this is a contradiction, since b was the (only) vertex resolving the vertices y and x , that is $d_G(b, y) \neq d_G(b, x)$.

Case 2. Assume that $z \in \{x, y\}$. Without loss of generality, assume $z = x$. Recall that by Lemma 1(ii) there is a third edge associated with the colour b , namely, $\{b, y\}$ in G_R . If there is yet another (fourth) edge of colour b , then we can assume that it is $\{b, z'\}$ where $z' \in V(G) \setminus R$. Due to Lemma 1(ii), this implies that there is an edge $\{z', x\}$ with $x, z' \in V(G) \setminus R$ of colour b , and we are done. Hence we may assume that we have only the three edges mentioned above. Because b is a basis forced vertex, the set $R[b \leftarrow y]$ is not a metric basis and, hence, $d_G(y, b) = d_G(y, x)$. Analogously, $R[b \leftarrow x]$ cannot be a metric basis. Hence, we obtain $d_G(x, b) = d_G(x, y)$. Now we have $d_G(b, x) = d_G(x, y) = d_G(b, y)$, a contradiction, since b resolves the pair x and y .

(ii) Let R be a metric basis (when R is only a resolving set, the claim follows by an analogous argument). Assume further that $r \in R$ and $\{r, x\}$ with $x \notin R$ is the only edge of colour associated with r (clearly, r is not a basis forced vertex). If we remove r from R , then the only pair of vertices that can have the same distances to all the elements of $R \setminus \{r\}$ is r and x . But in the set $R[r \leftarrow x]$ they have different distance to x . Therefore, the set $R[r \leftarrow x]$ is a metric basis in G . \square

The next result shows that if we have basis forced vertices in a graph G , then there are some limitations for the metric dimension $\dim(G)$.

Proposition 1. *If G is a graph such that $\text{bf}(G) \geq 1$, then $2 \leq \dim(G) \leq n - 4$.*

Proof. Assume that G is a graph with at least one basis forced vertex. Recall that paths are the only graphs with $\dim(G) = 1$. Hence, as the paths do not have basis forced vertices, it immediately follows that $\dim(G) \geq 2$. Let us look

at the claim $\dim(G) \leq n - 4$. Suppose to the contrary that $\dim(G) \geq n - 3$ (and $\text{bf}(G) \geq 1$). Let R be a metric basis of G . By Lemma 2(i), for each basis forced vertex u , there exist at least two edges associated with the colour u in G_R between vertices of $V(G) \setminus R$. Hence, $\dim(G) \leq n - 3$ as otherwise at most one edge can occur in $V(G) \setminus R$. Thus, $\dim(G) = n - 3$ and there are exactly three vertices, say x, y and z , in $V(G) \setminus R$. In addition, we know that there exists a unique basis forced vertex in G , say b . Indeed, if $\text{bf}(G) > 1$, then Lemma 2(i) would imply that $\dim(G) \leq n - 4$.

By Lemma 2(i), there exist at least two edges associated with the colour b in G_R between the vertices of $V(G) \setminus R$. Without loss of generality, we may assume that $\{x, y\}$ and $\{y, z\}$ are such edges. By Lemma 1(ii), this further implies that $\{x, z\}$ is also such an edge. Let r be a vertex in $R \setminus \{b\}$. Since R is a metric basis of G , there exists at least one edge in G_R associated with the colour r . Due to the fact that the edges between x, y and z are associated with the colour b , the edges of the colour r have to be of type $\{r, w\}$, where $w \in \{x, y, z\}$ due to claims (vi) and (v) in Lemma 1. Moreover, there exists exactly one such edge since otherwise a contradiction (with the fact that the edges between x, y and z are coloured with b) follows by Lemma 1(ii). For each $w \in \{x, y, z\}$, we define

$$R_w = \{w\} \cup \{u \in R \setminus \{b\} \mid \{u, w\} \in E(G_R)\}.$$

It is immediate that the sets $\{b\}$, R_x , R_y and R_z form a partition of $V(G)$. Furthermore, we have the following observations on R_w :

- (1) By Lemma 2(ii), the set $R[w \leftarrow v]$ is a metric basis for each $v \in R_w \setminus \{w\}$.
- (2) For each $r \in R_w \setminus \{w\}$, we have $d_G(b, w) = d_G(b, r)$ as the vertices r and w are solely resolved by r . Thus, the basis forced vertex b has the same distance to all $u \in R_w$.

Since the vertices x, y and z are resolved from each other by the vertex b , we may without loss of generality assume that $d_G(b, x) < d_G(b, y) < d_G(b, z)$. Thus, based on the partition $\{b\}$, R_x , R_y and R_z , we have that $d_G(b, x) = 1$, $d_G(b, y) = 2$ and $d_G(b, z) = 3$. Hence, by the observation (2) above, it happens $d_G(b, r_x) = 1$, $d_G(b, r_y) = 2$ and $d_G(b, r_z) = 3$ for all $r_x \in R_x \setminus \{x\}$, $r_y \in R_y \setminus \{y\}$ and $r_z \in R_z \setminus \{z\}$.

Let $bx'y'z$ be a shortest path of length 3 from b to z . It is immediate that $x' \in R_x$ and $y' \in R_y$; note that x' and y' can be x and y , respectively. By the observation (1), $R' = R[x' \leftarrow x]$ is a resolving set of G . As above, we may deduce that the edges $\{x', y\}$, $\{x', z\}$ and $\{y, z\}$ have colour b in $G_{R'}$. Moreover, in $G_{R'}$ there exists exactly one edge of colour y' , namely, $\{y, y'\}$. Therefore, by the observation (1), $R'' = R'[y' \leftarrow y]$ is a metric basis of G . Now $V(G) \setminus R'' = \{x', y', z\}$. Furthermore, $R''[b \leftarrow z]$ is a metric basis of G since $d_G(z, y') = 1$, $d_G(z, x') = 2$ and $d_G(z, b) = 3$. However, this contradicts the fact that b is a basis forced vertex. Thus, the claim follows. \square

Notice that there exist graphs attaining the upper bound of the proposition. For example, the graph G of order 6 in Fig. 1 satisfies $\text{bf}(G) = 2(> 1)$ and $\dim(G) = 2 = n - 4$.

3 The Main Bounds for the Maximum Number of Basis Forced Vertices

The main result of this section presents a bound for the number of basis forced vertices of a graph G in terms of the metric dimension $\dim(G)$ and the order n of G (see Theorem 1). As a special case, another bound for such quantity is given only in terms of the order of G (see Corollary 1).

A *cactus* is a graph in which no two cycles share an edge. In what follows, we want to show that a cactus graph appears as a certain subgraph of the colour graph G_R . To make the proof of Lemma 4 simpler, we present the following lemma, the proof of which is based on an iterative idea with an algorithmic flavour. The lemma shows that given a graph with two cycles sharing an edge, we can obtain a pair of cycles such that their intersection is a path (forming a so called theta graph). Recall that we define *cycle* as a closed walk with no repeated vertices. In the next lemma, we use the term *simple cycle* to emphasise the fact that cycles have no repeated vertices. Moreover, we “abuse” the notation and use the intersection notation $C_1 \cap C_2$ of two cycles (as subgraphs) to represent the subgraph induced by the vertices of the intersection $V(C_1) \cap V(C_2)$.

Lemma 3. *Let G be a graph and let C_1 and C_2 be simple cycles in G . If $C_1 \cap C_2$ contains at least one edge, then there exist a third cycle C_3 in G such that $C_1 \cap C_3$ is a nontrivial path (a path of length at least 2).*

Proof. The proof is given in the extended version of this article [7].

From now on, given a graph G and a resolving set R of G , we will consider a subgraph of G_R that will be “conveniently chosen” for our purposes. We consider a subgraph H_R of the colour graph G_R defined as follows. The vertex set of H_R is $V(H_R) = V(G) \setminus R$ and the edge set $E(H_R)$ is a (possibly not proper) subset of $E(G_R)$ satisfying that there are either 0 or 2 edges of each colour $r \in R$ in $E(H_R)$, i.e., there are either 0 or 2 edges of each color in H_R .

Despite the fact that several possible configurations for such subgraph H_R of G_R could exist, we next show that the connected components of each of them are bipartite cacti.

Lemma 4. *Let G be a graph and let $R \subseteq V(G)$ be a resolving set of G . Then the following claims follow for H_R .*

- (i) *The graph H_R is bipartite.*
- (ii) *The connected components of H_R are cacti.*

Proof. (i) Since H_R is a subgraph of G_R , if C is a cycle in H_R , then it is also a cycle in G_R . By Lemma 1(i), a colour that appears in a cycle of G_R appears at least twice in that cycle. Hence, by the choice of edges of H_R , a colour that appears in a cycle of H_R appears exactly twice in that cycle. Therefore, as all edges have a colour, all cycles in H_R have even length. It is well known that this implies that H_R is bipartite.

(ii) Suppose to the contrary that H_R is not composed of cacti, in other words, that there exist cycles in H_R that have edges in common. By Lemma 3, we may assume that C_1 and C_2 are cycles such that their intersection is a path, denoted $P = C_1 \cap C_2$. Now consider any colour r that appears on P . Clearly (as shown above), such colour r must appear exactly twice in the cycle C_1 , by the choice of the edges of H_R . If r appears in $C_1 \setminus P$, then C_2 is a cycle in which r appears only once, a contradiction. Therefore, the second occurrence of r must be in P as well.

Denote the vertices of C_1 by w_1, w_2, \dots, w_m so that vertices w_i and w_{i+1} are adjacent (with the notation that $w_{m+1} = w_1$) and assume that $V(P) = \{w_1, \dots, w_k\}$ where $k > 1$. Now consider the sequence of distances $d_G(r, w_1), d_G(r, w_2), \dots, d_G(r, w_m), d_G(r, w_1)$ for some $r \in R$. A difference in consecutive distances $d_G(r, w_i)$ and $d_G(r, w_{i+1})$ (say, $d_G(r, w_i) - d_G(r, w_{i+1}) = j \neq 0$) corresponds to an r -coloured edge $\{w_i, w_{i+1}\}$ in $E(H_R)$. If consecutive distances are equal, the corresponding edge in H_R has a colour other than r . By the choice of the edges of H_R , there can be at most two indices i and i' where consecutive distances differ. Since the first and last elements of the sequence are equal, we must have $d_G(r, w_i) = d_G(r, w_{i+1}) + j$ and $d_G(r, w_{i'}) = d_G(r, w_{i'+1}) - j$ for some $j \neq 0$.

For colours $s \in R$ that do not appear in P , we have $d_G(s, w_1) = d_G(s, w_2) = \dots = d_G(s, w_k)$. For colours $r \in R$ that do appear in P , we have $d_G(r, w_1) = d_G(r, w_k) + j - j = d_G(r, w_k)$, since r appears twice in P . Now the distinct vertices w_1 and w_k have $d_G(u, w_1) = d_G(u, w_k)$ for all $u \in R$, contradicting the assumption that R is a resolving set of G . Therefore, cycles in H_R do not have edges in common, in other words, H_R is composed of cacti. \square

The following result regarding the largest number of edges in a graph whose components are bipartite cacti shall be needed.

Lemma 5 ([12, Lemma 2]). *If H is a bipartite graph with $|V(H)| \geq 4$ such that all of its connected components are cacti, then $\frac{3}{4}|E(H)| + 1 \leq |V(H)|$.*

With the tools above in hand, we are then able to present our main theorem. By the notation $G[V(G) \setminus S]$ we mean the graph induced by the vertices of G not including the set $S \subseteq V(G)$.

Theorem 1. *Let G be a graph of order n with $\text{bf}(G) \geq 1$. Then*

$$\text{bf}(G) \leq \frac{2}{3}(n - \dim(G) - 1).$$

Proof. Let $R \subseteq V(G)$ be a metric basis of G . Let H_R be a subgraph of G_R (as previously described) with vertices $V(H_R) = V(G) \setminus R$ and for the edges of H_R , we select two edges of G_R with the colour b for each basis forced vertex b . Indeed, at least two such edges exist in $G_R[V(G) \setminus R]$ by Lemma 2(i) for each basis forced vertex. By definition, $|V(H_R)| = n - \dim(G)$ and $|E(H_R)| = 2\text{bf}(G)$. Since we assumed that $\text{bf}(G) \geq 1$, Proposition 1 gives us that $\dim(G) \leq n - 4$,

and consequently, $|V(H_R)| \geq 4$. By Lemma 4, the graph H_R is bipartite and its connected components are cacti. We can now use Lemma 5 to get

$$\frac{3}{4} \cdot 2\text{bf}(G) + 1 \leq n - \dim(G) \Rightarrow \text{bf}(G) \leq \frac{2}{3}(n - \dim(G) - 1),$$

which completes the proof. \square

In [1], Bagheri *et al.* study graphs with unique metrics basis, that is, graphs G satisfying $\dim(G) = \text{bf}(G)$. Among other results, they show that for any even $k \geq 2$ there exists a graph G such that $k = \text{bf}(G) = \dim(G) = 2(n-1)/5$, where n denotes the order of G . Furthermore, they state as an open question whether the metric dimension could be larger with respect to n . In the following corollary, we answer the question by stating that this is not possible.

Corollary 1. *Let G be a graph of order n . Then*

$$\text{bf}(G) \leq \frac{2}{5}(n-1).$$

Moreover, if $\text{bf}(G) = \frac{2}{5}(n-1)$, then G has a unique metric basis.

Proof. The first result follows immediately by the bound of Theorem 1, due to the fact that $\text{bf}(G) \leq \dim(G)$. Moreover, if $\text{bf}(G) = \frac{2}{5}(n-1)$, then we have

$$\frac{2}{5}(n-1) = \text{bf}(G) \leq \frac{2}{3}(n - \dim(G) - 1) \iff \dim(G) \leq \frac{2}{5}(n-1).$$

Therefore, $\dim(G) = \text{bf}(G)$ and G has a unique metric basis. \square

Notice that the bound of Corollary 1 can be attained as is shown in Sect. 4.

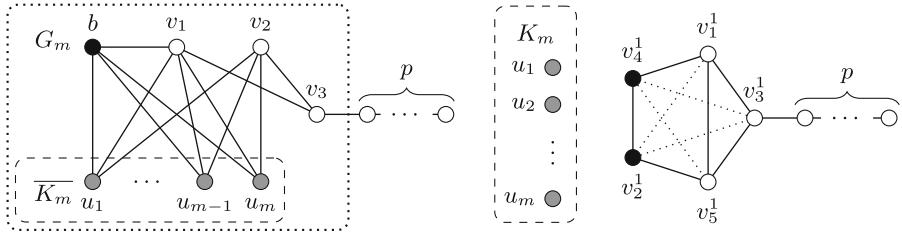
4 Realization of Possible Parameters

In the previous sections, we have considered the relations between $|V(G)| = n$, $\dim(G)$ and $\text{bf}(G)$. In particular, we obtained the following lower and upper bounds for $\dim(G)$:

- If $\text{bf}(G) = 1$, then $2 \leq \dim(G) \leq n-4$ by Proposition 1.
- If $\text{bf}(G) > 1$, then $\text{bf}(G) \leq \dim(G) \leq \lfloor n - \frac{3}{2}\text{bf}(G) - 1 \rfloor$ by Theorem 1.

In what follows, we show that $\dim(G)$ achieves all the values indicated by the previous inequalities when $\text{bf}(G)$ is even, or $\text{bf}(G) = 1$. For this purpose, we borrow the following result from [10].

Theorem 2 ([10, Theorem 2]). *Let G be a connected graph, and let $B \neq \emptyset$ be the set of basis forced vertices of G . Let $b \in B$, and let $v \in V(G) \setminus B$ be such that $d_G(b, v) = \max\{d_G(b, w) \mid w \in V(G) \setminus B\}$. Let P_p be the path $v_1 \dots v_p$. Let H be the graph with $V(H) = V(G) \cup V(P_p)$ and $E(H) = E(G) \cup E(P_p) \cup \{v, v_1\}$. Then, B is also the set of basis forced vertices of H and $\dim(H) = \dim(G)$.*



(a) The graph G_m is within the dotted line. The graph $G_{m,p}$ is obtained by attaching a path to v_3 .

(b) The graph $G_{q,m,p}$, with $q = 1$. The edges from and within K_m are omitted for illustrative purposes.

Fig. 2. Sketches of the constructions of Theorem 3. The black vertices are basis forced and the gray vertices are in some but not all metric bases.

We use Theorem 2 to grow the order of a graph without changing its metric dimension in the proof of the following theorem. The theorem discusses the possible values of the parameters $|V(G)|$, $\dim(G)$ and $\text{bf}(G)$.

Theorem 3. *Let f , k and n be positive integers.*

- If $f = 1$, then for each $k \in \{2, 3, \dots, n-4\}$ there exists a graph G such that $|V(G)| = n$, $\dim(G) = k$ and $\text{bf}(G) = f = 1$.
- If $f > 1$ and f is even, then for each $k \in \{f, f+1, \dots, \lfloor n - \frac{3}{2}f - 1 \rfloor\}$ there exists a graph G such that $|V(G)| = n$, $\dim(G) = k$ and $\text{bf}(G) = f$.

Proof. We will provide a method to construct a graph G with the desired values of $|V(G)|$, $\dim(G)$ and $\text{bf}(G)$.

First, we consider the case $f = 1$. By Proposition 1, we know that $2 \leq \dim(G) \leq n-4$. Consider the graph G_m as illustrated in Fig. 2(a), where $m \geq 2$. The m vertices in $\overline{K_m}$ are twins, hence a metric basis must contain at least $m-1$ of them. The set $V(\overline{K_m}) \setminus \{u_j\}$ is not a resolving set for any $j \in \{1, \dots, m\}$ since the vertices v_1 and v_2 are not resolved. It follows that $\dim(G_m) > m-1$. It is easy to verify that $(V(\overline{K_m}) \setminus \{u_j\}) \cup \{b\}$ is a resolving set of G_m for all $j \in \{1, \dots, m\}$. The sets $V(\overline{K_m})$, $(V(\overline{K_m}) \setminus \{u_j\}) \cup \{v_3\}$ and $(V(\overline{K_m}) \setminus \{u_j\}) \cup \{v_i\}$ are not resolving sets for any choice of j or $i \in \{1, 2\}$ since the pairs $\{v_1, v_2\}$, $\{v_1, v_2\}$ and $\{u_j, v_3\}$ are not resolved, respectively. Therefore, $\dim(G_m) = m$, the vertex b is basis forced, and the vertex v_3 is not basis forced and has $d_G(b, v_3) = 2 = \max\{d_G(b, w) \mid w \in V(G_m) \setminus \{b\}\}$. Choosing $m = k$, and applying Theorem 2 to the vertex v_3 with the path length $p = n - k - 4$, we get the graph $G_{m,p}$ with $m + 4 + p = n$ vertices (see Fig. 2(a)). By Theorem 2, the graph $G_{m,p}$ has the same metric dimension and basis forced vertices as G_m , and therefore, the graph $G_{m,p}$ has metric dimension $m = k$ and one basis forced vertex (namely, b). We assumed that $m \geq 2$ and, naturally, that $p \geq 0$. Hence, when m and p are chosen as defined, this construction realizes the values of k and n such that $k \geq 2$ and $n - k - 4 \geq 0$.

Now assume that $2 \leq f = 2q$ is even. Let $G_{q,m}$ be a graph such that $\overline{G_{q,m}} = P_5^1 \cup \dots \cup P_5^q \cup \overline{K_m}$ (these are disjoint unions), where $V(P_5^i) = \{v_1^i, v_2^i, v_3^i, v_4^i, v_5^i\}$, $V(\overline{K_m}) = \{u_1, \dots, u_m\}$ and $m \geq 1$. It has been shown (Lemma 22 in [9]) that all metric bases of $G_{q,m}$ are of the form $R = \bigcup_{i=1}^q \{v_2^i, v_4^i\} \cup (\overline{V(K_m)} \setminus \{u_j\})$ for some $j \in \{1, \dots, m\}$. There are $2q$ basis forced vertices in $G_{q,m}$ (vertices v_2^i and v_4^i for all $i \in \{1, \dots, q\}$), the metric dimension of $G_{q,m}$ is $2q + m - 1$, and particularly, the vertex v_3^1 is not basis forced. The vertices v_2^1 and v_3^1 satisfy the conditions for b and v in Theorem 2, respectively. Thus, we may attach a path of any length p to v_3^1 to obtain a graph $G_{q,m,p}$, illustrated in Fig. 2(b). Choosing $m = k - f + 1$ and $p = n - k - \frac{3}{2}f - 1$ gives us the graph $G_{q,m,p}$ with $5q + m + p = n$ vertices, $\dim(G_{q,m,p}) = 2q + m - 1 = k$, and $\text{bf}(G_{q,m,p}) = 2q = f$. The construction requires that $m \geq 1$ and $p \geq 0$. The choices of m and p are possible if $k - f + 1 \geq 1$ and $n - k - \frac{3}{2}f - 1 \geq 0$, or equivalently, if $f \leq k \leq n - \frac{3}{2}f - 1$, as promised. \square

In the full version of this paper [7], we will extend the previous result to the case when $f > 1$ is odd.

Acknowledgments. Ismael G. Yero has been partially supported by the Spanish Ministry of Science and Innovation through the grant PID2023-146643NB-I00. Ville Junnila, Tero Laihonen and Havu Miikonen have been partially supported by Academy of Finland grant number 338797. Anni Hakanen was supported by Turku Collegium for Science, Medicine and Technology (TCSMT).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Gh, B.B., Jannesari, M., Omoomi, B.: Unique basis graphs. *Ars Comb.* **129**, 249–259 (2016)
2. Buczkowski, P.S., Chartrand, G., Poisson, C., Zhang, P.: On k -dimensional graphs and their bases. *Period. Math. Hung.* **46**, 9–15 (2003). <https://doi.org/10.1023/A:1025745406160>
3. Chartrand, G., Eroh, L., Johnson, M.A., Oellermann, O.R.: Resolvability in graphs and the metric dimension of a graph. *Discrete Appl. Math.* **105**, 99–113 (2000). [https://doi.org/10.1016/S0166-218X\(00\)00198-0](https://doi.org/10.1016/S0166-218X(00)00198-0)
4. Claverol, M., et al.: Metric dimension of maximal outerplanar graphs. *Bull. Malays. Math. Sci. Soc.* **44**(4), 2603–2630 (2021). <https://doi.org/10.1007/s40840-020-01068-6>
5. Foster-Greenwood, B., Uhl, Ch.: Metric dimension of a direct product of three complete graphs. *Electron. J. Combin.* **31**, Paper No. 2.13 (2024). <https://doi.org/10.37236/12399>
6. Hakanen, A., Junnila, V., Laihonen, T.: The solid-metric dimension. *Theor. Comput. Sci.* **806**, 156–170 (2020). <https://doi.org/10.1016/j.tcs.2019.02.013>
7. Hakanen, A., Junnila, V., Laihonen, T., Miikonen, H., Yero, I.G.: On the Maximum Number of Vertices that Belong to Every Metric Basis (in preparation)

8. Hakanen, A., Junnila, V., Laihonen, T., Puertas, M.L.: On the metric dimensions for sets of vertices. *Discuss. Math. Graph Theory* **43**(1), 245–275 (2023). <https://doi.org/10.7151/dmgt.2367>
9. Hakanen, A., Junnila, V., Laihonen, T., Yero, I.G.: On vertices contained in all or in no metric basis. *Discrete Appl. Math.* **319**, 407–423 (2022). <https://doi.org/10.1016/j.dam.2021.12.004>
10. Hakanen, A., Junnila, V., Laihonen, T., Yero, I.G.: On the unicyclic graphs having vertices that belong to all their (strong) metric bases. *Discrete Appl. Math.* **353**, 191–207 (2024). <https://doi.org/10.1016/j.dam.2024.04.020>
11. Harary, F., Melter, R.: On the metric dimension of a graph. *Ars Combin.* **2**, 191–195 (1976)
12. Hernando, C., Mora, M., Pelayo, I.M.: Locating domination in bipartite graphs and their complements. *Discrete Appl. Math.* **263**, 195–203 (2019). <https://doi.org/10.1016/j.dam.2018.09.034>
13. Mashkaria, S., Ódor, G., Thiran, P.: On the robustness of the metric dimension of grid graphs to adding a single edge. *Discrete Appl. Math.* **316**, 1–27 (2022). <https://doi.org/10.1016/j.dam.2022.02.014>
14. Sedlar, J., Škrekovski, R.: Bounds on metric dimensions of graphs with edge disjoint cycles. *Appl. Math. Comput.* **396**, 125908 (2021). <https://doi.org/10.1016/j.amc.2020.125908>
15. Sedlar, J., Škrekovski, R.: Metric dimensions vs. cyclomatic number of graphs with minimum degree at least two. *Appl. Math. Comput.* **427**, 127147 (2022). <https://doi.org/10.1016/j.amc.2022.127147>
16. Sedlar, J., Škrekovski, R.: Vertex and edge metric dimensions of unicyclic graphs. *Discrete Appl. Math.* **314**, 81–92 (2022). <https://doi.org/10.1016/j.dam.2022.02.022>
17. Slater, P.J.: Leaves of trees. *Congr. Numer.* **14**, 549–559 (1975)
18. Tillquist, R.C., Lladser, M.E.: Low-dimensional representation of genomic sequences. *J. Math. Biol.* **79**(1), 1–29 (2019). <https://doi.org/10.1007/s00285-019-01348-1>
19. Tillquist, R.C., Frongillo, R.M., Lladser, M.E.: Getting the lay of the land in discrete space: a survey of metric dimension and its applications. *SIAM Rev.* **65**(4), 919–962 (2023). <https://doi.org/10.1137/21M1409512>
20. Wang, J., Tian, F., Liu, Y., Pang, J., Miao, L.: On graphs of order n with metric dimension $n - 4$. *Graphs Combin.* **39**, Paper No. 29 (2023). <https://doi.org/10.1007/s00373-023-02627-x>
21. Wu, J., Wang, L., Yang, W.: Learning to compute the metric dimension of graphs. *Appl. Math. Comput.* **432**, 127350 (2022). <https://doi.org/10.1016/j.amc.2022.127350>



Algorithms and Hardness Results for the (3, 1)-Cover Problem

Amirali Madani^{1(✉)}, Anil Maheshwari¹, Babak Miraftab¹, and Bodhayan Roy²

¹ Carleton University, Ottawa, ON, Canada

amiralmadani@cmail.carleton.ca, anil@scs.carleton.ca,

babakmiraftab@cunet.carleton.ca

² IIT Kharagpur, Kharagpur, India

Abstract. A connected graph has a (k, ℓ) -cover if each of its edges is contained in at least ℓ cliques of order k . Motivated by recent advances in extremal combinatorics and the literature on edge modification problems, we study the algorithmic version of the $(3, 1)$ -cover problem. Given a connected graph G , the $(k, 1)$ -cover problem is to find the minimum number of non-edges of G , whose addition to G yields a graph with a $(k, 1)$ -cover. We show that the $(3, 1)$ -cover problem is NP -complete for general graphs. Moreover, we show that it admits no polynomial-time constant-factor approximation algorithm unless $\mathbb{P} = \text{NP}$. However, we show that the $(3, 1)$ -cover problem can be solved in polynomial time when the input graph is chordal.

Keywords: Computational complexity · Graph algorithms · Optimal algorithms · Edge modification problems

1 Introduction

In recent years, research on *edge modification problems* has gained a lot of attention. The area of edge modification problems spans many definitions. Still, many such problems ask for the optimal way of editing an input graph G to another graph G' with a desired property, usually through edge additions to G [6]. The minimization objective is often defined in the literature as the number of added edges. Edge modification problems have been studied for many graph properties. For instance, some works have studied editing graphs into being Eulerian, regular, or having a specific degree sequence through edge additions and removals [8, 9, 11]. For a more comprehensive survey of edge modification problems, see [6].

Graphs with local covering conditions on edges or vertices have attracted significant interest in extremal graph theory. Burkhardt, Faber, and Harris [2] established asymptotically tight lower bounds on the number of edges in connected graphs where every edge lies in at least ℓ triangles. Chakraborti and

The first three authors are supported by NSERC. The last author is supported by the Science and Engineering Research Board (SERB) via the project MTR/2021/000474.

Loh [3] provided tight lower bounds, along with characterizations of extremal graphs, on the number of edges in graphs where every vertex belongs to a clique of order $k \geq 3$. Motivated by these results and the many applications of graphs with local edge covering conditions in big graph-based data analysis, Chakraborti et al. [4] introduced the concept of (k, ℓ) -covers. A connected graph G has a (k, ℓ) -cover if every edge of G lies in at least ℓ copies of K_k (a clique of order k). They proved tight lower bounds and structural characterizations of graphs with $(k, 1)$ -covers ($k \geq 3$) and graphs with $(3, 2)$ -covers. This motivates us to study the algorithmic version of the (k, ℓ) -cover problem.

1.1 New Results

We first present some definitions.

Definition 1. Let $G = (V, E)$ be a graph and let $E' \subseteq (V \times V) \setminus E$ be a set of non-edges of G . E' is a k -completion set of G if $G \cup E'$ has a $(k, 1)$ -cover.

Definition 2. The $(k, 1)$ -cover problem: Given a connected graph $G = (V, E)$ and two integers $k \geq 3$ and $t \geq 0$, does G have a k -completion set of size at most t ?

We study the $(3, 1)$ -cover problem in this paper. Our first result (Theorem 1) states that the $(3, 1)$ -cover problem is NP -complete for general graphs and admits no constant-factor approximation algorithm running in polynomial time unless $\mathbb{P} = \text{NP}$. However, in Theorem 3, we show that this problem can be solved in polynomial time when its input graph is restricted to the class of chordal graphs.

The remainder of this paper is organized as follows. In Sect. 2, we present the basic definitions and notations. Section 3 contains the hardness proof for the $(3, 1)$ -cover problem. In Sect. 4, we first present an optimal algorithm for the class of trees (Sect. 4.1, Proposition 1). We use the algorithm for trees to present an optimal algorithm for the class of chordal graphs in Sect. 4.2.

2 Preliminaries

A graph G is chordal if it does not have the cycle of length at least four as an induced subgraph. Throughout this paper, $G = (V, E)$ always serves as a connected graph with $n \geq 3$ vertices. Since we study the $(3, 1)$ -cover problem, we refer to 3-completion sets as completion sets in the remainder of this paper. For an edge $e \in E$, we say e is *unsaturated* if it is not contained in any triangles in G . For graph-theoretic and algorithmic notations and definitions not defined in the paper, we refer the readers to [1] and [5].

3 Hardness Results

We reduce the well-known problem of SET-COVER to the $(3, 1)$ -cover problem. The decision problem of SET-COVER is formally stated as follows. An instance

$(\mathcal{X}, \mathcal{F}, t)$ of SET-COVER consists of a finite set \mathcal{X} of items, a family of subsets \mathcal{F} of \mathcal{X} such that every item in \mathcal{X} belongs to at least one set from \mathcal{F} , and an integer t . Given an instance of SET-COVER, the problem is to determine whether there exists a subset $\mathcal{S} \subseteq \mathcal{F}$ with $|\mathcal{S}| \leq t$ such that the sets in \mathcal{S} cover all items of \mathcal{X} , i.e., $\bigcup_{S \in \mathcal{S}} S = \mathcal{X}$. We say that a subset $S \in \mathcal{F}$ *covers* its items, and each item $x_i \in \mathcal{X} \cap S$ is *covered* by S . It is well known that SET-COVER is NP-complete [10].

We now describe the reduction. For an instance $(\mathcal{X}, \mathcal{F}, t)$ of SET-COVER, construct a graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$. Initially, $\mathbb{V} = \emptyset$ and $\mathbb{E} = \emptyset$ (see Fig. 1).

1. For every set $S_j \in \mathcal{F}$, add a vertex S_j to \mathbb{V} . We refer to such vertices as *set vertices*
2. For every item $x_i \in \mathcal{X}$, add a subgraph I_i to \mathbb{G} . Each I_i is a disjoint union of $2|\mathcal{X}|$ isolated vertices. We refer to each I_i as an *item subgraph*
3. For each $x_i \in \mathcal{X}$ and every $S_j \in \mathcal{F}$, if $x_i \in S_j$, connect $S_j \in \mathbb{V}$ to all $2|\mathcal{X}|$ vertices of I_i , i.e., $\forall v \in V(I_i) : \mathbb{E} \leftarrow \mathbb{E} \cup \{(S_j, v)\}$.
4. For every edge (S_j, v) added in the previous step, add a new vertex w to \mathbb{V} by setting $\mathbb{V} \leftarrow \mathbb{V} \cup \{w\}$. Furthermore, set $\mathbb{E} \leftarrow \mathbb{E} \cup \{(S_j, w), (v, w)\}$. At the end of this step, all edges of \mathbb{G} are contained in triangles. We refer to these vertices w as *auxiliary vertices*
5. Add a vertex P to \mathbb{G} , $\mathbb{V} \leftarrow \mathbb{V} \cup \{P\}$. For every item subgraph I_i , connect every vertex of I_i to this new vertex, i.e., $\forall I_i \forall v \in V(I_i) : \mathbb{E} \leftarrow \mathbb{E} \cup \{(v, P)\}$. We refer to vertex P as *the common vertex*

Note that the unsaturated edges of \mathbb{G} are all of type (v, P) . Let x_i (S_j) be some item (set) with $x_i \in S_j$. Observe that adding the edge (S_j, P) to \mathbb{G} saturates all edges (v, P) for all $v \in V(I_i)$. Note that \mathbb{G} has exactly $|\mathcal{F}|$ set vertices, $2|\mathcal{X}|^2$ item vertices, one common vertex, and at most $2|\mathcal{X}|(|\mathcal{X}| \cdot |\mathcal{F}|) = 2|\mathcal{X}|^2 \cdot |\mathcal{F}|$ auxiliary vertices. Therefore, the size of the reduction graph is polynomial in $|\mathcal{X}| + |\mathcal{F}|$, as stated in the following observation:

Observation 1. *Given an instance $(\mathcal{X}, \mathcal{F}, t)$ of SET-COVER, the reduction graph \mathbb{G} has at most $\mathcal{O}(|\mathcal{X}|^2 \cdot |\mathcal{F}|)$ vertices. Moreover, due to the existence of the common vertex P , \mathbb{G} is connected.*

We present the following definition:

Definition 3. *Given an instance $(\mathcal{X}, \mathcal{F}, t)$ of SET-COVER, let E' be a completion set of its corresponding reduction graph \mathbb{G} . Let R be the set of all non-edges of \mathbb{G} between the set vertices and the common vertex, i.e., $R = \{(S_j, P) | \forall S_j \in \mathcal{F}\}$. If $E' \subseteq R$, we say E' is a *good completion set* of \mathbb{G} .*

We have the following lemma:

Lemma 1. *Any good completion set E' of \mathbb{G} corresponds to a set cover of size $|E'|$ for the corresponding SET-COVER instance.*

To prove the hardness result, we show that any completion set E' can be transformed to a good completion set E'' , where $|E''| \leq |E'|$, using Algorithm 1.

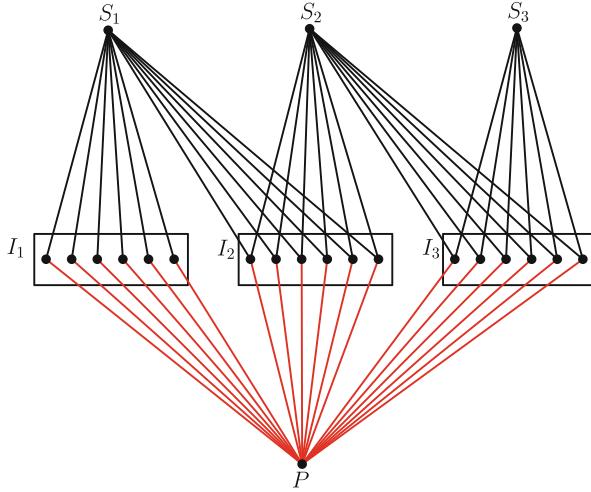


Fig. 1. An example of the reduction graph \mathbb{G} for $\mathcal{F} = \{S_1, S_2, S_3\}$, $\mathcal{X} = \{x_1, x_2, x_3\}$, $S_1 = \{x_1, x_2\}$, $S_2 = \{x_2, x_3\}$, and $S_3 = \{x_3\}$. Every black edge in this graph is contained in a triangle consisting of its endpoints plus one other auxiliary vertex omitted from this figure for simplicity (see Step 4 of the construction). Therefore, only the red edges of this graph are not contained in any triangles.

Algorithm 1

- 1: **Input:** \mathbb{G}, E' (a completion set of \mathbb{G})
 - 2: **Output:** E'' , a good completion set of \mathbb{G} with $|E''| \leq |E'|$
 - 3: **Initialization:** $E'' \leftarrow \emptyset$
 - 4: **Step 1:** For every item $x_i \in \mathcal{X}$, let $S_j \in \mathcal{F}$ be a set with $x_i \in S_j$. If E' has an edge (u, v) such that $\{u, v\} \subseteq V(I_i)$ for its corresponding item subgraph I_i , then add (S_j, P) to E'' , i.e., $E'' \leftarrow E'' \cup \{(S_j, P)\}$.
 - 5: **Step 2:** Add the *good* subset of E' to E'' , i.e., $E'' \leftarrow E'' \cup (E' \cap \{(S_j, P) \mid \forall S_j \in \mathcal{F}\})$.
 - 6: **Step 3:** For each item $x_i \in \mathcal{X}$, if I_i has an unsaturated edge $(u, P) \in \mathbb{E}$ (with $u \in V(I_i)$) in $\mathbb{G} \cup E''$, then add the edge (S_j, P) to E'' for some set $S_j \in \mathcal{F}$ with $x_i \in S_j$.
 - 7: **return** E''
-

Lemma 2. Algorithm 1 returns a good completion set E'' of \mathbb{G} in time polynomial in $(|\mathcal{X}| + |\mathcal{F}|)$ with $|E''| \leq |E'|$.

Proof. Since Algorithm 1 only adds edges of type (S_j, P) and continues until all edges of \mathbb{G} are saturated, it is easy to see that E'' is a good completion set of \mathbb{G} . Furthermore, it is also easy to see that Algorithm 1 runs in time polynomial in $(|\mathcal{F}| + |\mathcal{X}|)$.

We now show that $|E''| \leq |E'|$. To show this inequality, for every edge added to E'' , we *match* it to a unique edge in E' such that every edge in E' is matched

to at most one edge in E'' . As a result, $|E''| \leq |E'|$. Indeed, this matching is easy to see for Step 1 and Step 2 of Algorithm 1. In Step 1, if an edge (S_j, P) is added to E'' , we match it to some edge $(u, v) \in E'$ with $\{u, v\} \subseteq V(I_i)$ and $x_i \in S_j$. In Step 2, if some edge (S_j, P) is added to E'' (and was not added in the previous step), then we match it to its copy in E' , i.e., $(S_j, P) \in E'$. It is easy to see that in the first two steps, the edges of E'' are matched to unique edges from E' . To show this matching for Step 3, we prove the following claim:

Claim 1. *At the beginning of Step 3, if $\mathbb{G} \cup E''$ has an unsaturated edge (u, P) such that $u \in V(I_i)$ for some item x_i , then E' has at least $2|\mathcal{X}|$ edges that were not matched to any edge from E'' in Step 1 and Step 2.*

Proof. Suppose such an edge exists. Notice that for any two vertices $u, v \in V(I_i)$, we must have $(u, v) \notin E'$, because otherwise Algorithm 1 would have caught this edge in Step 1 and added an edge (S_j, P) for $x_i \in S_j$, saturating all such edges (u, P) . Similarly and using Step 2, we have $(S_j, P) \notin E'$ for any $S_j \in \mathcal{F}$ with $x_i \in S_j$. Since E' is a completion set of \mathbb{G} , it is easy to see that $2|\mathcal{X}|$ edges $\{(u, P) \mid \forall u \in V(I_i)\}$ were covered by at least $2|\mathcal{X}|$ edges in E' , not of the types removed in the first two steps. \square

We now conclude the proof of Lemma 2. If after Step 2 no such unsaturated edge (u, P) exists, then we are done. If such an edge exists, then using Claim 1, E' still has at least $2|\mathcal{X}|$ unmatched edges. Since at Step 3 we add at most $|\mathcal{X}|$ edges to E'' (one for each item in \mathcal{X}), we can easily match these edges to the ones described in Claim 1. \square

We restate a result on the inapproximability of SET-COVER:

Lemma 3. *[7, Corollary 4] For every $\varepsilon > 0$, it is NP-hard to approximate SET-COVER by a $((1 - \varepsilon) \cdot \ln |\mathcal{X}|)$ factor.*

Theorem 1. *The (3, 1)-cover problem is NP-complete. Moreover, it is NP-hard to approximate the (3, 1)-cover problem within a factor of c for any constant $c > 1$.*

Proof. We first show the NP-completeness. It is easy to see that the (3, 1)-cover problem belongs to NP since given a set of non-edges of an input graph G , it can be verified in polynomial time whether that set is a completion set of G of size at most t . We now show that the (3, 1)-cover problem is NP-hard. Let $\mathcal{I} = (\mathcal{X}, \mathcal{F}, t)$ be an instance of the SET-COVER problem. Construct the graph \mathbb{G} . Constructing this graph using Observation 1 takes polynomial time in $|\mathcal{X}|$ and $|\mathcal{F}|$. We claim that \mathcal{I} has a set cover of size at most t if and only if \mathbb{G} has a completion set of size at most t . Indeed, if \mathcal{I} has a set cover of size at most t , then we can construct a completion set for \mathbb{G} consisting of edges (S_j, P) for every set S_j in this set cover. Conversely, if \mathbb{G} has a completion set of size at most t , then using Lemma 2 and Algorithm 1, we can construct a good completion set E'' with $|E''| \leq |E'| \leq t$ which corresponds to a set cover of size $|E''| \leq t$ using Lemma 1. Therefore, the (3, 1)-cover problem is NP-hard.

We now show the approximability hardness. For the sake of contradiction, suppose $\mathbb{P} \neq \text{NP}$ and there exists a polynomial-time c -approximation algorithm A for the $(3, 1)$ -cover problem for some constant $c > 1$. For any instance $\mathcal{I} = (\mathcal{X}, \mathcal{F}, t)$ of the SET-COVER problem, let OPT_S be the size of the optimal set cover, and for its corresponding reduction graph \mathbb{G} , let OPT_G denote the size of its optimal completion set. Note that $\text{OPT}_S = \text{OPT}_G$. Every set cover for \mathcal{I} corresponds to a completion set for \mathbb{G} , so $\text{OPT}_G \leq \text{OPT}_S$. On the other hand, we also have $\text{OPT}_S \leq \text{OPT}_G$, because otherwise we would have $\text{OPT}_G < \text{OPT}_S$ and using Algorithm 1, Lemma 2, and Lemma 1, \mathcal{I} would have a set cover of size strictly less than OPT_S , a contradiction. Therefore, we have $\text{OPT}_G = \text{OPT}_S$.

Given algorithm A , we now show how to approximate any instance of SET-COVER within a factor of c in polynomial time. Given any such instance $(\mathcal{X}, \mathcal{F})$, we construct the reduction graph \mathbb{G} in $\mathcal{O}(|\mathcal{F}| \cdot |\mathcal{X}|^2)$ -time which is polynomial in $|\mathcal{F}| + |\mathcal{X}|$ (Observation 1). We run A on \mathbb{G} that gives us a completion set E' with $|E'| \leq c \times \text{OPT}_G$ in time polynomial in the size of \mathbb{G} and hence in $|\mathcal{X}|$ and $|\mathcal{F}|$. Using Algorithm 1, we can transform E' to another good completion set E'' with $|E''| \leq |E'|$ in time polynomial in $|\mathcal{X}|$ and $|\mathcal{F}|$. However, E'' is a good completion set that corresponds to a set cover with

$$|E''| \leq |E'| \leq c \times \text{OPT}_G = c \times \text{OPT}_S$$

Therefore, this results in a polynomial-time constant-factor approximation algorithm for SET-COVER. However, from Lemma 3, we know that for every $\varepsilon > 0$, it is NP-hard to devise an $((1 - \varepsilon) \cdot \ln |\mathcal{X}|)$ -approximation to SET-COVER, a contradiction. \square

4 An Optimal Algorithm for Chordal Graphs

This section presents an optimal algorithm for solving the $(3, 1)$ -cover problem on chordal graphs.

4.1 An Optimal Algorithm for Trees

Before presenting our main result, we briefly describe how we can optimally solve the $(3, 1)$ -cover problem when the input graph is a tree. We will later use the algorithm for trees to solve the problem for chordal graphs. We start by restating the following extremal result, which presents a tight lower bound on the number of edges of any n -vertex connected graph with a $(3, 1)$ -cover:

Theorem 2. [4, Theorem 1] Let G be a connected n -vertex graph ($n \geq 3$) with a $(3, 1)$ -cover. Then, the number of edges in G is at least $\lceil \frac{3(n-1)}{2} \rceil$.

Using the lower bound of Theorem 2, we now show that we can find an optimal completion set of any tree in linear time. Let $n \geq 3$ be any integer. Since every n -vertex tree has $n - 1$ edges, using Theorem 2, the size of the completion set of any n -vertex tree is at least $\lceil \frac{n-1}{2} \rceil$. Let $T = (V, E)$ be an n -vertex tree. We

partition the edges of E into $\lceil \frac{n-1}{2} \rceil$ subtrees. If $|E| = n - 1$ is even, all these subtrees are isomorphic to P_3 (the path on three vertices). If $|E|$ is odd, all but one subtree are isomorphic to P_3 , while the remaining subtree is isomorphic to K_2 . This partition can be obtained by extracting a copy of P_3 at each step rooted at the deepest leaf of T , removing its edges from T , and then recursing on the remaining tree. After obtaining this partition, we saturate each subtree by adding an edge, resulting in an optimal completion set of T of size $\lceil \frac{n-1}{2} \rceil$. We summarize our results on trees in the following.

Proposition 1. *Let $T = (V, E)$ be a tree with $|V| = n \geq 3$. In $\mathcal{O}(n)$ time, we can solve the (3, 1)-cover problem optimally for T by producing a completion set of size $\lceil \frac{n-1}{2} \rceil$.*

4.2 The Algorithm

We are now ready to describe our algorithm for chordal graphs. This section only considers connected chordal graphs with at least three vertices. Note that if an edge e is not a cut edge of a chordal graph G , it must belong to a cycle and, consequently, a triangle. We have the following notation.

Notation 1. *Let $G = (V, E)$ be a chordal graph. Denote by T_1, \dots, T_c the set of maximal connected subgraphs on the bridges of G .*

For convenience, we refer to these trees as *the trees of G* , see Fig. 2.

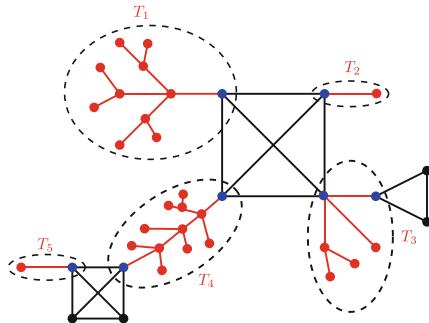


Fig. 2. An example of a chordal graph and its trees (depicted in red). The inner, outer, and boundary vertices are depicted in red, black, and blue, respectively. (Color figure online)

For a chordal graph G , we call the vertices that belong to some tree *the inner vertices* and those that belong to no trees *the outer vertices*. We refer to the vertices of G that belong to a tree and are incident to at least one non-bridge as the *boundary vertices*. Our algorithm for chordal graphs is described in Algorithm 2. Before proving the optimality of Algorithm 2, we present a few

notations. Let E' be any completion set of G . We denote by $E'(T_i, T_j)$ the set of edges of E' between the vertices of T_i and T_j , i.e., $E'(T_i, T_j) = \{(u, v) \in E' | u \in V(T_i), v \in V(T_j)\}$, where $i \neq j$.

Algorithm 2. An optimal algorithm for chordal graphs

- 1: **Input:** A connected chordal graph $G = (V, E)$ with $|V| \geq 3$
 - 2: **Output:** An optimal completion set E' of G
 - 3: **Initialization:** Find the trees T_1, \dots, T_c of G (see Notation 1), set $E' := \emptyset$.
 - 4: For every tree T_i with $|V(T_i)| \geq 3$, convert it to graph with a $(3, 1)$ -cover using the algorithm of Proposition 1. Update E' accordingly.
 - 5: For any tree T_i isomorphic to K_2 , let $V(T_i) = \{u, v\}$ and without loss of generality let u be a boundary vertex of G . Add (v, w) to E' , where $w \neq v$ is a neighbour of u in G .
 - 6: **return** E'
-

Lemma 4. *Let $G = (V, E)$ be any graph, and let E' be any optimal completion set of G . For every edge $e' \in E'$, there exists a triangle in $G \cup E'$ such that at least one edge of this triangle is unsaturated in G (from the set E).*

Proof. For the sake of contradiction, assume that there exists a non-empty subset $S \subseteq E'$ of edges that do not participate in any such triangles. Let $E'' = E' \setminus S$. It is easy to see that $G \cup E''$ has a $(3, 1)$ -cover with $|E''| < |E'|$, which contradicts the optimality of E' . \square

The next lemma states that one can construct an optimal completion set of G in which the endpoints of each edge in the completion set lie within some tree of G .

Lemma 5. *Let $G = (V, E)$ be any chordal graph, and let T_1, \dots, T_c denote the trees of G . Let E' be any optimal completion set of G . For any pair of distinct trees T_i and T_j of G with $|V(T_i)| \geq 3$ and $|V(T_j)| \geq 3$, we can transform E' to another optimal completion set by replacing all edges of $E'(T_i, T_j)$ with edges within $V(T_i)$ and $V(T_j)$.*

Proof. We prove this lemma by induction on $|E'(T_i, T_j)|$. For the base case, suppose $E'(T_i, T_j) = \{e_1\}$. By Lemma 4, e_1 must be a part of a triangle that contains an unsaturated edge of G . Since $|E'(T_i, T_j)| = 1$, it follows that this triangle consists of e_1 , an unsaturated edge from T_i or T_j , and an edge between two boundary vertices of G . Without loss of generality, assume that the unsaturated edge in this triangle belongs to T_i (i.e., $(u_1, v_1) \in E(T_i)$). We set $E' \leftarrow E' \setminus \{e_1\} \cup \{e'_1\}$, where e'_1 covers (u_1, v_1) in a triangle completely within $V(T_i)$, e.g., e'_1 can be between either u_1 (resp. v_1) and a distance-two neighbour of u_1 (resp. v_1) in T_i . It is easy to see that E' remains an optimal completion set with $E'(T_i, T_j) = \emptyset$. See Fig. 3 for an illustration.

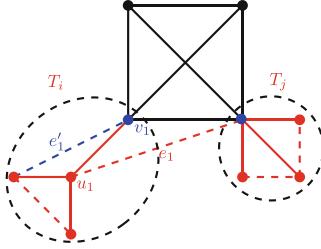


Fig. 3. The base case in the proof of Lemma 5: solid edges are the edges of E , and dashed edges are the edges of E' . We replace the red dashed edge e_1 with the blue dashed edge e'_1 in E' .

For the induction step, assume the lemma holds for any pair of distinct trees T_i and T_j such that $1 \leq |E'(T_i, T_j)| < d$. Now, consider an optimal completion set with $|E'(T_i, T_j)| = d$. We will show how to modify E' , without increasing its size while still keeping it a completion set, so that $|E'(T_i, T_j)| < d$. Once $|E'(T_i, T_j)|$ is reduced, we can apply the inductive hypothesis to complete the proof.

The procedure progresses in iterations. In each iteration ℓ , we remove an edge $e_\ell \in E'(T_i, T_j)$ from E' and add another edge e'_ℓ between two distance-two vertices within either $V(T_i)$ or $V(T_j)$ to E' , i.e., we set $E' \leftarrow E' \setminus \{e_\ell\} \cup \{e'_\ell\}$. We continue this process until $G \cup E'$ has no unsaturated edges $e \in E$. By Lemma 4, we do not need to worry about unsaturated edges in E' . If all edges in E are saturated, we can safely remove all unsaturated edges from E' using Lemma 4, yielding a smaller completion set, which contradicts the optimality of E' .

At the beginning of iteration ℓ , we have two subgraphs of T_i and T_j , denoted by $T_i^{\ell-1}$ and $T_j^{\ell-1}$, respectively. $T_i^{\ell-1}$ (resp. $T_j^{\ell-1}$) is the subgraph induced by the unsaturated edges from previous iterations in T_i (resp. T_j). For the first iteration, $T_i^0 = \emptyset$ and $T_j^0 = \emptyset$. As in the base case, we can find an edge $e_1 \in E'(T_i, T_j)$ that participates in a triangle containing an edge (u_1, v_1) from $E(T_i)$ or $E(T_j)$. Without loss of generality, assume $(u_1, v_1) \in E(T_i)$. We remove e_1 from E' , and if (u_1, v_1) becomes unsaturated, we cover it in a triangle by adding an appropriate edge e'_1 such that the resulting triangle lies completely within $V(T_i)$. We set $E' \leftarrow E' \setminus \{e_1\} \cup \{e'_1\}$, $T_i^1 = T_i^0 \cup (u_1, v_1)$ and $T_j^1 = T_j^0 \cup w$ where w is the endpoint of e_1 in $V(T_j)$. This operation does not increase the size of E' .

For any iteration $\ell > 1$, we first check if $G \cup E'$ has any unsaturated edges (from E) remaining. If no such edges exist, then $|E'(T_i, T_j)| < d$, and E' is a completion set, allowing us to apply the inductive hypothesis and complete the proof. If $(u_\ell, v_\ell) \in E$ is an unsaturated edge of $G \cup E'$, then, without loss of generality, assume that $(u_\ell, v_\ell) \in E(T_i)$. We find an edge $e_\ell \in E'(T_i, T_j)$ that is incident to (u_ℓ, v_ℓ) , remove it from E' , and cover (u_ℓ, v_ℓ) in a triangle by adding an edge e'_ℓ to E' such that the resulting triangle consists entirely of vertices from T_i , i.e., we set $E' \leftarrow E' \setminus \{e_\ell\} \cup \{e'_\ell\}$. Finally, we update the subtrees by setting

$T_i^\ell = T_i^{\ell-1} \cup (u_\ell, v_\ell)$ and $T_j^\ell = T_j^{\ell-1}$. The correctness of the described procedure is due to the following claim.

Claim 2. *At the start of each iteration $\ell \geq 2$, if the graph $G \cup E'$ contains any unsaturated edge $(u_\ell, v_\ell) \in E$, then $(u_\ell, v_\ell) \in E(T_i) \cup E(T_j)$. Additionally, there exists an edge $e_\ell \in E'(T_i, T_j)$ that is incident to (u_ℓ, v_ℓ) . At the end of every iteration $\ell \geq 1$, the subgraphs T_i^ℓ and T_j^ℓ remain connected subtrees of T_i and T_j , respectively.*

Proof. We prove this by induction on the number of iterations. The base case, $\ell = 1$, is straightforward, as the initial subgraphs are trivially connected. Now, assume the claim holds for every iteration up to $\ell - 1$. We will prove that it also holds for iteration ℓ .

By the induction hypothesis, at the beginning of iteration ℓ , the subgraphs $T_i^{\ell-1}$ and $T_j^{\ell-1}$ are still connected. Moreover, if $G \cup E'$ contains any unsaturated edge $(u_\ell, v_\ell) \in E$, then it must lie within $E(T_i)$ or $E(T_j)$, because we have only removed edges from $E'(T_i, T_j)$ in previous iterations. Without loss of generality, suppose $(u_\ell, v_\ell) \in E(T_i)$.

Since (u_ℓ, v_ℓ) is unsaturated, we know that at least one of its endpoints, u_ℓ or v_ℓ , must be a vertex of $T_i^{\ell-1}$, since it must have lost its triangle due to the edge removals of the previous iterations. Therefore, we have $|\{u_\ell, v_\ell\} \cap V(T_i^{\ell-1})| \geq 1$.

Next, we show that only one of u_ℓ or v_ℓ is in $V(T_i^{\ell-1})$, i.e., $|\{u_\ell, v_\ell\} \cap V(T_i^{\ell-1})| = 1$. Suppose, for the sake of contradiction, that both u_ℓ and v_ℓ belong to $V(T_i^{\ell-1})$. Since $(u_\ell, v_\ell) \notin E(T_i^{\ell-1})$ (as we have already covered all edges of $T_i^{\ell-1}$ in previous iterations by adding edges $e'_1, \dots, e'_{\ell-1}$), this would mean that there is now a cycle in $T_i^{\ell-1} \cup (u_\ell, v_\ell) \subseteq T_i$ ($T_i^{\ell-1}$ is a tree due to the induction hypothesis), contradicting the fact that T_i is a maximal connected subgraph on the bridges of G . Therefore, we must have $|\{u_\ell, v_\ell\} \cap V(T_i^{\ell-1})| = 1$.

Without loss of generality, assume $u_\ell \in V(T_i^{\ell-1})$ and $v_\ell \notin V(T_i^{\ell-1})$. Since (u_ℓ, v_ℓ) is unsaturated, it must have originally been part of a triangle involving another edge $e_{\ell'} \in E'(T_i, T_j)$ that was removed in a previous iteration. The edge $e_\ell \in E'(T_i, T_j)$ must still be present and incident to v_ℓ , because $v_\ell \notin V(T_i^{\ell-1})$ and, in previous iterations, we only removed edges between $V(T_i^{\ell-1})$ and $V(T_j^{\ell-1})$. We can now remove e_ℓ and cover (u_ℓ, v_ℓ) by adding a new edge e'_ℓ entirely within T_i , forming a triangle. Finally, since $u_\ell \in V(T_i^{\ell-1})$ and $v_\ell \notin V(T_i^{\ell-1})$, the subgraph $T_i^\ell = T_i^{\ell-1} \cup (u_\ell, v_\ell)$ remains connected. Similarly, T_j^ℓ also remains connected. An example for $\ell = 3$ is depicted in Fig. 4. □

Thus, the claim holds at the end of iteration ℓ , completing the proof. □

Due to the invariant of Claim 2, for every unsaturated edge, we remove one edge from $E'(T_i, T_j)$ and add another edge to E' to cover it in a triangle, and we never increase the size of E' . We continue until no unsaturated edges are left and $|E'(T_i, T_j)| < d$, at which point we apply the induction hypothesis. This finishes the proof of Lemma 5. □

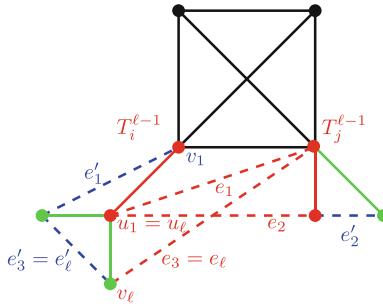


Fig. 4. An example of the induction step in the proof of Lemma 5 for $\ell = 3$: solid red edges are the edges of $T_i^{\ell-1}$ and $T_j^{\ell-1}$. Green edges belong to T_j and T_i but not to $T_j^{\ell-1}$ and $T_i^{\ell-1}$. We remove the red dashed edges and replace them with the blue dashed ones. In this example, $e_{\ell'} = e_1$. Observe how $T_i^{\ell-1} \cup (u_\ell, v_\ell)$ remains a tree.

We now prove our main result:

Theorem 3. *Let $G = (V, E)$ be a connected chordal graph on at least three vertices. Algorithm 2 produces an optimal completion set for G in $O(n + m)$ time, where $n = |V|$ and $m = |E|$.*

Proof. Let E' be any completion set of G . Due to Lemma 5, we can modify E' such that it remains optimal with $E'(T_i, T_j) = \emptyset$ for every T_i, T_j with $|V(T_i)| \geq 3$ and $|V(T_j)| \geq 3$. This can be done by applying the transformation of Lemma 5 to all such pairs of trees. Recall the definition of outer vertices. We can show the following claims analogously to Lemma 5:

Claim 3. *Any optimal completion set E' can be modified into another optimal completion set in which the endpoints of no edge in E' lie between an outer vertex of G and a vertex $v \in V(T_i)$ with $|V(T_i)| \geq 3$.*

Claim 4. *Let T_i, T_j be any two trees of G . Then, we can modify E' into another optimal completion set such that after adding the new completion set to G*

1. *If $|V(T_i)| \geq 3$ and $|V(T_j)| = 2$, then no triangles containing the edges of $E(T_i)$ contain a vertex from $V(T_j)$, and*
2. *Every tree T_j with $|V(T_j)| = 2$ is covered by one edge, exactly as described in line 5 of Algorithm 2.*

Both of these claims can be shown by removing any edges that do not conform to their respective conditions and fixing any resulting unsaturated edges from E similar to the transformation of Lemma 5. Using Lemma 5, Claim 3, and Claim 4, we can reduce the (3, 1)-cover problem on G to solving the problem locally for each tree of G . Therefore, Algorithm 2 is optimal.

As for the running time, all the trees of G can be located in $O(n + m)$ time by identifying the non-bridges in G . For each tree T_i of G , we can compute its optimal (3, 1)-cover in $O(|V(T_i)|)$ time by Proposition 1. Thus, in $O(n + m)$ time, we can construct an optimal (3, 1)-cover of G . \square

References

1. Bondy, J.A., Murty, U.S.R.: Graph Theory with Applications. American Elsevier Publishing Co., Inc., New York (1976)
2. Burkhardt, P., Faber, V., Harris, D.G.: Bounds and algorithms for graph trusses. *J. Graph Algorithms Appl.* **24**(3), 191–214 (2020)
3. Chakraborti, D., Loh, P.S.: Extremal graphs with local covering conditions. *SIAM J. Discret. Math.* **34**(2), 1354–1374 (2020)
4. Chakraborti, D., Madani, A., Maheshwari, A., Miraftab, B.: Sparse graphs with local covering conditions on edges. arXiv preprint [arXiv:2409.11216](https://arxiv.org/abs/2409.11216) (2024)
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press, Cambridge (2009)
6. Crespelle, C., Drange, P.G., Fomin, F.V., Golovach, P.A.: A survey of parameterized algorithms and the complexity of edge modification. *Comput. Sci. Rev.* **48**, 100556 (2023). <https://doi.org/10.1016/J.COSREV.2023.100556>
7. Dinur, I., Steurer, D.: Analytical approach to parallel repetition. In: Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing, pp. 624–633 (2014)
8. Golovach, P.A.: Editing to a connected graph of given degrees. *Inf. Comput.* **256**, 131–147 (2017). <https://doi.org/10.1016/J.IC.2017.04.013>
9. Goyal, P., Misra, P., Panolan, F., Philip, G., Saurabh, S.: Finding even subgraphs even faster. *J. Comput. Syst. Sci.* **97**, 1–13 (2018). <https://doi.org/10.1016/J.JCSS.2018.03.001>
10. Karp, R.M.: On the computational complexity of combinatorial problems. *Networks* **5**(1), 45–68 (1975)
11. Mathieson, L., Szeider, S.: Editing graphs to satisfy degree constraints: a parameterized approach. *J. Comput. Syst. Sci.* **78**(1), 179–191 (2012)



Extension Perfect Roman Domination

Kevin Mann^(✉) and Henning Fernau

Universität Trier, Fachbereich 4 – Abteilung Informatikwissenschaften,
54286 Trier, Germany
{mann,fernau}@uni-trier.de

Abstract. For an introductory single lecture into parameterized complexity classes, it appears to be necessary to introduce different problems to capture the (most important) different levels of the W-hierarchy. This puts an additional burden on the audience, as they also have to understand the different problems and not only the different complexity classes. In this paper, we will show that EXTENSION PERFECT ROMAN DOMINATION, EXT PRD for short, is a single problem that can be used for this introductory purpose. We can characterize the classes W[1], W[2] and W[3] and find problems in FPT, XP and those being para-NP-hard. Also, it can be used to explain how the choice of the parameter can influence the complexity status of the problem. One can even touch fine-grained complexity by establishing a run-time lower bound based on the k -Orthogonal Vector conjecture. EXT PRD, being a sibling to ROMAN DOMINATION, also comes with a nice story that can be also seen from the perspective of adventure games and might hence be appealing.

Keywords: (Perfect) Roman domination · Parameterized Complexity · W[3]

1 Introduction

In textbooks on Parameterized Complexity, as [9, 10, 13], the W-hierarchy is defined by WEIGHTED NORMALIZED SATISFIABILITY problems or variations thereof. For the classes W[1] and W[2], there are some examples for natural graph problems which are complete for such a class with respect to some parameter; often, one considers solution size as a parameter. For W[1]-completeness, examples include INDEPENDENT SET, PERFECT CODE or IRREDUNDANT SET [11]; for W[2]-completeness, DOMINATING SET is the standard example. W[1] and W[2] can be also characterized by Turing machine halting problems, namely, for single and multiple tapes, respectively, now parameterized by the number of steps [7]. However, W[3] is not that well known as the other two classes.

Before giving an example for a W[3]-complete problem, we have to understand the idea of *extension problems*. For a general definition of extension problems, we refer to [5]. Here, we will only discuss the extension version of minimization problems on graphs. Depending on the concrete problem (we choose to illustrate this in the following with DOMINATING SET in parentheses), a graph $G = (V, E)$ defines the search space $\text{presol}(G)$ of *pre-solutions* (for domination, $\text{presol}(G) = 2^V$) and a

set of *solutions* $\text{sol}(G) \subseteq \text{presol}(G)$ (dominating sets). For the extension version, we also need to define a partial order \preceq on $\text{presol}(G)$ (\subseteq for domination). The notion of a *minimal solution* is understood with respect to \preceq : $s \in \text{sol}(G)$ is called minimal if, for each $p \in \text{presol}(G) \setminus \{s\}$, $p \preceq s$ implies $p \notin \text{sol}(G)$. An instance of the extension version consists, apart from the graph G , in a pre-solution $p \in \text{presol}(G)$ (some vertex set). The question (called EXTENSION DOMINATING SET, or EXT DS for short) is if there exists a minimal $s \in \text{sol}(G)$ with $p \preceq s$. In general, these problems could be of help to speed up branching algorithms. For example, branching algorithms for the enumeration of all minimal solutions will implicitly create a pre-solution p , and then efficiently determine if any minimal solution exists that extends p would be very beneficial.

With respect to parameterized complexity, extension problems are interesting since EXT DS, parameterized by the size of the given pre-solution, is $\text{W}[3]$ -complete [5]. Interestingly, quite a number of problems that are also $\text{W}[3]$ -complete with respect to different parameters show up in database theory [18]. The particular parameterized complexity of EXT DS motivated us to look at extension variants of problems similar to DOMINATING SET (DS). In particular, we studied ROMAN DOMINATION (RD) that shows the same complexity as DS in many aspects. In the case of EXTENSION ROMAN DOMINATION, or EXT RD for short, $\text{presol}(G)$ is the set of all mappings $f : V \rightarrow \{0, 1, 2\}$ and $\text{sol}(G)$ is the set of all Roman dominating functions (Rdf for short). Now, \preceq is given by the partial order \leq describing the pointwise ordering. In [2], it was shown that EXT RD is polynomial-time solvable. This is surprising as this is quite different from EXT DS. This algorithm was also used to show that all minimal Rdf can be enumerated with polynomial delay and space. By giving a bijection that can be computed in polynomial time, in [17] it was shown that minimal perfect Roman dominating functions (pRdf for short) can also be enumerated with polynomial delay and space. The notion of perfect Roman domination was introduced in [15].

In this paper, we will study the parameterized complexity of EXTENSION PERFECT ROMAN DOMINATION (EXT PRD). First we will define important concepts in Sect. 2, including some (hi)story behind this problem. Then, Sect. 3 contains some combinatorial results. In the main part, different parameterizations of EXT PRD are studied, as surveyed in Fig. 1.

2 Preliminaries

General Notations. \mathbb{N} denotes the nonnegative integers and for $n \in \mathbb{N}$, $[n] := \{1, \dots, n\}$. For $A \subseteq X$, $\chi_A : X \rightarrow \{0, 1\}$ denotes its characteristic function. For $f, g : A \rightarrow \mathbb{N}$, $f \leq g$ iff for all $a \in A$, $f(a) \leq g(a)$ (pointwise). If A is finite, $\omega(f) := \sum_{a \in A} f(a)$ is the *weight* of $f : A \rightarrow \mathbb{N}$. If (X, \preceq) is a partial order, then $A \subseteq X$ is *upward-closed* if, for any $x, y \in X$, if $x \in A$ and $x \preceq y$, then $y \in A$. Let $G = (V, E)$ be a graph with the *vertex set* V and the *edge set* E . The (open) neighborhood of a $v \in V$ (in G) is defined as $N_G(u) := \{u \in V \mid \{v, u\} \in E\}$. $N_G[v] := \{v\} \cup N_G(v)$ denotes the closed neighborhood of v in G . A set $D \subseteq V$ is called a *dominating set* if for each $v \in V$, $|D \cap N[v]| \geq 1$. D is a *perfect code*

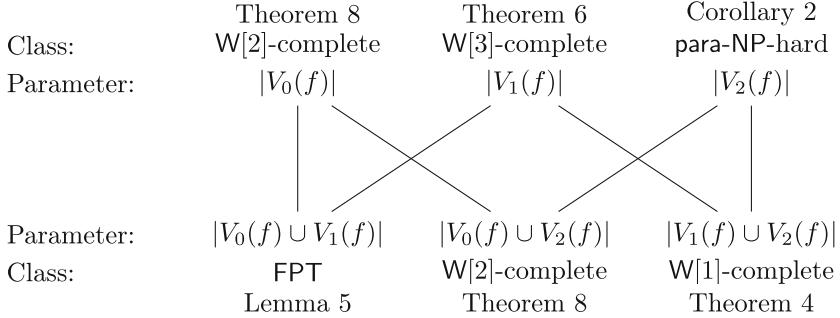


Fig. 1. Overview of the complexity results for EXT PRD.

if $|D \cap N[v]| = 1$ for all $v \in V$. For a set $X \subseteq V$ and two vertices $v \in X$ and $u \in V$, we say that u is a *private neighbor* of v if $N_G[u] \cap X = \{v\}$.

Roman Domination. $f : V \rightarrow \{0, 1, 2\}$ is called a *Roman dominating function* (Rdf) on G if, for each $v \in V$ with $f(v) = 0$, there is a $u \in N(u)$ with $f(u) = 2$. To simplify notation, let $V_i(f) := \{v \in V \mid f(v) = i\}$ for $i \in \{0, 1, 2\}$. A *perfect Roman dominating function* (pRdf) on G is a Rdff $: V \rightarrow \{0, 1, 2\}$ on G such that for each $v \in V_0(f)$, $|N(v) \cap V_2(f)| = 1$. We focus on the following problem.

Problem name: EXTENSION PERFECT ROMAN DOMINATION (EXT PRD)

Given: A graph $G = (V, E)$ and $f : V \rightarrow \{0, 1, 2\}$.

Question: Is there a (pointwise) minimal pRdfg on G with $f \leq g$?

Adventure Game Motivation. Consider a world where your task (as a player, you are ruling an empire) is to secure the life of your people against external attacks. Your strategy is to dispose your military forces, grouped in armies, in a way that each region of your empire can either defend itself or, if there is no army stationed on it, an army stationed on a neighboring region can come to aid without weakening that region. This defense strategy was reported to be used by the Roman Emperor Constantine the Great and this explains part of the problem name. In ancient times, not too much time should be wasted on extensive communication to clarify responsibilities. Thus, a defenseless region should know which region to ask for help. Also, we take into consideration that a defense infrastructure will not be built completely in the void; rather, we assume that certain forces have already been deployed, and now the task is to add more forces (and to build the defense infrastructure) so that finally the whole situation is satisfactory in the described sense. For reasons of efficiency, we require minimality in the sense that it is not possible to renounce any of the stationed troops without losing the defense capabilities defined in the strategy.

Observation 1. Let $G = (V, E)$ be a graph and $f : V \rightarrow \{0, 1, 2\}$.

- (1) If f is a pRdf, then for all $v \in V_2(f)$ and $u \in N(v) \cap V_0(f)$, u is a private neighbor of v .
- (2) If f is a Rdf such that for all $v \in V_2(f)$ and $u \in N(v) \cap V_0(f)$, u is a private neighbor of v , then f is a pRdf.

A final note on parameterized complexity: If P is some problem and κ is some parameter, we write $\kappa\text{-}P$ to denote the problem P parameterized by κ .

3 Combinatorial Results and an XP-Algorithm

Here, we want to collect some combinatorial results which will help us prove hardness and membership results. For one such result, we will give an XP-algorithm. But first, we discuss some properties of pRdfs with respect to \leq .

Lemma 1. *Let $G = (V, E)$ be graph, and $f, g, h : V \rightarrow \{0, 1, 2\}$ with $f \leq g \leq h$. If f, h are pRdfs but g is not, then there is a $u \in V$ such that $h - \chi_{\{u\}}$ is a pRdf.*

Proof. Since the Roman dominating functions are upward-closed with respect to \leq , g is a Rdf. This implies there is a $v \in V_0(g) \subseteq V_0(f)$ with $|N_G(v) \cap V_2(f)| = 1 < 2 \leq |N_G(v) \cap V_2(g)| \leq |N_G(v) \cap V_2(h)|$. Let $w \in V_2(f) \cap N_G(v)$ and $u \in (V_2(h) \cap N_G(v)) \setminus \{w\}$. Hence, $u \notin V_2(f)$. We want to show that $\tilde{h} := h - \chi_{\{u\}}$ is a pRdf. Assume the contrary. As $u \notin V_2(f)$, $f \leq \tilde{h}$. Thus, \tilde{h} is a Rdf. Therefore, there must be an $x \in V_0(\tilde{h})$ with $|N_G(x) \cap V_2(\tilde{h})| \neq 1$. Since \tilde{h} is a Rdf, there is a $y \in V_2(\tilde{h}) \cap N_G(x)$. Hence, $|N_G(x) \cap V_2(\tilde{h})| > 1$. By construction, $x \in V_0(\tilde{h}) = V_0(h)$ and $y \in V_2(\tilde{h}) = V_2(h) \setminus \{u\}$. Thus, $1 < |N_G(x) \cap V_2(\tilde{h})| \leq |N_G(x) \cap V_2(h)|$ would contradict the fact that h is a pRdf. Hence \tilde{h} is a pRdf. \square

Global minimal should be distinguished from local ones. In our setting of pRdfs, global minimality is just minimality, while a pRdf $f : V \rightarrow \{0, 1, 2\}$ is *locally minimal* if for each $u \in V_1(f) \cup V_2(f)$, $f - \chi_{\{u\}}$ is not a pRdf. Lemma 1 implies that locally minimal pRdfs are also globally minimal pRdfs, which means that minimality itself can be easily decided for pRdfs. We now recall a result.

Theorem 1 ([17]). *Let $G = (V, E)$ be a graph. A function $f : V \rightarrow \{0, 1, 2\}$ is a minimal pRdf iff the following three conditions are satisfied:*

[V_0] : $\forall v \in V_0(f) : |N_G(v) \cap V_2(f)| = 1$, [V_1] : $\forall v \in V_1(f) : |N_G(v) \cap V_2(f)| \neq 1$, and [V_2] : $\forall v \in V_2(f) : |N_G(v) \cap V_0(f)| \neq 0$.

Theorem 1 can be used to show some results for EXT PRD in the following.

Lemma 2. *Let $G = (V, E)$ be graph and $f : V \rightarrow \{0, 1, 2\}$. Then there exists a minimal pRdf $g : V \rightarrow \{0, 1, 2\}$ with $f \leq g$ and $V_2(f) = V_2(g)$ iff each $v \in V_2(f)$ has a private neighbor in $V_0(f)$ (with respect to G and $V_2(f)$) and $|N_G(u) \cap V_2(f)| \neq 1$ for each $u \in V_1(f)$.*

Proof. First we consider the if-part. Let $A = \{v \in V_0(f) \mid |N_G(v) \cap V_2(f)| \neq 1\}$. Define $g = f + \chi_A$. Clearly, $f \leq g$ and $V_2(f) = V_2(g)$. This leaves to show that g is a minimal pRdf. By definition of A , for each $v \in V_0(g) = V_0(f) \setminus A$, $1 = |N_G(v) \cap V_2(f)| = |N_G(v) \cap V_2(g)|$. Furthermore, for all $v \in V_1(g) = V_1(f) \cup A$, $1 \neq |N_G(v) \cap V_2(f)|$. Let $v \in V_2(g)$. By the requirement that $V_2(f) = V_2(g)$, for $v \in V_2(g)$, there exists a private neighbor $u \in V_0(f)$. Thus, $|N_G(u) \cap V_2(f)| = 1$

Algorithm 1. Solving instances of EXTprd

```

1: procedure EXTprdSOLVER( $G, f$ )
   Input: A graph  $G = (V, E)$  and a function  $f: V \rightarrow \{0, 1, 2\}$ .
   Output: Is there a minimal pRdf  $\tilde{f}$  with  $f \leq \tilde{f}$ ?
2:   for  $v \in V_2(f)$  do
3:      $x := \text{false}$ 
4:     for  $w \in N(v) \cap V_0(f)$  do
5:       if  $|N(w) \cap V_2(f)| = 1$  then  $x := \text{true}$ 
6:     if  $\neg x$  then return no
7:   for  $v \in V_1(f)$  do
8:     if  $|N(v) \cap V_2(f)| = 1$  then
9:       for  $u \in N[v] \setminus V_2(f)$  do // Try raising the value of  $u$  to 2.
10:        if EXTprd SOLVER( $G, f + (2 - f(u)) \cdot \chi_{\{u\}}$ ) then return yes
11:   return no
12: return yes

```

and $u \in V_0(f) \setminus A = V_0(g)$. Hence, $N_G(v) \cap V_0(f) \neq \emptyset$ for all $v \in V_2(f)$. In conclusion, g is a minimal pRdf.

For the only-if-part, assume there is a minimal pRdf g on G with $f \leq g$ and $V_2(f) = V_2(g)$. Hence, $V_0(g) \subseteq V_0(f)$ and $V_1(g) \subseteq V_1(f)$. Since g is a minimal pRdf, Conditions $[V_0]$ and $[V_2]$ of Theorem 1 hold. Thus each $v \in V_2(g) = V_2(f)$ has a private neighbor in $V_0(g) \subseteq V_0(f)$. For each $u \in V_1(g)$, $|N_G(u) \cap V_2(f)| = |N_G(u) \cap V_2(g)| \neq 1$. As $V_1(f) \subseteq V_1(g)$, this also holds for all $u \in V_1(f)$. \square

The proof gives also an algorithm to compute a minimal pRdf from the given function f : we only compute $A \subseteq V_0(f)$ as specified in the proof, which can be done in polynomial time.

Next, we will provide an explicit XP-algorithm, allowing a running time of $n^{\mathcal{O}(V_1(f))}$ on graphs of order n . In Sect. 4, we even prove W[1]-completeness. From this, one can also deduce membership in XP, but without having an implementable algorithm in hands. But more importantly, the Algorithm 1 implies some theoretical results which we need for the W[1]-membership proof and it is close to optimal under the k -Orthogonal Vector conjecture, see Remark 1. As testified in [18] for quite related problems (confer the discussions in [12]), also XP-algorithms of the proposed form could be useful in practical implementations.

Theorem 2. *Algorithm 1 is an XP algorithm for κ -EXT PRD with parameter $\kappa \in \{\omega(f), |V_1(f)|\}$. Let (G, f) be an instance of EXT PRD, with $G = (V, E)$, $f: V \rightarrow \{0, 1, 2\}$. If a minimal pRdf g is returned, then $|V_2(g)| \leq \omega(f)$.*

Proof. Let $G = (V, E)$ be graph and $f: V \rightarrow \{0, 1, 2\}$. Since $|V_1(f)| \leq \omega(f)$, we only need to consider $|V_1(f)|$ as parameterization. At first we explain the algorithm. The idea is to modify f such that it fulfills the conditions of Lemma 2. Therefore, we look if each $v \in V_2(f)$ has a private neighbor in $V_0(f)$. If this is not the case, then there exists no minimal pRdf bigger than f . If each vertex

in $V_2(f)$ has a private neighbor in $V_0(f)$, we look if there exists a $v \in V_1(f)$ with $|N_G(v) \cap V_2(f)| = 1$. If this is not the case, then f itself already fulfills the condition of Lemma 2. Otherwise, we go through all vertices $u \in N_G[v] \setminus V_2(f)$ and try the algorithm with $f_u = f + (2 - f(u)) \cdot \chi_{\{u\}}$, changing the value of f to 2 on argument u . Since we only return yes (or a minimal pRdf) if f verifies the condition of Lemma 2 and we only increase f , there exists a minimal pRdf $g : V \rightarrow \{0, 1, 2\}$ with $f \leq g$ if we return yes (or a pRdf).

Conversely, assume there is a minimal pRdf g with $f \leq g$. Hence, $V_2(f) \subseteq V_2(g)$ and $V_0(g) \subseteq V_0(f)$. We prove that the algorithm will return a minimal pRdf by an induction argument on $\omega(g - f)$. Details can be found in the appendix.

Now, we consider the running time of the algorithm. Checking if each vertex in $V_2(f)$ has a private neighbor in $V_0(f)$ can be done in polynomial time. Further, we can test in polynomial time if there exists a $u \in V_1(f)$ with $|N_G(u) \cap V_2(f)| = 1$. If this is the case, then we go through $w \in N_G[u] \setminus V_2(f)$ and run the algorithm on f_w . Clearly $V_0(f_w) \subseteq V_0(f)$ and $V_1(f_w) \subseteq V_1(f)$ and $V_2(f) \cup \{w\} = V_2(f_w)$. Together with $|N_G(u) \cap V_2(f_w)| > 1$, this implies that $|N_G(u) \cap V_2(h)| \neq 1$ will hold for all $h : V \rightarrow \{0, 1, 2\}$ with $f_w \leq h$. Thus, we add only one vertex to $V_2(f)$ per vertex in $V_1(f)$. As we never add a vertex to $V_1(f)$, the recursion tree has at most $|V_1(f)|$ many nodes between the root and a leaf. This also proves the bound on the size of $V_2(h)$ for a solution h returned by the algorithm. As there are at most $|V|$ choices for w , we call the recursive function at most $n^{|V_1(f)|}$ times. Hence, this is an XP algorithm. \square

The proof of the last theorem implies the following corollary.

Corollary 1. *Let $G = (V, E)$ be a graph and $f : V \rightarrow \{0, 1, 2\}$ with $k := \omega(f)$. If there exists a minimal pRdf h on G with $f \leq h$, then there exists a minimal pRdf g with $|V_2(g) \setminus V_2(f)| \leq |V_1(f)|$, $|V_2(g)| \leq k$ and $f \leq g$.*

The very existence of an XP-algorithm with respect to $|V_1(f)|$ is interesting, as the similarly looking parameterized problem $|V_1(f)|$ -EXTENSION ROMAN HITTING FUNCTION is para-NP-hard (see [12]). We also need this result to prove W[1]-membership in the next section, so that we cannot derive XP-membership without the considerations of this section.

The next two lemmas collect properties of a minimal pRdf g that extends a given function $f : V \rightarrow \{0, 1, 2\}$. The first one gives a sufficient and necessary condition for f such that g exists, and the second one upper-bounds $|V_2(g)|$.

Lemma 3. *Let $G = (V, E)$ be a graph and $f : V \rightarrow \{0, 1, 2\}$. There exists a minimal pRdf g on G with $f \leq g$ iff there exists a $V' \subseteq V$ with $|V'| \leq \omega(f)$ and $V_2(f) \subseteq V'$ such that each $v \in V'$ has a private neighbor in $V_0(f) \setminus V'$ and $|N_G(u) \cap V'| \neq 1$ for each $u \in V_1(f) \setminus V'$.*

Proof. Let g be a minimal pRdf on G with $f \leq g$. By Corollary 1, there exists a minimal pRdf g' on G with $f \leq g'$ and $|V_2(g')| \leq k := \omega(f)$. As $f \leq g'$, $V_2(f) \subseteq V' := V_2(g')$, $V_0(g') \subseteq V_0(f) \setminus V'$ and $V_1(f) \setminus V' \subseteq V_1(g')$. Since g' is minimal pRdf, each $v \in V'$ has a neighbor in $V_0(g') \subseteq V_0(f) \setminus V'$. By the definition

of pRdf, this neighbor is private. For $u \in V_1(f) \setminus V' \subseteq V_1(g')$, $|N_G(u) \cap V'| \neq 1$. Therefore, V' satisfies the condition.

Let V' be a set that satisfies the condition. Then $f' : V \rightarrow \{0, 1, 2\}$ with $V_0(f') = V_0(f) \setminus V'$, $V_1(f') = V_1(f) \setminus V'$ and $V_2(f') = V'$ fulfills the conditions of Lemma 2. Hence, there exists a minimal pRdf on G with $f \leq f' \leq g$. \square

Lemma 4. *Let $G = (V, E)$ be a graph and $f : V \rightarrow \{0, 1, 2\}$ a function. For a minimal pRdf on G with $f \leq g$, $|V_2(g)| \leq |V_0(f)|$.*

Proof. As mentioned before, $V_0(g) \subseteq V_0(f)$. Further, $|N_G(v) \cap V_2(g)| = 1$ for each $v \in V_0(g)$, by Theorem 1. Thus, there is a function $\phi : V_0(g) \rightarrow V_2(f)$ mapping a vertex to its unique neighbor in $V_2(f)$. By Theorem 1, ϕ is surjective. Hence, $|V_2(g)| \leq |V_0(g)| \leq |V_0(f)|$. \square

4 W[1]-Completeness for $\omega(f)$ -Ext PRD

We show W[1]-completeness of Ext PRD, parameterized by the weight of the pre-solution. As the reduction is even polynomial, this also proves NP-hardness of the underlying decision problem. In the reduction, we use IRREDUNDANT SET. A set $I \subseteq V$ is called *irredundant* if each vertex in I has a private neighbor.

Problem name: IRREDUNDANT SET

Given: A graph $G = (V, E)$ and $k \in \mathbb{N}$

Question: Is there an irredundant set I on G with $|I| = k$?

In [11] it is shown that IRREDUNDANT SET, parameterized by k , is W[1]-complete.

Theorem 3. Ext PRD is NP-complete and $\omega(f)$ -Ext PRD is W[1]-hard.

Proof. For the NP-membership, we simply guess the value of each vertex and check the conditions of Theorem 1. Let $G = (V, E)$ be a graph and $k \in \mathbb{N}$. Define for each $i \in [k+1]$, $V_i = \{v_i \mid v \in V\}$ and $G' = (V', E')$ with

$$\begin{aligned} V' &:= \{a, b, c, d\} \cup \{u_1, \dots, u_k\} \cup \bigcup_{i=1}^{k+1} V_i, \\ E' &:= \{\{a, b\}, \{c, d\}\} \\ &\quad \cup \{\{a, u_i\}, \{v_i, u_i\}, \{v_i, c\}, \{v_i, w_{k+1}\} \mid v \in V, w \in N_G[v], i \in [k]\}. \end{aligned}$$

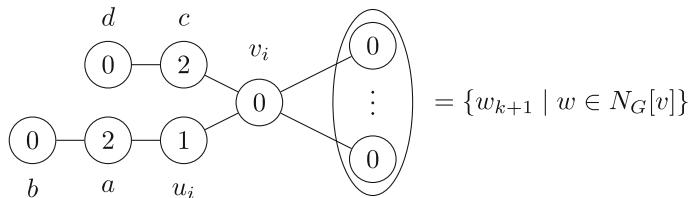


Fig. 2. Construction for Theorem 3, for each $v \in V$ and $i \in [k]$.

We also need $f : V' \rightarrow \{0, 1, 2\}$ with

$$V_0(f) = \{b, d\} \cup \bigcup_{i=1}^{k+1} V_i, \quad V_1(f) = \{u_1, \dots, u_k\} \text{ and } V_2(f) = \{a, c\}.$$

This implies $\omega(f) = k+4$. See Fig. 2. We have to prove that G has an irredundant set iff (G', f) has an extension. For details, see the appendix. \square

In the previous proof, we find $|V_2(f)| = 2$. This implies the following corollary. Notice: the first part of it will be improved in Sect. 5.

Corollary 2. $|V_1(f)|$ -EXT PRD is W[1]-hard; $|V_2(f)|$ -EXT PRD is para-NP-hard.

For W[1]-completeness, we now show W[1]-membership of $\omega(f)$ -EXT PRD by a reduction to SHORT NON-DETERMINISTIC TURING MACHINE COMPUTATION, or STMC, which is a well-known W[1]-complete problem when parameterized by the number of steps, see [6, 7, 10].

Problem name: Steps-STMC

Given: A nondeterministic one-tape Turing machine TM, a word w and $k \in \mathbb{N}$

Parameter: k

Question: Does TM accept w in at most k steps?

Theorem 4. $\omega(f)$ -EXT PRD is W[1]-complete.

Proof (Sketch). By Theorem 3, we only need to prove W[1]-membership. Therefore, let $G = (V, E)$ be a graph and $f : V \rightarrow \{0, 1, 2\}$ be a function with $V_1(f) := \{u^1, \dots, u^\ell\}$ and $V_2(f) = \{v^1, \dots, v^{\ell'}\}$ (where $\ell, \ell' \in \mathbb{N}$ and $\ell + 2\ell' = k$).

The idea of our nondeterministic Turing machine (NTM for short) is to guess at most k new vertices in $V_2(g)$ for a pRdfg with $f \leq g$. After this, we check if the vertices satisfy the conditions of Lemma 3. For the private neighborhood condition, we guess for each vertex which vertex is the private neighbor and check if this vertex is the private neighbor and if the vertex is in $V_0(f) \setminus V_2(g)$. For the second condition, we go for each $u \in V_1(f) \setminus V_2(g)$ through the vertices and count the number of neighbors up to two. The very technical construction of the NTM can be found in the appendix. \square

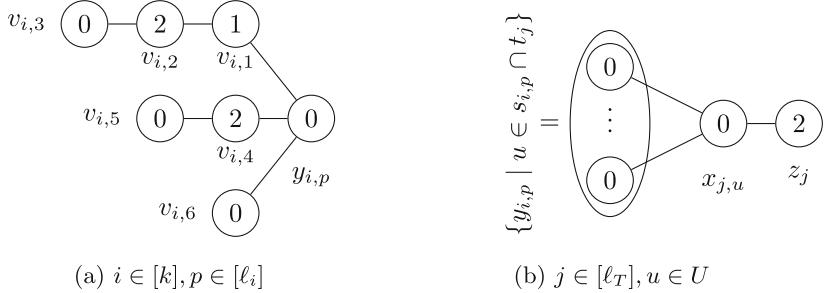
5 W[3]-Completeness for $|V_1(f)|$ -Ext PRD

The main goal of this section is to show that $|V_1(f)|$ -EXT PRD is W[3]-complete. We will use the problem MULTICOLORED INDEPENDENT FAMILY.

Problem name: MULTICOLORED INDEPENDENT FAMILY (MULTINDFAM)

Given: A $(k+1)$ -tuple $(\mathcal{S}_1, \dots, \mathcal{S}_k, \mathcal{T})$ of subsets of 2^U on the common universe U , i.e., $(U, \mathcal{S}_1), \dots, (U, \mathcal{S}_k), (U, \mathcal{T})$ are $k+1$ many simple hypergraphs.

Question: Are there $S_1 \in \mathcal{S}_1, \dots, S_k \in \mathcal{S}_k$ s.t. all $T \in \mathcal{T}$ satisfy $T \not\subseteq \bigcup_{i=1}^k S_i$?

**Fig. 3.** Construction for Theorem 5.

MULTINDFAM looks like a very technical problem, but it is used (with some variations) quite frequently in various applications, linking it (*e.g.*) to questions in database theory [18]. k -MULTINDFAM is W[3]-complete.

Theorem 5. $|V_1(f)|$ -EXT PRD is W[3]-hard.

Proof. Let $(S_1, \dots, S_k, \mathcal{T})$ be a $(k+1)$ -tuple of subsets of 2^U on the common universe U . Define $\ell_i := |S_i|$, $\ell_T := |\mathcal{T}|$, $S_i = \{S_{i,1}, \dots, S_{i,\ell_i}\}$ and $\mathcal{T} = \{T_1, \dots, T_{\ell_T}\}$ for $i \in [k]$. For $i \in [k]$ and $j \in [\ell_T]$, let $V_i := \{v_{i,p} \mid p \in [6]\}$, $Y_i := \{y_{i,p} \mid p \in [\ell_i]\}$, and $X_j := \{x_{j,u} \mid u \in t_j\}$. Define $G = (V, E)$ (also see Fig. 3) with

$$\begin{aligned} V &:= \left(\bigcup_{i=1}^k V_i \cup Y_i \right) \cup \left(\bigcup_{j=1}^{\ell_T} X_j \cup \{z_j\} \right) \\ E &:= \left(\bigcup_{i=1}^k \{\{v_{i,p}, v_{i,p+1}\} \mid p \in \{1, 2, 4\}\} \cup \{\{y_{i,p}, v_{i,q}\} \mid p \in [\ell_i], q \in [1, 4, 6]\} \right. \\ &\quad \left. \cup \{\{y_{i,p}, x_{j,u}\} \mid p \in [\ell_i], j \in [\ell_T], u \in S_{i,p} \cap T_j\} \right) \cup \{\{z_j, x_{j,u}\} \mid j \in [\ell_T], u \in T_j\}. \end{aligned}$$

Furthermore, let $f := \sum_{i=1}^k (\chi_{\{v_{i,1}\}} + 2 \cdot \chi_{\{v_{i,2}, v_{i,4}\}})$. In total, $|V| \leq |U| \cdot \ell_T + 6k + \sum_{i=1}^k \ell_i$, with $|V_1(f)| = k$. The correctness proof of the construction is quite technical and hence omitted. \square

Theorem 6. $|V_1(f)|$ -EXT PRD is W[3]-complete.

Also for membership, we use a reduction to MULTINDFAM. Hence, instead of k -MULTINDFAM, $|V_1(f)|$ -EXT PRD could be seen as a good problem characterizing W[3] and hence providing good intuition for this class.

Proof. Let $G = (V, E)$ be a graph and $f \in \{0, 1, 2\}$. Define $V_{1,0} = \{v \in V_1(f) \mid |N(v) \cap V_2(f)| = 0\}$, $V_{1,1} = \{v \in V_1(f) \mid |N(v) \cap V_2(f)| = 1\}$, $V' := V_{1,0} \cup V_{1,1}$ and $V_{1,2} = \{v \in V_1(f) \mid |N(v) \cap V_2(f)| \geq 2\}$. We want to use this to construct a MULTINDFAM instance which is equivalent to the EXT PRD instance (G, f) .

Let $U = \{x_{u,v} \mid \{v, u\} \in E, u \in V_0(f) \setminus \{v\}\} \cup \{\tau_v \mid v \in V \setminus V_2(f)\} \cup \{\mu_v, \gamma_v \mid v \in V_{1,0}\}$ be the universe. For $v \in V'$ and $u \in N(v) \setminus V_2(f)$, define $S_{u,v} := \{\tau_u\} \cup \{\gamma_w \mid w \in N(u) \cap V_{1,0}\} \cup \{x_{w_1,w_2} \mid w_1 \in V_0(f) \cap N[u], w_2 \in N(w_1) \setminus \{u\}\}$. Set $Z_v := \{\mu_v, \gamma_v\}$ with $v \in V_{1,0}$. We define t_v for $v \in V$ by a case distinction: If $v \in V_2(f)$, $T_v := \{x_{u,v} \mid u \in V_0(f) \cap (N(v) \setminus [V_2(f) \setminus \{v\}])\}$. For $v \notin V_2(f)$, define $T_v := \{\tau_v\} \cup \{x_{u,v} \mid u \in V_0(f) \cap (N(v) \setminus N[V_2(f)])\}$. Let $\mathcal{T} := \{Z_v \mid v \in V_{0,1}\} \cup \{T_v \mid v \in V\}$. For $v \in V_{1,0}$, define $\mathcal{S}_v := \{\{\mu_v\}, S_{v,v}\} \cup \{S_{u,v} \cup S_{w,v} \mid u, w \in N(v)\}$, and for $v \in V_{1,1}$, define $\mathcal{S}_v := \{S_{u,v} \mid u \in N[v] \setminus V_2(f)\}$.

We will shortly explain the idea behind the elements. In order to do this, let $g : V \rightarrow \{0, 1, 2\}$ be a minimal pRdf with $f \leq g$ and $S_v \in \mathcal{S}_v$ (for $v \in V'$) be the corresponding solution for the MULTINDFAM instance.

- If there is a $v \in V'$ with $S_v = S_{u,v}$, then $g(u) = 2$.
- If there is a $v \in V_{1,0}$ with $S_v = \{\mu_v\}$, then $g(v) = 1$ and $N(v) \cap V_2(g) = \emptyset$.
- If there is a $v \in V_{1,0}$ with $\gamma_v \in \bigcup_{u \in V'} S_u$, then $|N(v) \cap V_2(g)| \geq 1$.
- If there is a $v \in V \setminus V_2(f)$ with $\tau_v \in \bigcup_{u \in V'} S_u$, then $g(v) = 2$.
- If there are $v \in V, w \in V_0(f) \setminus \{v\}$ such that $x_{w,v} \in \bigcup_{u \in V'} S_u$, then w cannot be the private neighbor of v .

It is easy to see that $|\mathcal{T}| \leq 2|V|$, $|U| \leq (|V| + 3)|V|$ and $|\mathcal{S}_v| = |V|^2 + 2$.

Claim. \mathcal{S}_v for $v \in V'$ together with \mathcal{T} is a yes-instance of MULTINDFAM iff(G, f) is a yes-instance of EXT PRD.

The very technical proof of the claim is omitted due to its length. Since $|V'| \leq |V_1(f)|$, this is an FPT-reduction. \square

Remark 1. Schirneck proved in [18, Lemma 4.15] that a relatively naïve brute-force XP-algorithm for MULTINDFAM cannot be significantly improved unless the so-called k -Orthogonal Vectors conjecture fails (cf. [14]). Looking at the proof of the cited lemma and at our proof of Theorem 5, one can see that we can also rule out $\mathcal{O}(n^{|V_1(f)|-\varepsilon})$ algorithms for solving $|V_1(f)|$ -EXT PRD instances of order n , for any $\varepsilon > 0$, unless the k -Orthogonal Vectors conjecture fails. Hence, our problem also allows us to speak about certain aspects of fine-grained complexity. We are currently working on a variant of Algorithm 1 that runs in time $\mathcal{O}(n^{\max\{|V_1(f)|, c\}})$, for some constant c .

6 W[2]-Completeness for $|V_0(f)|$ -Ext PRD

The main goal of this section is to prove W[2]-completeness of this parameterized problem. For space reasons, most proofs had to be omitted. We start with some observations. From Lemma 4, we can provide a simple XP algorithm with the parameterization $|V_0(f)|$. We know that if there exists a minimal pRdf g with $f \leq g$ then $|V_2(g)| \leq |V_0(f)|$. Therefore, we guess the at most $|V_0(f)|$ many vertices in $V_2(g)$ and use Lemma 2 on the new function.

Nonetheless, Algorithm 1 is also an XP algorithm with respect to the parameterization $|V_0(f)|$. The running time result follows as on one path of the branching tree we can only add $|V_0(f)|$ vertices to $V_2(g)$. Otherwise, not each vertex in $V_2(f)$ will have a private neighbor in $V_0(f)$; also cf. the argument of Lemma 4. Therefore, the depth of the branching tree is at most $|V_0(f)|$.

Lemma 5. $|V_0(f) \cup V_1(f)|$ -EXT PRD and $\omega(2-f)$ -EXT PRD are in FPT. Both parameterizations admit a linear kernel.

The Kernelization Algorithm is Based on Inequality Chains and is Rather Simple. For W[2]-membership, we use a reduction using the problem SHORT BLIND NON-DETERMINISTIC MULTI-TAPE TURING MACHINE COMPUTATION which was introduced by Cattan  o and Perdrix in [6]. In that paper, they have also shown W[2]-completeness of this Turing machine problem. The difference between a *blind* multi-tape nondeterministic and a normal multi-tape nondeterministic Turing machine is that the transitions can be independent of the symbols that are in the cells under the current head positions, *i.e.*, the Turing machine may, but need not read the cell contents, and in this sense, it may be blind.

Theorem 7. $|V_0(f)|$ -EXT PRD $\in W[2]$.

The proof combines the simulation of Theorem 4 with the known reduction from k -DOMINATING SET to Steps-STMC; it uses $|V_1(f)|+1$ many tapes and employs blindness in particular at the very end, when it accepts if the blank symbol is under each tape head. It uses Lemma 3 for the correctness proof.

For proving W[2]-hardness, we use Index-MULTICOLORED DOMINATING SET (MULTDS) which is known to be W[2]-complete, see [16]. We chose to call the parameter *index* here as it refers to the number of classes of the partition.

Problem name: Index-MULTICOLORED DOMINATING SET (MULTDS)

Given: A graph $G = (V, E)$, $k \in \mathbb{N}$ and a partition W_1, \dots, W_k of V

Parameter: k

Question: Is there a dominating set $D \subseteq V$ with $|W_i \cap D| = 1$ for each $i \in [k]$?

Theorem 8. $|V_0(f)|$ -EXT PRD is W[2]-complete even on bipartite graphs.

7 Conclusion

In this paper, we provided some parameterized completeness results with respect to the W-hierarchy. $\omega(f)$ -EXT PRD is W[1]-complete while $|V_0(f)|$ -EXT PRD and $|V_0(f) \cup V_2(f)|$ -EXT PRD are W[2]-complete. For $|V_1(f)|$ -EXT PRD, we showed W[3]-completeness. Furthermore, we found a linear kernel for $(|V| - \omega(f))$ -EXT PRD and a para-NP-hardness proof for the parameter $|V_2(f)|$. The amount of different complexity results is significant and makes EXT PRD a good candidate for exploring several levels of the W-hierarchy with a single problem. This implies the question if this richness of complexity also prevails for extension

versions of other Roman domination variants, like Roman- $\{2\}$ -domination (also known as Italian domination) [8] or double Roman domination [1,3,4].

Starting out with this single problem into the world of parameterized complexity has also some further advantages: (a) It is clear from the very beginning that selecting parameters is an art, and the parameter is usually not coming with the optimization problem. In particular, the parameter need not be a bound on the solution size. (b) Parameterized problems are introduced that can be relevant for building reductions. Namely, the given pre-solution gives possibilities to “annotate” the instance, similar to the classes that make MULTICOLORED INDEPENDENT SET one of the favorite problems to use for W[1]-hardness proofs.

References

1. Abdollahzadeh Ahangar, H., Chellali, M., Sheikholeslami, S.M.: On the double Roman domination in graphs. *Discret. Appl. Math.* **232**, 1–7 (2017)
2. Abu-Khzam, F.N., Fernau, H., Mann, K.: Minimal Roman dominating functions: extensions and enumeration. *Algorithmica* **86**, 1862–1887 (2024)
3. Banerjee, S., Henning, M.A., Pradhan, D.: Algorithmic results on double Roman domination in graphs. *J. Comb. Optim.* **39**(1), 90–114 (2020)
4. Beeler, R.A., Haynes, T.W., Hedetniemi, S.T.: Double Roman domination. *Discret. Appl. Math.* **211**, 23–29 (2016)
5. Casel, K., Fernau, H., Ghadikolaei, M.K., Monnot, J., Sikora, F.: On the complexity of solution extension of optimization problems. *Theor. Comput. Sci.* **904**, 48–65 (2022)
6. Cattanéo, D., Perdrix, S.: The parameterized complexity of domination-type problems and application to linear codes. In: Gopal, T.V., Agrawal, M., Li, A., Cooper, S.B. (eds.) TAMC 2014. LNCS, vol. 8402, pp. 86–103. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06089-7_7
7. Cesati, M.: The Turing way to parameterized complexity. *J. Comput. Syst. Sci.* **67**, 654–685 (2003)
8. Chellali, M., Haynes, T.W., Hedetniemi, S.T., McRae, A.A.: Roman 2-domination. *Discret. Appl. Math.* **204**, 22–28 (2016)
9. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Cham (1999)
10. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science, Springer, Cham (2013)
11. Downey, R.G., Fellows, M.R., Raman, V.: The complexity of irredundant set parameterized by size. *Discret. Appl. Math.* **100**, 155–167 (2000)
12. Fernau, H., Mann, K.: Hitting the Romans. Technical report, abs/2302.11417, arXiv, Cornell University (2023). <https://doi.org/10.48550/arXiv.2302.11417>
13. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Cham (2006)
14. Gao, J., Impagliazzo, R., Kolokolova, A., Williams, R.: Completeness for first-order properties on sparse structures with algorithmic applications. *ACM Trans. Algorithms* **15**(2), 23:1–23:35 (2018)
15. Henning, M.A., Klostermeyer, W.F., MacGillivray, G.: Perfect Roman domination in trees. *Discret. Appl. Math.* **236**, 235–245 (2018)
16. Lackner, M., Pfandler, A.: Fixed-parameter algorithms for finding minimal models. In: Brewka, G., Eiter, T., McIlraith, S.A. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR. AAAI Press (2012)

17. Mann, K., Fernau, H.: Perfect Roman domination: aspects of enumeration and parameterization. In: Rescigno, A.A., Vaccaro, U. (eds.) IWOCA 2024. LNCS, vol. 14764, pp. 354–368. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-63021-7_27
18. Schirneck, M.: Enumeration algorithms in data profiling. Ph.D. thesis, University of Potsdam, Germany (2022). <https://publishup.uni-potsdam.de/frontdoor/index/index/docId/55672>

A Sub-quadratic Algorithm for the Minsum One Sink Location Problem on Balanced Binary Tree Networks

Jannatul Maowa^(✉) and Robert Benkoczi[✉]

Department of Mathematics and Computer Science, University of Lethbridge,
Lethbridge, AB, Canada

{jannatul.maowa,robert.benkoczi}@uleth.ca

Abstract. We focus on the problem of locating one sink on balanced binary tree networks with uniform edge capacities, all while minimizing the total evacuation time for all evacuees (*minsum criterion*). The challenge with sink location problems is modeling congestion, which determines evacuation time during a major disaster. Sub-quadratic algorithms to locate one sink that minimizes the minsum objective exist for path and cycle networks. Designing a sub-quadratic algorithm to locate one sink on a tree network has been an open problem for about ten years. Our algorithm has a time complexity of $O(n \log^2 n)$, where n is the number of vertices in the network. We achieve this result by introducing two new ideas. First, we define the concept of *modified clusters*, which captures the change in the congestion information computed during the pre-processing phase, to account for new evacuation paths. Second, we devise a simple and efficient method to determine the costs associated with every potential sink, using the partial information gathered from the modified clusters that are accessible at several network nodes.

Keywords: Evacuation planning · Dynamic flow in networks · Minsum · Sink location problem · Balanced binary tree

1 Introduction

Sink location problems generalize facility location problems and are used to model large-scale evacuations. The evacuation network is modeled by a so-called *dynamic network*, a term introduced by Ford and Fulkerson [4] in the context of dynamic network flows. Positive weights are assigned to the network's vertices and represent the number of evacuees located at the vertex. The edges have a positive length, which determines the travel time through the edge, and a capacity, which limits the number of evacuees who can enter the edge in the unit of time. Once the order for evacuation is given, all evacuees travel towards one of possibly several *sink nodes* in the network. The sink nodes represent shelters,

locations where the evacuees are considered to be safe from the calamity. The evacuation time for an evacuee is between the moment the evacuation order is given and the moment the evacuee arrives at the sink.

In this paper, we propose an algorithm to compute the location of one sink node in a balanced binary tree network with uniform edge capacities to minimize the total evacuation time, also known as the *minsum* objective. Our result has an important theoretical value as it is the first algorithm to break the quadratic time complexity barrier for the minsum objective for a special class of tree networks. The only sub-quadratic algorithms for minsum currently known are for path networks [5], which can be viewed as trees with maximum diameter, in some sense. By focusing on the special case of trees with a small diameter, we show that we can tackle networks with a structure significantly different from path networks. Solving the problem efficiently for arbitrary trees is challenging, but we hope that the techniques introduced here may be useful in more general settings as well. Our result may be useful in practice, too. Currently, very little is known about computing the sink location problem in arbitrary networks approximately. One approach is to enumerate different types of spanning trees and run algorithms for tree networks as a sub-routine, therefore efficient algorithms for the problem on tree networks become essential. We mention that our algorithm is not excessively complicated and may be implemented in practice.

The challenge of optimizing the minsum objective lies in its lack of structure. Algorithms that minimize the evacuation completion time (minmax criterion or the evacuation time for the last evacuee) have been more successful because they exploit an essential property of the minmax objective, unimodality. This property is not satisfied by the minsum objective and, as a consequence, algorithms for minsum problems are forced to compute the cost for the objective function for every candidate sink node and select the optimum cost.

Despite this challenge, we show that the cost for the objective function can be efficiently computed using two new ideas. First, we apply a standard pre-computation step in the rooted tree network that calculates the sets of congested evacuees from every sub-tree of the network as these evacuees travel towards the root of the sub-tree. In the literature, these subsets of evacuees are called *clusters*. We can compute, for each of the pre-computed clusters, a so-called *modified cluster* that accounts for the possibly additional congestion experienced by the evacuees on their downward path. Our second contribution is to design an effective cost accounting scheme that obtains the total cost of the complement tree without using any additional data structures or pointers other than the standard fractional cascading technique of Chazelle and Guibas [2,3].

This is the first non-trivial result on the minsum problem for tree networks, and we hope that our techniques will facilitate the development of further results for this challenging problem.

2 Literature Review

The most widespread evacuation model considered in the literature is the so-called continuous model, where evacuees are infinitesimal quantities represented

globally by the weight assigned to a node. In addition, the evacuation flow is confluent, which means that at any given node that is not a sink, there is exactly one edge that carries the flow out of the node. Confluent flow models are more easily implemented in the field during large-scale evacuations. Our problem adopts these constraints, too.

Many results are known for sink location problems on the path and tree networks with the minmax objective. The excellent review paper by Higashikawa and Katoh [6] provides an overview of these results. For the minsum problem, an $O(n)$ algorithm for the 1-sink location problem on path networks with uniform capacity exists [5]. Benkoczi et al. [1] provide $O(kn \log^3 n)$ and $O(kn \log^4 n)$ time algorithms for uniform and arbitrary capacities, respectively. This result was improved by Highashikawa et al. [7] to $O(\min\{kn \log^3 n, n2^{O(\sqrt{\log k \log \log n})} \log^3 n\})$ and $O(\min\{kn \log^2 n, n2^{O(\sqrt{\log k \log \log n})} \log^2 n\})$ for arbitrary and uniform capacities, respectively. The 1-sink location problem for trees with uniform capacity can be solved trivially in $O(n^2)$ time using a technique called *flattening* that transforms the tree and a fixed sink node into a path network. This paper presents an $O(n \log^2 n)$ time algorithm for the minsum one sink location problem on balanced binary trees with uniform capacities.

3 Problem Definition

Let $T = (V, E)$ be a tree with the set of vertices V and the set of edges E . Let $N = (T, c, \tau, w, l)$ be a dynamic flow network [4] with the underlying undirected graph being the tree T , where $c \in \mathbb{R}^+$ is the (uniform) edge capacity representing an upper bound on the flow traversing an edge in the unit of time, $\tau \in \mathbb{R}^+$ is the time it takes evacuees to travel one unit of distance and it is the same along all edges and for all evacuees, $w : V \rightarrow \mathbb{R}^+$ is the supply function representing the total number of evacuees situated at a vertex, and $l : E \rightarrow \mathbb{R}^+$ is the function associating a non-negative length to each edge. The model assumes that all evacuees begin evacuation at the same moment.

Given an edge (u, v) with length l and w evacuees situated on vertex u , it takes $\frac{w}{c}$ units of time until all w evacuees enter the edge of capacity c . Once on the edge just outside node u , it takes τl units of time until the evacuees traverse the edge to reach node v .

A standard representation used in facility location is to consider each edge (u, v) as a continuum of points so that we can refer to a point x on edge (u, v) and extend the distance from x to u and v in a natural way [5]. If we consider some reference point x on the network (either on a vertex or on an edge), we can define the rate of the evacuation flow through that point as a function of time. For the uniform capacity sink model, the value of the rate is either c or 0.

Definition 1. *Given a reference point x on the network, a maximal, contiguous interval of time for which the evacuation flow is non-zero at x , is called a cluster.*

The maximal, contiguous interval of time for which the evacuation rate is non-zero and constant is called a section. We call the collection of clusters, separated by gaps in evacuation flow, and ordered by time, a cluster sequence.

For sink location problems with uniform capacity, a cluster is also a section, and we use these terms interchangeably in this paper.

Minsum 1-sink problem: Given a dynamic flow network $N = (T, c, \tau, w, l)$, find a sink vertex v from tree T so that the total evacuation time of all evacuees to sink v is minimized.

We consider network T to be a rooted balanced binary tree, whose height is $O(\log n)$, where n represents the number of vertices. Given an internal node v of T , we arbitrarily label one of its children nodes as the left child and the other, if it exists, as the right child. We define point v^{-L} to be the point on the edge between v and the left child of v , so that the distance between v and v^{-L} is 0. Similarly, we define point v^{-R} to be the point on the edge between v and the right child of v , so that the distance between v and v^{-R} is 0. We define the point v^+ to be the point on the edge between v and the parent of v so that the distance between v and v^+ is 0. Therefore, if w evacuees are located at vertex v and they evacuate at point v^+ , for example, the last evacuee will reach v^+ after $\frac{w}{c}$ units of time.

Cost of a Cluster/Section: A useful concept connects the cost for the location of a sink node and the cluster sequence whose reference point is at the sink node. For a cluster that starts at time τ_1 and ends at time τ_2 , the weight w of the cluster equals $c(\tau_2 - \tau_1)$. The contribution of this cluster to the total cost is given by the expression $Z = \tau_1 w + \frac{w^2}{2c}$, and the total cost of the solution is given by the total cost of all clusters in the cluster sequence [5].

4 An Efficient Algorithm for Balanced Binary Trees

Our algorithm computes the cost for each vertex considered a sink and selects the vertex with the smallest cost as the solution. We compute the costs efficiently, first by pre-computing a list of cluster sequences associated with every node in the tree. The cluster sequence at some node v concerns the evacuation of all evacuees in sub-tree T_v as they exit node v towards the root. To compute the cost when v is the sink, we use the precomputed cluster sequences stored at the nodes from v to the root of the tree to account for the interactions between these evacuees as they move towards sink v . Instead of updating the cluster sequences to reflect this interaction, we compute a list of *differences* for each node on the path from v to the root, and we recover the updated cost by collecting the updates from every node on the path from v to the root.

To pre-compute the list L_v of cluster sequences stored at a vertex v , we use a standard bottom-up linear merging algorithm. Let the lists L_{u_1} and L_{u_2} represent cluster sections of two sibling nodes u_1 and u_2 computed for the reference point u_1^+ and u_2^+ respectively. Let v be the parent of u_1 and u_2 and L_v the

cluster sequence for sub-tree T_v and reference v^+ . Here, l_{u_1} and l_{u_2} represent the distance from u_1 and u_2 to v respectively. For each section $S_{u_1} \in L_{u_1}$, the start and end time of the section is denoted by $S_{u_1}^-$ and $S_{u_1}^+$. When computing the clusters for parent node v for which the reference point is v^+ , we need first to translate each section in L_{u_1} (resp. L_{u_2}), by adding τl_{u_1} (resp. τl_{u_2}) to each section. For each section $S_{u_1} \in L_{u_1}$, the translated start and end time will be $S_{u_1}^- + \tau l_{u_1}$ and $S_{u_1}^+ + \tau l_{u_1}$.

When the evacuees from the children u_1 and u_2 reach parent node v and continue evacuating towards reference v^+ , they might interact with each other and congestion can occur. For example, consider two sections S_{u_1} and S_{u_2} computed for the same reference point, where $S_{u_1}^- \leq S_{u_2}^-$ and $S_{u_1} \cap S_{u_2} \neq \emptyset$. In this case, we have congestion, and then two sections merge into a single section S . The parameters for the new section $S = S_{u_1} + S_{u_2}$ are defined as follows,

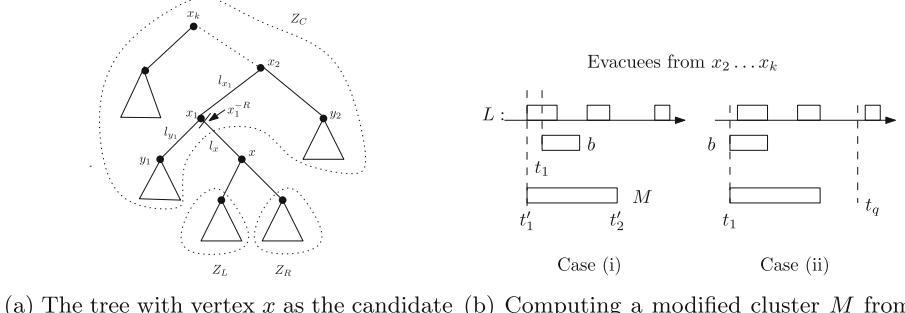
$$\begin{aligned} S^- &= S_{u_1}^- \\ S^+ &= S_{u_1}^- + (S_{u_1}^+ - S_{u_1}^-) + (S_{u_2}^+ - S_{u_2}^-) \end{aligned}$$

We can use standard arrays to store the cluster sequences L_v , and the entire merging process will take $O(n \log n)$ time in total.

Our challenge is to obtain the cost of component Z_C (Fig. 1(a)). If x is the sink node and x_1, \dots, x_k are the nodes on the path from x to the root x_k (k is $O(\log n)$), then the evacuees from $T_{y_{k-1}}$ may interact with the evacuees from T_{y_k} as they move towards sink node x . Rather than merging the cluster sequences L_{y_k} and $L_{y_{k-1}}$, we compute for each cluster in $L_{y_{k-1}}$ its final configuration that accounts for any possible interactions with x_{k-1} , x_k , and L_{y_k} . We call this updated cluster, *the modified cluster*. We update every cluster from $L_{y_{k-1}}$ in this fashion and we obtain a new list, which we call *the modified cluster sequence*.

We show the following: we can compute the modified cluster sequences in total time that is sub-quadratic in n ; we can re-use the information contained by modified cluster sequences to compute the cost for several sink nodes. However, each modified cluster sequence contains only partial information about the component Z_C . Information about the time spanning the gaps between modified clusters is not available, and we need to query other modified cluster sequences stored at vertices on the path towards the root of the tree, to find out. Moreover, since modified clusters contain information only about the evacuees arriving from the root node, this information could be invalidated as the evacuees travel down in the tree. Newer information will be contained by cluster sequences down in the tree, in this case. We devised an efficient cost accounting mechanism that takes care of the relevant information only.

Computing the Modified Clusters. Given a list of clusters L and a scalar t representing time, we denote by $L + t$ the list L whose clusters have been translated by the time value t . If v is a vertex, we abuse the notation $L \cup v$ to denote the list of clusters L which is merged with the cluster generated by the evacuees at node v .



(a) The tree with vertex x as the candidate sink
(b) Computing a modified cluster M from a base cluster section b at node x_1

Fig. 1. A balanced binary tree with notation details and a modified sequence

Consider Fig. 1(a) where x is the sink node and x_1, \dots, x_k are nodes on the path from x to the root. The modified cluster sequence stored at vertex x_1 which is used to compute the cost of sink x contains the clusters from L_{y_1} , modified to account for the interaction with x_1 and all the other evacuees coming from x_2, \dots, x_k . We denote this list $M_L(x_1)$ (a similar list denoted $M_R(x_1)$ will be stored at x_1 and will account for the interaction of L_x with the evacuees from x_1 and all the other evacuees coming from x_2, \dots, x_k).

We first merge the cluster corresponding to x_1 with the cluster sequence $L_{y_1} + \tau l_{y_1}$ to obtain a new cluster sequence denoted L'_{y_1} . We denote the list of cluster sections L'_{y_1} by B . This is easy and can be done in amortized constant time because the evacuees from x_1 arrive before any evacuees from L_{y_1} . Let $b \in B$ be a cluster from cluster sequence L'_{y_1} . We call this cluster, the *base cluster*. We want to determine the modified cluster M , that contains the evacuees from B and any other evacuees coming from x_2, \dots, x_k who have possibly collided with B and are now part of M . This computation can be completed in poly-log of n time, using binary search in two steps.

Consider Fig. 1(b) where $b \in B$ is the base cluster and the cluster sequence on top represents the evacuees arriving from x_2, \dots, x_k denoted by list L . Let t_1 the start of b computed in reference to x_1^{-R} (see Fig. 1(a)). We want to determine the start t'_1 and end t'_2 of modified cluster M .

Step 1 Find the beginning t'_1 of modified cluster M . We use binary search in L with target t_1 to determine if t_1 intersects a cluster from L (either case (i)) or (case (ii) in Fig. 1(b)). In case (i), t'_1 is determined by the start of the intersected cluster. In case (ii), $t'_1 = t_1$.

Step 2 We can determine t'_2 using binary search in L with target a query time value t_q . If we pre-compute list L by storing prefix sums of cluster weights, we can determine the total weight of clusters between times t'_1 and t_q in L . Let W be this weight. Then, we can determine if $t'_2 \leq t_q$ or not using the following comparison.

- If $W + b \leq (t_q - t'_1)c$, then there must be gaps between t'_1 and t_q and therefore $t'_2 \leq t_q$.

- Otherwise, $t'_2 > t_q$. This comparison can guide the binary search procedure to find t'_2 .

There are still several observations that need to be addressed:

- Obs. 1 The cluster sequence L used in the search procedure above, does not exist. Computing L is what we try to avoid. But we can apply the procedure to the modified cluster sequence stored at x_2 and appropriately translated for the reference point x_1^{-R} . Since the information in the modified cluster sequence is incomplete, we must repeat the search for the modified cluster sequence stored at x_3, \dots, x_k .
- Obs. 2 The weight query in the modified cluster sequence at nodes x_3, \dots, x_k may return weights already included in the modified list stored at x_2 and other nodes closer to x . We can avoid duplication of weights if we compute the prefix sum not of the full weight of each modified cluster but only of the weight of evacuees originating from the base clusters. Thus, the prefix sum of the modified cluster sequence stored at some node x_i will add only the weight corresponding to evacuees from $L_{y_i} \cup x_i$.
- Obs. 3 The modified cluster M may collide with the next cluster from the base cluster sequence. This situation is easy to detect, and, if it happens, we can continue the binary search process for the end of the modified cluster, t'_2 , using the weight of the next base cluster added to the value of b .
- Obs. 4 It is unclear how we can choose the values for the target time t_q for the binary search procedure. We can avoid this complication using the fractional cascading technique of [2,3]. This technique allows us to synchronize the k modified cluster sequences stored at x_1, \dots, x_k by time. Thus, we can replace the $O(k)$ binary searches using time as a target with a single binary search routine using indices, followed by look-ups in the remaining modified cluster sequences.

The computation of modified clusters is a top-down approach. We will start computing the modified cluster from the root, which is $M_L(x_k)$ (resp. $M_R(x_k)$) for the reference point x_k^{-R} (resp. x_k^{-L}). We then proceed with the children of x_k, x_{k-1}, \dots , etc. Assume that we want to compute the modified cluster sequence for the node x_1 and x is the sink (see Fig. 1(a)). For the reference point x_1^{-R} (resp. x_1^{-L}), the modified cluster sequence is denoted by $M_L(x_1)$ (resp. $M_R(x_1)$). To compute $M_L(x_1)$ we need to start looking up the modified cluster sequence $M_L(x_2)$ or $M_R(x_2)$ based on the relation of x_1 and x_2 . If x_1 is the left child (resp. right child) then we need to look up in $M_R(x_2)$ (resp. $M_L(x_2)$). Then look up modified cluster sequences for the node on the path x_3, \dots, x_k will continue in the same way. Therefore, we simplify the notation $M_L(x_i)$ or $M_R(x_i)$ by $M(x_i)$ in the algorithms. We formalize the computations in Algorithms 1 and 2.

The list of modified cluster sections for the node x_1 can be computed using Algorithm 1 where the first parameter B represents the list of base cluster sections and the second a list of modified cluster sections for the nodes x_2, \dots, x_k .

Algorithm 1. Computing the list of Modified Cluster Sections(x_1)

Input : A list of base sections B , list of modified cluster sections $M(x_2), \dots, M(x_k)$ for the reference point x_1^{-R}

Output : A list of modified cluster sections M_L for the node x_1

```

1: procedure MODSEQ( $B, M(x_2), \dots, M(x_k)$ )
2:   while  $B$  is not empty do
3:      $b \leftarrow \text{dequeue}(B)$ 
4:      $t_1 \leftarrow \text{start time of } b$ 
5:      $\triangleright \text{find the start time of the modified section and store in variable startTime} \triangleleft$ 
6:      $S \leftarrow \text{findInterval}(t_1, M(x_2), \dots, M(x_k))$ 
7:      $\text{startTime} \leftarrow \min(\text{start time of } S, t_1)$ 
8:      $\triangleright \text{See Obs. 3 in the page 8 for the reason of maintaining variable}$ 
       $\text{baseWeight} \triangleleft$ 
9:      $\text{baseWeight} \leftarrow \text{weight of base section } b$ 
10:     $\triangleright \text{find end time of the modified section} \triangleleft$ 
11:     $M \leftarrow \text{findEndTime}(\text{startTime}, \text{baseWeight}, b, M(x_2), \dots, M(x_k))$ 
12:    while  $B$  is not empty and the start time of  $\text{Top}(B) \leq$  the end time of  $M$  do
13:       $b \leftarrow \text{dequeue}(B)$ 
14:       $\text{baseWeight} \leftarrow \text{baseWeight} + \text{weight of } b$ 
15:       $M \leftarrow \text{findEndTime}(\text{startTime}, \text{baseWeight}, b, M(x_2), \dots, M(x_k))$ 
16:       $M_L \leftarrow \text{enqueue}(M)$ 
17:   return  $M_L$ 

```

Algorithm 2. Computing a Modified Cluster Section

```

1: procedure FINDENDTIME( $\text{startTime}, \text{baseWeight}, b, M(x_2), \dots, M(x_k)$ )
2:    $\triangleright \text{start time of } M \text{ denoted by } t'_1, \text{ for details see Step 1 on the page 7} \triangleleft$ 
3:    $t'_1 \leftarrow \text{startTime}$ 
4:    $S \leftarrow \text{findInterval}(\text{startTime}, M(x_2), \dots, M(x_k))$ 
5:    $i \leftarrow \text{index}(S)$ 
6:    $left \leftarrow i$ 
7:    $right \leftarrow \text{Last index of the list } M(x_2)$ 
8:    $t_1 \leftarrow \text{start time of } S$ 
9:   while  $left \leq right$  do
10:     $j \leftarrow (left + right)/2$ 
11:     $\triangleright \text{See Obs. 4 in the page 8} \triangleleft$ 
12:    Do binary search with fractional cascading with the index  $i$  and  $j$  in
         $M(x_2), \dots, M(x_k)$  to accumulate weight  $w$ 
13:     $t_2 \leftarrow \text{end time of } j^{\text{th}} \text{ section}$ 
14:     $\triangleright \text{end time of } M \text{ denoted by } t'_2, \text{ for details see Step 2 on the page 7} \triangleleft$ 
15:     $t'_2 \leftarrow w + \text{baseWeight}$ 
16:    if  $t'_2 < (t_2 - t_1)c$  then
17:       $right \leftarrow j - 1$ 
18:    else
19:       $left \leftarrow j + 1$ 
20:   return  $M$ 

```

To compute the list of modified cluster sections $M_L(x_1)$, at node x_1 for the reference point x_1^{-R} as shown in Fig. 1(a), the algorithm will start searching from the modified cluster sections at node x_2 to the root x_k .

For each base section, $b \in B$, the Algorithm 1 will find the modified section M by using the Algorithm 2. In Algorithms 1, the procedure *findInterval* in line 6 will return a section S from the modified cluster sections on the node x_2, \dots, x_k for the time t_1 as explained in Step 1 on the page 7. The procedure *findEndTime* in line 11 will return the modified section M . The start time of M is *startTime* but the end time of M will be decided by the Algorithm 2.

The returned modified section M will be compared with the next base section. If M interacts with the next base section, then both sections will join together and the weight of the next base section will be updated to *baseWeight* which is mentioned in line 14, and repeat the step until base cluster sequences B are not empty. Otherwise, M will be enqueued to the list of the modified clusters M_L and repeat all the steps of Algorithm 1 again until base cluster sequences B are not empty.

We design Algorithm 2, to compute the end time of a modified cluster section, which is explained in Step 2 in the page 7. Fractional cascading is used by Algorithm 2 in the line 12. It is a technique used to expedite a series of binary searches in a series of related data structures for the same value, as introduced in two papers [2, 3] by Chazelle and Guibas in 1986. The total time for a query for fractional cascading is $k + \log n$, where k is the number list to search.

Computing the Cost from the Modified Cluster Sequences. Once we compute the modified cluster sequences for every node with Algorithms 1 and 2, we can compute the cost of component Z_C . For any cluster section with weight w and start time τ_1 , the cost Z can be computed by

$$Z = \frac{w^2}{2c} + \tau_1 w$$

We compute the two terms for cost, $z_1 = \frac{w^2}{2c}$ and $z_2 = \tau_1 w$, separately because the techniques we use to avoid over-counting are different for these terms. The cost component z_1 involves weight and capacity. We can maintain the sum of the term z_1 for each of the modified cluster sequences we compute. However, some modified clusters already contain the evacuees from modified clusters stored at ancestor nodes, thus the total of the z_1 terms would over-estimate the cost. Theorem 1 shows how we avoid this over-estimation.

For clarity, we conceptually represent the relationship between clusters in the modified cluster sequence stored at vertices along the path between x_1 and the root x_k (Fig. 1(a)). Consider the forest representation of all searches in Fig. 2 where each node represents a section in modified cluster sequences and each section is assigned a cost of $Z(s)$. The square nodes represent merged sections of circle nodes, which are identified by the Algorithm 2. The diagram shows a rooted forest where the edge between a parent node s and a child node s' indicates that modified cluster s contains evacuees from s' . Thus, the cost term $Z(s')$ for

cluster s' should not be added to the total cost as it is already included in the cost of $Z(s)$. We can maintain another cost named *actual cost* A by summing up only the cost of square sections for each modified cluster sequence to avoid adding the section's cost twice to the total cost. Level 0 indicates clusters from the modified cluster sequence at vertex x_1 ; Level 1 depicts the modified cluster sequence stored at x_2 , etc.

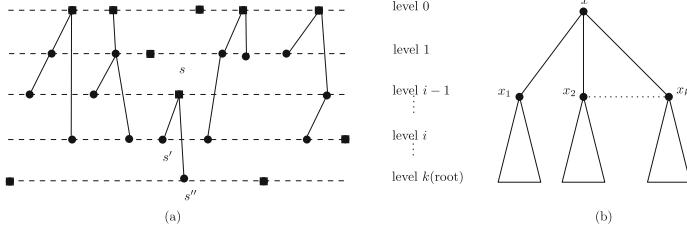


Fig. 2. (a) Illustration of how sections are modified and (b) Tree representation of a modified cluster section

Figure 2(b) depicts one tree from the rooted forest in sub-figure (a) where x is the root and x_1, \dots, x_k are the children of x . Let $Z(x)$ be a value we can associate with any node x in the tree. We want to be able to compute $Z(x)$ only for the root of the tree, but without traversing the tree. Our algorithm can add terms for all clusters in a modified cluster sequence but does not maintain a tree structure such as one from Fig. 2(b). We prove the following.

Theorem 1. *Let x be the root of a tree T_x and x_1, \dots, x_k its children. Let $Z(v)$ be a cost term associated with every node v of T_x . For every node v , we define an actual cost $A(v)$ as follows, where $A(T_v) = \sum_{u \in T_v} A(u)$.*

$$A(v) = \begin{cases} Z(v), & \text{if } v \text{ is a leaf} \\ Z(v) - A(T_{v_1}) - A(T_{v_2}) - \dots - A(T_{v_l}), & \text{if } v \text{ is internal node} \\ & \text{with children } v_1, \dots, v_l \end{cases}$$

Then,

$$A(T_x) = \sum_{v \in T_x} A(v) = Z(x)$$

Proof. For any tree with a height of 0, there exists an isolated vertex x . Therefore, the total cost and actual cost will be equal, i.e., $A(x) = Z(x)$ which holds the claim of the theorem.

Now, if the claim is true for any tree with height $\leq h$, then it must be true for any tree with height $\leq h + 1$.

Assume a rooted tree T_x has a height of $h + 1$ and x is a root. The tree T_x has k children x_i where $i = 1, 2, \dots, k - 1, k$ [see Fig. 2]. As the height of T_x is

$h+1$, the height of T_{x_i} is $\leq h$. At node x , the actual cost of the modified section can now be calculated by

$$\begin{aligned} A(x) &= Z(x) - A(T_{x_1}) - A(T_{x_2}) - \cdots - A(T_{x_k}) \\ &= Z(x) - Z(x_1) - Z(x_2) - \cdots - Z(x_k) \triangleright x_i \text{ are leaf nodes} \end{aligned}$$

Here,

$$\begin{aligned} A(T_x) &= A(T_{x_1}) + A(T_{x_2}) + \cdots + A(T_{x_k}) + A(x) \\ &= Z(x_1) + Z(x_2) + \cdots + Z(x_k) + Z(x) - Z(x_1) - Z(x_2) - \cdots - Z(x_k) \\ &= Z(x) \end{aligned}$$

Theorem 1 allows us to compute the suitable cost terms z_1 for each modified cluster if we maintain the corresponding total actual cost. Unfortunately, the cost component z_2 involves weight and a reference point. The modified cluster sequences $M_L(x_1)$ for the node x_1 are computed for the reference point x_1^{-R} . To compute the cost of the sink x , we need to translate the reference point of the modified cluster sections $M_L(x_1)$ from x_1^{-R} to x . Similarly, we need to translate the reference points of other modified cluster sections of the node from $x_2 \dots x_k$ to x . It is not clear how Theorem 1 can help maintain the total of terms z_2 if a cost adjustment is necessary, and this adjustment depends on the level of the tree.

We address this issue by choosing a virtual reference point p^∞ when calculating a modified cluster's cost term z_2 . The distance of point p^∞ to the root is chosen so that any possible sink node is not farther from the root than p^∞ . We need to subtract the cost to adjust the pre-computed cost term z_2 to the actual sink location. Fortunately, the cost value we need to subtract is the same for all modified clusters and does not depend on the level of the modified cluster. It depends on the distance between p^∞ and the sink node. We can, therefore, apply Theorem 1 with cost term $Z(x)$ computed for z_2 with the reference point p^∞ .

5 Complexity Analysis

We compute the list of cluster sections for each node using a standard approach similar to, but simpler than Mamada's procedure [8]. Every node can contribute at most one section to the list of cluster sections stored at the nodes on the path towards the root. The merging operation takes constant time per cluster section. Since the height of a balanced binary tree with n nodes is $O(\log n)$, the pre-processing phase takes $O(n \log n)$ time.

We can compute the modified cluster sections for each node as explained in Sect. 4. The modified cluster sections can be computed by using the Algorithms 1 and 2. Here, Algorithm 2 will perform a binary search for each base section $b \in B$. However, the binary search uses fractional cascading, costing $O(\log n)$ time to query all $O(\log n)$ modified cluster sequences. Therefore, to compute modified cluster sections, each base section will take $O(\log n)$ time. Since there are $O(n \log n)$ base sections in total, the computational cost for the modified clusters is $O(n \log^2 n)$.

6 Conclusion

In this paper, we present several algorithmic techniques that allow us to significantly improve the time complexity of the minsum 1-sink location problem on a balanced binary tree network. Minsum sink location problems for tree networks are notoriously difficult. No result other than the obvious quadratic algorithms has been known for over a decade.

We have shown that the pre-processed cluster sections can be efficiently adjusted to account for new evacuation paths in the tree. This is a fundamental operation for many algorithms that tackle sink location problems in networks that are more complicated than paths. We believe that our idea of modified clusters and our cost accounting techniques, which do not require maintaining complicated data structures, can be extended to handle the 1-sink minsum location problem with arbitrary capacity, and may reduce the effort to obtain sub-quadratic algorithms for arbitrary networks. At the opposite spectrum from trees with small diameters are the caterpillar networks. We conjecture that a sub-quadratic algorithm for caterpillars, combined with the results from this paper, will open the way to a sub-quadratic algorithm for arbitrary trees.

References

1. Benkoczi, R., Bhattacharya, B., Higashikawa, Y., Kameda, T., Katoh, N.: Minsum k-sink problem on path networks. *Theor. Comput. Sci.* **806**, 388–401 (2020)
2. Chazelle, B., Guibas, L.J.: Fractional cascading: I. A data structuring technique. *Algorithmica* **1**, 133–162 (1986). <https://api.semanticscholar.org/CorpusID:12745042>
3. Chazelle, B., Guibas, L.J.: Fractional cascading: II. Applications. *Algorithmica* **1**, 163–191 (1986). <https://api.semanticscholar.org/CorpusID:11232235>
4. Ford, L.R., Jr., Fulkerson, D.R.: Constructing maximal dynamic flows from static flows. *Oper. Res.* **6**(3), 419–433 (1958)
5. Higashikawa, Y., Golin, M.J., Katoh, N.: Multiple sink location problems in dynamic path networks. *Theor. Comput. Sci.* **607**, 2–15 (2015)
6. Higashikawa, Y., Katoh, N.: A survey on facility location problems in dynamic flow networks. *Rev. Socionetwork Strategies* **13**, 163–208 (2019)
7. Higashikawa, Y., Katoh, N., Teruyama, J., Watase, K.: Almost linear time algorithms for minsum k-sink problems on dynamic flow path networks. *Theor. Comput. Sci.* **873**, 87–113 (2021)
8. Mamada, S., Uno, T., Makino, K., Fujishige, S.: An $O(n \log^2 n)$ algorithm for the optimal sink location problem in dynamic tree networks. *Discret. Appl. Math.* **154**(16), 2387–2401 (2006)



Two Step Graph Protection Game

Aakash Ghosh^{1(✉)} and Saraswati Girish Nanoti^{2(✉)}

¹ Tata Institute of Fundamental Research, Mumbai, India

aakash.ghosh@tifr.res.in

² Indian Institute of Technology Gandhinagar, Palaj, India

nanoti_saraswati@iitgn.ac.in

<https://www.tifr.res.in/>, <http://www.iitgn.ac.in>

Abstract. The *two-step graph protection game* is a two-player game played on an n -vertex graph G , where one player is *the attacker* and the second player is *the defender*. The defender has k guards which he places on some vertices of the graph at the beginning of the game. In each move, the attacker attacks an edge. The defender moves the guards in such a manner that each guard can move at most two steps (along two adjacent edges) without retracing his previous step and one of the guards must move along the attacked edge. If such a movement is not possible after a finite sequence of attacks, the attacker wins. If the defender is able to defend the graph for an infinite sequence of attacks, the defender wins. We define the minimum number of guards for which the defender has a winning strategy, as the *two-step protection number* of a graph G and denote it by $\text{tsn}(G)$.

We show that it is NP-hard to decide whether $\text{tsn}(G) \leq k$ for a given graph G and integer k and this problem is also W[2]-hard parameterized by k . We compute the $\text{tsn}(G)$ for some graph classes, i.e., stars, paths, cycles, complete graphs, complete bipartite graphs and split graphs. We also generalize these results to a game where the guards can move for at most r steps (without retracing) where r is less than the diameter of the graph.

1 Introduction

Graph protection or graph security games have their roots in the time of the ancient Roman Empire (as mentioned in [15]) where people studied attack-defense strategies for actual warfare. A graph could be used to depict a situation where vertices could represent a set of safe houses or places and roads connecting them could be represented by edges. Depending on the situation, there could be a restriction on how the guards or soldiers can move from one place to another. The concept of *Roman Domination* or *weak Roman domination* involves the study of a function from $V \rightarrow \{0, 1, 2\}$, where every vertex u with 0 value has at least one neighbour v with $f(v) = 2$ (for *Roman Domination*) or $f(v) > 0$ (for

This work was done when the first author was a student at Indian Institute of Science Education and Research Kolkata. The second author acknowledges support from CSIR.

weak Roman domination). Such kind of problems have been widely studied and some examples of papers are [11, 13] and [17], to name a few.

Some infinite variations of these games are also well studied. These games are typically two-player games, where one player is the attacker and the second player is the defender. A survey on ETERNAL VERTEX COVER and ETERNAL DOMINATING SET problems which are examples of such games can be found in [15].

The ETERNAL VERTEX COVER problem is a dynamic variation of the vertex cover problem. The motivation behind this problem can be understood with the help of a practical example discussed in [5]. Suppose we have a network where nodes are represented by vertices and wires/links connecting the nodes are represented by edges. Suppose we have an adversary who is destroying the links, and we need to send guard from one node to its neighbour to repair the destroyed link (along the destroyed link), what is the least number of guards required so that we can keep on doing this for as long as needed? Note that each guard only moves one step at a time (from one node to an adjacent node) and all the other guards can stay on their initial position or shift to an adjacent node. The attacks of the adversary are not pre-decided and whenever they find a link with both the endpoints unoccupied they can destroy it, and we would not be able to repair it. We will define the problem formally in Sect. 2. The minimum number of guards which can be used to defend the graph G for any infinite sequence of attacks is called $\text{evc}(G)$.

Our Results. In [16], a variant of the game called NEW ETERNAL VERTEX COVER is introduced, where the guards can move for at most two steps without retracing; but the attack needs to be defended in the first step. In this case, the minimum number of guards required to protect G is known as $\text{nevc}(G)$. Here, we study a variant where the guards are allowed to move for at most two steps without retracing and the attack may be defended in either the first or the second step. We denote the minimum number of guards required in this case by $\text{tsn}(G)$, and call this the two-step protection number. Note that this number can be much smaller than $\text{mvc}(G)$ whereas $\text{nevc}(G) \geq \text{mvc}(G)$ (for all graphs G) as discussed in some of the next sections. Here $\text{mvc}(G)$ denotes the size of the smallest vertex cover of the graph G .

The TWO STEP PROTECTION NUMBER problem asks, given a graph G , to determine $\text{tsn}(G)$. We show that the problem is NP-hard even when the input graph has diameter at most five (Theorem 1). From the same reduction, we can conclude that the problem is W[2]-hard when parameterized by $\text{tsn}(G)$ (Theorem 2). The problem remains hard when the guards can move more than two steps (Theorem 3). We also compute the values for $\text{tsn}(G)$ for stars, complete graphs, paths and cycles and compare these results with the ETERNAL VERTEX COVER and the NEW ETERNAL VERTEX COVER problem.

Related Work. The Eternal Vertex Cover problem was introduced by Klostermeyer and Mynhardt in 2009 in [14]. In the same paper [14], they gave the result that the ETERNAL VERTEX COVER NUMBER of G for any graph G lies between $\text{mvc}(G)$ and $2\text{mvc}(G)$. The characterization of the graphs G which satisfy the

upper bound for $\text{evc}(G)$ was also given in the same paper [14]. It was shown that the ETERNAL VERTEX COVER problem is NP-hard even on bipartite graphs of small diameter in [6] and the problem is FPT parameterized by the number of guards as shown in [12]. In [3, 4], and [7], this question of finding the graphs G which attain the lower bound for $\text{evc}(G)$ is addressed and [4] and [3] give such a characterization for a family of graphs which include chordal graphs and biconnected internally triangulated planar graphs. In [8] and [5] the authors use these type of structural characterizations to give a polynomial-time algorithm to calculate the $\text{evc}(G)$ for a graph G belonging to some special graph classes such as maximal outerplanar graphs and cactus graphs. The ETERNAL VERTEX COVER problem for generalized trees was studied in [1]. In [16], a characterization of bipartite graphs for which the evc number is same as the size of the minimum sized vertex cover is given.

2 Preliminaries

Let $G(V, E)$ be a simple, finite and undirected graph with n vertices and m edges unless mentioned otherwise. The vertex set and the edge set of G will be denoted by $V(G)$ and $E(G)$ respectively. A *path* on n vertices is a graph P with vertices $\{v_1, v_2, \dots, v_n\}$ such that $E(P) = \bigcup_{i=1}^{n-1} (v_i, v_{i+1})$. A *cycle* on n vertices is a graph C with vertices $\{v_1, v_2, \dots, v_n\}$ such that $E(C) = \bigcup_{i=1}^{n-1} (v_i, v_{i+1}) \cup (v_1, v_n)$. A graph is said to be a *clique* if every pair of its vertices have an edge between them. The set of all vertices v such that $uv \in E(G)$ is denoted by $N(u)$ and is said to be the *open neighbourhood* of u . The set $N(u) \cup \{u\}$ is said to be the *closed neighbourhood* of u . A graph $G(V, E)$ is said to be *connected* if for every two distinct vertices $u, v \in V$, there exists a path between u and v .

A subset S of $V(G)$ is said to be a *vertex cover* of G if for every edge $(u, v) \in E$, either $u \in S$ or $v \in S$. The size of the smallest vertex cover of the graph G is called the *minimum vertex cover number* of G and denoted by $\text{mvc}(G)$. A subset S of $V(G)$ is said to be an *independent set* of G if no edge of $E(G)$ has both its endpoints in S . A subset S of $V(G)$ is said to be a *dominating set* of G if for every vertex $v \in V$, we have $N[v] \cap S \neq \emptyset$. If $u \in N[v] \cap S$ then we say that u dominates v or v is dominated by u .

A graph G whose vertices can be partitioned into two sets such that one set induces a clique C and the other induces an independent set I is called a *split graph*. A vertex in I is called global if it is adjacent to all the vertices in $V(C)$. Without loss of generality, we can shift one of the global vertices to C and assume that I contains no global vertex.

For an instance of size n and parameter k , a problem which can be solved in $f(k)n^{O(1)}$ where f is an arbitrary computable function independent of n is called FPT (fixed parameter tractable).

For more terminologies on graphs the reader is directed to [10] and for the notion of parameterized reduction and W hierarchy, the reader is directed to [9] and for other definitions in computational complexity, the reader is directed to [2].

In the ETERNAL VERTEX COVER problem, we have a two-player game on a graph $G(V, E)$. One player, i.e., the “attacker” attacks an edge of the graph in each of her moves and the other player, i.e., the “defender” moves the guards to their neighbouring vertices in such a manner that at least one guard moves along the attacked edge. Some of the other guards may remain on their original vertex. If such a move is not possible, the attacker wins. The defender needs to defend an infinite sequence of attacks for him to win. The minimum number of guards for which the defender has a winning strategy, is called the *eternal vertex cover number* of the graph G (denoted by $\text{evc}(G)$). In the first variant of this problem only one guard per vertex is allowed after the completion of each move whereas in the second variant there is no such constraint. The values for $\text{evc}(G)$ mentioned in this paper are applicable for both the variants.

In [16], the NEW ETERNAL VERTEX COVER problem was defined where the guards can move for at most two steps without retracing but the attack needs to be defended in the first step. Here, if retracing is allowed, then the problem becomes the same as the VERTEX COVER problem. It is easy to see that $\text{tsn}(G) \leq \text{nevc}(G)$, and we have examples where it is much smaller in Sect. 4.

In [14], it is stated that the eternal vertex cover number of G ($\text{evc}(G)$) for any graph G lies between $\text{mvc}(G)$ and $2\text{mvc}(G)$. It was shown that the ETERNAL VERTEX COVER problem is NP-hard even on bipartite graphs of small diameter in [6] and the problem is FPT parameterized by the number of guards as shown in [12]. In [16], it was shown that $\text{nevc}(G)$ lies between $\text{mvc}(G)$ and $\text{mvc}(G) + 1$. Also, it was shown that $\text{nevc}(G) = \text{mvc}(G)$ if and only if either G has no degree 1 vertex; or for every degree 1 vertex v of G there exists a minimum sized vertex cover S_v of G which contains v . Since there exists an FPT algorithm for vertex cover, this means that the NEW ETERNAL VERTEX COVER problem is also FPT parameterized by solution size. We will show that this is unlikely to be the case for the Two STEP PROTECTION NUMBER problem (and its generalized version described in Sect. 5), i.e., we will show that this problem is W[2]-hard parameterized by solution size.

3 Complexity of Computing the Two Step Protection Number of a Graph G

The *two step graph protection game* is a two player game played on an n -vertex graph G , where one player is *the attacker* and the second player is *the defender*. The defender has k guards which he places on some vertices of the graph at the beginning of the game. In each move, the attacker attacks an edge. The defender moves the guards in such a manner that each guard can move at most two steps (along two adjacent edges) without retracing his previous step and one of the guards must move along the attacked edge. If such a movement is not possible after a finite sequence of attacks, the attacker wins. If the defender is able to defend the graph for an infinite sequence of attacks, the defender wins. We define the minimum number of guards for which the defender has a winning strategy, as the *two step protection number* of a graph G and denote it by $\text{tsn}(G)$.

Note that we allow multiple guards on one vertex while rearranging the guards, but the final reconfiguration after each attack must have one guard per vertex.

In this section we will show that it is NP-hard and W[2]-hard parameterized by solution size to determine whether k guards are sufficient to protect a given input graph G from an infinite sequence of attacks.

We define the problem formally as follows:

TWO-STEP PROTECTION NUMBER

Input: A graph G , an integer k

Question: Is it possible to defend graph G from an infinite sequence of attacks using k guards in the *two-step protection game*?

We will show that the TWO STEP PROTECTION NUMBER problem is NP-hard by using a reduction from the DOMINATING SET problem on split graphs. This problem is hard even when the given split graph is connected and we will assume this throughout that the given split graph is connected. This problem is defined as shown:

DOMINATING SET ON SPLIT GRAPHS

Input: A split graph G (with a clique C and an independent set I), an integer k

Question: Does there exist a set $S \subset V(G)$ of size at most k such that for each $v \in V(G)$, there exists $u \in S$ such that $u \in N[v]$?

We will describe the construction of the equivalent instance and the proof of the reduction in detail in the next few paragraphs. Without loss of generality, we will assume that $k > 1$.

Lemma 1. *Given an instance of DOMINATING SET on a split graph G and an integer k , we can construct an equivalent instance of TWO STEP PROTECTION NUMBER with $k + 1$ guards in polynomial time.*

Proof. Before describing the construction of the reduced instance and showing the equivalence between the given problem instance and the newly constructed problem instance, we state a structural result regarding the YES instances of the DOMINATING SET problem on split graphs. This is a well known result in the literature and we are deferring the proof to the full version.

Proposition 1. *If a split graph G with clique C and independent set I is a YES instance of the DOMINATING SET problem with solution size k , then G has a k -sized subset S of $V(C)$ such that S is a dominating set of G .*

Now we describe the construction of the equivalent instance for the TWO STEP PROTECTION NUMBER from the given instance of DOMINATING SET on a split graph G with a clique C and an independent set I .

The Construction: For constructing the new graph H , begin with a copy of G . We will refer to the subgraph induced by the copies of the clique and in the independent set in H as C_H and I_H , respectively. Now add a pendant vertex (degree-one neighbour) to each vertex in I_H and denote subgraph induced by the set of these newly added vertices by P_H . Note that the graph induced by I_H and P_H is a matching (a set of disjoint edges) of size $|I_H|$. Now add a new vertex N which is adjacent to all the vertices in C_H . Add a new leaf vertex L which is adjacent to N . This completes the construction. A figure demonstrating the construction is shown in the full version.

Sometimes we will refer to the copy of a vertex by the same name, and it will be clear from the context whether we mean a vertex from G or a vertex from H . Also, sometimes when we have a vertex u in H such that u is a copy of a vertex v in G , we will refer to v as a copy of u in H . Note that H is not a split graph and the diameter of H is at most 5. It is clear that H can be constructed from G in polynomial time.

For a vertex $v \in V(P_H)$, we will denote its unique neighbour u in $V(I_H)$ by *parent* of v and we will refer to v as a *child* of u . For a vertex v in $V(I_H)$, we will refer to its neighbour (say u) in $V(C_H)$ as a *parent* of v and we will refer to v as a *child* of u . Note that a vertex in $V(P_H)$ has a unique parent and a vertex in $V(I_H)$ has a unique child. But the vertices in $V(I_H)$ may have multiple parents (in $V(C_H)$) and the vertices in $V(C_H)$ may have multiple children.

We say that a vertex $v \in V(P_H)$ is a *grandchild* of $u \in V(C_H)$ if there exists $w \in V(I_H)$ such that w is a child of u and v is the (unique) child of w . In this case, we will also refer to the vertex u as a *grandparent* of v . Notice that a vertex in $V(C_H)$ may have multiple grandchildren, and similarly, a vertex in $V(P_H)$ may have multiple grandparents. However, for a given vertex $u \in V(C_H)$ and a fixed child w of u , there will be a unique grandchild v of u such that w is the parent of v . We will refer to v as the grandchild of u through w .

We show that G has a dominating set of size k ($k > 1$) if and only if H can be protected using $k + 1$ guards against an infinite sequence of attacks.

The Forward Direction: Suppose (G, k) is a YES instance of the DOMINATING SET problem, then by Proposition 1 we have $S \subseteq V(C)$ such that S is a dominating set of G . Let $T \subseteq V(C_H)$ be the set of vertices of H which are copies of vertices of S . We give a winning strategy for the defender using $k + 1$ guards.

In this strategy, we will maintain the invariant that T is always occupied before each attack. After that, we will describe how each attack can be defended (while restoring the invariant). In order to show that this invariant can be maintained, we first observe the following: (and the proof is deferred to the full version).

Proposition 2. *Any guard which is not on a vertex of T , can reach some vertex of T in at most two steps.*

Now consider any configuration where all the vertices in T have a guard and any one arbitrary vertex which is not in T has a guard. We will show how to

defend any attack while reaching a configuration such that all the vertices in T have a guard (and one guard is on any arbitrary vertex not in T). We will refer to the guard outside T as the *extra* guard. We know that there exists $t \in T$ such that the extra guard (wherever he is) can move to t in at most two steps. We will refer to t as a *reachable* vertex. This vertex may not be unique for a given position of the extra guard. Suppose an edge whose both endpoints are occupied is attacked, then the guards exchange their position and the configuration remains the same. Hence we will consider the edges where at least one of the endpoints is unoccupied. A sketch of this strategy is shown in a figure in the full version.

Case I: An edge uv where $u \in V(I_H)$ and $v \in V(P_H)$ is attacked.

Since S is a dominating set in G , the copy of u in G has a neighbour in S (say w). The guard on the copy of w (in T) moves to u and then to v , thus defending the attack. If a reachable vertex t is the same as the copy of w in T , then the extra guard moves to t and we are done. If not, then the guard on t moves to the copy of w (in one step) and the extra guard moves to t and we are done. We have a configuration where all the vertices in T have a guard and the extra guard is on v .

Case II: An edge uv where $u \in V(C_H)$ and $v \in V(I_H)$ is attacked.

If $u \in T$, then the guard on u moves to v , thus defending the attack. If u is a reachable vertex, then the extra guard moves to u (in at most two steps), otherwise the guard on a reachable vertex t moves to u in one step and the extra guard moves to t in at most two steps. Thus we reach a configuration with all the vertices in T occupied and the extra guard on v .

If $u \notin T$, then again the guard on a reachable vertex t moves to u and then to v , thus defending the attack. The extra guard moves to t in at most two steps. Thus we again reach a configuration with all the vertices in T occupied and the extra guard on v .

Case III: An edge uv where $u, v \in V(C_H)$ is attacked.

If one of u and v belongs to T (and the other does not) i.e. suppose $u \in T$ and $v \notin T$, the guard on u moves to v , thus defending the attack. If u is a reachable vertex, then the extra guard moves to u , otherwise the guard on the reachable vertex t moves to u and the extra guard moves to t . Thus the guards occupy all the vertices of T and the extra guard is on v in the resulting configuration.

If neither u nor v belongs to T , then the guard from a reachable vertex t moves to u and then to v thus defending the attack. The extra guard moves to t in at most two steps. Thus the guards occupy all the vertices of T and the extra guard is on v in the resulting configuration.

Case IV: An edge uN is attacked where $u \in V(C_H)$ and N is the new vertex which is adjacent to all the vertices of C_H .

If $u \in T$, the guard on u moves to N defending the attack. If u is a reachable vertex, the extra guard moves to u in at most two steps and we are done. If not, the guard on the reachable vertex t moves to u and the extra guard moves to t

in at most two steps. Thus the resulting configuration has all the vertices in T occupied and the extra guard on N .

If $u \notin T$, the guard on the reachable vertex t moves to u and then to N . The extra guard moves to t in at most two steps. Thus the resulting configuration has all the vertices in T occupied and the extra guard on N .

Case V: The edge NL is attacked.

If the extra guard is on N or L then the guard moves from N to L (or L to N). All the other guards stay on their position. Thus we have a configuration with all the vertices in T occupied and the extra guard on N or L .

If the extra guard is on some other vertex, let $t \in T$ be the reachable vertex. The guard on t moves to N and then to L , defending the attack. Now the extra guard moves to t in at most two steps. Thus we have a configuration with all the vertices in T occupied and the extra guard on N or L .

Hence we have shown that if G has a dominating set of size k , then H can be defended for an infinite sequence of attacks using $k+1$ guards.

The Reverse Direction: We need to show that if H has a winning strategy using at most $k+1$ guards, then G has a dominating set of size at most k . Consider a configuration of H with at most $k+1$ guards. If a guard is not present on either N or L , then we attack the edge NL . Then some guard must come to L . Therefore, we can assume without loss of generality that H has a winning strategy starting from a configuration with a guard on L or N .

Now we claim that we can form a dominating set of G of size at most k using the positions of all the guards on $V(H) \setminus \{N, L\}$. Note that since $k > 1$, there is at least one guard on $V(H) \setminus \{N, L\}$. Let T be the set of vertices with guards in $V(H) \setminus \{N, L\}$, let $S = \emptyset$. Note that $|T| \leq k$. For every $u \in T \cap V(C_H)$, add the copy of u in G to S . For every $u \in T \cap V(I_H)$, add the copy of a parent of u in G to S . For every $u \in T \cap V(P_H)$, add the copy of any grandparent of u to S . If some vertex selected in this procedure is already in S , do not add anything in S . (This may happen because the parent of a vertex in I_H or the grandparent of a vertex in P_H may already be added in S because it was also the parent of some other vertex in I_H or the grandparent of some other vertex in P_H). It is clear that $S \subset C$ and the size of S is at least 1 and less than or equal to the size of T , and therefore it is at most k .

Note that all vertices of C are dominated by S , since $S \subset C$ and size of S is at least 1, then all the vertices in $C \setminus S$ have at least one neighbour in S . Now suppose S does not form a dominating set of G , then there exists a vertex in I which does not have a neighbour in S . Let $u \in I$ be the vertex which does not have a neighbour in S . Attack the edge joining the copy of u (in I_H) to its child (say v) in P_H . To defend this attack, there must be at least guard on v , or its parent or one of its grandparents. But if this was the case, we would have added the copy of one of the grandparents of v to S and thus the vertex u would have had a neighbour in S , which is a contradiction. Thus G has a dominating set of size at most k . \square

Since we have a polynomial time reduction from the DOMINATING SET problem on connected split graphs to the TWO STEP PROTECTION NUMBER on graphs of diameter at most 5 as described in Lemma 1, we have proved the theorem given next.

Theorem 1. *The TWO STEP PROTECTION NUMBER problem is NP-hard on graphs of diameter at most 5.*

Since the DOMINATING SET problem on connected split graphs is also W[2]-hard with the solution size as parameter, the above reduction also implies that the TWO STEP PROTECTION NUMBER problem is W[2]-hard parameterized by the number of guards.

Theorem 2. *It is W[2]-hard to compute the TWO STEP PROTECTION NUMBER of a graph G of diameter at most 5 (parameterized by the solution size).*

4 The Two Step Protection Number for Some Graph Classes

In this section, we describe the two-step protection number for some special classes of graphs.

Complete Graphs and Complete Bipartite Graphs. The values for $\text{mvc}(G)$ and $\text{evc}(G)$ for complete graphs and complete bipartite graphs are well-known and the values of $\text{nevc}(G)$ of these classes can be found in [16] and also can be verified easily. In the case of complete graphs, we have $\text{nevc}(G) = \text{evc}(G) = \text{mvc}(G) = n - 1$. This is an example where $\text{tsn}(G)$ is much smaller than the $\text{evc}(G)$ as well as the $\text{nevc}(G)$ whereas in the case of stars, $\text{nevc}(G) = \text{evc}(G) = 2$ and $\text{tsn}(G) = \text{mvc}(G) = 1$. In case of any complete bipartite graph $G = K_{m,n}$, with $m \leq n$ we have $\text{mvc}(G) = \text{nevc}(G) = m$; $\text{evc}(G) = m + 1$ and $\text{tsn}(G) = 1$.

Lemma 2. *If G is a complete graph K_n or a complete bipartite graph $K_{m,n}$ or a complete k -partite graph for $k \geq 3$, we have $\text{tsn}(G) = 1$.*

Split Graphs. For connected split graphs it was proved that $\text{evc}(G) = \text{mvc}(G)$ or $\text{evc}(G) = \text{mvc}(G) + 1$ in [4]. Also, $\text{mvc}(G) = |C|$ or $\text{mvc}(G) = |C| - 1$. Therefore, $\text{nevc}(G)$ is also equal to $|C|$ or $|C| + 1$ because $\text{mvc}(G) \leq \text{nevc}(G) \leq \text{mvc}(G) + 1$. If $|C|$ is large, then split graphs are one example of a graph class where $\text{tsn}(G)$ is much smaller than $\text{mvc}(G)$ (thus also $\text{nevc}(G)$ and $\text{evc}(G)$).

Lemma 3. *Given a connected split graph G with clique C and independent set I such that I has no global vertex and $|C| > 1, |I| \geq 1$, we have $\text{tsn}(G) \leq 2$. Also $\text{tsn}(G) = 1$ if and only if all the vertices in I have the same neighbourhood in C and exactly one non-neighbour in C .*

We defer the proof of Lemma 2 and Lemma 3 to the full version.

Paths and Cycles. For paths and cycles the sizes of $\text{tsn}(G)$, $\text{evc}(G)$, $\text{mvc}(G)$ and $\text{nevc}(G)$ are all $O(n)$. We have the lemma given below and we defer the proof to the full version.

Lemma 4. *For a graph G which is a cycle on n vertices, we have $\text{tsn}(G) = \lceil \frac{n}{4} \rceil$. For a graph G which is a path on n vertices, we have $\text{tsn}(G)$ equal to $\lfloor \frac{n}{2} \rfloor$.*

5 Moving More Than Two Steps

In this section, we generalise the problem as follows: We have a two-player game played on an n -vertex graph G where one player is *the attacker* and the second player is *the defender*. The defender has k guards which he places on some vertices of the graph at the beginning of the game. In each move, the attacker attacks an edge. The defender moves the guards in such a manner that each guard can move at most r steps for some fixed $r \in \mathbb{N}$ (along a path of r edges) without retracing his previous step and one of the guards must move along the attacked edge. If such a movement is not possible after a finite sequence of attacks, the attacker wins. If the defender is able to defend the graph for an infinite sequence of attacks, the defender wins. We define the minimum number of guards for which the defender has a winning strategy, as the r -step protection number of a graph G and denote it by $\text{rsn}(G)$.

If $r > d$ where d is the diameter of a graph G , then the problem becomes trivial as only one guard is sufficient for defending the graph G . If the guard is present on a vertex w and the edge uv is attacked and if $v = w$ then the guard goes from v to u . Otherwise the distance between u and w is at most d which is at most $r - 1$. Thus the guard moves from w to u via the shortest path in at most $r - 1$ steps. If v lies on this path then we are done. If not, then the guard moves from u to v . Still, the total number of steps is not more than r . Each subsequent attack can be defended using essentially the same strategy.

Thus, the problem is interesting only when $r \leq d$. It is easy to see that $\text{rsn}(G) = 1$ for complete bipartite as well as complete k -partite graphs for any $k \geq 3$ which is the same as the value of $\text{tsn}(G)$ for these graphs. In general, it is clear that we always have $\text{rsn}(G) \leq \text{tsn}(G)$ for all graphs G . We have a lemma for connected split graphs and we defer the proof to the full version.

Lemma 5. *For a connected split graph G with clique C and independent set I and at most one edge, $\text{rsn}(G) = 1$ for $r \geq 3 \in \mathbb{N}$.*

We will show that this problem is NP-hard for some fixed $r \geq 3 \in \mathbb{N}$ on $r - 1$ -partite graphs of diameter $3r - 1$ and it is also W[2]-hard parameterized by the number of guards. The reduction is quite similar to the reduction in Sect. 3. (Note that the case when $r = 2$ has already been studied in Sect. 3 and the case $r = 1$ is the same as the ETERNAL VERTEX COVER problem. Therefore, now we assume $r \geq 3$.)

Lemma 6. *Given an instance of DOMINATING SET on a split graph G and an integer k , we can construct an equivalent instance of r -STEP PROTECTION NUMBER with $k + 1$ guards in polynomial time for a given fixed $r \geq 3 \in \mathbb{N}$.*

Theorem 3. *The r -STEP PROTECTION NUMBER problem is NP-hard and W[2]-hard parameterized by solution size for any fixed $r \geq 3 \in \mathbb{N}$ on $(r-1)$ -partite graphs of diameter $3r-1$.*

The proof of Lemma 6 is deferred to the full version and Theorem 3 is implied by Lemma 6.

6 Conclusion

In this paper, we define a variant of the ETERNAL VERTEX COVER problem where the guards are allowed to move for at most two steps without retracing. We show that the TWO STEP PROTECTION NUMBER problem is NP-hard and W[2]-hard parameterized by the solution size even on graphs of diameter 5. We also show that the same result holds when the guards are allowed to move for r steps without retracing for $r-1$ -partite graphs of diameter $3r-1$. The reductions work even if retracing is allowed. It will be interesting to find graphs which require less number of guards compared to $\text{tsn}(G)$ if retracing is allowed. We also give the value of the two-step protection number for split graphs, complete bipartite graphs, paths and cycles. The complexity of the TWO STEP PROTECTION NUMBER problem is unknown on bipartite graphs and even on trees. Also, there can be other variants with intermediate crossing of guards not allowed, or guards have to move for exactly two steps and so on. The characterization of graphs for which $\text{rsn}(G) = \text{tsn}(G)$ or $\text{tsn}(G) = \text{mvc}(G)$ or $\text{tsn}(G) = \text{nevc}(G)$ could be some more directions for future research.

Acknowledgement. The authors thank Dr Neeldhara Misra for her useful and helpful suggestions and discussions.

References

1. Araki, H., Fujito, T., Inoue, S.: On the eternal vertex cover numbers of generalized trees. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **98-A**(6), 1153–1160 (2015)
2. Arora, S., Barak, B.: Computational Complexity: A Modern Approach. Cambridge University Press, Cambridge (2009)
3. Babu, J., Chandran, L.S., Francis, M., Prabhakaran, V., Rajendraprasad, D., Warrier, J.N.: On graphs with minimal eternal vertex cover number. In: Pal, S.P., Vijayakumar, A. (eds.) CALDAM 2019. LNCS, vol. 11394, pp. 263–273. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11509-8_22
4. Babu, J., Sunil Chandran, L., Francis, M.C., Prabhakaran, V., Rajendraprasad, D., Warrier, N.J.: On graphs whose eternal vertex cover number and vertex cover number coincide. *Discret. Appl. Math.* (2021, in press)
5. Babu, J., Krishnan, K., Prabhakaran, V., Warrier, N.: Eternal vertex cover number of maximal outerplanar graphs. arxiv, Discrete Mathematics (2022)
6. Babu, J., Misra, N., Nanoti, S.G.: Eternal vertex cover on bipartite graphs. In: Computer Science – Theory and Applications, pp. 64–76. Springer, Cham (2022)

7. Babu, J., Prabhakaran, V.: A new lower bound for the eternal vertex cover number of graphs. *J. Comb. Optim.* **27**–39 (2021)
8. Babu, J., Prabhakaran, V., Sharma, A.: A linear time algorithm for computing the eternal vertex cover number of cactus graphs. *arXiv, Discrete Mathematics* (2020)
9. Cygan, M., et al.: Parameterized Algorithms. Springer, Cham (2015)
10. Diestel, R.: Graph Theory. Springer, Cham (2017)
11. Fernau, H.: Roman domination: a parameterized perspective. *Int. J. Comput. Math.* **85**, 25–38 (2006)
12. Fomin, F.V., Gaspers, S., Golovach, P.A., Kratsch, D., Saurabh, S.: Parameterized algorithm for eternal vertex cover. *Inf. Process. Lett.* **110**(16), 702–706 (2010)
13. Henning, M.A.: Defending the roman empire from multiple attacks. *Discret. Math.* **271**, 101–115 (2003)
14. Klostermeyer, W.F., Mynhardt, C.M.: Edge protection in graphs. *Australas. J. Comb.* **45**, 235–250 (2009)
15. Klostermeyer, W.F., Mynhardt, C.M.: Protecting a graph with mobile guards. *Applicable Anal. Discret. Math.* **10**(1), 1–29 (2016)
16. Misra, N., Nanoti, S.G.: Spartan bipartite graphs are essentially elementary. In: MFCS. LIPIcs, vol. 272, pp. 68:1–68:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2023)
17. Valveny, M., Pérez-Rosés, H., Rodríguez-Velázquez, J.A.: On the weak roman domination number of lexicographic product graphs. *Discret. Appl. Math.* **263**, 257–270 (2019). Special Issue: 10th Andalusian Meeting on Discrete Mathematics



Bipartite Domination in Graphs: Complexity and Algorithms

Bhawani Sankar Panda¹, Subhasmita Joshi², and Dalu Jacob^{2(✉)}

¹ Department of Computer Science and Engineering, The LNM Institute of Information Technology, Jaipur 302031, India
bhawani.panda@lnmiit.ac.in

² Department of Mathematics, Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110016, India
{maz238454,dalu{jacob}@maths.iitd.ac.in}

Abstract. A set $D \subseteq V$ of vertices of a graph $G = (V, E)$ is called a *dominating set* of G if for every vertex $u \in V \setminus D$, there exists a vertex $v \in D$ such that $uv \in E(G)$. A dominating set D is called a *bipartite dominating set* if $G[D]$, the subgraph induced by D , is bipartite. The *domination number* of G is the minimum cardinality among all dominating sets of G and it is denoted by $\gamma(G)$. The *bipartite domination number* of G is the minimum cardinality among all bipartite dominating sets of G and it is denoted by $\gamma_{bip}(G)$. The MIN DOM problem is to find a dominating set of minimum cardinality of a given graph G and DECIDE DOM is the decision version of the MIN DOM problem. Similarly, the MIN BIP-DOM problem is to find a bipartite dominating set of minimum cardinality of a given graph G and DECIDE BIP-DOM is the decision version of the MIN BIP-DOM problem. In this paper, we initiate the algorithmic study of the MIN BIP-DOM problem. First, we study the complexity difference between the MIN DOM problem and the MIN BIP-DOM problem. The difference between $\gamma_{bip}(G)$ and $\gamma(G)$ of a graph G can be arbitrarily large. However, the DECIDE BIP-DOM problem is the same as the DECIDE DOM problem for bipartite graphs and hence is NP-complete for bipartite graphs, as DECIDE DOM problem is known to be NP-complete for bipartite graphs. We show that DECIDE BIP-DOM is NP-complete for non-bipartite graphs as well. Finally, we propose polynomial-time algorithms for the MIN BIP-DOM problem for interval graphs and block graphs.

Keywords: Domination · Bipartite domination · Block graphs · Block-cut tree

1 Introduction

The graphs considered in this paper are finite, simple, and undirected. A set $D \subseteq V$ of a graph $G = (V, E)$ is called a *dominating set* of G if every vertex

B. S. Panda—On Leave from IIT Delhi.

$v \in V \setminus D$ is adjacent to a vertex in D . The *domination number* of $G = (V, E)$ is the minimum cardinality among all dominating sets of G and it is denoted by $\gamma(G)$. The MIN DOM problem is to find a dominating set of cardinality $\gamma(G)$ and DECIDE DOM is the decision version of the MIN DOM, which is stated below:

DECIDE DOM

Instance: A graph $G = (V, E)$ and a positive integer $k \leq |V|$.

Question: Does there exist a dominating set of cardinality at most k ?

The field of domination and its variants have garnered considerable attention in the literature [7–9]. The fundamental variations of domination include *connected domination* [8], *independent domination* [6], *total domination* [4], *paired domination* [8], etc. In this paper, we are interested in the variant called *bipartite domination*. A *bipartite dominating set* of a graph $G = (V, E)$ is a subset, D of V such that D is a dominating set of G and $G[D]$, the subgraph induced by D , is bipartite. Let $H = (V', E')$ be a maximal bipartite subgraph of a graph $G = (V, E)$. Then, clearly V' is a bipartite dominating set of G . Hence, every graph admits a bipartite dominating set. The *bipartite domination number* of G , denoted by $\gamma_{bip}(G)$, is the minimum cardinality of a bipartite dominating set of G . The MIN BIP-DOM is to find a bipartite dominating set of minimum cardinality and DECIDE BIP-DOM is the decision version of the MIN BIP-DOM, as stated below.

DECIDE BIP-DOM

Instance: A graph $G = (V, E)$ and a positive integer $k \leq |V|$.

Question: Does there exist a bipartite dominating set of cardinality at most k ?

The concept of bipartite domination was introduced in 2022 by Bachstein et al. [2], where they obtained some bounds on γ_{bip} , involving chromatic number. In 2022, Bachstein et al. [1] introduced a related parameter, called a lower bipartite number of a graph, which is the minimum order of a maximal induced bipartite subgraph. Further, in [12] and [11], the bipartite domination number of the *join* graph, *fan* graph, and *Mycielski* graph of path and cycle graphs have been studied. The DECIDE BIP-DOM is the same as the DECIDE DOM problem for bipartite graphs and hence NP-complete for bipartite graphs as DECIDE DOM problem is known to be NP-complete for bipartite graphs [3]. However, note that the difference between the bipartite domination number and the domination number of a non-bipartite graph can be arbitrarily large.

In this paper, we initiate the complexity and algorithmic study of the bipartite domination problem in graphs. The main contributions of the paper are summarized as follows:

1. We show that DECIDE BIP-DOM is NP-complete for non-bipartite graphs as well.
2. We study the complexity difference between MIN DOM and MIN BIP-DOM.
3. We observe that the difference between the bipartite domination number and the domination number of a graph can be arbitrarily large. However, we show that for an interval graph G , $\gamma(G) = \gamma_{bip}(G)$, and hence, the bipartite domination problem can be solved in $O(n + m)$ time for interval graphs.

4. We propose a polynomial-time algorithm for the MIN BIP-DOM for block graphs.

2 Preliminaries

In this section, we give some pertinent definitions. Let $G = (V, E)$ be a finite, simple, and undirected graph, and let G^c denote the complement of G . Let n and m denote the cardinality of V and E , respectively. The *open neighborhood* of a vertex v in G is defined as, $N(v) = \{u \in V : uv \in E\}$ and the *closed neighborhood* is defined as, $N[v] = \{v\} \cup N(v)$. For $S \subseteq V$, $G[S] = (S, E_S)$, where $E_S = \{uv : uv \in E \text{ and } u, v \in S\}$, denotes the *subgraph induced by S* . A set of vertices in which every pair of distinct vertices are non-adjacent is called an *independent set* and if every pair of distinct vertices in S are adjacent, then $G[S]$ is called a *clique*. We use the standard notation $[k] = \{1, 2, \dots, k\}$.

A graph $G = (V, E)$ is called a *bipartite graph* if the vertex set V can be partitioned into two disjoint sets X and Y such that every edge in E has one end vertex in X and the other in Y . A graph $G = (V, E)$ is said to be a *chordal graph* if every cycle of length at least four has a chord, where a *chord* is an edge joining two non-consecutive vertices of the cycle. A graph $G = (V, E)$ is an *interval graph* if there exists a bijection $f : V \rightarrow \mathcal{F}$, where \mathcal{F} is a family of $|V|$ intervals on a real line such that $uv \in E$ if and only if $f(u) \cap f(v) \neq \emptyset$. An *interval ordering* of any graph $G = (V, E)$ is an ordering of vertices in V , say (v_1, v_2, \dots, v_n) such that for any $i < j < k$ and $v_i v_k \in E$ imply $v_j v_k \in E$. It is well-known that a graph is an interval graph if and only if it admits an interval ordering. A graph $G = (V, E)$ is said to be *split graph* if the vertex set V can be partitioned into two sets, say C and I such that $G[C]$ is a clique and I is an independent set of G .

For a graph $G = (V, E)$ and a vertex $v \in V$, $G - v$ is the graph $G[V \setminus \{v\}]$. Let G be a connected graph. A maximally connected subgraph H of G is said to be a *block* of G if $H - v$ is connected for every $v \in V(H)$. A graph G is said to be a *block graph* if every block of G is a complete graph. A vertex v in G is said to be a *cut vertex* if $G - v$ has more number of components than G . Let G be a graph and let (v_1, v_2, \dots, v_n) be an ordering of $V(G)$ obtained by performing a Breadth First Search (BFS) traversal of G starting from v_1 . Then the ordering $(v_n, v_{n-1}, \dots, v_1)$ is known as a *reverse BFS ordering* of G . Clearly, a reverse BFS ordering of a graph can be found in linear time.

3 NP-completeness Results

The NP-completeness result of DECIDE BIP-DOM follows from that of DECIDE DOM for bipartite graphs [3]. Note that $\gamma(G) \leq \gamma_{bip}(G)$ for every graph G . For a bipartite graph G , $\gamma_{bip}(G) = \gamma(G)$. However, the difference between $\gamma_{bip}(G)$ and $\gamma(G)$ can be arbitrarily large for a non-bipartite graph G .

Observation 1. *For every positive integer r , there exists a graph G such that $\gamma_{bip}(G) - \gamma(G) > r$.*

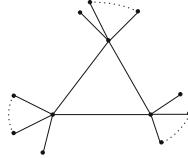


Fig. 1. A graph G for which the difference $\gamma_{bip}(G) - \gamma(G)$ is arbitrarily large.

Proof. Let G be the graph in Fig. 1 with $r + 2$ pendant edges incident at each vertex of the triangle. It is not difficult to verify that $\gamma_{bip}(G) = r + 4$ and $\gamma(G) = 3$. Therefore, we have $\gamma_{bip}(G) - \gamma(G) = r + 1 > r$. \square

In this section, we show that DECIDE BIP-DOM is NP-complete even for non-bipartite graphs. For the reduction, we use the vertex cover decision problem, which is known to be NP-complete [5]. The DECIDE VERTEX COVER is defined as follows:

DECIDE VERTEX COVER

Instance: A graph $G = (V, E)$ and a positive integer $k \leq |V|$.

Question: Does there exist a vertex cover of G with cardinality at most k ?

We prove the following theorem.

Theorem 1. DECIDE BIP-DOM is NP-complete for non-bipartite graphs.

Proof. It is easy to verify that DECIDE BIP-DOM \in NP. Now, we give a polynomial reduction to the problem from DECIDE VERTEX COVER. Given an instance (G, k) of DECIDE VERTEX COVER, we construct an instance (G', k') of DECIDE BIP-DOM, where $k' = n + k$ and the construction of the graph G' is defined as follows:

Let $V(G) = \{v_1, v_2, \dots, v_n\}$ and $E(G) = \{e_1, e_2, \dots, e_m\}$. Let G_s denote the graph obtained from G by subdividing each edge of G exactly once. Since each of the newly introduced vertex in G_s corresponds to an edge in G , we may write $V(G_s) = V(G) \cup E(G)$ and $E(G_s) = \{v_i e_j, e_j v_k : v_i, v_k \in V(G) \text{ and } e_j = v_i v_k \in E(G)\}$. Now, let H be the graph given in Fig. 2c with vertices x, y, z and p . Introduce n copies of the graph H , where for each $i \in [n]$. In the i^{th} copy of H , let x_i, y_i , and z_i be the vertices that constitute the triangle in H and p_i be the pendant vertex adjacent to z_i in H . Finally, for each i , merge the vertex p_i with the vertex v_i in G_s . We then call the resultant graph to be G' . Refer to Fig. 2 for an illustration.

We then have the following claim.

Claim 1. The graph G has a vertex cover of cardinality at most k if and only if G' has a bipartite dominating set of cardinality at most $n + k$.

Proof of the Claim: Let V_c be a vertex cover of G of cardinality at most k . Consider the set $B = \{z_i : i \in [n]\} \cup V_c$. It is not difficult to verify that B is a dominating set of G' with cardinality at most $n + k$. Also, $G'[B]$ is bipartite (by the definition of G'). Thus, B is a bipartite dominating set.

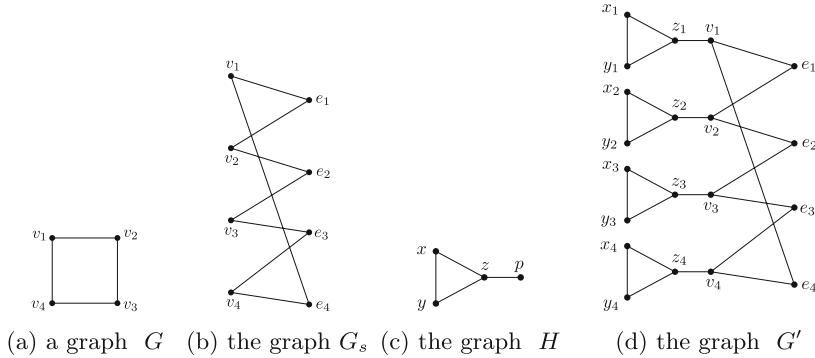


Fig. 2. An illustration of the construction of G' from the graph G in the proof of Theorem 1

Conversely, suppose that B' is a bipartite dominating set of G' of cardinality at most $n + k$. Since B' is a dominating set of G' , for each $i \in [n]$, at least one of the vertices from the set $\{x_i, y_i, z_i\}$ should be present in B' . If x_i or y_i is in B' for some $i \in [n]$, then replace it with z_i . Also if $e_j \in B'$ for some $j \in [m]$ then replace $e_j \in B'$ with one of its neighbors in G' . Let the resultant set be B . It can be seen that B is also a bipartite dominating set of G' of cardinality at most $n + k$. Further, note that $\{z_i : i \in [n]\} \subseteq B$ and therefore, B contains at most k vertices from V . Now, it is easy to verify that $B \cap V$ is a vertex cover of size at most k . This completes the proof of the claim. \square

Therefore, DECIDE BIP-DOM is NP-complete for non-bipartite graphs. Hence the theorem. \square

4 Complexity Difference between Domination and Bipartite Domination Problems

In this section, we show that the bipartite domination problem differs from the domination problem in terms of computational complexity. The DECIDE DOM is known to be NP-complete for split graphs [3]. However, we show that MIN BIP-DOM can be solved in polynomial time for split graphs.

First, we have the following observation which follows from the definitions of split graphs and bipartite dominating sets.

Observation 2. *Let $G = (C \cup I, E)$ be a split graph. Then there exists a minimum bipartite dominating set D of G such that $1 \leq |D \cap C| \leq 2$.*

We then prove the following theorem.

Theorem 2 (★). *The MIN BIP-DOM can be solved in $O(n^3)$ time for split graphs.*

The proofs of the statements marked with a (★) are omitted due to space constraints.

As a counterpart to Theorem 2, we construct a family of graphs for which MIN DOM is polynomial-time solvable but DECIDE BIP-DOM is NP-complete.

Let \mathcal{F} be the set of all graphs. We now define the following family of graphs.

Definition 1. *The family of graphs \mathcal{F}' is defined to be the class of graphs, $H = G \cdot K_n^c$, obtained from each $G \in \mathcal{F}$ by adding n pendant vertices to every vertex of G , where $|V| = n$. i.e. let $G = (V, E) \in \mathcal{F}$, where $V = \{v_1, v_2, \dots, v_n\}$. Then $\mathcal{F}' = \{H = (V', E')\}$, where $V' = V \cup_{i=1}^n \{x_{i_1}, x_{i_2}, \dots, x_{i_n}\}$ and $E' = E \cup_{i=1}^n \{v_i x_{i_j} : j \in [n]\}$.*

Next, we show that MIN DOM is polynomial-time solvable for the family \mathcal{F}' .

Theorem 3 (★). *Let $H \in \mathcal{F}'$ be such that $H = G \cdot K_n^c$, then $\gamma(H) = n$.*

We then have the following corollary.

Corollary 1. *MIN DOM is solvable in polynomial time for the family \mathcal{F}' .*

Now, in order to show that DECIDE BIP-DOM is NP-complete for the family \mathcal{F}' , we need the INDUCED BIPARTITE SUBGRAPH problem which is known to be NP-complete [10].

INDUCED BIPARTITE SUBGRAPH

Instance: A graph $G = (V, E)$ and a positive integer k .

Question: Does there exist an induced bipartite subgraph of G on at least k vertices?

We then show that DECIDE BIP-DOM is NP-complete for the class \mathcal{F}' .

Theorem 4. *DECIDE BIP-DOM is NP-complete for the class \mathcal{F}' .*

Proof. Clearly, DECIDE BIP-DOM belongs to NP. To show the hardness, we give a polynomial reduction from the INDUCED BIPARTITE SUBGRAPH. Given an instance (G, k) of INDUCED BIPARTITE SUBGRAPH, we deduce an instance (G', k') of DECIDE BIP-DOM such that $G' = G \cdot K_n^c \in \mathcal{F}'$ (refer to Definition 1) and $k' = k + n(n - k)$. We now claim the following.

Claim 2. *The graph G has an induced bipartite subgraph of order at least k if and only if G' has a bipartite dominating set of cardinality at most $k + n(n - k)$.*

Proof of the Claim: Suppose G_1 is the induced bipartite subgraph of G such that $|V(G_1)| = k' \geq k$. Recall the definition of G' . Let $B = V(G_1) \cup_i \{x_{i_1}, x_{i_2}, \dots, x_{i_n} : v_i \notin V(G_1)\}$. It is easy to verify that B is a bipartite dominating set of G' with cardinality $k' + (n - k')n$. Since $k' \geq k$ and $n \geq 1$, we have $k'(1 - n) + n^2 \leq k(1 - n) + n^2$. This implies that $|B| \leq k + n(n - k)$.

Conversely, suppose that B is a bipartite dominating set of G' with cardinality at most $k + n(n - k)$. Let $B = B' \cup B''$, where $B' = B \cap V(G)$ and $B'' = B \setminus B'$.

Note that $|B| = |B'| + |B''| = |B'| + (n - |B'|)n$ (the second equality is due to the fact that B is a dominating set of G') and $|B| \leq k + n(n - k)$. This implies

that $|B'| + (n - |B'|)n \leq k + n(n - k)$, which further yields that $|B'| \geq k$. Now, since B is a bipartite dominating set of G' and $B' \subseteq B$ this implies that $G[B']$ is an induced bipartite subgraph of G of order at least k . Thus, the claim follows. \square

Hence, DECIDE BIP-DOM is NP-complete for the graph class \mathcal{F}' . \square

Therefore, from Theorem 3 and Theorem 4, we can conclude that MIN DOM is polynomial-time solvable for the graph class \mathcal{F}' , whereas DECIDE BIP-DOM is NP-complete for \mathcal{F}' .

5 Polynomial-time Algorithms for Interval Graphs and Block Graphs

In this section, we show that MIN BIP-DOM can be solved in polynomial time for interval graphs and block graphs.

5.1 Interval Graphs

We prove the following theorem.

Theorem 5. *Every minimum dominating set of an interval graph is a bipartite dominating set.*

Proof. Let $G = (V, E)$ be an interval graph and let $\sigma = (v_1, v_2, \dots, v_n)$ be an interval ordering of G . Let D be a minimum dominating set of G . To show $G[D]$ is bipartite in G , it is sufficient to show that $G[D]$ is C_3 -free (since interval graphs form a subclass of chordal graphs and chordal graphs are $C_n (n > 3)$ -free). For the sake of contradiction, assume that $G[D]$ has a C_3 induced by the vertices say, v_i, v_j and v_k in G . Without loss of generality, we can assume that $i < j < k$ with respect to σ . Thus, by the definition of interval ordering, for an $l \geq i$, if $v_l \in N[v_i] \cup N[v_j]$ then v_l is adjacent to v_k . Therefore, any vertex v_l for $l \geq i$, which is dominated by v_i or v_j in G is also dominated by v_k in G . Now for the vertices $v_l \in N[v_i] \cup N[v_j]$ such that $l \leq i$, we discuss the following two cases:

For a vertex $v_q \in V(G)$, let $\min(v_q)$ denote the minimum index of the vertices in $N[v_q]$ with respect to σ . Note that for $i \leq j$, $\min(v_i)$ can be smaller or larger than $\min(v_j)$. (For instance, for the graph in Fig. 3, $\min(v_1) = \min(v_3) = 1$ and $\min(v_2) = \min(v_4) = \min(v_5) = 2$). We then have two cases.

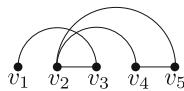


Fig. 3. An example of interval graph along with an interval ordering

Case-1: $\min(v_i) < \min(v_j)$:

Clearly, for each p such that $\min(v_i) \leq p \leq i$, we have v_p adjacent to v_i and hence dominated by v_i . This together with a previous observation implies that $D \setminus \{v_j\}$ is a dominating set in G , a contradiction to the minimality of D .

Case-2: $\min(v_i) \geq \min(v_j)$:

For each p such that $\min(v_j) \leq p \leq i$, we have v_p adjacent to v_j and hence dominated by v_j . As in the previous case, this implies that $D \setminus \{v_i\}$ is a dominating set in G , a contradiction.

As we have a contradiction in both cases, we can conclude that every dominating set for an interval graph is also a bipartite dominating set. \square

For any interval graph G , Theorem 5 implies that $\gamma(G) \geq \gamma_{bip}(G)$. Since, we always have $\gamma(G) \leq \gamma_{bip}(G)$, this implies that $\gamma(G) = \gamma_{bip}(G)$. Further, since the MIN DOM is solvable in linear time for interval graphs [7], the MIN BIP-DOM can also be then solved in linear time for interval graphs.

5.2 Block Graphs

In this section, we design a polynomial-time algorithm for solving MIN BIP-DOM for block graphs. Note that as in Observation 1, the difference, $\gamma(G) - \gamma_{bip}(G)$ can be arbitrarily large even when G is a block graph (for example, see Fig. 1). First, we give some preliminary concepts on block graphs.

Note that in a block graph, two blocks intersect at a cut vertex and a cut vertex has one or more blocks incident to it. Due to this relation, we have a nice representation of a block graph, known as *block-cut tree*. Let $G = (V, E)$ be a block graph and \mathcal{B} be the set of blocks in G and \mathcal{C} be the set of cut vertices in G . The *block-cut tree* of G is a tree $T^* = (V^*, E^*)$, where

$$V^* = \mathcal{B} \cup \mathcal{C} \text{ and } E^* = \{(B, v) : B \in \mathcal{B} \text{ and } v \in \mathcal{C} \text{ such that } v \in V(B)\}.$$

Note that all leaves of a block-cut tree T^* are block vertices. i.e. from \mathcal{B} . The block-cut tree, T^* of a block graph G can be constructed using depth-first search in linear time [5]. If G has exactly one block, then G is a complete graph and $\gamma_{bip}(G) = 1$. Hence, we can assume that G is a connected block graph with at least two blocks. Here, we use a dynamic programming approach to find the bipartite domination number of block graphs. In particular, we process the *blocks* and *cut-vertices* of the input graph G in a bottom-up order with respect to the block-cut tree of G . Note that each block in G is a clique and therefore, any bipartite dominating set of G can not contain more than two vertices from each block. This fact is summarized in the following observation.

Observation 3. *Let G be a block graph. Then, every bipartite dominating set of G contains at most two vertices from each block of G .*

For a cut vertex u of a block graph G , let T_u^* denote the sub-tree of T^* rooted at u and let $G_u = G[\{v \in V(B) : B \in V(T_u^*) \cap \mathcal{B}\}]$. Now, we define the following sets:

U^Y : a minimum bipartite dominating set of G_u , which contains u .

U^N : a minimum bipartite dominating set of G_u , which does not contain u .

\bar{U} : a minimum bipartite dominating set of $G_u - u$.

Further, we are interested in the parameters, $u^Y = |U^Y|$, $u^N = |U^N|$, and $\bar{u} = |\bar{U}|$.

We consider T^* to be rooted at a cut vertex, say r . Hence, by the definition of G_r , we have $G_r = G$ and therefore, $\gamma_{bip}(G) = \min\{r^Y, r^N\}$. Now, before defining the recurrence relation for the above parameters, we observe some properties of bipartite dominating sets for block graphs. The observation below easily follows from Observation 3 and the definitions of U^Y, U^N , and \bar{U} .

Observation 4. *For a cut vertex u of G , let the blocks B_1, B_2, \dots, B_l be the children of u in T^* . Then,*

- (a) $|U^Y \cap V(B_i - u)| \leq 1$, for each $i \in [n]$.
- (b) $|U^N \cap V(B_j)| \geq 1$, for some $j \in [l]$ and $|U^N \cap V(B_i)| \leq 2$, for each $i \in [l]$.
- (c) $|\bar{U} \cap V(B_i)| \leq 2$, for each $i \in [l]$.

From Observation 4(b), it is clear that the bipartite dominating set U^N contains at least one vertex (other than u) from at least one of the child block of u , say B_j , for some $j \in [l]$, and for each child $B_i, i \neq j$, we have either $|U^N \cap V(B_i)| = 0$ or $|U^N \cap V(B_i)| = 1$ or $|U^N \cap V(B_i)| = 2$. Consequently, we have many different possible cases to handle while computing the parameter u^N . Therefore, if we try to write the recurrence for u^N , by only involving the existing parameters: v^Y, v^N , and \bar{v} , where v is a cut vertex belonging to a child block of u (in other words, grandchildren of u in T^*), the recurrence formula for u^N becomes quite complicated. Hence, for the sake of convenience, we introduce the following parameters for each block vertex in T^* as well.

Let u be a cut vertex in G . Note that the children of u in T^* correspond to blocks containing u in G . Let B be a non-leaf child vertex of u in T^* and let C be the set of cut vertices of $B - u$ in G . We then define the following:

B^0 : a minimum bipartite dominating set of $G_u - u$ such that $|B^0 \cap C| = 0$.

B^1 : a minimum bipartite dominating set of $G_u - u$ such that $|B^1 \cap C| = 1$.

B^2 : a minimum bipartite dominating set of $G_u - u$ such that $|B^2 \cap C| = 2$.

For a cut-vertex u in G , let \mathcal{L}_u denote the set of children of u that are leaves in T^* and let \mathcal{N}_u denote the set of children of u that are non-leaves in T^* . (Note that the elements of \mathcal{L}_u and \mathcal{N}_u are blocks containing u in G). Then, we have three kinds of cut vertices in the block-cut tree, T^* as defined below:

- (a) *Type-1* if all the children of u in T^* are leaves, i.e. $\mathcal{N}_u = \emptyset$,
- (b) *Type-2* if all the children of u in T^* are non-leaves, i.e. $\mathcal{L}_u = \emptyset$, and
- (c) *Type-3* if $\mathcal{L}_u \neq \emptyset$ and $\mathcal{N}_u \neq \emptyset$.

We now have the following lemmas for finding the recurrence relation of the parameters defined for block vertices and each type of cut vertices.

Lemma 1 (★). Let G be a block graph and T^* its block-cut tree. If u is a Type-1 cut vertex of T^* then we have $u^Y = 1$ and $u^N = \bar{u} = |\mathcal{L}_u|$.

Lemma 2 (★). Let G be a block graph and T^* its block-cut tree. For a cut vertex u of G , let B be a non-leaf child vertex of u in T_u^* , and let C be the set of cut vertices of $B - u$ in G such that for each $v \in C$, we have computed the parameters v^Y, v^N , and \bar{v} correctly. Then we have,

$$\begin{aligned} |B^0| &= \begin{cases} 1 + \sum_{v \in C} \bar{v} & , \text{ if } B \text{ contains at least one non-cut vertex of } G \\ \sum_{v \in C} v^N & , \text{ otherwise} \end{cases} \\ |B^1| &= \min_{w \in C} \left\{ w^Y + \sum_{v \in C, v \neq w} \bar{v} \right\} \\ |B^2| &= \begin{cases} \min_{w_1, w_2 \in C} \left\{ w_1^Y + w_2^Y + \sum_{v \in C, v \neq w_1, w_2} \bar{v} \right\} & , \text{ if } |C| \geq 2 \\ \infty & , \text{ otherwise} \end{cases} \end{aligned}$$

Lemma 3 (★). Let G be a block graph and T^* its block-cut tree. Let u be a Type-2 cut vertex of T^* such that for each non-leaf child B of u in T^* , we have computed the parameters B^0, B^1 , and B^2 correctly. Then we have,

$$\begin{aligned} u^Y &= 1 + \sum_{B \in \mathcal{N}_u} \min\{|B^0|, |B^1|\} \\ u^N &= \min_{B \in \mathcal{N}_u} \left\{ \min\{|B^1|, |B^2|\} + \sum_{B' \in \mathcal{N}_u, B' \neq B} \min\{|B'^0|, |B'^1|, |B'^2|\} \right\} \\ \bar{u} &= \sum_{B \in \mathcal{N}_u} \min\{|B^0|, |B^1|, |B^2|\} \end{aligned}$$

Lemma 4 (★). Let G be a block graph and T^* its block-cut tree. Let u be a Type-3 cut vertex of T^* such that for each non-leaf child B of u in T^* , we have computed the parameters B^0, B^1 , and B^2 correctly. Then we have,

$$\begin{aligned} u^Y &= 1 + \sum_{B \in \mathcal{N}_u} \min\{|B^0|, |B^1|\} \\ u^N &= |\mathcal{L}_u| + \sum_{B \in \mathcal{N}_u} \min\{|B^0|, |B^1|, |B^2|\} \\ \bar{u} &= |\mathcal{L}_u| + \sum_{B \in \mathcal{N}_u} \min\{|B^0|, |B^1|, |B^2|\} \end{aligned}$$

Based on the above four lemmas, we now design Algorithm 1, which computes the bipartite domination number for a given block graph $G = (V, E)$ with at least two blocks.

Theorem 6. Given a block graph $G = (V, E)$, Algorithm 1 computes $\gamma_{bip}(G)$ in $O(n^3)$ time.

Proof. Correctness of the algorithm: Given a block graph G , let T^* be its block-cut tree, rooted at a cut vertex r and $\alpha = (v_1, v_2, \dots, v_s = r)$ be the reverse BFS ordering of T^* . Note that for some $j \in [s]$, if v_j is a leaf in T^* then $v_j \in \mathcal{B}$ and the block representing v_j is a leaf-block in G . In this case, Algorithm 1 does not compute anything in the j^{th} iteration. Now, for each $i \in [s]$, we claim the following.

Claim 1.

- (a) If $v_i \in \mathcal{C}$ then the algorithm computes the parameters v_i^Y, v_i^N , and \bar{v}_i correctly.
- (b) If $v_i \in \mathcal{B}$ is a non-leaf vertex in T^* such that $v_i = B$, where B is the block in G representing v_i in G then the algorithm computes the parameters $|B^0|, |B^1|$, and $|B^2|$ correctly.

Proof of the Claim: We prove this by induction on i .

Since α is a reverse BFS ordering of T^* , v_1 is a leaf in T^* and $v_1 \in \mathcal{B}$. Thus the claim is trivially true. Let v_i be the minimum indexed vertex in α that is not a leaf in T^* . Again, using the fact that α is a reverse BFS ordering, it is not difficult to see that $v_i \in \mathcal{C}$ and v_i is a *Type-1* cut vertex. Therefore, by Lemma 1, Algorithm 1 computes the parameters, v_i^Y, v_i^N , and \bar{v}_i correctly, and this proves the base case of the claim.

Algorithm 1: MIN-BIP-DOM-BLOCK(G)

Input: A block graph $G = (V, E)$ with at least two blocks.

Output: $\gamma_{bip}(G)$.

begin

 Compute the block-cut tree T^* of G rooted at a cut vertex r of G ;

 Compute a reverse BFS ordering $\alpha = (v_1, v_2, \dots, v_s = r)$ of T^* ;

for $i = 1$ to s **do**

if v_i is a *Type-1* cut vertex of T^* **then**

 | Compute the parameters v_i^Y, v_i^N , and \bar{v}_i as in Lemma 1;

end

else if v_i is a *Type-2* cut vertex of T^* **then**

 | Compute the parameters v_i^Y, v_i^N , and \bar{v}_i as in Lemma 3;

end

else if v_i is a *Type-3* cut vertex of T^* **then**

 | Compute the parameters v_i^Y, v_i^N , and \bar{v}_i as in Lemma 4;

end

else if v_i is a non-leaf block vertex of T^* ; let $v_i = B$, where B is the

 block in G representing the vertex v_i in T^* **then**

 | Compute the parameters $|B^0|, |B^1|$, and $|B^2|$ as in Lemma 2;

end

end

$\gamma_{bip}(G) = \min\{r^Y, r^N\}$, where $r = v_s$ is the root of T^* ;

return $\gamma_{bip}(G)$;

end

Now, assume that for all $j < i$, the claim holds. Consider the vertex v_i . If v_i is a leaf in T^* then again, the claim is trivially true. Therefore, we can assume that v_i is a non-leaf vertex in T^* . We now have the following cases:

Case-1: $v_i \in \mathcal{C}$:

If v_i is a *Type-1* cut-vertex then by Lemma 1, Algorithm 1 computes the parameters, v_i^Y, v_i^N , and \bar{v}_i correctly. Note that a reverse BFS on the rooted tree T^* visits a parent vertex in T^* only after visiting all its children. Thus, by the induction hypothesis, for each non-leaf child v_j of v_i , the corresponding parameters for v_j are computed correctly. Now, if v_i is a *Type-2* (respectively, *Type-3*) cut vertex, then by Lemma 3 (respectively, Lemma 4), v_i^Y, v_i^N , and \bar{v}_i are computed correctly (note that in the assumptions of Lemma 3 and Lemma 4, we need to know only the the correct values of the parameters for the non-leaf children of v_i in T^*).

Case-2: $v_i \in \mathcal{B}$:

Let $v_i = B$, where B is the block in G representing v_i . Recall that v_i is a non-leaf vertex in T^* . Let C be the set of cut vertices of $B - v_k$, where v_k is the parent of v_i in T^* . Clearly, all the vertices in C are children of v_i in T^* and each of them is a non-leaf vertex in T^* (as they are cut-vertices in G). Now, again by the induction hypothesis, for each $v_j \in C$, the corresponding parameters for v_j are computed correctly. Therefore, by Lemma 2, the algorithm finds $|B^0|, |B^1|$, and $|B^2|$ correctly. Hence, the claim.

Recall that $\gamma_{\text{bip}}(G) = \min\{r^Y, r^N\}$. Since $r \in \mathcal{C}$, by Claim 1(a), Algorithm 1 computes r^Y and r^N correctly. This concludes the correctness of the algorithm.

Running time of the algorithm: The block-cut tree, T^* rooted at a cut vertex r of G can be computed in $O(n+m)$ time. The number of vertices in the block-cut tree T^* is also $O(n)$. Now, we compute the reverse BFS ordering, α of T^* , which takes $O(n)$ time. It can be seen that a single iteration of the algorithm takes only $O(n^2)$ time (by Lemmas 1, 2, 3, and 4). Therefore, we can conclude that Algorithm 1 runs in $O(n^3)$ time.

This completes the proof of the theorem. \square

6 Conclusion

In this paper, we proposed polynomial-time algorithms for the MIN BIP-DOM problem for interval graphs and block graphs. It would be interesting to design polynomial time algorithms for MIN BIP-DOM problem for chordal graphs.

References

1. Bachstein, A., Goddard, W.: The lower bipartite number of a graph. *Math. Pannon.* **29**(1), 49–56 (2023)
2. Bachstein, A., Goddard, W., Henning, M.: Bipartite domination in graphs. *Math. Pannon.* **28**, 118–126 (2022)
3. Bertossi, A.A.: Dominating sets for split and bipartite graphs. *Inf. Process. Lett.* **19**(1), 37–40 (1984)

4. Cockayne, E.J., Dawes, R.M., Hedetniemi, S.T.: Total domination in graphs. *Networks* **10**(3), 211–219 (1980)
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT press, Cambridge (2022)
6. Goddard, W., Henning, M.A.: Independent domination in graphs: a survey and recent results. *Disc. Math.* **313**(7), 839–854 (2013)
7. Haynes, T.W., Hedetniemi, S., Slater, P.: Fundamentals of Domination in Graphs. CRC Press, Boca Raton (2013)
8. Haynes, T.W., Hedetniemi, S.T., Henning, M.A. (eds.): Topics in Domination in Graphs. DM, vol. 64. Springer, Cham (2020)
9. Hedetniemi, S.T., Haynes, T.W., Slater, P.J.: Domination in Graphs: Advanced Topics. Marcel Dekker, New York (1998)
10. Lewis, J.M., Yannakakis, M.: The node-deletion problem for hereditary properties is NP-complete. *J. Comput. Syst. Sci.* **20**(2), 219–230 (1980)
11. Pelias, W.P., Cabahug, I.S., Jr.: Bipartite domination number of mycielski graph of some graph families. *Asian Res. J. Math.* **19**(5), 41–50 (2023)
12. Pelias, W.P., Cabahug, I.S., Jr.: Bipartite domination in some classes of graphs. *Asian Res. J. Math.* **19**(3), 8–17 (2023)



Packing Sets of Paths, Stars and Triangles: Tractability and Approximability

Santiago Valdés Ravelo^(✉) and Flávio K. Miyazawa

Institute of Computing, University of Campinas, Campinas, Brazil
`{santiago, fkm}@ic.unicamp.br`

Abstract. The Maximum Weighted Graph Set Packing (MAXGP) problem is defined as follows: given a graph G , a set \mathcal{G} of subgraphs of G , and a weight function $\omega : \mathcal{G} \rightarrow \mathbb{Q}^+$, the objective is to find a subset $S \subseteq \mathcal{G}$ that maximizes $\sum_{H \in S} \omega(H)$ and ensures that each pair of elements in S are edge-disjoint. This work introduces MAXGP and three specific cases of the problem, where \mathcal{G} contains only subgraphs from a particular class: stars (MAXSP), paths (MAXPP), or triangles (MAXTP).

For the unweighted versions of these problems (where $\omega(H) = 1$ for all $H \in \mathcal{G}$), we derived several intractability and inapproximability results. We proved that both unweighted MAXSP and unweighted MAXPP are W[1]-hard and APX-hard, even when the maximum degree of G is restricted to 3. Additionally, we demonstrated that these two problems are not fixed-parameter tractable (FPT) when parameterized by the treewidth or the maximum degree of G , but they do admit fixed-parameter algorithms when parameterized by both treewidth and maximum degree. For MAXTP, we showed the existence of a fixed-parameter algorithm when parameterized solely by the treewidth of G .

Moreover, we proved that the unweighted versions of MAXSP, MAXPP, and MAXTP are NP-hard on planar graphs. Using our FPT results, we proposed polynomial-time approximation schemes (PTASs) for these versions on planar graphs.

Keywords: Graph packing · Disjoint edges · Weighted packing · Fixed-parameter tractability · Treewidth · Polynomial-time approximation scheme

1 Introduction

Graph packing problems have been extensively studied by the scientific community due to their theoretical complexity and practical applications. Many of these problems involve finding a maximum set S of subgraphs of a given graph

Supported by São Paulo Research Foundation (FAPESP) grants 2015/11937-9, 2022/06728-5, and 2022/05803-3, and National Council for Scientific and Technological Development (CNPq) grants 404315/2023-2, and 313146/2022-5.

G , where the subgraphs belong to a specific class (e.g., stars, paths, triangles, cliques, trees) and are either pairwise disjoint or edge-disjoint [3, 5, 6]. Most of these problems are NP-hard, prompting the development of approximation and fixed-parameter algorithms for general or special cases.

When, in addition to G , the problem is given a set \mathcal{G} of possible subgraphs to be packed, along with a weight function $\omega : \mathcal{G} \rightarrow \mathbb{Q}^+$ associated with the subgraphs, the problem becomes the Maximum Weighted Graph Set Packing (MAXGP). This generalization might increase the problem's difficulty and encompasses several previously studied problems. For instance, the Edge-Disjoint Triangle Packing (ETP) is a special case of MAXGP where \mathcal{G} consists of all triangles in G , and $\omega(H) = 1$ for each $H \in \mathcal{G}$.

Although graph packing problems have been widely studied in the computer science literature, there has been no theoretical results focused on packing a given set of subgraphs. The only previous works where the input includes a specific collection of subgraphs dealt with minimum exact edge covering problems [8, 9]. This motivated our investigation into the Maximum Weighted Graph Set Packing (MAXGP) problem and three of its specific cases: MAXSP, where \mathcal{G} contains only stars; MAXPP, where \mathcal{G} contains only paths; and MAXTP, where \mathcal{G} contains only triangles. Below, we summarize the main contributions of this paper:

- We prove that MAXSP, MAXPP, and MAXTP are APX-hard on general graphs and NP-hard on planar graphs, even with bounded degree. Additionally, we establish that MAXSP and MAXPP are W[1]-hard.
- We present a $2^{O(k \cdot \Delta^\ell + 2^{\Delta^\ell})} \cdot n$ -time algorithm for MAXGP on graphs with n vertices, treewidth k , and maximum degree Δ , where the longest path in each subgraph of the collection is bounded by ℓ . From this result, we derive a $2^{O(k \cdot \Delta^2 + 2^{\Delta \Delta})} \cdot n$ -time algorithm for MAXSP, and a $2^{O(k^3)} \cdot n$ -time algorithm for MAXTP.
- Furthermore, we propose a PTAS (Polynomial-Time Approximation Scheme) for MAXGP on planar graphs with bounded maximum degree and bounded longest path for the subgraphs in the collection. Specifically, we provide a PTAS for MAXSP on planar graphs with bounded degree, and for MAXTP on any planar graph.

The remainder of this document is organized as follows. First, we introduce basic graph notation and formally define the problems we study. In Sect. 2, we present NP-hardness, inapproximability, and intractability results for each problem. Section 3 proposes fixed-parameter algorithms, which are then used in Sect. 4 to derive approximation results. Finally, in Sect. 5, we give some conclusions and discuss open questions for future research.

1.1 Graph Notation and Problems Definitions

In this section, we formally define the problems studied in this document. First, we introduce the basic graph notation necessary for their specification. Given a graph G , let V_G denote its set of vertices and E_G its set of edges. We focus

on two key graph relations: the subgraph relation and the edge-disjoint relation. A graph H is a **subgraph** of a graph G (denoted $H \subseteq G$) if each vertex of H is a vertex of G and every edge of H is also an edge of G (i.e., $V_H \subseteq V_G$ and $E_H \subseteq E_G$).

Given two graphs H_1 and H_2 , we say they are **edge-disjoint** if they share no edges (i.e., $E_{H_1} \cap E_{H_2} = \emptyset$). Note that edge-disjoint graphs may still share vertices.

With these concepts in place, we now define the primary problem addressed in this research.

Problem 11. *Maximum Weighted Graph Set Packing (MAXGP)*

Input: A tuple $\langle G, \mathcal{G}, \omega \rangle$, where G is a graph, \mathcal{G} is a set of subgraphs of G , and $\omega : \mathcal{G} \rightarrow \mathbb{Q}^+$ is a non-negative weight function defined on the elements of \mathcal{G} .

Output: A set $S \subseteq \mathcal{G}$ of pairwise edge-disjoint subgraphs that maximizes the weighted sum of its elements (i.e., for every pair of distinct graphs $H_1, H_2 \in S$, the graphs are edge-disjoint, and S maximizes $\sum_{H \in S} \omega(H)$).

As we will see in the following sections, solving the above problem is challenging, even when considering specific classes of graphs. Therefore, we focus on three particular cases, where the set \mathcal{G} contains only one of the following graph classes: stars, paths, or triangles.

A **k -star** is a graph with $k + 1$ vertices: a central vertex u and k leaves. The edges are defined between u and each leaf (i.e., the set of edges is $\{\langle u, v \rangle \mid v \text{ is one of the } k \text{ leaves}\}$).

A **path** of length k (or a **k -length path**) is a graph with $k + 1$ vertices, where the vertices are ordered as $u_1 u_2 \dots u_{k+1}$, and edges are defined between consecutive vertices (i.e., the set of edges is $\{\langle u_i, u_{i+1} \rangle \mid 1 \leq i \leq k\}$).

A **triangle** is a graph with exactly three vertices and three edges, one edge between each pair of vertices.

The following three definitions specify the particular cases of MAXGP that we study.

Problem 12. *Maximum Weighted Star Set Packing (MAXSP). The particular case of MAXGP where \mathcal{G} only contains stars of G .*

Problem 13. *Maximum Weighted Path Set Packing (MAXPP). The particular case of MAXGP where \mathcal{G} only contains paths of G .*

Problem 14. *Maximum Weighted Triangle Set Packing (MAXTP). The particular case of MAXGP where \mathcal{G} only contains triangles of G .*

Since we analyze whether these problems are Fixed-Parameter Tractable (FPT) with respect to treewidth and develop algorithms based on this parameter, we provide a brief definition. A **tree decomposition** of a graph G is a pair $\mathcal{T} = \langle T, \{X_t\}_{t \in V_T} \rangle$, where T is a tree and each node t of T is associated with a set of vertices $X_t \subseteq V_G$ called a bag. The decomposition must satisfy the following conditions: (i) the union of all bags equals V_G (i.e., $\bigcup_{t \in V_T} X_t = V_G$);

(ii) every edge of G has both endpoints in at least one bag (i.e., $\forall uv \in E_G, \exists t \in V_T : u, v \in X_t$); and (iii) for every vertex u in G , the bags containing u induce a connected subtree of T (i.e., $\forall u \in V_G, T_u = \{t \in V_T \mid u \in X_t\}$ induces a connected subtree of T). The size of the largest bag minus one, in a tree decomposition of G is called the **width** of the decomposition, and the **treewidth** of G is the minimum width of any tree decomposition of G .

The next section discusses the complexity, intractability, and approximation bounds for these problems.

2 Complexity

MAXGP generalizes MAXSP, MAXPP, and MAXTP, making it at least as hard, intractable, and inapproximable as its specific cases. Therefore, in this section, we focus on analyzing MAXSP, MAXPP, and MAXTP. Specifically, we examine the unweighted versions of these problems, where the objective is to maximize the cardinality of the solution rather than the weighted sum of its elements. Thus, we assume that MAXSP, MAXPP, and MAXTP refer to their unweighted variants.

We begin our complexity study with MAXSP, which appears to be the least tractable of the three problems. Next, we analyze MAXPP. For both of these problems, we propose reductions from two well-known NP-hard problems: Maximum Independent Set (MIS) and Maximum 3-Dimensional Matching (MAX3DM). These reductions allow us to establish not only proofs of NP-hardness but also fixed-parameter intractability results and approximation bounds. Finally, we discuss the complexity of MAXTP, primarily based on previous studies of a specific case of this problem.

Before presenting our findings, we formally define MIS and MAX3DM.

Problem 21. *Maximum Independent Set (MIS).*

Input: A graph G .

Output: A maximum cardinality set of vertices $I \subseteq V_G$, such that there is no edge of G with both endpoints in I .

Problem 22. *Maximum 3-Dimensional Matching (MAX3DM).*

Input: A tuple $\langle A, B, C, T \rangle$, where A , B , and C are pairwise disjoint sets, and $T \subseteq A \times B \times C$.

Output: A maximum cardinality pairwise disjoint set $T \subseteq T$.

2.1 MAXSP

As mentioned before, MAXSP appears to be quite difficult to solve. We will demonstrate that the problem is APX-hard even when G is a tree or a general graph with maximum degree bounded by 3. First, we prove the following theorem by a simple reduction from MIS.

Theorem 1. MAXSP is NP-hard even on graphs with maximum degree at most 3, W[1]-hard with respect to the size of an optimal solution, and, denoting by n the number of vertices in an arbitrary instance, for any $\epsilon > 0$, there is no polynomial time $n^{1-\epsilon}$ -approximation algorithm for MAXSP unless $P = NP$.

Despite the hardness established for MAXSP, Theorem 1 does not give any light on the problem's difficulty when G belongs to a simpler class of graphs. Specifically, is MAXSP NP-hard on trees? Since MIS is solvable in polynomial time on trees [4], we cannot address this question using the previous reduction. However, we demonstrate that MAXSP remains NP-hard even on trees by a reduction from MAX3DM.

Theorem 2. MAXSP is APX-hard even on instances $\langle G, \mathcal{T} \rangle$ where G is a star, \mathcal{G} is a collection of 3-stars, and each edge of G appears in at most three elements of \mathcal{G} .

2.2 MAXPP

Following the analysis of MAXSP, this section presents polynomial reductions from MIS and MAX3DM to MAXPP, which yield results regarding the problem's NP-hardness, inapproximability, and intractability. As in the previous section, the first result is obtained by a reduction from MIS.

Theorem 3. MAXPP is NP-hard, W[1]-hard with respect to the size of an optimal solution, and, denoting by $\langle G, \mathcal{G} \rangle$ an arbitrary instance, for any $\epsilon > 0$, there is no polynomial time $|\mathcal{G}|^{1-\epsilon}$ -approximation algorithm for MAXPP unless $P = NP$, even if the maximum degree of G is bounded by 3 and the lengths of the paths in \mathcal{G} are bounded by 12.

The result above does not provide any insights into the hardness of MAXPP on graphs with bounded treewidth. Therefore, we present an alternative reduction from MAX3DM, showing that MAXPP remains NP-hard even when the graph G is planar with treewidth bounded by 2, and the paths in \mathcal{G} have lengths of at most 4.

Theorem 4. MAXPP is APX-hard even on instances $\langle G, \mathcal{G} \rangle$, where G is a planar graph with treewidth at most 2, and \mathcal{G} is a collection of 4-length paths.

2.3 MAXTP

MAXTP appears to be the most tractable version of MAXGP among those studied in this document. Its complexity is established through a straightforward reduction from the NP-hard Edge-Disjoint Triangle Packing problem (ETP), defined below.

Problem 23. Edge-Disjoint Triangle Packing problem (ETP).

Input: A graph G .

Output: A maximum cardinality pairwise edge-disjoint set of triangles S of G .

Theorem 5. MAXTP is NP-hard even on planar graphs with maximum degree 5, and APX-hard on general graphs with maximum degree 4.

Proof. Notice that ETP is a specific case of MAXTP, where \mathcal{G} contains all triangles of G . Since a graph G with n vertices can have at most $O(n^3)$ triangles, any instance of ETP can be transformed in polynomial time into an equivalent instance of MAXTP. This transformation preserves both the complexity and the approximation factor.

In [3], the authors proved that ETP is NP-hard even for planar graphs with a maximum degree of 5, and APX-hard for general graphs with a maximum degree of 4. \square

3 FPT Algorithm

Theorems 1, 2, 3, and 4 demonstrate that MAXSP and MAXPP are NP-hard when the input graph has a treewidth bounded by 2 or a maximum degree bounded by 3. Thus, unless P = NP, these problems are not fixed-parameter tractable (FPT) when parameterized by either treewidth or maximum degree alone. Consequently, we consider parameterization with the combination of these two parameters. Additionally, for MAXPP, we impose that the paths in the collection have lengths less than or equal to a fixed integer value $\ell > 0$, since for $\ell \leq 12$ the problem is NP-hard even with a maximum degree bounded by 3, and for $\ell \leq 4$, MAXPP remains NP-hard even with a treewidth bounded by 2.

This section presents a fixed-parameter algorithm using both treewidth and maximum degree as parameters for a specific case of MAXGP, which generalizes MAXSP and MAXPP. We will also explain how this approach can be adapted for treewidth parameterization in MAXTP.

First, we define a particular tree decomposition and establish some notation.

3.1 Nice Tree Decomposition and Some Notation

Our approach considers a **nice tree decomposition** $\mathcal{T} = \langle T, \{X_t\}_{t \in V_T} \rangle$ of the input graph G . In this decomposition, T is a binary tree, and each of its nodes belongs to one of the following four categories [4]: (i) **leaf node**: a leaf of T ; (ii) **introduce node**: a node $t \in V_T$ with a single child $t' \in V_T$, such that $X_t = X_{t'} \cup \{u\}$ for some $u \in V_G \setminus X_{t'}$; (iii) **forget node**: a node $t \in V_T$ with a single child $t' \in V_T$, such that $X_t = X_{t'} \setminus \{u\}$ for some $u \in X_{t'}$; and (iv) **join node**: a node $t \in V_T$ with two children $t_1, t_2 \in V_T$, such that $X_t = X_{t_1} = X_{t_2}$.

Given a graph G with treewidth k and n vertices, there exists a $2^{O(k)}n$ time algorithm that produces a tree decomposition of G with width at most $2k + 1$ [7]. Furthermore, from such a decomposition, one can derive a nice tree decomposition of G with the same width, containing $O(k \cdot n)$ nodes, where the leaves are empty bags, in $O(k^2 \cdot n)$ time [4]. Therefore, we assume that we are provided with a nice tree decomposition of G that has width at most $2k + 1$,

$O(k \cdot n)$ nodes, and the leaves are associated to empty bags. It is important to note that the time required to compute this decomposition ($2^{O(k)}n$) will be included in the overall complexity of the algorithm.

Additionally, consider the following notation for a node $t \in V_T$: G_t denotes the subgraph of G induced by the vertices in the bags of the subtree of T rooted at t ; \mathcal{G}_t denotes the subset of \mathcal{G} containing the elements that are subgraphs of G_t ; and for $\ell \in \mathbb{N}$, E_t^ℓ denotes the subset of edges of G_t with at least one endpoint at a distance no greater than ℓ from some node in the bag X_t . Additionally, given any subset $Y \subseteq \mathcal{G}$, $E(Y) = \bigcup_{H \in Y} E_H$.

3.2 Treewidth Parameterization

The main result of this section is as follows:

Theorem 6. *Any instance $\langle G, \mathcal{G}, \omega \rangle$ of MAXGP can be solved within $2^{O(k \cdot \Delta^\ell + 2^{\Delta^\ell})} \cdot n$ time, where G is a graph with n vertices, treewidth k , and maximum degree Δ , and the longest path between two vertices of any element of \mathcal{G} is ℓ .*

Proof. Let $\langle G, \mathcal{G}, \omega \rangle$ be an instance of MAXGP, where n denotes the number of vertices in G , k its treewidth, Δ its maximum degree, ℓ the length of the longest path in any element of \mathcal{G} , and $\langle T, \{X_t\}_{t \in V_T} \rangle$ a nice tree decomposition of G with width at most $2k + 1$, $O(k \cdot n)$ nodes, and leaves associated with empty bags.

For each node $t \in V_T$, any element of \mathcal{G}_t incident to a vertex in X_t has all its edges in E_t^ℓ . Therefore, for each subset S of E_t^ℓ , we consider an optimal solution that uses all edges in S and does not utilize any edge in $E_t^\ell \setminus S$. We denote by $\pi[t, S]$ the optimal value of MAXGP for the instance $\langle G_t, \mathcal{G}_t, \omega \rangle$ that includes all edges in S and excludes the edges in $E_t^\ell \setminus S$. Below, we show how to compute the dynamic table π at a node $t \in V_T$ for a set $S \subseteq E_t^\ell$:

- *Leaf node.* If t is a leaf node, then $E_t^\ell = \emptyset$, which implies $\pi[t, \emptyset] = 0$.
- *Introduce node.* If t is an introduce node, let $t' \in V_T$ be its child, $u \in V_G$ the only vertex in $X_t \setminus X_{t'}$, S_u the subset of edges of S incident to u , $\mathcal{G}_t^{S_u}$ the subset of \mathcal{G}_t containing elements that have at least one edge in S_u and no edges in $E_t^\ell \setminus S$, and $2^{\mathcal{G}_t^{S_u}}$ the set of all pairwise edge-disjoint subsets of $\mathcal{G}_t^{S_u}$ that utilize each edge in S_u .
 - If $S_u = \emptyset$, then $\pi[t, S] = \pi[t', S]$, since no edge incident to u can be included in the solution.
 - If $2^{\mathcal{G}_t^{S_u}} = \emptyset$, then $\pi[t, S] = -\infty$ because no packing of elements in \mathcal{G}_t can use all edges of S without using edges from $E_t^\ell \setminus S$.
 - If $2^{\mathcal{G}_t^{S_u}} \neq \emptyset$, we iterate through each subset $Y \in 2^{\mathcal{G}_t^{S_u}}$ that covers all edges in S_u (i.e., $S_u \subseteq E(Y)$). If Y is part of a solution, the remainder must be an optimal packing Y' that utilizes each edge in $S \setminus E(Y)$ while avoiding any edges in $E_t^\ell \setminus (S \setminus E(Y))$. Since $S_u \subseteq E(Y)$, all edges in $S \setminus E(Y)$ belong to $E_{t'}^\ell$, meaning that Y' must be a solution for $\pi[t', S \setminus E(Y)]$.

Therefore, we conclude:

$$\pi[t, S] = \begin{cases} \max_{Y \in 2^{\mathcal{G}_t^{S_u}}} \{\omega(Y) + \pi[t', S \setminus E(Y)]\} & \text{if } 2^{\mathcal{G}_t^{S_u}} \neq \emptyset \\ \pi[t', S] & \text{if } S_u = \emptyset \\ -\infty & \text{otherwise.} \end{cases}$$

- *Forget node.* If t is a forget node, let $t' \in V_T$ be its child, and $u \in V_G$ the only vertex in $X_{t'} \setminus X_t$. A solution for $\pi[t, S]$, in addition to using all edges of S , may utilize some edges in $E_{t'}^\ell \setminus E_t^\ell$. Thus, such a solution can be computed by iterating over each subset $Y \subseteq (E_{t'}^\ell \setminus E_t^\ell)$ and selecting the one that maximizes $\pi[t', S \cup Y]$. Hence, we have:

$$\pi[t, S] = \max_{Y \subseteq (E_{t'}^\ell \setminus E_t^\ell)} \{\pi[t', S \cup Y]\}.$$

- *Join node.* If t is a join node, let $t_1, t_2 \in V_T$ be its children. The only edges that G_{t_1} and G_{t_2} may share are those with both endpoints in $X_t = X_{t_1} = X_{t_2}$. Therefore, a solution for $\pi[t, S]$ is obtained by combining a solution for $\pi[t_1, S_1]$ and another for $\pi[t_2, S_2]$, such that $S = S_1 \cup S_2$ and $S_1 \cap S_2 = \emptyset$. Consequently, we can iterate over each subset $S' \subseteq S$, selecting the one that maximizes $\pi[t_1, S'] + \pi[t_2, S \setminus S']$, yielding:

$$\pi[t, S] = \max_{S' \subseteq S} \{\pi[t_1, S'] + \pi[t_2, S \setminus S']\}.$$

Now, we analyze the computational effort required to obtain π for each $t \in V_T$ and $S \subseteq E_t^\ell$. If t is a leaf node, computing $\pi[t, S]$ takes $\mathcal{O}(1)$ time. If t is an introduce node, since each element of $\mathcal{G}_t^{S_u}$ must include edges within distance ℓ from u , the number of elements in $\mathcal{G}_t^{S_u}$ cannot exceed 2^{Δ^ℓ} . Consequently, iterating over all its subsets results in a complexity of $\mathcal{O}(2^{2^{\Delta^\ell}})$.

If t is a forget node, the time complexity is $\mathcal{O}(2^{\Delta^\ell})$, since $E_{t'}^\ell \setminus E_t^\ell$ contains only edges within distance ℓ from u , and the number of such edges is at most Δ^ℓ . For a join node, the time complexity is determined by the number of subsets of S , which is $\mathcal{O}(2^{|S|}) = \mathcal{O}(2^{|E_t^\ell|})$.

Finally, since $|V_T| = \mathcal{O}(k \cdot n)$ and $|E_t^\ell| \leq \Delta^\ell \cdot |X_t| \leq 2\Delta^\ell \cdot k + \Delta^\ell$, the overall time complexity for computing π is $2^{\mathcal{O}(k \cdot \Delta^\ell + 2^{\Delta^\ell})} \cdot n$. □

The theorem above can be directly applied to MAXPP, while for MAXSP, the result can be further refined. In the case of MAXSP, at any introduce node t , the number of elements in $\mathcal{G}_t^{S_u}$ is bounded by the sum of possible stars centered at u , which is $\sum_{i=1}^{\Delta} \binom{\Delta}{i} = \mathcal{O}(2^\Delta)$, along with the possible stars where u is a leaf, which is bounded by $\mathcal{O}(2^\Delta \Delta)$. Additionally, the longest path in any star is at most 2. Thus:

Corollary 1. *Any instance $\langle G, \mathcal{G}, \omega \rangle$ of MAXSP can be solved within $2^{\mathcal{O}(k \cdot \Delta^2 + 2^\Delta \Delta)} \cdot n$ time, where G is a graph with n vertices, treewidth k , and maximum degree Δ .*

Similar to stars, the longest path in any triangle is 2, so MAXTP does not require the parameter ℓ . Moreover, we can apply a simple modification to the strategy used for MAXGP to adapt it for MAXTP without utilizing Δ as a parameter. Specifically, we consider E_t^0 instead of E_t^2 .

If t is an introduce node, any triangle in \mathcal{G}_t that contains u must have its other vertices in X_t . Consequently, $\mathcal{G}_t^{S_u}$ contains at most $\mathcal{O}(|X_t|^2) = \mathcal{O}(k^2)$ triangles. If t is a forget node, the edges in $E_{t'}^0 \setminus E_t^0$ are those in $E_{t'}^0$ that are adjacent to u , which cannot be more than $\mathcal{O}(|X_{t'}|^2) = \mathcal{O}(k^2)$. For leaf nodes or join nodes, we can apply a similar analysis as previously discussed.

By implementing these ideas, we obtain a time complexity of $2^{\mathcal{O}(k^3)} \cdot n$ for solving MAXTP.

Corollary 2. *Any instance $\langle G, \mathcal{G}, \omega \rangle$ of MAXTP can be solved within $2^{\mathcal{O}(k^3)} \cdot n$ time, where G is a graph with n vertices, and treewidth k .*

The results presented in this section facilitate the development of approximation algorithms for cases where the input graph is planar. A discussion of these algorithms will follow in the next section.

4 Polynomial-Time Approximation Schemes on Planar Graphs

The unweighted versions of MAXGP, MAXSP, MAXPP, and MAXTP are NP-hard on planar graphs, even with bounded degree and, in the case of MAXPP, when the lengths of the paths in the collection are also bounded. Therefore, unless P = NP, efficient (polynomial-time) algorithms for solving these problems do not exist. A viable approach is to trade optimality for efficiency by developing algorithms that guarantee high-quality solutions. This section discusses approximation algorithms for MAXGP, MAXSP, MAXPP, and MAXTP, beginning with some brief definitions.

For a maximization problem Π , an **α -approximation algorithm** produces a solution for any instance I of Π whose value is at least $\alpha \cdot OPT(I)$ in $|I|^{\mathcal{O}(1)}$ time, where $OPT(I)$ denotes the optimal value of I . If, for every fixed $\epsilon > 0$, there exists a $(1 - \epsilon)$ -approximation algorithm for Π , we say the problem admits a *Polynomial-Time Approximation Scheme* (PTAS).

By combining a strategy introduced by [1] with the FPT results on treewidth presented earlier, we obtain a PTAS for the unweighted versions of the studied problems when the input graph is planar.

Theorem 7. *For each $\epsilon > 0$, there exists a $(1 - \epsilon)$ -approximation algorithm for unweighted MAXGP that runs in $2^{\mathcal{O}(\frac{\ell \cdot \Delta \cdot \ell}{\epsilon} + 2^{\Delta \cdot \ell})} \cdot n$ time on instances $\langle G, \mathcal{G} \rangle$ where G is a planar graph with n vertices and maximum degree Δ , and ℓ an upper bound for the longest path in any graph of \mathcal{G} .*

Proof. Let $\langle G, \mathcal{G} \rangle$ be an instance of the unweighted MAXGP, where G is a planar graph with n vertices and maximum degree Δ , and let k denote the outerplanarity index of G . Additionally, assume that the longest path in any subgraph of \mathcal{G} is bounded by a constant $\ell > 0$. Inspired by the approach of [1] for planar graphs, we propose the following method.

Given an integer $\delta > 0$, for each $0 \leq i < \delta$, let G_δ^i represent the subgraph of G formed by removing all edges with one endpoint in any layer $j \equiv i \pmod{\delta}$ and the other in the layer $j + 1$. Let \mathcal{G}_δ^i denote the subset of \mathcal{G} consisting of the subgraphs of G_δ^i . Notice that each component of G_δ^i is δ -outerplanar, which implies that its treewidth is at most $3\delta - 1$ [2].

Using Theorem 6, we can obtain an optimal solution S_δ^i for MAXGP with instance $\langle G_\delta^i, \mathcal{G}_\delta^i \rangle$ in $2^{O(\delta \cdot \Delta \ell + 2^{\Delta \ell})} \cdot n$ time. Finally, we return the solution $S_\delta^* = \arg \max_{S \in \{S_\delta^i | 0 \leq i < \delta\}} \{|S|\}$, leading to an overall time complexity of $2^{O(\delta \cdot \Delta \ell + 2^{\Delta \ell})} \cdot n$.

Let S^* be an optimal solution for the unweighted MAXGP instance $\langle G, \mathcal{G} \rangle$. Since each element of \mathcal{G} contains edges from at most ℓ layers, there exists $0 \leq i < \delta$ such that S^* includes at most $\frac{2\ell}{\delta} \cdot |S^*|$ elements not found in \mathcal{G}_δ^i . Thus, for this i , we have $|S_\delta^i| \geq (1 - \frac{2\ell}{\delta}) \cdot |S^*|$, which implies that $|S_\delta^*| \geq (1 - \frac{2\ell}{\delta}) \cdot |S^*|$. By defining $\epsilon = \frac{2\ell}{\delta}$, we concludes the Theorem's result. \square

Corollaries 1 and 2 allow us to refine the previous result, yielding specific PTASs for unweighted MAXSP and MAXTP.

Corollary 3. *For each $\epsilon > 0$, there exists a $(1 - \epsilon)$ -approximation algorithm for unweighted MAXSP that runs in $2^{O(\frac{\Delta^2}{\epsilon} + 2^{\Delta \Delta})} \cdot n$ time on instances $\langle G, \mathcal{G} \rangle$ where G is a planar graph with n vertices and maximum degree bounded by a constant $\Delta > 0$.*

Corollary 4. *For each $\epsilon > 0$, there exists a $(1 - \epsilon)$ -approximation algorithm for unweighted MAXSP that runs in $2^{O(\frac{1}{\epsilon^3})} \cdot n$ time on instances $\langle G, \mathcal{G} \rangle$ where G is a planar graph with n vertices.*

5 Conclusions

In this work, we introduced MAXGP and its specific cases: MAXSP, MAXPP, and MAXTP. We demonstrated that solving any of these problems is highly challenging by proving that they belong to the NP-hard complexity class. Furthermore, all of them are APX-hard, and some are W[1]-hard. Despite these negative results, we also achieved positive outcomes, including fixed-parameter and approximation algorithms for all the studied problems.

We designed algorithms parameterized by treewidth and maximum degree for MAXSP and for MAXGP when the longest path of the elements in \mathcal{G} is bounded by a constant ℓ . Observe that, unless P = NP, neither MAXSP nor the mentioned version of MAXGP are fixed-parameter tractable (FPT) with only treewidth or only maximum degree as parameters. Thus, a parameterized

algorithm for these problems must consider a combination of those parameters or different parameters altogether. On the other hand, for MAXTP, we obtained a fixed-parameter algorithm parameterized by treewidth. Additionally, from these FPT algorithms, we derived PTASs for the unweighted versions of the problems on planar graphs. All these versions are NP-hard and do not admit FPTAS (Fully Polynomial-Time Approximation Schemes), even on planar graphs; therefore, PTASs represent the best approximation ratio that can be achieved for such problems.

In future research, we aim to investigate the tractability of these problems by considering other structural graph parameters. Another promising direction is the exploration of approximation ratios for general graphs and the weighted cases of the problems in planar graphs. Additionally, we can study other versions of the problems in relation to specific classes of graphs in \mathcal{G} , such as trees, cliques, or cycles.

References

1. Baker, B.S.: Approximation algorithms for NP-complete problems on planar graphs. In: 24th Annual Symposium on Foundations of Computer Science (sfcs 1983), pp. 265–273 (1983). <https://doi.org/10.1109/SFCS.1983.7>
2. Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.* **209**(1), 1–45 (1998). [https://doi.org/10.1016/S0304-3975\(97\)00228-4](https://doi.org/10.1016/S0304-3975(97)00228-4)<https://www.sciencedirect.com/science/article/pii/S0304397597002284>
3. Caprara, A., Rizzi, R.: Packing triangles in bounded degree graphs. *Inf. Process. Lett.* **84**(4), 175–180 (2002). [https://doi.org/10.1016/S0020-0190\(02\)00274-0](https://doi.org/10.1016/S0020-0190(02)00274-0)<https://www.sciencedirect.com/science/article/pii/S0020019002002740>
4. Cygan, M., et al.: Parameterized Algorithms. Springer, Heidelberg (2015). <https://doi.org/10.1007/978-3-319-21275-3>
5. Holyer, I.: The NP-completeness of some edge-partition problems. *SIAM J. Comput.* **10**(4), 713–717 (1981). <https://doi.org/10.1137/0210054><https://doi.org/10.1137/0210054>
6. Huang, C.C., Mari, M., Mathieu, C., Schewior, K., Vygen, J.: An approximation algorithm for fully planar edge-disjoint paths. *SIAM J. Disc. Math.* **35**(2), 752–769 (2021). <https://doi.org/10.1137/20M1319401><https://doi.org/10.1137/20M1319401>
7. Korhonen, T.: A single-exponential time 2-approximation algorithm for treewidth. In: 2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS), pp. 184–192 (2022). <https://doi.org/10.1109/FOCS52979.2021.00026>
8. Ravelo, S.V.: Minimum constellation covers: hardness, approximability and polynomial cases. *J. Comb. Optim.* **41**(3), 603–624 (2021). <https://doi.org/10.1007/s10878-021-00698-1><https://link.springer.com/article/10.1007/s10878-021-00698-1>
9. Ravelo, S.V., Fernandes, C.G.: Complexity and approximability of minimum path-collection exact covers. *Theor. Comput. Sci.* **942**, 21–32 (2023). <https://doi.org/10.1016/j.tcs.2022.11.022><https://www.sciencedirect.com/science/article/pii/S0304397522007034>



Structural Parameterization of Minus Domination

Sangam Balchandar Reddy^(✉) and Anjeneya Swami Kare

School of Computer and Information Sciences, University of Hyderabad,
Hyderabad, India
`{21mcp14, askcs}@uohyd.ac.in`

Abstract. Given a graph $G = (V, E)$, a minus dominating function $f : V \rightarrow \{-1, 0, 1\}$ is a labelling of vertices such that $\sum_{v \in N[u]} f(v) \geq 1$, for each $u \in V$. The weight of f is the sum of $f(u)$ over all the vertices $u \in V$. The objective of MINUS DOMINATING SET problem is to compute a minus dominating function of minimum weight. The problem is known to be NP-complete even on split graphs and bipartite graphs. The parameterized complexity of the problem for the parameter treewidth is a long standing open question. In this paper, we answer this by proving that the problem is W[1]-hard for the parameter distance to disjoint paths, which is larger than treewidth and feedback vertex set number. Later, we show that no polynomial kernel exists for the parameter vertex cover number. For the parameter weight, we prove that the problem is W[2]-hard on bipartite graphs and W[1]-hard on circle graphs, respectively. In addition, we provide an FPT algorithm for the parameter feedback edge set number.

Keywords: Minus dominating set · Distance to disjoint paths · Vertex cover · Circle graphs · Feedback edge set

1 Introduction

DOMINATING SET (DS) problem is one of the well-studied NP-complete problems. Given a graph $G = (V, E)$, a dominating function $f : V \rightarrow \{0, 1\}$ is a labelling of vertices such that $\sum_{v \in N[u]} f(v) \geq 1$, for each $u \in V$. The weight of a dominating function f is $\sum_{u \in V} f(u)$. The goal is to obtain a minimum weighted dominating function f . Many variants of the problem such as total domination [17], connected domination [20], roman domination [3] and signed domination [6] have been studied in the literature. In this work, we study the MINUS DOMINATING SET (MDS) problem which was introduced by Dunbar et al. [7]. Given a graph $G = (V, E)$, a minus dominating function $f : V \rightarrow \{-1, 0, 1\}$ is a labelling of vertices such that $\sum_{v \in N[u]} f(v) \geq 1$, for each $u \in V$. The weight of a minus dominating function f is $\sum_{u \in V} f(u)$. The objective of the MDS problem is to compute a minus dominating function f with minimum weight.

We formally define the problem as follows.

Problem. MINUS DOMINATING SET

Input. An instance $I = (G, k)$, where $G = (V, E)$ is an undirected graph, and an integer k .

Output. Does there exist a function $f : V \rightarrow \{0, 1, -1\}$ such that $\sum_{u \in V} f(u) \leq k$ and $\sum_{v \in N[u]} f(v) \geq 1$ for all $u \in V$?

A dominating function uses the labels $\{0, 1\}$, a signed dominating function uses the labels $\{-1, 1\}$ where as the minus dominating function uses $\{-1, 0, 1\}$.

1.1 Related Work

MINUS DOMINATING SET problem is NP-complete even on split graphs [8], chordal bipartite graphs [7, 12] and planar graphs with maximum degree four [5]. The problem is linear-time solvable on trees [7], strongly chordal graphs [12] and chain interval graphs [15]. Recently, Faria et al. [8] have presented polynomial-time algorithms for the problem on cographs, distance hereditary graphs and graphs of bounded rank-width.

The parameterized complexity of MINUS DOMINATING SET problem was also studied at various occasions. It is known that the DOMINATING SET is W[2]-hard parameterized by the solution size [19], whereas MDS parameterized by weight is para-NP-hard [8]. MDS is fixed-parameter tractable for d -degenerate graphs parameterized by the size of the minus dominating set and d , where the size is the number of vertices with label 1 [8]. Lin et al. [13] have proved that the problem is FPT on subcubic graphs parameterized by weight. DS is known to be FPT for the structural parameters treewidth [4] and distance to cluster [11]. When it comes to MDS, Bhyravarapu et al. [1] have proved that the problem is FPT parameterized by twin cover, neighbourhood diversity and the combined parameter distance to cluster and the size of the largest clique. They have also shown that the problem has an XP-algorithm for distance to cluster.

MINUS DOMINATING SET problem is known to be APX-hard on graphs with maximum degree 7 [13]. There are several combinatorial bounds obtained for various graph classes such as graphs with small degree ($\Delta \leq 4$) [5], bipartite graphs [14] and regular graphs [16]. Unless P = NP, the weight of a minus dominating function cannot be approximated in polynomial time within a factor $1 + \epsilon$, for some $\epsilon > 0$, even for graphs with maximum degree at most four [5].

1.2 Our Results

It was asked in [1, 8] regarding the parameterized complexity of MINUS DOMINATING SET problem parameterized by treewidth. In Sect. 3, we provide a parameterized reduction to prove that the problem is W[1]-hard. In fact, we not only prove that the problem is W[1]-hard parameterized by treewidth but for a larger parameter distance to disjoint paths. We prove this by reducing a W[1]-hard problem, Multidimensional Relaxed Subset Sum to MDS.

In Sect. 4, we consider the kernelization aspects of the problem. It is known that MINUS DOMINATING SET problem is fixed-parameter tractable parameterized by vertex cover number [1] but whether there exists a polynomial kernel is not known. We answer this question by providing a polynomial parameter transformation from Red-Blue Dominating Set (RBDS) to MDS. As RBDS does not admit a polynomial kernel, the same result holds for MDS parameterized by vertex cover number.

Later in Sect. 5, we provide a parameterized reduction from DOMINATING SET problem parameterized by solution size to MINUS DOMINATING SET problem parameterized by weight. DS is W[2]-hard on bipartite graphs parameterized by the solution size [18] and W[1]-hard on circle graphs parameterized by the solution size [2]. As we provide a parameterized reduction from DS to MDS preserving the graph class, we obtain that MDS is W[2]-hard on bipartite graphs parameterized by weight and W[1]-hard on circle graphs parameterized by weight.

As the MINUS DOMINATING SET problem is W[1]-hard for feedback vertex set number and treewidth, we shift our focus to a larger parameter feedback edge set number. In Sect. 6, we present a fixed-parameter tractable algorithm for MDS parameterized by feedback edge set number. We make use of the known linear time algorithm for MDS on trees [7].

2 Preliminaries

We consider only simple, finite and connected graphs. Let $G = (V, E)$ be a graph with V as the vertex set and E as the edge set. The sets $N(u)$ and $N[u]$ consists of the neighbours and the closed neighbours of a vertex u , respectively. The open neighbourhood of a set $T \subseteq V$ is denoted by $N(T)$ and the closed neighbourhood by $N[T]$. $N(T) = \bigcup_{u \in T} N(u)$ and $N[T] = \bigcup_{u \in T} N[u]$. The degree of a vertex u is represented by $d(u)$ and $d(u) = |N(u)|$. A vertex of degree one is called a pendant vertex. Let f be a minimum weighted minus dominating function. We use the term **label** for a vertex u to denote the value that f maps to, i.e., $f(u)$. For a vertex u , **labelSum** is defined as the sum of the labels of all the vertices in its closed neighbourhood, i.e., $\sum_{v \in N[u]} f(v)$. For a set T , **weight** of T is the sum of the labels of all the vertices in the set, i.e., $\sum_{u \in T} f(u)$. Other than this, we use the standard notations as defined in [21].

We use $\mathcal{O}^*(f(n))$ to denote the time complexity of the form $\mathcal{O}(f(n) \cdot n^{\mathcal{O}(1)})$. A problem is considered to be *fixed-parameter tractable* w.r.t. a parameter k , if there exists an algorithm with running time $\mathcal{O}^*(f(k))$, where f is some computable function. For more information on *parameterized complexity* and *graph theory*, we refer the reader to [4] and [21], respectively.

Observation 1 (¹). Let $u \in V$ with $d(u) = 1$ and $v \in N(u)$, then $f(u) \neq -1$ and $f(v) \neq -1$.

¹ Due to the space limit, the proofs of statements marked with a will appear in the full version of the paper.

3 W[1]-Hardness Parameterized by Distance to Disjoint Paths

In this section, we study the parameterized complexity of the MINUS DOMINATING SET problem for the parameter distance to disjoint paths.

Definition 1. For a graph $G = (V, E)$, the parameter distance to disjoint paths is the cardinality of the smallest set $D \subseteq V$ such that each component in $G[V \setminus D]$ is a path.

We provide a reduction from a well known W[1]-hard problem, Multidimensional Relaxed Subset Sum (MRSS). The problem MRSS is defined as follows.

Problem. Multidimensional Relaxed Subset Sum

Input. An integer k , a set $S = \{s_1, s_2, \dots, s_n\}$ of vectors with $s_i \in \mathbb{N}^k$ for every i with $1 \leq i \leq n$, a target vector $t \in \mathbb{N}^k$ and an integer m .

Parameter. $k + m$

Output. Does there exist a subset $S' \subseteq S$ with $|S'| \leq m$ such that $\sum_{s \in S'} s \geq t$?

It is known that MRSS is W[1]-hard parameterized by $k + m$, even when all integers in the input are in unary [10]. We reduce the problem to MDS to prove that the problem is W[1]-hard parameterized by distance to disjoint paths.

Construction. Consider an instance $I = (k, m, S, t)$ of MRSS. We construct an instance $I' = (G, k')$ of the MDS problem in the following way.

- We create two sets of vertices $\{u_1, u_2, \dots, u_k\}$ and $\{v_1, v_2, \dots, v_k\}$.
- We introduce a set D_j with $t(j)$ vertices, for each $j \in \{1, 2, \dots, k\}$.
- We introduce a set F_j with $t(j)$ vertices, for each $j \in \{1, 2, \dots, k\}$. We make two pendant vertices adjacent to each vertex of F_j . The set containing the pendant vertices of F_j is denoted by P_j .
- We make two pendant vertices adjacent to each vertex of $\bigcup_{j \in \{1, 2, \dots, k\}} (u_j \cup v_j)$. The pendant vertices adjacent to u_j are denoted by r_j^1 and r_j^2 . The pendant vertices adjacent to v_j are denoted by r_j^3 and r_j^4 .
- For each $j \in \{1, 2, \dots, k\}$, we make u_j adjacent to each vertex of the set D_j and v_j adjacent to each vertex of the sets D_j and F_j .
- For each $s_i \in S$, we create a path of $3 \cdot \max(s_i) + 2$ vertices. We denote the vertices of the path with $p_i^1, p_i^2, \dots, p_i^t$, where $t = 3 \cdot \max(s_i) + 2$.
- We classify the vertices of the path into three sets A_{s_i}, B_{s_i} and C_{s_i} . Let $A_{s_i} = \bigcup_{l \in \{1, 2, \dots, \max(s_i)+1\}} a_i^l$ where $a_i^l = p_i^{3l-2}$, $B_{s_i} = \bigcup_{l \in \{1, 2, \dots, \max(s_i)+1\}} b_i^l$ where $b_i^l = p_i^{3l-1}$ and $C_{s_i} = \bigcup_{l \in \{1, 2, \dots, \max(s_i)\}} c_i^l$ where $c_i^l = p_i^{3l}$.
- For each $s_i \in S$ and $j \in \{1, 2, \dots, k\}$, we make u_j adjacent to exactly $s_i(j)$ vertices in C_{s_i} arbitrarily.

This concludes the construction of the reduced instance I . See Fig. 1 for an illustration.

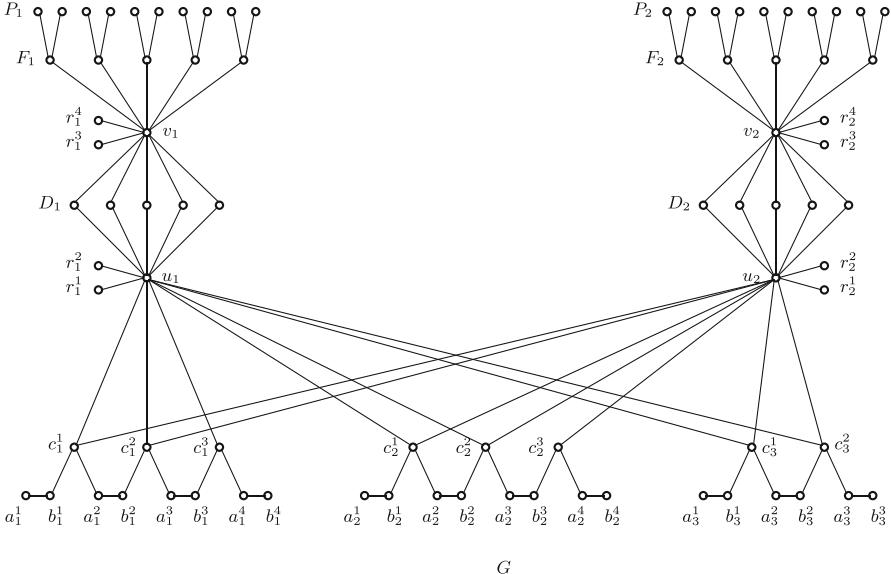


Fig. 1. Reduced instance of MDS constructed from MRSS instance $S = \{(3, 2), (2, 3), (2, 2)\}, t = (5, 5), k = 2, m = 2$.

We set $k' = \sum_{s_i \in S} (\max(s_i) + 1) + 2k + m$ and the set corresponding to distance to disjoint paths is $\bigcup_{j \in \{1, 2, \dots, k\}} (u_j \cup v_j)$ of size $2k$.

Lemma 1. If (k, m, S, t) is a yes instance of MRSS then there exists a minus dominating function f of weight at most k' .

Proof. Let $S' \subseteq S$ such that $|S'| \leq m$ and $\sum_{s_i \in S'} s_i \geq t$.

The corresponding minimum weighted minus dominating function f is given as follows.

$$\begin{aligned} f(w) &= 0 \text{ for } w \in \{\bigcup_{j \in \{1, 2, \dots, k\}} (P_j \cup \{r_j^1, r_j^2, r_j^3, r_j^4\}) \cup \bigcup_{s_i \in S'} (B_{s_i} \cup \bigcup_{l \in \{2, 3, \dots, \max(s_i)\}} a_i^l) \cup \\ &\quad a_i^l\} \cup \bigcup_{s_i \notin S'} (A_{s_i} \cup C_{s_i})\} \\ f(w) &= 1 \text{ for } w \in \{\bigcup_{j \in \{1, 2, \dots, k\}} (u_j \cup v_j \cup F_j) \cup \bigcup_{s_i \in S'} (C_{s_i} \cup \bigcup_{l \in \{1, \max(s_i)+1\}} a_i^l) \cup \\ &\quad \bigcup_{s_i \notin S'} B_{s_i}\} \\ f(w) &= -1 \text{ for } w \in \{\bigcup_{j \in \{1, 2, \dots, k\}} D_j\} \end{aligned}$$

- All the vertices of F_j have a label of one and all the vertices of P_j have a label of zero. This results in each vertex of P_j having a labelSum of exactly one.
- Each vertex of F_j has a labelSum of two due to the positive labels of the vertex itself and its neighbour v_j .
- Vertex v_j has a labelSum of one, the positive weight from F_j and the negative weight from D_j will cancel out and the positive label of v_j will contribute to its positive labelSum.

- Each vertex of D_j has a labelSum of one. This is due to the positive labels of its two neighbours u_j and v_j .
- Vertex u_j has a labelSum of at least one, the positive weight of its neighbours from $\bigcup_{s_i \in S} C_{s_i}$ is at least the negative weight from D_j . As u_j itself has a positive label, the labelSum of u_j will remain positive.
- Vertices r_j^1 and r_j^2 have positive labelSum as their only neighbour u_j has a label of one.
- Vertices r_j^3 and r_j^4 have positive labelSum as their only neighbour v_j has a label of one.
- Vertex a_i^l has a neighbour b_i^l with label one, if $s_i \notin S$. Vertex a_i^l itself has a label of one and for $l \geq 2$, a_i^l has a neighbour c_i^{l-1} with label one, if $s_i \in S$.
- Vertex b_i^l itself has a label of one, if $s_i \notin S$. For $l \leq \max(s_i)$, b_i^l has a neighbour c_i^l with a label of one and for $l = \max(s_i) + 1$, b_i^l has a neighbour a_i^l with a label of one, if $s_i \in S$.
- As all the vertices $\{u_1, u_2, \dots, u_j\}$ have a label of one, for each $s_i \in S$, each vertex in C_{s_i} has a positive labelSum.

Each vertex $w \in V(G)$ has a positive labelSum and the weight of the minus dominating function f is $k' = \sum_{s_i \in S} (\max(s_i) + 1) + 2k + m$. \square

Observation 2 (♣). Let f be a minimum weighted minus dominating function. For each $j \in \{1, 2, \dots, k\}$, both u_j and v_j gets a label of one and the pendant vertices r_j^1, r_j^2, r_j^3 and r_j^4 gets a label of zero in f .

Observation 3 (♣). Let f be a minimum weighted minus dominating function. For each $j \in \{1, 2, \dots, k\}$, each vertex in F_j gets a label of one and the vertices of P_j gets a label of zero in f .

Lemma 2 (♣). Let f be a minimum weighted minus dominating function. For each $s_i \in S$, the weight of $A_{s_i} \cup B_{s_i} \cup C_{s_i}$ is at least $\max(s_i) + 1$ in f .

Observation 4 (♣). Let the weight of the minus dominating function f be $\sum_{s_i \in S} (\max(s_i) + 1) + 2k + m$. Then, for $j \in \{1, 2, \dots, k\}$, each vertex in D_j gets a label of -1 in f .

Lemma 3. If there exists a minus dominating function f of weight at most k' then (k, m, S, t) is a yes instance of MRSS.

Proof. Let f be a minus dominating function of weight $k' = \sum_{s_i \in S} (\max(s_i) + 1) + 2k + m$.

- Due to Observation 2, Observation 3 and Observation 4, we fix the labels of each vertex in P_j to 0, F_j to 1 and D_j to -1 for each $j \in \{1, 2, \dots, k\}$. We also set the labels $u_j = 1, v_j = 1, r_j^1 = 0, r_j^2 = 0, r_j^3 = 0$ and $r_j^4 = 0$ for each $j \in \{1, 2, \dots, k\}$.
- After assigning the labels to the vertices of the sets P_j, F_j and D_j and the vertices $u_j, v_j, r_j^1, r_j^2, r_j^3$ and r_j^4 , vertex u_j has a negative labelSum of $-t(j)$. In order for the vertex u_j to have a positive labelSum, we must consider having positive labels for its neighbours in $\bigcup_{s_i \in S} C_{s_i}$. The labelSum of its neighbours from $\bigcup_{s_i \in S} C_{s_i}$ must be at least $t(j)$.

- At this point, we are left with a threshold of $\sum_{s_i \in S} (\max(s_i) + 1) + m$. If we choose to have labels as zero for the vertices of any C_{s_i} then we would need a weight of $\max(s_i) + 1$ and if we choose to have labels as one for the vertices of any C_{s_i} , then we would need a weight of $\max(s_i) + 2$, an extra weight of one for each such set.
- If there exist m sets from C_{s_i} those bring positive labelSum to each u_j then we conclude that there exists m sets from S that surpass t . \square

This completes the proof of the following theorem.

Theorem 1. MINUS DOMINATING SET problem parameterized by distance to disjoint paths is W[1]-hard.

The parameters feedback vertex set number, treewidth, clique width and pathwidth are bounded by a function of the parameter distance to disjoint paths. With this direct relation between the parameters, we have the following theorem.

Theorem 2. MINUS DOMINATING SET problem is W[1]-hard when parameterized by any of the following parameters: (a) feedback vertex set number (b) treewidth (c) clique width (d) pathwidth.

4 No Polynomial Kernel Parameterized by Vertex Cover Number

In this section, we study regarding the non-existence of a polynomial kernel for the parameter vertex cover number.

Definition 2. Given a graph $G = (V, E)$, vertex cover number is the cardinality of the smallest set $C \subseteq V$ such that $V \setminus C$ is an independent set.

We provide a reduction from Red-Blue Dominating Set (RBDS) to prove that no polynomial kernel exists for the parameter vertex cover number. The problem RBDS is defined as follows.

Problem. Red-Blue Dominating Set

Input. A bipartite graph $G = (X \cup Y, E)$, and an integer k

Output. Does there exist a subset $S \subseteq X$ of size at most k that dominates Y ?

Theorem 3 ([9]). RBDS parameterized by $|Y|$ does not admit a polynomial kernel unless coNP \subseteq NP/poly.

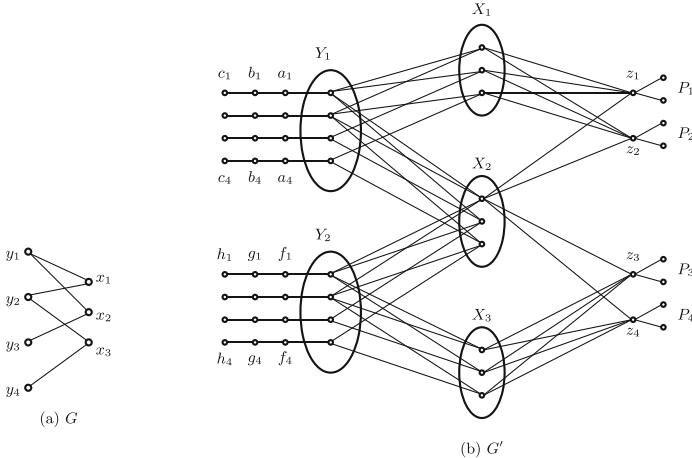


Fig. 2. (a) a graph G with $k = 2$ and (b) the graph G' with $k' = 13$

We provide a polynomial parameter transformation (PPT) from RBDS parameterized by $|Y|$ to MDS parameterized by the vertex cover number.

Construction. Consider an instance $I = (G, k)$ of the RBDS, we construct the corresponding MDS instance $I' = (G', k')$ as follows. Take two distinct copies Y_1 and Y_2 of Y and three distinct copies X_1, X_2 and X_3 of X . The vertex corresponding to $u \in X$ is denoted by X_1^u, X_2^u and X_3^u in X_1, X_2 and X_3 respectively. Similarly, the vertex corresponding to $u \in Y$ is denoted by Y_1^u and Y_2^u in Y_1 and Y_2 respectively. We make each vertex $u \in Y_1$ adjacent to the vertices, $X_1^{N_G(u)}$ in X_1 and $X_2^{N_G(u)}$ in X_2 . We also make each vertex $u \in Y_2$ adjacent to the vertices, $X_2^{N_G(u)}$ in X_2 and $X_3^{N_G(u)}$ in X_3 . We add a path of length three to each vertex of Y_1 and Y_2 . The vertices of the path adjacent to Y_1 are denoted using a_i, b_i, c_i where $A = \bigcup_{i \in \{1, 2, \dots, |Y|\}} a_i$, $B = \bigcup_{i \in \{1, 2, \dots, |Y|\}} b_i$, $C = \bigcup_{i \in \{1, 2, \dots, |Y|\}} c_i$. The vertices of the set A in the path are adjacent to the vertices of the set Y_1 . The vertices of the path adjacent to Y_2 are denoted using f_i, g_i, h_i where $F = \bigcup_{i \in \{1, 2, \dots, |Y|\}} f_i$, $G = \bigcup_{i \in \{1, 2, \dots, |Y|\}} g_i$, $H = \bigcup_{i \in \{1, 2, \dots, |Y|\}} h_i$. The vertices of the set F in the path are adjacent to the vertices of the set Y_2 . We create four vertices z_1, z_2, z_3 and z_4 . We make z_1 and z_2 adjacent to each vertex of X_1 . We make z_3 and z_4 adjacent to each vertex of X_3 . We pick a set of $|X_2| - k$ vertices from X_2 arbitrarily and make them adjacent to the vertices z_1, z_2, z_3 and z_4 . We add two pendant vertices to each of z_1, z_2, z_3 and z_4 . The pendant vertices adjacent to the vertex z_j are denoted using the set P_j . This completes the construction of the reduced instance G' . For more details, see Fig. 2 for an illustration.

We set $k' = 2|Y| - |X| + 2k + 4$ and the vertex cover $C = Y_1 \cup Y_2 \cup B \cup G \cup \{z_1, z_2, z_3, z_4\}$ of size $4|Y| + 4$.

Lemma 4 (♣). *If there exists a subset $S \subseteq X$ of size at most k that dominates Y then there exists a minus dominating function f of weight at most k' .*

Lemma 5 (♣). *If there exists a minus dominating function f of weight at most k' then there exists a subset $S \subseteq X$ of size at most k that dominates Y .*

With this, we arrive at the following theorem.

Theorem 4. MINUS DOMINATING SET problem does not admit a polynomial kernel parameterized by vertex cover number unless $\text{coNP} \subseteq \text{NP/poly}$.

5 Hardness Results on Bipartite and Circle Graphs Parameterized by Weight

In this section, we study the parameterized complexity of the MINUS DOMINATING SET problem on bipartite and circle graphs.

Definition 3. *A graph is a circle graph if and only if there exists a corresponding circle model in which the chords represent the vertices of the graph and two chords intersect only if the corresponding vertices are adjacent.*

We prove that the problem is W[2]-hard on bipartite graphs parameterized by weight and W[1]-hard on circle graphs parameterized by weight.

Theorem 5 ([18]). DOMINATING SET problem on bipartite graphs is W[2]-hard parameterized by solution size.

Theorem 6 ([2]). DOMINATING SET problem on circle graphs is W[1]-hard parameterized by solution size.

Construction. Given an instance $I = (G, k)$ of the DOMINATING SET problem, we transform into an instance $I' = (G', k)$ of the MINUS DOMINATING SET problem as follows. For each vertex $u \in V(G)$, we create a gadget as follows. We introduce two vertices w_1 and w_2 and make them adjacent to u . We create three sets of five vertices each, which are denoted by X_1 , X_2 and X_3 . We also create four vertices y_1, y_2, y_3 and y_4 . Vertices of the set X_1 are made adjacent to w_1, y_1 and y_2 . Vertices of the set X_2 are made adjacent to w_1, w_2, y_1, y_2, y_3 and y_4 . Vertices of the set X_3 are made adjacent to w_2, y_3 and y_4 . We add two pendant vertices to each of y_1, y_2, y_3 and y_4 . The pendant vertices adjacent to y_1, y_2, y_3 and y_4 are denoted by P_1, P_2, P_3 and P_4 , respectively. We also introduce another vertex v , which is made adjacent to w_1 and w_2 . Vertex v has two pendant vertices adjacent to it. We use P_v to denote the pendant vertices adjacent to v . This concludes the construction of I' . See Fig. 3 for an illustration.

Lemma 6 (♣). *If G has a dominating set of size at most k then G' has a minus dominating function of weight at most k .*

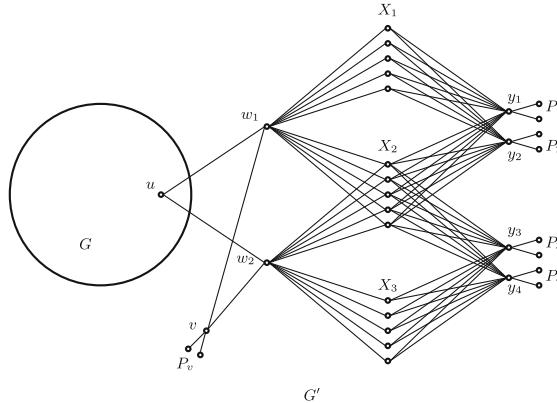


Fig. 3. Construction of G' from G

Lemma 7 (♣). *If G' has a minus dominating function of weight at most k then G has a dominating set of size at most k .*

Claim 1 (♣). *If G is a bipartite graph then G' is a bipartite graph.*

From Theorem 5 and Claim 1, we have the following result.

Theorem 7. MINUS DOMINATING SET problem on bipartite graphs is $W[2]$ -hard parameterized by weight.

Claim 2 (♣). *If G is a circle graph then G' is a circle graph.*

Similarly, from Theorem 6 and Claim 2, we obtain the following theorem.

Theorem 8. MINUS DOMINATING SET problem on circle graphs is $W[1]$ -hard parameterized by weight.

6 FPT Algorithm for Feedback Edge Set Number

In this section, we study the parameterized complexity of the problem for the parameter feedback edge set number.

Definition 4. *Given a graph $G = (V, E)$, the parameter feedback edge set number is the cardinality of the smallest set of edges $F \subseteq E$ such that $G[V, E \setminus F]$ is a spanning tree of G .*

The parameter feedback edge set number can be obtained in linear time via depth-first or breadth-first search. For the rest of this paper, we assume that the set corresponding to the feedback edge set number is given to us. From Theorem 2, we know that for the feedback vertex set number the problem is $W[1]$ -hard whereas for the feedback edge set number we present an FPT algorithm.

Let F denote the feedback edge set and $k = |F|$. Let T denote the tree $G[V, E \setminus F]$ and U be the vertices with an endpoint in F . We guess the labels for all the vertices of U . The number of vertices in U is at most $2k$. Hence, we have at most 3^{2k} unique guesses. For each such guess, we compute the optimal weight for the tree T . While computing the optimal weight for T , for each vertex in T , we also need to consider the labelSum of its closed neighbours in U from the original graph G . A subset of vertices of T have initial labelSum (due to the vertices of U) and a subset of vertices (U) of T are prelabelled. Hence, we use the term *special tree* to refer to T .

Theorem 9 ([7]). MINUS DOMINATING SET problem on trees is linear-time solvable.

We adopt the approach given in Theorem 9 to solve the MDS problem on the *special tree* T .

Theorem 10 (♣). Given a special tree T , minimum weighted minus dominating function can be computed in linear time.

We compute the optimal weight for the *special tree* T with the help of Theorem 10 in linear time. Hence, we have proved the following theorem.

Theorem 11. MINUS DOMINATING SET problem can be solved in time $\mathcal{O}^*(9^k)$, where k is the feedback edge set number.

In order to obtain an FPT algorithm for the feedback edge set number, we get rid of a bounded number of edges from G that results in a spanning tree. The proposed approach for the feedback edge set number will not analogously work for the feedback vertex set number as the edges incident to the vertices corresponding to feedback vertex set are not bounded.

7 Conclusion

We conclude the paper with the following open questions. The parameterized complexity of MINUS DOMINATING SET problem for the other structural parameters such as distance to path, distance to cluster, distance to co-cluster and modular width is open. The existence of a polynomial kernel for the parameter distance to clique could be investigated. The study on parameterized approximation algorithms for the problem could be initiated with the parameters distance to disjoint paths, feedback vertex set and treewidth. Obtaining non-trivial exact algorithms for the problem is also an interesting research direction.

References

1. Bhyravarapu, S., Kanesh, L., Mohanapriya, A., Purohit, N., Sadagopan, N., Saurabh, S.: On the parameterized complexity of minus domination. In: Fernau, H., Gaspers, S., Klasing, R. (eds.) SOFSEM 2024: Theory and Practice of Computer Science, pp. 96–110. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-52113-3_7

2. Bousquet, N., Gonçalves, D., Mertzios, G.B., Paul, C., Sau, I., Thomassé, S.: Parameterized domination in circle graphs. In: Golumbic, M.C., Stern, M., Levy, A., Morgenstern, G. (eds.) Graph-Theoretic Concepts in Computer Science, pp. 308–319. Springer, Heidelberg (2012)
3. Cockayne, E.J., Dreyer, P.A., Hedetniemi, S.M., Hedetniemi, S.T.: Roman domination in graphs. *Disc. Math.* **278**(1), 11–22 (2004)
4. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer, Switzerland (2015)
5. Damaschke, P.: Minus domination in small-degree graphs. *Disc. Appl. Math.* **108**(1), 53–64 (2001)
6. Dunbar, J., Hedetniemi, S., Henning, M., Slater, P.: Signed domination in graphs. *Graph Theory Comb. Appl.* **1**, 311–322 (1995)
7. Dunbar, J., Goddard, W., Hedetniemi, S., McRae, A., Henning, M.A.: The algorithmic complexity of minus domination in graphs. *Disc. Appl. Math.* **68**(1), 73–84 (1996)
8. Faria, L., Hon, W.K., Kloks, T., Liu, H.H., Wang, T.M., Wang, Y.L.: On complexities of minus domination. *Disc. Optim.* **22**, 6–19 (2016)
9. Fomin, F.V., Lokshtanov, D., Saurabh, S., Zehavi, M.: Kernelization: Theory of Parameterized Preprocessing. Cambridge University Press, Cambridge (2019)
10. Ganian, R., Klute, F., Ordyniak, S.: On structural parameterizations of the bounded-degree vertex deletion problem. *Algorithmica* **83**(1), 297–336 (2021)
11. Goyal, D., Jacob, A., Kumar, K., Majumdar, D., Raman, V.: Structural parameterizations of dominating set variants. In: Fomin, F.V., Podolskii, V.V. (eds.) Computer Science - Theory and Applications, pp. 157–168. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-90530-3_14
12. Lee, C.M., Chang, M.S.: Variations of y-dominating functions on graphs. *Disc. Math.* **308**(18), 4185–4204 (2008)
13. Lin, J.Y., Liu, C.H., Poon, S.H.: Algorithmic aspect of minus domination on small-degree graphs. In: Xu, D., Du, D., Du, D. (eds.) Computing and Combinatorics, pp. 337–348. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21398-9_27
14. Liu, H., Sun, L.: On the minus domination number of graphs. *Czechoslov. Math. J.* **54**(4), 883–887 (2004)
15. Lu, C.L., Peng, S.L., Tang, C.: Efficient minus and signed domination in graphs. *Theor. Comput. Sci.* **301**, 381–397 (2003). [https://doi.org/10.1016/S0304-3975\(02\)00594-7](https://doi.org/10.1016/S0304-3975(02)00594-7)
16. Matoušek, J.: Lower bound on the minus-domination number. *Disc. Math.* **233**(1), 361–370 (2001)
17. Henning, M.A., A.Y.: Total Domination in Graphs. Springer, Switzerland (2013)
18. Raman, V., Saurabh, S.: Short cycles make w-hard problems hard: fpt algorithms for w-hard problems in graphs with no short cycles. *Algorithmica (New York)* **52**, 203–225 (2008). <https://doi.org/10.1007/s00453-007-9148-9>
19. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Springer, Switzerland (2013)
20. Sampathkumar, E., Walikar, H.: The connected domination number of a graph. *J. Math. Phys. Sci.* **13** (1979)
21. West, D.B.: Introduction to Graph Theory, 2nd edn. Pearson, Chennai (2015)



An Algebraic Characterization of Strong Graphs

Pablo Romero^{1,2(✉)}

¹ Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay
promero@fing.edu.uy

² Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires,
Buenos Aires, Argentina

Abstract. Let G be a connected simple graph on n vertices and m edges. Denote $N_i^{(j)}(G)$ the number of spanning subgraphs of G having precisely i edges and not more than j connected components. The graph G is *strong* if $N_i^j(G) \geq N_i^j(H)$ for each pair of integers $i \in \{0, 1, \dots, m\}$ and $j \in \{1, 2, \dots, n\}$ and each connected simple graph H on n vertices and m edges. The graph G is *Whitney-maximum* if for each connected simple graph H on n vertices and m edges there exists a polynomial $P_H(x, y)$ with nonnegative coefficients such that $W_G(x, y) - W_H(x, y) = (1 - xy)P_H(x, y)$, where W_G and W_H stand for the Whitney polynomial of G and H . In this work it is proved that a graph is strong if and only if it is Whitney-maximum. Consequently, the 0-element conjecture proposed by Boesch [J. Graph Theory 10 (1986), 339–352] is true when restricted to graph classes in which Whitney-maximum graphs exist.

Keywords: Whitney polynomial · Strong graph · 0-element conjecture

1 Historic Motivation

During the half of the 20th century there was a growing interest in two problems that are different in appearance, namely, the 4-coloring problem and the design of networks with maximum reliability. Some key references in both problems are given in the following paragraphs and the interplay between them is explained, which is the main source of inspiration for this work.

On the one hand Birkhoff [1], interested in the 4-coloring problem, defined a univariate polynomial known as the chromatic polynomial for planar graphs. Later works by Whitney [14] and Tutte [13] considered generalizations of the chromatic polynomial using bivariate polynomials. The work of Tutte [13] linked the theory of spanning trees, electrical networks, and graph coloring. In his Ph.D. thesis, he proved the universality property of his eponymous polynomial. Essentially, each graph invariant that obeys the deletion-contraction formula can be obtained by a special evaluation of the Tutte polynomial. The interested reader can consult the recent book [4] for a rich historic account of the Tutte polynomial and an updated reference in this topic.

On the other hand Moore and Shannon [10] wanted to design highly reliable computers using imperfect electronic components such as relays. The work of Moore and Shannon is considered a point of departure in the study of network reliability. The concept of uniformly most reliable graphs was later introduced by Boesch [2]. The existence and construction of uniformly most reliable graphs is a current research topic. Boesch et al. [3] found all uniformly most reliable graphs of corank up to 3. Curiously enough, the first classes of uniformly most reliable graphs were determined by Kelmans [9] some years before the definition was formally established in print. In his work he showed that if we remove an arbitrary matching to the complete graph then a uniformly most reliable graph is obtained. The reader can consult the recent survey [12] for further details.

Even though the classical reliability problem aims to find the probability that a graph is connected subject to edge failures, Kelmans [9] studied a much more general problem in which we are given a positive integer k and the aim is to find the probability that a graph has at most k connected components after each of its edges is independently deleted with given probabilities. This generalization due to Kelmans has not been further explored. The counting problems for the study of graphs having multiple connected components are even harder than the ones that appear in the classical reliability problem faced by Boesch, Moore and Shannon.

Joint works of Kahl [7], and Kahl and Luttrell [8], established links between uniformly most reliable graphs and the Tutte polynomial by means of a novel concept of a *Tutte-maximum graph*. Using the universality property of the Tutte polynomial it follows that Tutte-maximum graphs are uniformly most reliable graphs. Kahl and Luttrell [8] then proceeded to recover the results obtained by Boesch et al. on the existence of uniformly most reliable graphs of corank up to 3 but using algebraic methods. Furthermore, Kahl [7] showed that each Tutte-maximum graph simultaneously attains the extremal value (i.e., minimum or maximum) of several real-valued graph invariants among all graphs with a prescribed number of vertices and edges.

The goal of this work is to further explore the interplay between counting problems coming from network reliability analysis and algebraic methods that are available using the Tutte polynomial or Whitney polynomial. This article is organized as follows. A background including some concepts and preliminary results is given in Sect. 2. The main result of this work is given in Sect. 3, where it is proved that a graph is strong if and only if it is Whitney-maximum. In Sect. 4 it is proved that the 0-element conjecture proposed by Boesch [2] is true when restricted to graph classes in which a Whitney-maximum graph exists.

2 Background

In this section we include the concept of Tutte-maximum graph [8] and some concepts on network reliability. We will also include the 0-element conjecture proposed by Boesch [2] as well as preliminary results which will be used in the sequel.

Let G be a graph on n vertices, m edges and $\kappa(G)$ connected components. The *corank* of G , denoted $c(G)$, equals $m - n + \kappa(G)$. The *rank* of G , denoted $r(G)$, equals $n - \kappa(G)$. Let $\mathcal{S}(G)$ be the set of all spanning subgraphs of G . The *Tutte polynomial* of G is denoted $T_G(x, y)$ and is defined as follows,

$$T_G(x, y) = \sum_{H \in \mathcal{S}(G)} (x - 1)^{r(G) - r(H)} (y - 1)^{c(H)}. \quad (1)$$

The *Whitney polynomial* of G is defined as $W_G(x, y) = T_G(x + 1, y + 1)$, i.e.,

$$W_G(x, y) = \sum_{H \in \mathcal{S}(G)} x^{r(G) - r(H)} y^{c(H)}. \quad (2)$$

A bivariate polynomial $P(x, y)$ is *nonnegative* if there exists nonnegative integers p and q and nonnegative real numbers a_{ij} such that $P(x, y) = \sum_{i=0}^p \sum_{j=0}^q a_{ij} x^i y^j$.

Let $\mathcal{C}_{n,m}$ be the set of all connected simple graphs on n vertices and m edges. The concept of Tutte-maximum graphs introduced by Kahl and Luttrell [8] is a point of departure in the construction of algebraic methods to find uniformly most reliable graphs.

Definition 1. For each pair of graphs G and H in $\mathcal{C}_{n,m}$ we write $H \preceq G$ when $T_G(x, y) - T_H(x, y) = (x + y - xy)P_H(x, y)$ for some nonnegative polynomial $P_H(x, y)$. The graph G is *Tutte-maximum* if $H \preceq G$ for each H in $\mathcal{C}_{n,m}$.

Now, let us present key concepts from network reliability. Let G be any graph in $\mathcal{C}_{n,m}$. For each $k \in \{1, 2, \dots, n\}$ and each $p \in [0, 1]$, the k -*reliability* of G at p , denoted $R_G^{(k)}(p)$, is the probability that G has at most k connected components after each of its edges is independently retained with probability p . For each $i \in \{0, 1, \dots, m\}$, let $N_i^{(k)}(G)$ be the number of spanning subgraphs H in G composed by i edges having at most k connected components. Clearly,

$$R_G^{(k)}(p) = \sum_{i=0}^m N_i^{(k)}(G) p^i (1-p)^{m-i}. \quad (3)$$

If $k \in \{1, 2, \dots, n\}$ then G is a k -*uniformly most reliable graph* (k -UMRG) if $R_G^{(k)}(p) \geq R_H^{(k)}(p)$ for each H in $\mathcal{C}_{n,m}$ and p in $[0, 1]$. The graph G is a *uniformly most reliable graph* [2] if it is 1-UMRG. Theorem 1 is a consequence of the universality property of the Tutte-polynomial [4].

Theorem 1. If G is any graph in $\mathcal{C}_{n,m}$ and $p \in (0, 1)$ then

$$R_G^{(1)}(p) = p^{n-1} (1-p)^{m-n+1} T_G \left(1, \frac{1}{1-p} \right)$$

As corollary, each Tutte-maximum graph is uniformly most reliable [8].

A graph G is a *0-element* in $\mathcal{C}_{n,m}$ if $N_i^{(1)}(G) \geq N_i^{(1)}(H)$ for each i in $\{0, 1, \dots, m\}$ and each H in $\mathcal{C}_{n,m}$. From Eq. (3) it is clear that each 0-element is uniformly most reliable. The converse is an unresolved conjecture proposed in 1986 by Boesch [2].

Conjecture 1 (Boesch [2]). Each uniformly most reliable graph in $\mathcal{C}_{n,m}$ is a 0-element in $\mathcal{C}_{n,m}$.

The concept of a 0-element in $\mathcal{C}_{n,m}$ was introduced by Boesch [2] and its name is explained in the following. For each G in $\mathcal{C}_{n,m}$ we define $\mu_i(G) = \binom{m}{i} - N_i^{(1)}(G)$ where $i \in \{0, 1, \dots, m\}$. Denote $\mu(G) = (\mu_0(G), \mu_1(G), \dots, \mu_m(G))$. Two graphs G and H in $\mathcal{C}_{n,m}$ are *equivalent* if $\mu(G) = \mu(H)$. The class of equivalent graphs in $\mathcal{C}_{n,m}$ equipped with the lexicographic order among tuples $\mu(G)$ is a partially ordered set. A minimum element, if any, is a 0-element as defined by Boesch.

If G has at least k vertices then its *k -order edge connectivity* of G , denoted $\lambda^{(k)}(G)$, equals the minimum number of edges that must be removed to G to obtain a spanning subgraph H such that $\kappa(H) > k$. The maximization of the 1-edge connectivity of a graph was studied by Harary [6]. The second-order edge connectivity of G was discussed by Goldsmith et al. [5].

For each k in $\{1, 2, \dots, n\}$, $t_k(G)$ denotes the number of spanning forests in G composed by k trees. The number $t_1(G)$ is known as the *tree-number* of G . A graph G in $\mathcal{C}_{n,m}$ is called *t -optimal* if $t_1(G) \geq t_1(H)$ for all H in $\mathcal{C}_{n,m}$. The determination of t -optimal graphs in each nonempty class $\mathcal{C}_{n,m}$ is still unresolved; the reader can find a characterization of some t -optimal graphs in [11].

Several graph invariants are evaluations of the Tutte or Whitney polynomial. As an example, let us derive the *forest generating function* of a graph G in $\mathcal{C}_{n,m}$ defined by $\sum_{i=0}^{n-1} t_{i+1}(G)x^i$. If $\mathcal{F}(G)$ denotes the set of all forests of G then

$$W_G(x, 0) = \sum_{H \in \mathcal{F}(G)} x^{\kappa(H)-1} = \sum_{i=0}^{n-1} t_{i+1}(G)x^i,$$

where we used that there are precisely $t_i(G)$ forests in G composed by i trees. Then, for each $i \in \{1, 2, \dots, n\}$,

$$t_i(G) = \frac{\partial^{i-1} W}{\partial x^{i-1}}(0, 0).$$

In particular, the tree-number $t_1(G)$ of a graph G equals $W_G(0, 0)$. The reader can find a list of real-valued graph invariants that are evaluations of the Tutte-polynomial in the recent book [4]. It is worth to remark that a great variety of graph invariants are either maximized or minimized among all graphs in $\mathcal{C}_{n,m}$ by Tutte-maximum graphs; see [8] for further details.

3 Main Result

Boesch [2] discussed the existence of a 0-element in $\mathcal{C}_{n,m}$. Let us consider the following generalization.

Definition 2. A graph G in $\mathcal{C}_{n,m}$ is strong if $N_i^{(k)}(G) \geq N_i^{(k)}(H)$ for each $H \in \mathcal{C}_{n,m}$ and each pair of integers $i \in \{0, 1, \dots, m\}$ and $k \in \{1, 2, \dots, n\}$.

By Eq. (3), each strong graph is k -UMRG for all $k \in \{1, 2, \dots, n\}$. As a consequence, the existence and construction of strong graphs is a relevant topic in network reliability in the wide sense as presented by Kelmans [9]. In this section we will find an algebraic characterization of strong graphs. Consider the following definition inspired by the definition of Tutte-maximum graphs.

Definition 3. For each pair of graphs G and H in $\mathcal{C}_{n,m}$ we write $H \preceq_W G$ when $W_G(x, y) - W_H(x, y) = (1 - xy)Q_H(x, y)$ for some nonnegative polynomial $Q_H(x, y)$. The graph G is Whitney-maximum if $H \preceq_W G$ for each H in $\mathcal{C}_{n,m}$.

All the properties satisfied by Whitney-maximum graphs are also satisfied by Tutte-maximum graphs. In fact, the following lemma holds.

Lemma 1. Each Tutte-maximum graph is Whitney-maximum.

Proof. Let G be a Tutte-maximum graph in $\mathcal{C}_{n,m}$ and H be any graph in $\mathcal{C}_{n,m}$. As G is Tutte-maximum, there exists some nonnegative polynomial $P(x, y)$ such that $T_G(x, y) - T_H(x, y) = (x + y - xy)P(x, y)$. Consequently,

$$\begin{aligned} W_G(x, y) - W_H(x, y) &= T_G(x + 1, y + 1) - T_H(x + 1, y + 1) \\ &= ((x + 1) + (y + 1) - (x + 1)(y + 1))P(x + 1, y + 1) \\ &= (1 - xy)P(x + 1, y + 1). \end{aligned}$$

As $P(x, y)$ is nonnegative, $P(x + 1, y + 1)$ is also nonnegative. The lemma follows. \square

A natural question that arises is whether or not each Whitney-maximum graph is Tutte-maximum. Let G and H be the graphs depicted in Fig. 1.

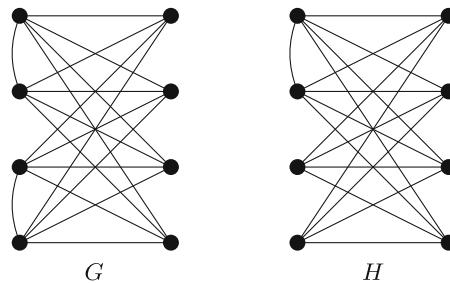


Fig. 1. Graphs G and H .

Computation shows that G is the only Whitney-maximum graph in $\mathcal{C}_{8,18}$ up to isomorphism. However,

$$T_G(x, y) - T_H(x, y) = (x + y - xy)P(x, y),$$

where $P(x, y) = 4xy^5 + x^3y^2 + 4x^2y^3 + 12xy^4 + 2x^3y + 13x^2y^2 + 24xy^3 - x^3 + x^2y + 9xy^2 - 8y^3 - 4x^2 - 12xy - 19y^2 - 7x - 15y - 4$. As $P(x, y)$ has negative coefficients, G is not Tutte-maximum disproving the converse of Lemma 1.

Now, let us study the relationship between strong graphs and Whitney-maximum graphs. The following technical lemma will be useful.

Lemma 2. *Let G be any graph in $\mathcal{C}_{n,m}$. For each $k \in \{1, 2, \dots, n\}$ and each $i \in \{0, 1, \dots, m\}$,*

$$N_i^{(k)}(G) = \sum_{j=\max\{1, n-i\}}^k \frac{1}{(j-1)!(i-n+j)!} \frac{\partial^{2j+i-n-1} W_G}{\partial x^{j-1} y^{i-n+j}}(0, 0).$$

Proof. Let G be any graph in $\mathcal{C}_{n,m}$ and i and k as in the statement. Define $N_{i,j}(G)$ as the number of spanning graphs of G composed precisely by i edges and j connected components. Observe that $N_i^{(k)}(G) = \sum_{j=1}^k N_{i,j}(G)$ and, further, if $j < n - i$ then $N_{i,j}(G) = 0$. Therefore,

$$N_i^{(k)}(G) = \sum_{j=1}^k N_{i,j}(G) = \sum_{j=\max\{1, n-i\}}^k N_{i,j}(G). \quad (4)$$

Let j be any positive integer such that $j \geq n - i$. By definition, $N_{i,j}(G)$ equals the coefficient of $x^{j-1} y^{i-n+j}$ in $W_G(x, y)$. Consequently,

$$N_{i,j}(G) = \frac{1}{(j-1)!(i-n+j)!} \frac{\partial^{2j+i-n-1} W_G}{\partial x^{j-1} y^{i-n+j}}(0, 0). \quad (5)$$

The lemma follows replacing Eq. (5) into (4). \square

Lemma 3. *If G and H are two graphs in $\mathcal{C}_{n,m}$ such that $H \preceq_W G$ then, for each $i \in \{0, 1, \dots, m\}$ and each $k \in \{1, 2, \dots, n\}$, $N_i^{(k)}(H) \leq N_i^{(k)}(G)$.*

Proof. Let G and H be two graphs in $\mathcal{C}_{n,m}$ and let i and k as in the statement. Define $Q(x, y) = 1 - xy$. As $H \preceq_W G$, there exists some nonnegative polynomial $P(x, y)$ such that $W_G(x, y) - W_H(x, y) = P(x, y)Q(x, y)$. Let j be any integer in $\{1, 2, \dots, k\}$ such that $i - n + j \geq 0$. Observe that the second and higher order derivatives of $Q(x, y)$ with respect to x are equal to 0. Then, by Leibniz rule,

$$\begin{aligned} \frac{\partial^{j-1}(PQ)}{\partial x^{j-1}}(x, y) &= \sum_{\ell=0}^{j-1} \binom{j-1}{\ell} \frac{\partial^\ell P}{\partial x^\ell}(x, y) \frac{\partial^{j-1-\ell} Q}{\partial x^{j-1-\ell}}(x, y) \\ &= \frac{\partial^{j-1}P}{\partial x^{j-1}}(x, y)Q(x, y) + (j-1) \frac{\partial^{j-2}P}{\partial x^{j-2}}(x, y) \frac{\partial Q}{\partial x}(x, y), \end{aligned} \quad (6)$$

where we use the convention that the second term on the right hand side of Eq. (6) is 0 when $j = 1$.

If we now derive $i - n + j$ times expression (6) with respect to y followed by an evaluation at $(x, y) = (0, 0)$,

$$\begin{aligned} \frac{\partial^{2j+i-n-1}(PQ)}{\partial x^{j-1}\partial y^{i-n+j}}(0, 0) &= \sum_{\ell=0}^{i-n+j} \binom{i-n+j}{\ell} \frac{\partial^{j-1+\ell}P}{\partial x^{j-1}\partial y^\ell}(0, 0) \frac{\partial^{i-n+j-\ell}Q}{\partial y^{i-n+j-\ell}}(0, 0) \\ &+ (j-1) \sum_{\ell=0}^{i-n+j} \binom{i-n+j}{\ell} \frac{\partial^{j-2+\ell}P}{\partial x^{j-2}\partial y^\ell}(0, 0) \frac{\partial^{i-n+j-\ell+1}Q}{\partial x\partial y^{i-n+j-\ell}}(0, 0) \\ &= \frac{\partial^{2j+i-n-1}P}{\partial x^{j-1}\partial y^{i-n+j}}(0, 0) - (j-1)(i-n+j) \frac{\partial^{2j+i-n-3}P}{\partial x^{j-2}\partial y^{i-n+j-1}}(0, 0), \end{aligned}$$

where we used Leibniz rule and the facts that $\frac{\partial Q}{\partial y^s}(0, 0) = 0$ for each $s \geq 1$ and that $\frac{\partial Q}{\partial x\partial y}(0, 0) = -1$. Now, by Lemma 2,

$$\begin{aligned} N_i^{(k)}(G) - N_i^{(k)}(H) &= \sum_{j=\max\{1, n-i\}}^k \frac{1}{(j-1)!(i-n+j)!} \frac{\partial^{2j+i-n-1}(W_G - W_H)}{\partial x^{j-1}\partial y^{i-n+j}}(0, 0) \\ &= \sum_{j=\max\{1, n-i\}}^k \frac{1}{(j-1)!(i-n+j)!} \frac{\partial^{2j+i-n-1}(PQ)}{\partial x^{j-1}\partial y^{i-n+j}}(0, 0) \\ &= \sum_{j=\max\{1, n-i\}}^k \frac{1}{(j-1)!(i-n+j)!} \cdot \\ &\quad \left(\frac{\partial^{2j+i-n-1}P}{\partial x^{j-1}\partial y^{i-n+j}}(0, 0) - (j-1)(i-n+j) \frac{\partial^{2j+i-n-3}P}{\partial x^{j-2}\partial y^{i-n+j-1}}(0, 0) \right) \\ &= \frac{1}{(k-1)!(i-n+k)!} \frac{\partial^{2k+i-n-1}P}{\partial x^{k-1}\partial y^{i-n+k}}(0, 0) \geq 0, \end{aligned}$$

where we identified a telescopic sum in the last equality and we used that $P(x, y)$ is nonnegative in the inequality. Consequently, $N_i^{(k)}(H) \leq N_i^{(k)}(G)$, as required. \square

Lemma 4. *Each strong graph is Whitney-maximum.*

Proof. Let G be a strong graph in $\mathcal{C}_{n,m}$ and let G' be any graph in $\mathcal{C}_{n,m}$. The proof strategy is the following. First, we will construct a bijective mapping φ from $\mathcal{S}(G)$ to $\mathcal{S}(G')$ such that $|E(\varphi(H))| = |E(H)|$ and $\kappa(\varphi(H)) \geq \kappa(H)$ for each H in $\mathcal{S}(G)$. Then, we will prove that for each spanning subgraph H in $\mathcal{S}(G)$ it holds that $x^{r(G)-r(H)}y^{c(H)} - x^{r(G')-r(\varphi(H))}y^{c(\varphi(H))} = (1-xy)P_H(x, y)$ for some nonnegative polynomial $P_H(x, y)$. The lemma will follow by summing the contributions of all spanning subgraphs H in $\mathcal{S}(G)$ and the fact that the addition of nonnegative polynomials is also a nonnegative polynomial.

Now, we will construct the bijective mapping φ satisfying the aforementioned conditions. Consider, for each $i \in \{0, 1, \dots, m\}$ and each $j, k \in \{1, 2, \dots, n\}$, the following sets,

$$\begin{aligned}\mathcal{S}_{i,j}(G) &= \{H : H \in \mathcal{S}(G), |E(H)| = i, \kappa(H) = j\}, \\ \mathcal{S}_i^{(k)}(G) &= \{H : H \in \mathcal{S}(G), |E(H)| = i, \kappa(H) \leq k\}.\end{aligned}$$

The sets $\mathcal{S}_{i,j}(G')$ and $\mathcal{S}_i^{(k)}(G')$ are defined analogously (replacing G by G'). Observe that $N_i^{(k)}(G) = |\mathcal{S}_i^{(k)}(G)|$ and $\mathcal{S}_i^{(k)}(G) = \cup_{j=1}^k \mathcal{S}_{i,j}(G)$. As G is strong, $N_i^{(1)}(G) \geq N_i^{(1)}(G')$ and the set $\mathcal{S}_i^{(1)}(G)$ has at least as many elements as $\mathcal{S}_i^{(1)}(G')$. Pick some set $\mathcal{G}_i^{(1)}$ in $\mathcal{S}_i^{(1)}(G)$ such that $|\mathcal{G}_i^{(1)}| = |\mathcal{S}_i^{(1)}(G')|$. Similarly, as $N_i^{(2)}(G) \geq N_i^{(2)}(G')$, the set $\mathcal{S}_i^{(2)}(G)$ has at least as many elements as $\mathcal{S}_i^{(2)}(G')$. Equivalently, $|\mathcal{S}_i^{(2)}(G) - \mathcal{G}_i^{(1)}| \geq |\mathcal{S}_i^{(2)}(G') - \mathcal{S}_i^{(1)}(G')| = |\mathcal{S}_{i,2}(G')|$. Consequently, there exists some set $\mathcal{G}_i^{(2)}$ in $\mathcal{S}_i^{(2)}(G) - \mathcal{G}_i^{(1)}$ such that $|\mathcal{G}_i^{(2)}| = |\mathcal{S}_{i,2}(G')|$. If we repeat this process we will construct, for each i in $\{0, 1, \dots, m\}$, some disjoint sets $\mathcal{G}_i^{(1)}, \dots, \mathcal{G}_i^{(n)}$ such that $|\mathcal{G}_i^{(j)}| = |\mathcal{S}_{i,j}(G')|$ for each $j \in \{1, 2, \dots, n\}$, where $\mathcal{G}_i^{(j)} \subseteq \mathcal{S}_i^{(j)}(G)$. Further, observe that the sets $\mathcal{G}_i^{(1)}, \dots, \mathcal{G}_i^{(n)}$ are not only disjoint but also define a partition of $\mathcal{S}_i(G)$, since

$$|\mathcal{S}_i(G)| \geq \sum_{j=1}^n |\mathcal{G}_i^{(j)}| = \sum_{j=1}^n |\mathcal{S}_{i,j}(G')| = |\mathcal{S}_i(G')| = |\mathcal{S}_i(G)|$$

thus the previous expression is a chain of equalities. Consequently, the set of spanning subgraphs in G can be written as follows: $\mathcal{S}(G) = \cup_{i=0}^m \mathcal{S}_i(G) = \cup_{i=0}^m \cup_{j=1}^n \mathcal{G}_i^{(j)}$, and clearly, $\mathcal{S}(G') = \cup_{i=0}^m \cup_{j=1}^n \mathcal{S}_{i,j}(G')$. By construction, we know that $|\mathcal{G}_i^{(j)}| = |\mathcal{S}_{i,j}(G')|$ for each $i \in \{0, 1, \dots, m\}$ and $j \in \{1, 2, \dots, n\}$. Then, for each $i \in \{0, 1, \dots, m\}$ and $j \in \{1, 2, \dots, n\}$ there exists some bijective mapping $\varphi_{i,j}$ from $\mathcal{G}_i^{(j)}$ to $\mathcal{S}_{i,j}(G')$. Define the mapping $\varphi : \mathcal{S}(G) \rightarrow \mathcal{S}(G')$ such that, for each H in $\mathcal{G}_i^{(j)}$, $\varphi(H) = \varphi_{i,j}(H)$. In the following we will denote $H' = \varphi(H)$.

Now, we will prove that for each H in $\mathcal{S}(G)$, $\kappa(H') \geq \kappa(H)$. In fact, as H belongs to $\mathcal{S}(G)$, there exists $i \in \{0, 1, \dots, m\}$ and $j \in \{1, 2, \dots, n\}$ such that $H \in \mathcal{G}_i^{(j)}$, and $H' \in \mathcal{S}_{i,j}(G')$. On the one hand, $\mathcal{G}_i^{(j)} \subseteq \mathcal{S}_i^{(j)}$, and $\kappa(H) \leq j$. On the other hand, as $H' \in \mathcal{S}_{i,j}$, we have that $\kappa(H') = j$. Consequently, $\kappa(H') \geq \kappa(H)$.

Let us define, for each spanning graph H in $\mathcal{G}_i^{(j)}$, the function $f_H(x, y)$ given by $x^{r(G)-r(H)}y^{c(H)} - x^{r(G')-r(H')}y^{c(H')}$.

By construction $|E(H)| = |E(H')| = i$ and $\kappa(H') \geq \kappa(H)$. Define s as $\kappa(H') - \kappa(H)$. As both G and G' are connected on n vertices, $r(G) = r(G')$ and $r(G') - r(H') = r(G) - r(H) + s$. Additionally, $c(H') = i - n + \kappa(H') = i - n + \kappa(H) + s = c(H) + s$. If $s = 0$ then $f_H(x, y) = 0$ and defining $P_H(x, y) = 0$ we get that $f_H(x, y) = (1 - xy)P(x, y)$ where $P(x, y)$ is nonnegative. Otherwise,

$$\begin{aligned}
f_H(x, y) &= x^{r(G)-r(H)}y^{c(H)} - x^{r(G')-r(H')}y^{c(H')} \\
&= x^{r(G)-r(H)}y^{c(H)}(1 - (xy)^s) = (x^{r(G)-r(H)}y^{c(H)}) \sum_{\ell=0}^{s-1} (xy)^\ell (1 - xy) \\
&= P_H(x, y)(1 - xy),
\end{aligned}$$

where $P_H(x, y) = x^{r(G)-r(H)}y^{c(H)} \sum_{\ell=0}^{r-1} (xy)^\ell$ is a nonnegative polynomial. Then,

$$W_G(x, y) - W_{G'}(x, y) = \sum_{H \in \mathcal{S}(G)} f_H(x, y) = (\sum_{H \in \mathcal{S}(G)} P_H(x, y))(1 - xy).$$

As the addition of nonnegative polynomials is nonnegative, the lemma follows. \square

Theorem 2. *A graph is Whitney-maximum if and only if it is strong.*

Proof. Let G be a graph in $\mathcal{C}_{n,m}$. First, assume G is Whitney-maximum. Let H be any graph in $\mathcal{C}_{n,m}$. As G is Whitney-maximum, $H \preceq_W G$. By Lemma 3 we know that $N_i^{(k)}(G) \geq N_i^{(k)}(H)$ for each $i \in \{0, 1, \dots, m\}$ and each $k \in \{1, 2, \dots, n\}$. As H is an arbitrary graph in $\mathcal{C}_{n,m}$ we conclude that G is a strong graph. By Lemma 4 we know that each strong graph is Whitney-maximum. The theorem follows. \square

4 Applications

Remark 1 gives closed forms for the k -order edge connectivity and the number of spanning forests composed by k trees of G as a function of its coefficients $N_i^{(k)}(G)$.

Remark 1. If k is a positive integer and G is a graph in $\mathcal{C}_{n,m}$ such that $n \geq k$, then the following inequalities hold,

- (i) $\lambda^{(k)}(G) = \min\{x : x \geq 0, N_{m-x}^{(k)}(G) < \binom{m}{m-x}\}$,
- (ii) $t_k(G) = N_{n-k}^{(k)}(G)$.

We are in position to prove the following result.

Theorem 3. *Each Whitney-maximum graph G in $\mathcal{C}_{n,m}$ is k -UMRG for all $k \in \{1, 2, \dots, n\}$. For each $k \in \{1, 2, \dots, n\}$, the graph G has the maximum k -edge order connectivity as well as the maximum number of spanning forests composed by k trees.*

Proof. Let G be any Whitney-maximum graph in $\mathcal{C}_{n,m}$. By Theorem 2 we know that G is strong. Therefore, by Eq. (3) it follows that G is k -UMRG for each $k \in \{1, 2, \dots, n\}$.

Now, let H be any graph in $\mathcal{C}_{n,m}$. As G is Whitney-maximum, $H \preceq_W G$. Therefore, Lemma 3 gives that $N_i^{(k)}(H) \leq N_i^{(k)}(G)$ for each $k \in \{1, 2, \dots, n\}$ and each $i \in \{0, 1, \dots, m\}$. The second sentence of the statement follows from Remark 1. The theorem follows. \square

Conjecture 1 is still unresolved. Nevertheless, such conjecture is true when restricted to graph classes $\mathcal{C}_{n,m}$ for which a Whitney-maximum graph exists.

Proposition 1. *If there exists some Whitney-maximum graph in $\mathcal{C}_{n,m}$ then each uniformly most reliable graph in $\mathcal{C}_{n,m}$ is a 0-element in $\mathcal{C}_{n,m}$.*

Proof. Let G be a Whitney-maximum graph in $\mathcal{C}_{n,m}$. By Theorem 3 we know that G is strong. Therefore, G is a 0-element and uniformly most reliable as well. Let H be any uniformly most reliable graph in $\mathcal{C}_{n,m}$. It is enough to prove that H is a 0-element. If H is isomorphic to G then we are done. Otherwise, as both graphs G and H are uniformly most reliable it follows that $R_G^{(1)}(p) = R_H^{(1)}(p)$ for all $p \in [0, 1]$ and consequently, $N_i^{(1)}(G) = N_i^{(1)}(H)$ for each $i \in \{0, 1, \dots, m\}$. As G is a 0-element it follows that H is also a 0-element as required. \square

Acknowledgments. This work is partially supported by City University of New York project entitled *On the problem of characterizing graphs with maximum number of spanning trees*, grant number 66165-00. The author wants to thank Dr. Martín Safe as well as the anonymous reviewers for their helpful comments that improved the presentation of this manuscript.

References

1. Birkhoff, G.D.: A determinant formula for the number of ways of coloring a map. *Ann. Math.* **14**(1/4), 42–46 (1912)
2. Boesch, F.: On unreliability polynomials and graph connectivity in reliable network synthesis. *J. Graph Theory* **10**(3), 339–352 (1986)
3. Boesch, F., Li, X., Suffel, C.: On the existence of uniformly optimally reliable networks. *Networks* **21**(2), 181–194 (1991)
4. Ellis-Monaghan, J., Moffatt, I.: *Handbook of the Tutte Polynomial and Related Topics*. CRC Press, Boca Raton (2022)
5. Goldsmith, D., Manvel, B., Faber, V.: Separation of graphs into three components by the removal of edges. *J. Graph Theory* **4**(2), 213–218 (1980)
6. Harary, F.: The maximum connectivity of a graph. *Proc. Natl. Acad. Sci.* **48**(7), 1142–1146 (1962)
7. Kahl, N.: Extremal graphs for the Tutte polynomial. *J. Comb. Theory Ser. B* **152**, 121–152 (2022)
8. Kahl, N., Luttrell, K.: On maximum graphs in Tutte polynomial posets. *Disc. Appl. Math.* **339**, 78–88 (2023)
9. Kelmans, A.: On graphs with randomly deleted edges. *Acta Math. Hungar.* **37**(1–3), 77–88 (1981)
10. Moore, E., Shannon, C.: Reliable circuits using less reliable relays. *J. Franklin Inst.* **262**(3), 191–208 (1956)
11. Petingi, L., Rodríguez, J.: A new technique for the characterization of graphs with a maximum number of spanning trees. *Disc. Math.* **244**(1), 351–373 (2002)
12. Romero, P.: Uniformly optimally reliable graphs: a survey. *Networks* **80**(4), 466–481 (2022)
13. Tutte, W.T.: A contribution to the theory of chromatic polynomials. *Can. J. Math.* **6**, 80–91 (1954)
14. Whitney, H.: The coloring of graphs. *Ann. Math.* **33**(4), 688–718 (1932)



There are Finitely Many Uniformly Most Reliable Graphs of Corank 5

Pablo Romero^{1,2(✉)}

¹ Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay

² Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires,
Buenos Aires, Argentina
promero@fing.edu.uy

Abstract. If G is a simple graph and $\rho \in [0, 1]$, the reliability $R_G(\rho)$ is the probability of G being connected after each of its edges is removed independently with probability ρ . A simple graph G is a *uniformly most reliable graph* (UMRG) if $R_G(\rho) \geq R_H(\rho)$ for every $\rho \in [0, 1]$ and every simple graph H on the same number of vertices and edges as G . Boesch conjectured that, if n and m are such that there exists a connected simple graph on n vertices and m edges, then there also exists a UMRG on the same number of vertices and edges. Some counterexamples to Boesch's conjecture appeared in the literature. It is known that Boesch's conjecture holds whenever the corank, defined as $c = m - n + 1$, is at most 4 (and the corresponding UMRGs are fully characterized). Ath and Sobel conjectured that Boesch's conjecture holds whenever the corank c is between 5 and 8, provided that the number of vertices is at least $2c - 2$. It is known that for each positive integer s there is no uniformly most reliable graph of corank 5 and $12s + 4$ vertices. Here it is proved that there are only finitely many uniformly most reliable graphs of corank 5. This is in strong contrast with classes of graphs whose corank is not greater than 4 for which uniformly most reliable graphs always exist.

Keywords: Reliability · Uniformly most reliable graph · Corank

1 Introduction

If G is a simple graph and $\rho \in [0, 1]$, the *reliability of G with failure probability ρ* , denoted $R_G(\rho)$, is the probability of G being connected after each of its edges is removed independently with probability ρ . Given positive integers n and m such that $n - 1 \leq m \leq \binom{n}{2}$, the question arises as to whether there exists a graph G in the class $\mathcal{C}_{n,m}$ of connected simple graphs on n vertices and m edges such that $R_G(\rho) \geq R_H(\rho)$ for every H in $\mathcal{C}_{n,m}$ and every $\rho \in [0, 1]$. Such a simple graph G is called a *uniformly most reliable graph* (UMRG). This concept was introduced in 1986 by Boesch in his seminal article [2].

The corank of each graph G in $\mathcal{C}_{n,m}$ is defined as $m - n + 1$. We will also say that the corank of the class $\mathcal{C}_{n,m}$ is $m - n + 1$. Regarding dense graphs, it is known that there exists infinitely many classes $\mathcal{C}_{n,m}$ in which there is no UMRG

thus Boesch's conjecture was disproved [3, 5, 6]. Regarding sparse graphs, in each nonempty class $\mathcal{C}_{n,m}$ having corank at most 4 there exists precisely one UMRG. The reader is invited to consult the survey [7] for further details.

Ath and Sobel [1] proposed the following weaker conjecture in 2000.

Conjecture 1 (Ath-Sobel [1]). If a nonempty class $\mathcal{C}_{n,m}$ has corank $c \in \{5, 6, 7, 8\}$ and $n \geq 2c - 2$, then $\mathcal{C}_{n,m}$ contains at least one UMRG.

Furthermore, explicit candidates for such UMRGs were also proposed in [1]. Conjecture 1 was disproved when $c = 5$.

Theorem 2 ([8]). *For each positive integer s there is no UMRG in the class $\mathcal{C}_{12s+4, 12s+8}$.*

It is known [6] that there is precisely one UMRG in each of the classes of corank 5 and n vertices where $n \in \{5, 6, 7, 8\}$. By Theorem 2, there exists infinitely many classes of corank 5 with no UMRG. Here, we will prove that there are only finitely many classes $\mathcal{C}_{n,m}$ of corank 5 having some UMRG. As each class $\mathcal{C}_{n,m}$ is finite, there are only finitely many UMRG having corank 5. This is in strong contrast with classes of graphs whose corank is not greater than 4 for which UMRG always exist.

This article is organized as follows. A proof strategy is given in Sect. 2. A background including some concepts from graph theory as well as preliminary results is presented in Sect. 3. The concept of vertex-fair graph is introduced in Sect. 4. The main result of this work is given in Sect. 5.

2 Proof Strategy

Let G be a graph in $\mathcal{C}_{n,m}$. An *edge-cut* of G is a set F of edges of G such that $G - F$ is disconnected. A *k-edge-cut* is an edge-cut of size k . We denote $\mu_k(G)$ the number of k -edge-cuts in G . Clearly, for each $\rho \in [0, 1]$,

$$R_G(\rho) = 1 - \sum_{k=0}^m \mu_k(G) \rho^k (1-\rho)^{m-k}.$$

The following theorem can be proved using elementary calculus.

Theorem 3 (Brown and Cox [3]). *Let G and H be graphs in $\mathcal{C}_{n,m}$.*

- (i) *If there exists $i \in \{0, 1, \dots, m\}$ such that $\mu_k(G) = \mu_k(H)$ for all $k \in \{0, 1, \dots, i-1\}$ and $\mu_i(G) < \mu_i(H)$, then there exists some $\delta > 0$ such that $R_G(\rho) > R_H(\rho)$ for every $\rho \in (0, \delta)$.*
- (ii) *If there exists $j \in \{0, 1, \dots, m\}$ such that $\mu_k(G) = \mu_k(H)$ for all $k \in \{j+1, j+2, \dots, m\}$ and $\mu_j(G) < \mu_j(H)$, then there exists some $\delta > 0$ such that $R_G(\rho) > R_H(\rho)$ for every $\rho \in (1-\delta, 1)$.*

Consider any nonempty class $\mathcal{C}_{n,m}$ and any graph G in $\mathcal{C}_{n,m}$. If $k > m - n + 1$ then, clearly, $\mu_k(G) = \binom{m}{k}$. Let $t(G)$ be the number of spanning trees of G . Observe that $\mu_{m-n+1}(G) = \binom{m}{m-n+1} - t(G)$.

Definition 4. A graph H in $\mathcal{C}_{n,m}$ is t -optimal if $\mu_{m-n+1}(H) \leq \mu_{m-n+1}(G)$ for each G in $\mathcal{C}_{n,m}$.

The following remark is essential for our proof strategy of nonexistence. It is just an application of Theorem 3(ii) using $j = m - n + 1$.

Remark 5. Each uniformly most reliable graph in $\mathcal{C}_{n,m}$ is t -optimal.

Now, consider any nonempty class $\mathcal{C}_{n,m}$. Sort all graphs G in $\mathcal{C}_{n,m}$ using the lexicographic order among all tuples $(\mu_0(G), \mu_1(G), \dots, \mu_m(G))$. Define $\mathcal{C}_{n,m}^0$ as $\mathcal{C}_{n,m}$ and $\mathcal{C}_{n,m}^{i+1} = \{G : G \in \mathcal{C}_{n,m}^i, \forall H \in \mathcal{C}_{n,m}^i, \mu_{i+1}(G) \leq \mu_{i+1}(H)\}$ for each $i \in \{0, 1, \dots, m-1\}$. As $\mathcal{C}_{n,m}$ is finite and nonempty, each set $\mathcal{C}_{n,m}^i$ is finite and nonempty as well. Our proof strategy is summarized in the following lemma.

Lemma 6. If $\mathcal{C}_{n,m}$ is nonempty, $\mathcal{C}_{n,m}^j = \{G\}$ for some positive integer j and G is not t -optimal, then there is no UMRG in $\mathcal{C}_{n,m}$.

Proof. As G is not t -optimal, by Remark 5 we know that G is not UMRG. Let H be any other graph in $\mathcal{C}_{n,m}$. As $\mathcal{C}_{n,m}^j = \{G\}$, there exists some integer $i \in \{0, 1, \dots, j-1\}$ such that $\mu_k(G) = \mu_k(H)$ for all $k \in \{0, 1, \dots, i-1\}$ but $\mu_i(G) < \mu_i(H)$. By Theorem 3(i), there exists some $\delta > 0$ such that $R_G(\rho) > R_H(\rho)$ for every $\rho \in (0, \delta)$. Consequently, H is not UMRG. The lemma follows. \square

In Sect. 5 we will find the only graph G_n in $\mathcal{C}_{n,n+4}$ such that $\mathcal{C}_{n,n+4}^4 = \{G_n\}$. Then we will find a graph H_n in each set $\mathcal{C}_{n,n+4}$ such that $\mu_5(H_n) < \mu_5(G_n)$ for all $n \geq n_0$. The main theorem will then follow from Lemma 6.

3 Background

In this section we include some concepts that already appeared in [8] as well as some preliminary results that will be useful for our purpose.

All graphs in this work are finite and undirected. We denote the path, the cycle, and the complete graph on n vertices by P_n , C_n , and K_n , respectively. The Wagner graph W and the cube graph Q are depicted in Fig. 1. For each positive integer p such that $p \geq 2$, the Möbius graph, denoted M_p , consists of C_{2p} plus p edges each one joining opposite vertices in C_{2p} .

The *edge-connectivity* of a graph $G \neq K_1$, denoted $\lambda(G)$, is the minimum k such that G has a k -edge-cut. Let F be an edge-cut of G . We say F separates a set S of vertices of G if F contains all edges with precisely one endpoint in S and no edge with both endpoints in S . Notice that F may also contain some edges with no endpoints in S . If $S = \{v\}$ for some vertex v of G , we say that F is *vertex-separating*. If $S = \{u, v\}$ where u and v are the endpoints of an edge e of G , we say that F is *edge-separating*. An edge-cut is *nontrivial* if it is neither vertex-separating nor edge-separating. If S induces a graph H in G , we say F is *H -separating*.

Let G be a graph with no loops. An edge e is *incident* to a vertex v if v is an endpoint of e . The *degree* $d_G(v)$ of a vertex v of G is the number of edges incident

to it. We say G is *cubic* if all its vertices have degree 3. Two edges are *nonincident*, *incident*, or *parallel* if they share precisely 0, 1, or 2 endpoints, respectively. By *subdividing k times* an edge with endpoints x and y , we mean replacing the edge xy by $k+1$ edges $xv_1, v_1v_2, \dots, v_{k-1}v_k, v_ky$, where v_1, v_2, \dots, v_k are k new vertices of degree 2 each.

A simple graph G is *2-connected* if it has at least 3 vertices, it is connected, and $G - v$ is connected for all v in $V(G)$. Let G be a 2-connected simple graph having more edges than vertices. A *chain* γ of G is the edge set of a path P in G , where all internal vertices of P (if any) have degree 2 in G and P has two distinct endpoints of degree greater than 2 in G each. The *endpoints* of γ are those of P and γ is *incident* to a vertex v if v is one of its endpoints. The *internal vertices* of γ are those of P . By *removing* γ from G , we mean removing the edges and internal vertices of γ (but not its endpoints). The graph that results by removing γ from G is denoted $G \ominus \gamma$. If \mathcal{H} is a set of chains of G , we denote $G \ominus \mathcal{H}$ the graph that arises from G by removing all the chains in \mathcal{H} . By *collapsing* γ we mean removing γ and adding an edge with the same endpoints as γ . We denote $\Gamma(G)$ the set of all chains of G . The *distillation* of G , denoted $D(G)$, is the graph that arises from G by collapsing all of its chains. Clearly, G arises from $D(G)$ by a sequence of (possibly zero) subdivisions.

Let G be in $\mathcal{C}_{n,m}$. If $k \in \{0, 1, \dots, m\}$, we say G is *min- μ_k* if $\mu_k(G) \leq \mu_k(H)$ for every H in $\mathcal{C}_{n,m}$. Theorem 7 restricts our study to 2-connected graphs.

Theorem 7 (Wang [9]). *Let G be a simple graph on n vertices and m edges such that $m > n$. If G is min- μ_k for some $k \in \{\lambda(G), \lambda(G) + 1, \dots, m - n + 1\}$, then G is 2-connected.*

If c is a positive integer, $[c]$ denotes the set $\{1, 2, \dots, c\}$. If k is a nonnegative integer, then the family of all the subsets of a set S with cardinality k is denoted $\binom{S}{k}$. Let G be a 2-connected simple graph having more edges than vertices. Let $\Gamma(G)$ be the set of all chains of G . Moreover, if k is a nonnegative integer, let $\Gamma^{(k)}(G)$ be the family of all subsets of $\Gamma(G)$ of size k ; i.e., $\Gamma^{(k)}(G) = \binom{\Gamma(G)}{k}$. We also let

$$\Gamma_-^{(k)}(G) = \{\mathcal{H} \in \Gamma^{(k)}(G) : G \ominus \mathcal{H} \text{ is disconnected}\}.$$

The following lemma gives a closed form for of k -edge-cuts of any 2-connected simple graph having more edges than vertices.

Lemma 8 ([8]). *For each 2-connected graph G in $\mathcal{C}_{n,m}$ such that $m > n$ and each $k \in \{0, 1, \dots, m\}$,*

$$\mu_k(G) = \binom{m}{k} - \sum_{\mathcal{H} \in \Gamma^{(k)}(G)} \prod_{\gamma \in \mathcal{H}} \ell(\gamma) + \sum_{\mathcal{H} \in \Gamma_-^{(k)}(G)} \prod_{\gamma \in \mathcal{H}} \ell(\gamma). \quad (1)$$

If the graph G in Lemma 8 has precisely t chains and the lengths of all its chains are $\ell_1, \ell_2, \dots, \ell_t$, then the second term of the right-hand side of (1) is

$$\sum_{\mathcal{H} \in \Gamma^{(k)}(G)} \prod_{\gamma \in \mathcal{H}} \ell(\gamma) = \sum_{J \in \binom{[t]}{k}} \prod_{i \in J} \ell_i. \quad (2)$$

If $\ell_1 + \ell_2 + \cdots + \ell_t$ equals to a fixed value m , the right-hand side of (2) is maximized when the tuple $(\ell_1, \ell_2, \dots, \ell_t)$ is fair as defined below. This maximality result is stated in Lemma 10.

Definition 9 ([8]). A tuple $(x_1, x_2, \dots, x_t) \in \mathbb{Z}_+^t$ is fair if $|x_i - x_j| \leq 1$ for all $i, j \in \{1, 2, \dots, t\}$. A graph G is fair if it is a 2-connected simple graph having more edges than vertices such that the tuple whose entries are the lengths of all the chains in G is fair.

Lemma 10 ([8]). Let k and t be integers such that $2 \leq k \leq t$ and let

$$\phi_t^{(k)}(\ell_1, \ell_2, \dots, \ell_t) = \sum_{J \in \binom{[t]}{k}} \prod_{i \in J} \ell_i \quad \text{for each } (\ell_1, \ell_2, \dots, \ell_t) \in \mathbb{Z}_+^t.$$

Let m be such that $m \geq t$ and let $L_{t,m} = \{(\ell_1, \ell_2, \dots, \ell_t) \in \mathbb{Z}_+^t : \ell_1 + \ell_2 + \cdots + \ell_t = m\}$. The following assertions hold.

- (i) The maximum of $\phi_t^{(k)}(\ell_1, \ell_2, \dots, \ell_t)$ as $(\ell_1, \ell_2, \dots, \ell_t)$ ranges over $L_{t,m}$ is attained precisely at those tuples that are fair.
- (ii) Let $\Phi_t^{(k)}(m)$ be the maximum attained by $\phi_t^{(k)}(\ell_1, \ell_2, \dots, \ell_t)$ in $L_{t,m}$. If $2 \leq k < t$ then $\Phi_t^{(k)}(m) < \Phi_{t+1}^{(k)}(m)$.

Definition 11 ([8]). Let G be a 2-connected simple graph on more edges than vertices. Let $\{f_1, f_2, \dots, f_k\}$ be a k -edge-cut of $D(G)$. For each $i \in \{1, 2, \dots, k\}$, let γ_i be the chain of G corresponding to the edge f_i of $D(G)$. We say a k -edge-cut $\{e_1, e_2, \dots, e_k\}$ of G is induced by $\{f_1, f_2, \dots, f_k\}$ if $e_i \in \gamma_i$ for each $i \in \{1, 2, \dots, k\}$. Moreover,

- (i) if $\{f_1, f_2, \dots, f_k\}$ is vertex-separating, then the k -edge-cut $\{e_1, e_2, \dots, e_k\}$ is called Type-V;
- (ii) if $\{f_1, f_2, \dots, f_k\}$ is edge-separating but not vertex-separating, then the k -edge-cut $\{e_1, e_2, \dots, e_k\}$ is called Type-E;
- (iii) if $\{f_1, f_2, \dots, f_k\}$ is nontrivial, then the k -edge-cut $\{e_1, e_2, \dots, e_k\}$ is called Type-N.

The number of Type-V, Type-E, and Type-N k -edges-cuts of G is denoted $\mu_k^V(G)$, $\mu_k^E(G)$, and $\mu_k^N(G)$, respectively. The total number of induced k -edge-cuts of G is denoted $\mu_k^I(G)$; i.e.,

$$\mu_k^I(G) = \mu_k^V(G) + \mu_k^E(G) + \mu_k^N(G).$$

Notice that, by the definition of $\mu_k^I(G)$, it coincides with the third term of the right-hand side of (1), i.e.,

$$\mu_k^I(G) = \sum_{\mathcal{H} \in \Gamma_-^{(k)}(G)} \prod_{\gamma \in \mathcal{H}} \ell(\gamma). \tag{3}$$

This fact combined with Lemma 10 leads to the following result.

Proposition 12 ([8]). *Let k and t be positive integers such that $2 \leq k \leq t$ and let \mathcal{S} be a nonempty set of fair graphs on n vertices and m edges such that $m > n$ and having precisely t chains. Then, as G ranges over \mathcal{S} , the minimum of $\mu_k(G)$ is attained precisely in the same graphs G where the minimum of $\mu_k^I(G)$ is attained.*

4 Minimization of 2-Edge-Cuts and 3-Edge-Cuts

Let p and i be nonnegative integers such that $p \geq 2$. The purpose of this section is two-fold. First, we characterize the set $\mathcal{C}_{2p+i,3p+i}^2$ consisting of all min- μ_2 graphs in $\mathcal{C}_{2p+i,3p+i}$. Then, we find the set $\mathcal{C}_{2p+i,3p+i}^3$ consisting of all graphs with minimum number of 3-edge-cuts among all graphs in $\mathcal{C}_{2p+i,3p+i}^2$.

The following lemma can be proved using handshaking.

Lemma 13. *Let p and i be nonnegative integers such that $p \geq 2$. If G is a 2-connected graph in $\mathcal{C}_{2p+i,3p+i}$ then all the following statements hold,*

- (i) *Its distillation $D(G)$ has at most $3p$ edges and $\delta(D(G)) \geq 3$.*
- (ii) *If $D(G)$ has precisely $3p$ edges then $D(G)$ is cubic.*
- (iii) *The graph G has at most $3p$ chains. The equality holds if and only if $D(G)$ is cubic.*

Lemma 14 follows just finding a simultaneous minimization of each of the 3 terms that appear on the right hand side of Eq. (1). The first term does not depend on the choice of the 2-connected graph G . By Lemma 10, the second term is minimum precisely when G is a fair graph and has the greatest number of chains (i.e., $D(G)$ is a cubic graph in $\mathcal{C}_{2p,3p}$), while the third term is minimum precisely when the set $\Gamma_-^{(2)}(G)$ is empty, or equivalently, when $\lambda(D(G)) = 3$.

Lemma 14. *Let p and i be nonnegative integers such that $p \geq 2$. Among all 2-connected graphs G in $\mathcal{C}_{2p+i,3p+i}$, the number $\mu_2(G)$ is minimum if and only if G is a fair graph whose distillation $D(G)$ is a cubic graph in $\mathcal{C}_{2p,3p}$ such that $\lambda(D(G)) = 3$.*

The following concept will be used to characterize the set $\mathcal{C}_{2p+i,3p+i}^3$ for each pair of nonnegative integers p and i such that $p \geq 2$.

Definition 15. *A fair graph is vertex-fair if the sum-length of the chains that are incident to any fixed vertex is a fair tuple.*

Remark 16. Möbius graph M_p has only vertex-trivial 3-edge cuts thus $\mu_3(M_p) = 2p$ and M_p is a cubic min- μ_3 graph in $\mathcal{C}_{2p,3p}$. For each nonnegative integer i there exists some vertex-fair graph G_{2p+i} in $\mathcal{C}_{2p+i,3p+i}$ whose distillation is M_p .

The following technical lemma will give us all Type-V 3-edge-cuts for each graph G in $\mathcal{C}_{2p+i,3p+i}^2$.

Lemma 17. Let p and i be nonnegative integers such that $p \geq 2$. Let r and s be the only integers such that $r \in \{0, 1, \dots, 3p - 1\}$ and $3p + i = 3ps + r$. For each $j \in \{0, \dots, 3\}$, define q_j such that $q_j(s) = (s + 1)^j s^{3-j}$. Denote $S_{p,r} = \{(x_1, \dots, x_{2p}) \in \{0, \dots, 3\}^{2p} : x_1 + \dots + x_{2p} = 2r\}$ and let

$$g_s(x_1, \dots, x_{2p}) = \sum_{j=1}^{2p} q_{x_j}(s) \quad \text{for each } (x_1, \dots, x_{2p}) \in S_{p,r}.$$

Then, the inequalities $q_3(s) > q_2(s) > q_1(s) > q_0(s)$ hold and the function $g_s(x_1, \dots, x_{2p})$ attains its minimum in $S_{p,r}$ precistuples (x_1, \dots, x_{2p}) in $S_{p,r}$ which are fair.

Let G be any graph in $\mathcal{C}_{2p+i,3p+i}^2$ and let v_1, v_2, \dots, v_{2p} be the $2p$ vertices that are endpoints of some chain in G . If we call x_i to the number of incident chains of v_i whose lengths are $s + 1$, then G has precisely $g_s(x_1, x_2, \dots, x_{2p})$ Type-V 3-edge-cuts. If G has the least number of Type-V 3-edge-cuts among all graphs in $\mathcal{C}_{2p+i,3p+i}^2$ and no Type-E neither Type-N 3-edge-cuts, then G will have the least value of μ_3^1 in $\mathcal{C}_{2p+i,3p+i}^2$. Lemma 18 then follows from Proposition 12.

Lemma 18. Let p and i be nonnegative integers such that $p \geq 2$. The number $\mu_3(G)$ is minimum among all graphs G in $\mathcal{C}_{2p+i,3p+i}^2$ if and only if G is vertex-fair and $D(G)$ is a cubic graph in $\mathcal{C}_{2p,3p}$ having only vertex-trivial 3-edge-cuts.

5 Main Result

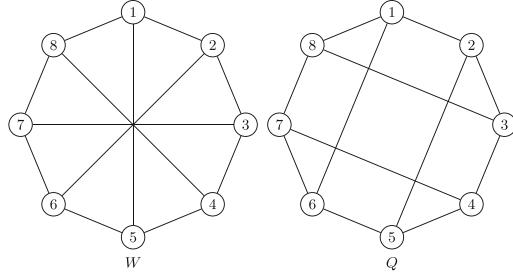
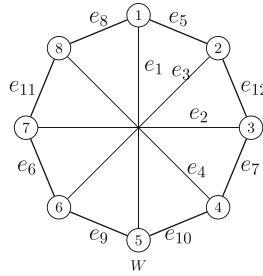
In this section we will prove that there are finitely many uniformly most reliable graphs of corank 5. Let n be any integer such that $n \geq 8$. First, we will find the only graph G_n in $\mathcal{C}_{n,n+4}$ such that $\mathcal{C}_{n,n+4}^4 = \{G_n\}$. Then we will construct, for each integer n such that $n \geq 13$, a graph H_n in $\mathcal{C}_{n,n+4}$. Finally, we will prove that there exists some integer n_0 such that $\mu_5(H_n) < \mu_5(G_n)$ for all $n \geq n_0$. The main theorem will then follow from Lemma 6.

Now, let us find $\mathcal{C}_{n,n+4}^3$. Choosing $p = 4$ and $i = n - 2p$ in Lemma 18 gives that a graph G belongs to $\mathcal{C}_{n,n+4}^3$ if and only if G is vertex-fair and $D(G)$ is a cubic graph in $\mathcal{C}_{8,12}$ having only vertex-trivial 3-edge-cuts. Bussemake [4] shows that there are only 5 nonisomorphic cubic graphs in $\mathcal{C}_{8,12}$. There are precisely 2 such graphs having only vertex-trivial 3-edge-cuts, namely, Wagner graph W and the cube graph Q which are depicted in Fig. 1. We obtained the following result.

Lemma 19. Let n be an integer such that $n \geq 8$. A graph G belongs to $\mathcal{C}_{n,n+4}^3$ if and only if G is vertex-fair and $D(G) \in \{Q, W\}$.

The following concept will be useful to define the only graph G_n such that $\mathcal{C}_{n,n+4}^4 = \{G_n\}$ for each $n \geq 8$.

Definition 20. Let H be any 2-connected simple cubic graph and let Y be a set of edges of H . We denote $H \odot_s Y$ the graph that arises from H by subdividing s times each edge in Y and $s-1$ times each edge not in Y . A vertex-fair subdivision of H is any vertex-fair graph isomorphic to $H \odot_s Y$ for some $Y \subseteq E(H)$.

**Fig. 1.** Wagner graph W and the cube Q .**Fig. 2.** Edges e_1, e_2, \dots, e_{12} in the Wagner graph W .

Definition 21. Define, for each integer n such that $n \geq 8$, the vertex-fair subdivision G_n of W as follows. Label the edges of W as in Fig. 2. Let r and s be the unique integers such that $n+4 = 12s+r$ and $r \in \{0, \dots, 11\}$. If $r = 0$, let $X_0 = \emptyset$; if $r = 8$, let $X_8 = \{e_1, e_2, e_3, e_4, e_6, e_8, e_{10}, e_{12}\}$; otherwise, $X_r = \{e_1, e_2, \dots, e_r\}$. Then, G_n is defined as $W \odot_s X_r$.

By Lemma 19, each graph in $\mathcal{C}_{n,n+4}^3$ is either a vertex-fair subdivision of Q or a vertex-fair subdivision of W . In order to determine $\mathcal{C}_{n,n+4}^4$ we will find closed forms for $\mu_4^V(G)$, $\mu_4^E(G)$, and $\mu_4^N(G)$. The following definition will be considered, where the labels of the vertices and edges of Q and W are given in Fig. 1.

Definition 22. Let G be any graph in $\mathcal{C}_{n,n+4}^3$ such that $G = D(G) \odot_s X$.

- For each $i \in \{0, 1, 2, 3\}$, we will denote $p_i(G)$ the number of vertices in $D(G)$ that are incident to precisely i edges of X .
- For each $j \in \{0, 1, 2, 3, 4\}$, we will denote $q_j(G)$ the number of edges in $D(G)$ that are incident to precisely j edges of X .
- If $k \in \{1, 2, 3\}$ and $D(G) = Q$, then $z_k(G) = |F_k \cap X|$, where $F_1 = \{23, 45, 67, 81\}$, $F_2 = \{12, 38, 47, 56\}$, and $F_3 = \{16, 25, 34, 78\}$.
- If $k \in \{1, 2, 3\}$ and $D(G) = W$, then $z_k(G) = |E_j \cap X|$, where $E_1 = \{12, 34, 56, 78\}$, $E_2 = \{23, 45, 67, 81\}$, and $E_3 = \{15, 26, 37, 48\}$.

Using the definition of Type-V, Type-E and Type-N edge-cuts and the fact that each graph G in $\mathcal{C}_{n,n+4}^3$ is either a vertex-fair subdivision of Q or a vertex-fair subdivision of W , the following lemma can be derived.

Lemma 23. *Let n be an integer such that $n \geq 8$ and let r and s be the unique integers such that $r \in \{0, 1, \dots, 11\}$ and $n + 4 = 12s + r$. If $G \in \mathcal{C}_{n,n+4}^3$, then*

$$\mu_4^V(G) = \sum_{i=0}^3 p_i(G)(s+1)^i s^{3-i}(n+4-3s-i), \quad (4)$$

$$\mu_4^E(G) = \sum_{i=0}^4 q_i(G)(s+1)^i s^{4-i}, \quad (5)$$

$$\mu_4^N(G) = \sum_{i=1}^3 (s+1)^{z_i(G)} s^{4-z_i(G)}, \text{ when } D(G) = Q, \quad (6)$$

$$\mu_4^N(G) = \sum_{i=1}^2 (s+1)^{z_i(G)} s^{4-z_i(G)}, \text{ when } D(G) = W. \quad (7)$$

Consider, for each integer n such that $n \geq 8$, the only integers r and s given in the statement of Lemma 23. As each graph in $\mathcal{C}_{n,n+4}^3$ is either a vertex-fair subdivision of Q or a vertex-fair subdivision of W , we can obtain for each r and s the list of all nonisomorphic graphs in $\mathcal{C}_{n,n+4}^3$. Then, we can find the value $\mu_4^I(G)$ given by the sum $\mu_4^V(G) + \mu_4^E(G) + \mu_4^N(G)$ to obtain the graph G in $\mathcal{C}_{n,n+4}^3$ which minimizes $\mu_4^I(G)$. As all graphs in $\mathcal{C}_{n,n+4}^3$ are fair, we can finally apply Proposition 12 to obtain the graph G in $\mathcal{C}_{n,n+4}^3$ which minimizes $\mu_4(G)$. Using the previous methodology, the following result can be proved.

Proposition 24. *For each positive integer n such that $n \geq 8$, $\mathcal{C}_{n,n+4}^4 = \{G_n\}$.*

Now we will construct, for each integer n such that $n \geq 13$, a graph H_n in $\mathcal{C}_{n,n+4}$. Finally, we will prove that there exists some positive integer n_0 such that $\mu_5(H_n) < \mu_5(G_n)$ for all $n \geq n_0$.

Definition 25. *Let $\ell_1, \ell_2, \dots, \ell_{12}$ be positive integers. Denote $W(\ell_1, \ell_2, \dots, \ell_{12})$ the graph that arises from W by subdividing $\ell_i - 1$ times the edge e_i of W for each $i \in \{1, 2, \dots, 12\}$. For each $n \geq 13$, let r and s be the unique integers such that $r \in \{0, 1, \dots, 11\}$ and $n + 4 = 12s + r$. Define $\ell_i = s + 1$ if $e_i \in X_r$, or $\ell_i = s$ otherwise. Observe that G_n is precisely $W(\ell_1, \ell_2, \dots, \ell_{12})$. Define H_n as $W(\ell'_1, \ell'_2, \dots, \ell'_{12})$ where $\ell'_1 = \ell_1 + 1$, $\ell'_5 = \ell_5 - 1$, and $\ell'_i = \ell_i$ when $i \in \{1, 2, \dots, 12\} - \{1, 5\}$.*

Clearly, H_n has as many vertices and edges as G_n . The definition of H_n makes sense since we must have that $\ell'_i \geq 1$ for each $i \in \{1, 2, \dots, 12\}$, and $\ell'_5 \geq 1$ only when $n \geq 13$. Notice that H_n is not a fair graph. Our goal is to prove that there exists some positive integer n_0 such that $\mu_5(H_n) < \mu_5(G_n)$ for

all $n \geq n_0$. As both G_n and H_n are 2-connected graphs on more edges than vertices, by Lemma 8 and Eqs. (2) and (3),

$$\mu_5(G_n) = \binom{n+4}{5} - \phi_{12}^{(5)}(\ell_1, \ell_2, \dots, \ell_{12}) + \mu_5^I(G_n), \quad (8)$$

$$\mu_5(H_n) = \binom{n+4}{5} - \phi_{12}^{(5)}(\ell'_1, \ell'_2, \dots, \ell'_{12}) + \mu_5^I(H_n). \quad (9)$$

In achieving our goal, we need to find the number of 5-edge-cuts in W .

Lemma 26 ([8]). *Each nontrivial 5-edge-cut of W is either P_3 -separating or C_4 -separating.*

Let G be either G_n or H_n . Denote $\mu_5^{P_3}(G)$ (resp. $\mu_5^{C_4}(G)$) the number 5-edge-cuts of G induced by P_3 -separating (resp. C_4 -separating) edge-cuts of $D(G)$. By Lemma 26,

$$\mu_5^I(G) = \mu_5^V(G) + \mu_5^E(G) + \mu_5^{P_3}(G) + \mu_5^{C_4}(G). \quad (10)$$

Let us find $\mu_5(G_n) - \mu_5(H_n)$ by replacing (10) into (8) and (9),

$$\begin{aligned} \mu_5(G_n) - \mu_5(H_n) &= \phi_{12}^{(5)}(\ell'_1, \ell'_2, \dots, \ell'_{12}) - \phi_{12}^{(5)}(\ell_1, \ell_2, \dots, \ell_{12}) \\ &\quad + (\mu_5^V(G_n) - \mu_5^V(H_n)) + (\mu_5^E(G_n) - \mu_5^E(H_n)) \\ &\quad + (\mu_5^{P_3}(G_n) - \mu_5^{P_3}(H_n)) + (\mu_5^{C_4}(G_n) - \mu_5^{C_4}(H_n)) \end{aligned} \quad (11)$$

The following 5 lemmas give closed forms for each of the 5 terms that appear on the right-hand side of Eq. (11). The proof of Lemma 27 is straightforward. The rationale behind the proofs of Lemmas 28, 29, 30 and 31 is similar. First, we identify which 5-edge-cuts in W we must consider. Then, we count the respective number of induced 5-edge-cuts for both G_n and H_n . Finally, we take the difference $\mu_5^X(G_n) - \mu_5^X(G)$ where X equals V , E , P_3 , or C_4 , respectively.

Lemma 27. *For each integer n such that $n \geq 13$,*

$$\phi_{12}^{(5)}(\ell'_1, \ell'_2, \dots, \ell'_{12}) - \phi_{12}^{(5)}(\ell_1, \ell_2, \dots, \ell_{12}) = (\ell_5 - \ell_1 - 1) \sum_{J \in \binom{[12] - \{1, 5\}}{3}} \prod_{i \in J} \ell_i. \quad (12)$$

Lemma 28. *For each integer n such that $n \geq 13$,*

$$\begin{aligned} \mu_5^V(G_n) - \mu_5^V(H_n) &= \ell_3 \ell_6 \ell_9 \ell_{10} + \ell_4 \ell_7 \ell_9 \ell_{10} - \ell_2 \ell_3 \ell_7 \ell_{12} - \ell_3 \ell_6 \ell_9 \ell_{12} \\ &\quad + (\ell_1 + 1 - \ell_5)(\ell_2 \ell_7 \ell_{12} + \ell_4 \ell_7 \ell_{10} + \ell_3 \ell_6 \ell_9 + \ell_2 \ell_6 \ell_{11} + \ell_4 \ell_8 \ell_{11}) \\ &\quad - (\ell_1 + 1 - \ell_5)(\ell_8 \ell_9 \ell_{10} + \ell_3 \ell_8 \ell_{12} + \ell_4 \ell_8 \ell_{11}) \\ &\quad + \ell_3 \ell_{12} \left(\sum_{\{i,j\} \in \binom{[12] - \{1, 3, 5, 12\}}{2}} \ell_i \ell_j + (\ell_1 + 1 - \ell_5) \sum_{i \in [12] - \{1, 3, 5, 12\}} \ell_i \right) \\ &\quad + \ell_9 \ell_{10} ((\ell_1 + 1 - \ell_5) \sum_{i \in [12] - \{1, 5, 9, 10\}} \ell_i - \sum_{\{i,j\} \in \binom{[12] - \{1, 5, 9, 10\}}{2}} \ell_i \ell_j) \\ &\quad + \ell_8 (\ell_1 + 1 - \ell_5) \sum_{\{i,j\} \in \binom{[12] - \{1, 5, 8\}}{2}} \ell_i \ell_j. \end{aligned} \quad (13)$$

Lemma 29. For each integer n such that $n \geq 13$,

$$\begin{aligned} \mu_5^E(G_n) - \mu_5^E(H_n) &= \ell_8\ell_9\ell_{10}(n+4-\ell_1-\ell_5-\ell_8-\ell_9-\ell_{10}) \\ &\quad + \ell_6\ell_9\ell_{12}(n+5-\ell_3-2\ell_5-\ell_6-\ell_9-\ell_{12}) \\ &\quad + \ell_2\ell_3\ell_7(n+5-\ell_2-\ell_3-2\ell_5-\ell_7-\ell_{12}) \\ &\quad - \ell_3\ell_8\ell_{12}(n+4-\ell_1-\ell_3-\ell_5-\ell_8-\ell_{12}) \\ &\quad - \ell_3\ell_6\ell_{10}(n+3-2\ell_1-\ell_3-\ell_6-\ell_9-\ell_{10}) \\ &\quad - \ell_4\ell_7\ell_9(n+3-2\ell_1-\ell_4-\ell_7-\ell_9-\ell_{10}) \\ &\quad + \ell_4\ell_{11}(\ell_1+1-\ell_5)(n+4-\ell_1-\ell_4-\ell_5-\ell_8-\ell_{11}). \end{aligned} \quad (14)$$

Lemma 30. For each integer n such that $n \geq 13$,

$$\begin{aligned} \mu_5^{P_3}(G_n) - \mu_5^{P_3}(H_n) &= (\ell_1+1-\ell_5)(\ell_2\ell_4\ell_6 + \ell_6\ell_{10}\ell_{12} + \ell_7\ell_{10}\ell_{11}) \\ &\quad + \ell_2\ell_3\ell_4\ell_{10} + \ell_3\ell_6\ell_7\ell_{11} + \ell_2\ell_6\ell_7\ell_9 + \ell_4\ell_7\ell_8\ell_9 \\ &\quad + \ell_3\ell_6\ell_8\ell_{10} + \ell_2\ell_9\ell_{11}\ell_{12} + \ell_4\ell_9\ell_{10}\ell_{11} \\ &\quad - \ell_2\ell_3\ell_7\ell_8 - \ell_2\ell_4\ell_9\ell_{12} - \ell_3\ell_4\ell_6\ell_7 - \ell_2\ell_3\ell_{10}\ell_{11} \\ &\quad - \ell_3\ell_4\ell_{11}\ell_{12} - \ell_6\ell_8\ell_9\ell_{12} - \ell_7\ell_8\ell_9\ell_{11}. \end{aligned} \quad (15)$$

Lemma 31. For each integer n such that $n \geq 13$,

$$\mu_5^{C_4}(G_n) - \mu_5^{C_4}(H_n) = \ell_7\ell_9\ell_{11}(n+5-2\ell_5-\ell_7-\ell_9-\ell_{11}). \quad (16)$$

Proposition 32. There exists some positive integer n_0 such that, for all $n \geq n_0$, $\mu_5(H_n) < \mu_5(G_n)$.

Proof. For each $n \geq 13$, let r and s be unique integers such that $r \in \{0, 1, \dots, 11\}$ and $n+4 = 12s+r$. Consider, for each $j \in \{1, 2, 3, 4, 5\}$, the sequence $a_n^{(j)}$ defined as follows:

$$\begin{aligned} a_n^{(1)} &= \phi_{12}^{(5)}(\ell'_1, \ell'_2, \dots, \ell'_{12}) - \phi_{12}^{(5)}(\ell_1, \ell_2, \dots, \ell_{12}), \\ a_n^{(2)} &= \mu_5^V(G_n) - \mu_5^V(H_n), \\ a_n^{(3)} &= \mu_5^E(G_n) - \mu_5^E(H_n), \\ a_n^{(4)} &= \mu_5^{P_3}(G_n) - \mu_5^{P_3}(H_n), \\ a_n^{(5)} &= \mu_5^{C_4}(G_n) - \mu_5^{C_4}(H_n). \end{aligned}$$

Observe that $\ell_1+1-\ell_5 \in \{1, 2\}$, and $\ell_i \in \{s, s+1\}$ for all $i \in \{1, 2, \dots, 12\}$. By Lemma 27, $a_n^{(1)} \in O(n^3)$. Similarly, by Lemmas 28, 29 and 30, $a_n^{(j)} \in O(n^3)$ for each $j \in \{2, 3, 4\}$. Additionally, by Lemma 31 we can see that $a_n^{(5)} = 7(\left\lfloor \frac{n}{12} \right\rfloor)^4 + q_n$ for some sequence $q_n \in O(n^3)$. Consequently, the difference $\mu_5(G_n) - \mu_5(H_n)$, which equals $\sum_{j=1}^5 a_n^{(j)}$, tends to infinity with respect to n thus there exists some n_0 such that $\mu_5(G_n) - \mu_5(H_n) > 0$ for all $n \geq n_0$, as required. \square

Replacing each of the expressions (12)–(16) into Eq. (11) gives that $\mu_5(G_n) - \mu_5(H_n) > 0$ for each integer n such that $n \geq 167$.

We are in position to prove the main result of this work.

Theorem 33. *There are finitely many UMRG of corank 5.*

Proof. Let n be any positive integer such that $n \geq n_0$, where n_0 is the least positive integer satisfying Proposition 32. It is enough to show that there is no UMRG in $\mathcal{C}_{n,n+4}$. On the one hand, consider the graph G_n given by Definition 21. By Proposition 24, we know that $\mathcal{C}_{n,n+4}^4 = \{G_n\}$. On the other hand, consider the graph H_n given by Definition 25. As $n \geq n_0$, by Proposition 32 we conclude that $\mu_5(H_n) < \mu_5(G_n)$. The statement follows by Lemma 6. \square

Acknowledgments. This work is partially supported by City University of New York project entitled *On the problem of characterizing graphs with maximum number of spanning trees*, grant number 66165-00. The author wants to thank Dr. Martín Safe for his helpful comments that improved the presentation of this manuscript.

References

1. Ath, Y., Sobel, M.: Some conjectured uniformly optimal reliable networks. *Prob. Eng. Inf. Sci.* **14**, 375–383 (2000)
2. Boesch, F.T.: On unreliability polynomials and graph connectivity in reliable network synthesis. *J. Graph Theory* **10**(3), 339–352 (1986)
3. Brown, J.I., Cox, D.: Nonexistence of optimal graphs for all terminal reliability. *Networks* **63**(2), 146–153 (2014)
4. Bussemaker, F., Cobeljic, S., Cvetkovic, D., Seidel, J.: Cubic graphs on ≤ 14 vertices. *J. Comb. Theory Ser. B* **23**(2–3), 234–235 (1977)
5. Kelmans, A.: On graphs with randomly deleted edges. *Acta Math. Hung.* **37**(1–3), 77–88 (1981)
6. Myrvold, W., Cheung, K.H., Page, L.B., Perry, J.E.: Uniformly-most reliable networks do not always exist. *Networks* **21**(4), 417–419 (1991)
7. Romero, P.: Uniformly optimally reliable graphs: a survey. *Networks* **80**(4), 466–481 (2022)
8. Romero, P., Safe, M.D.: Least corank for the nonexistence of uniformly most reliable graphs. *Procedia Comput. Sci.* **223**, 88–95 (2023)
9. Wang, G.: A proof of Boesch's conjecture. *Networks* **24**(5), 277–284 (1994)



Helly Number, Radon Number and Rank in Δ -Convexity on Graphs

Bijo S. Anand¹ , Arun Anil² , Manoj Changat² , Revathy S. Nair³ ,
and Prasanth G. Narasimha-Shenoi^{4,5}

¹ Department of Mathematics, Sree Narayana College,
Punalur, Kollam 691305, Kerala, India

² Department of Futures Studies, University of Kerala,
Thiruvananthapuram 695581, Kerala, India

mchangat@gmail.com

³ Department of Mathematics, Mar Ivanios College, University of Kerala,
Thiruvananthapuram 695015, Kerala, India

⁴ Department of Mathematics, Government College Chittur,
Palakkad 678104, Kerala, India

⁵ Department of Collegiate Education, Government of Kerala,
Thiruvananthapuram 695033, Kerala, India

Abstract. This article discusses Δ -convexity on simple connected graphs. We establish general bounds for the Helly number, Radon number, and rank with respect to Δ -convexity on graphs. Additionally, we give the exact values for the Helly number and Radon number for chordal graphs, as well as the rank for block graphs.

Keywords: Convexity · Helly number · Radon number · Rank

AMS Sub. Classification: 52A35 · 05C38 · 05C69

1 Introduction

Convexity is one of the most fascinating concepts in mathematics, with connections to several fields such as optimization, combinatorics, clustering systems, and more. For further details, see [15]. van de Vel [18] developed a comprehensive axiomatic treatment of convexity theory, which has served as a foundational framework for many researchers studying convexity and its parameters.

For elaborate studies on convexity properties, refer [18]. Several distinct notions of convexity have been introduced and investigated in the context of graphs. Pioneers such as Degreef, Doignon, Jamison, Reay, Sierksma, and van de Vel investigated different combinatorial parameters, namely, the Carathéodory, Helly, Radon and exchange numbers of convex sets for the axiomatic convexity; see [7, 12, 17, 18]. In particular, Sierksma introduced the exchange number in 1975, [16].

Various authors have explored different types of convexities on graphs. For example, the geodesic convexity, see [8, 11, 14]; for monophonic convexity, see [5, 10, 13]; for P_3 -convexity, see [4]; and for triangle-path convexity, see [6, 9].

A more recently studied convexity is Δ -convexity, introduced in [1], where the authors proved that computing the Δ -convex hull of a general graph is NP-complete and explained various polynomial-time algorithms for some graph families like chordal, dually chordal and cographs.

The Helly number, Radon number, and Rank are fundamental concepts in convexity spaces that offer insights into their interconnectedness, separability, and dimensionality. The Helly number measures the degree of overlap among convex sets, the Radon number reflects the separability of points, and the rank indicates the maximum size of convexly independent sets (dimension of convex sets). These concepts are interrelated and have significant implications for the study of convexity spaces. Understanding them is crucial in combinatorial geometry, topology, and optimization. In this article, we determine the combinatorial parameters, Helly, Radon and the rank of the Δ -convexity in graphs.

Before discussing the Δ -convexity, we formally define a convexity space. A finite *convexity space* is a pair (X, \mathcal{C}) where X is a finite set and \mathcal{C} is a collection of subsets of X such that; $\emptyset, X \in \mathcal{C}$ and \mathcal{C} is closed under intersections. The elements in \mathcal{C} are said to be the *convex sets*. For a subset S of X , the smallest convex set containing S is called the convex hull of S and is denoted as $\langle S \rangle$.

Given a convexity \mathcal{C} on X and a subset S of X , the set S is *Helly dependent* (or, *H-dependent*) provided $\bigcap_{a \in S} \langle S \setminus \{a\} \rangle \neq \emptyset$, and it is *Helly independent* (or,

H-independent) otherwise. The set S is *Radon dependent* (or, *R-dependent*) if there is a partition $\{S_1, S_2\}$ of S such that $\langle S_1 \rangle \cap \langle S_2 \rangle \neq \emptyset$. In these circumstances, $\{S_1, S_2\}$ is called a *Radon partition* of S . If no such partition exists, then S is *Radon (R-) independent*. A set S is said to be *Carathéodory dependent* $\langle S \rangle \subseteq \bigcup_{a \in S} \langle S \setminus \{a\} \rangle$ and it is *Carathéodory independent* or *C-independent* otherwise.

A subset S of X is said to be a *hull* set of X if $\langle S \rangle = X$. A subset S of a convex structure X is *convexly independent* provided $a \notin \langle S \setminus \{a\} \rangle$ for each $a \in S$. It is *convexly dependent* otherwise.

A convex structure X gives rise to the following numbers $h(X), r(X), c(X)$, $d(X) \in \{0, 1, \dots, \infty\}$ determined by the following prescription. For each n with $0 \leq n < \infty$, $h(X) \leq n$ if and only if each finite set $S \subseteq X$ with $|S| > n$ is H-dependent; $r(X) \leq n$ if and only if each finite set $S \subseteq X$ with $|S| > n$ is R-dependent; $c(X) \leq n$ if and only if each finite set $S \subseteq X$ with $|S| > n$ is C-dependent; $d(X) \leq n$ if and only if each finite set $S \subseteq X$ with $|S| > n$ is convexly dependent; where $h(X)$ is the *Helly number* of X , $r(X)$ is the *Radon number* of X , and $d(X)$ is the *rank* of X .

In Δ -convexity, the parameters Helly, Radon, Carathéodory and the rank are respectively denoted as $h_\Delta(X)$, $r_\Delta(X)$, $c_\Delta(X)$ and $d_\Delta(X)$ or simply h, r, c, d if there is no confusion.

For $S \subseteq V(G)$, the Δ -interval $[S]$ of S is the set of all vertices in S together with those vertices in V that are adjacent to any two adjacent vertices in S . A set S is said to be Δ -convex if $[S] = S$ and the Δ -convex hull of S denoted by $\langle S \rangle$, is the smallest Δ -convex set containing S . In [3], the authors investigate the Δ -number and Δ -convexity number, providing, in particular, the exact value for the convexity number of block graphs with diameter ≤ 3 , as well as for graph products, specifically the strong and lexicographic products of two graphs, [2].

Throughout this article, the graphs under consideration are connected, finite, simple and undirected, denoted by $G = (V, E)$. A graph G on n vertices is said to be *complete* if every vertex is adjacent to every other vertex and denoted by K_n . A complete graph K_n is called a *triangle* when $n = 3$. G is said to be triangle-free if G does not contain K_3 . A subset of vertices of a graph is *independent* in a simple graph if the vertices are pairwise non-adjacent. The *independence number* $\alpha = \alpha(G)$ of a graph G is the cardinality of a maximum independent set of vertices. A *chordal graph* is one in which all cycles of four or more vertices have a chord, which is an edge that is not a part of the cycle but connects two non adjacent vertices of the cycle. A *block graph* is a graph in which every block is complete. For more graph-theoretic notions that are not defined formally, see [19].

In this paper, for the basic definitions and results on axiomatic convexity, explained in this introductory section, we refer to [18]. The Helly number, the Radon number, and the rank of the Δ -convexity are discussed in Sects. 2, 3 and 4, respectively.

Theorem 1 [18]. *The following holds for all convex structures.*

1. (*Levi inequality*) $h \leq r$.
2. (*Eckhoff-Jamison inequality*) $r \leq c(h - 1) + 1$ if $h \neq 1$ or $c < \infty$

Remark 1 [18]. Any convex structure satisfies $d \geq \max\{h, c, r\}$

Lemma 1 [1]. *If G is a 2-connected chordal graph, then every pair of adjacent vertices form a hull set of G .*

2 Helly Number

In this section, we investigate the Helly number within the context of Δ -convexity. Due to the complex nature of triangle formations in graphs, determining the exact Helly number for Δ -convexity presents a significant challenge. However, we establish an upper bound for the Helly number based on the number of triangles in the graph and prove that this bound is the least upper bound. Furthermore, we explicitly derive the Helly number for chordal graphs.

We begin with the following observations that naturally arise from the definitions of Δ -convexity and Helly-independent sets:

Observation 1. *If G is a triangle-free graph, then the Helly number, $h_\Delta(G) = |V(G)|$.*

Observation 2. If G is a complete graph K_n for $n > 2$, then $h_\Delta(G) = 2$.

The following property follows directly from the definition of a Helly dependent set:

Remark 2. If S is a set and $w \in S$ such that $w \in \langle S \setminus \{w\} \rangle$, then S is Helly dependent.

Next, we will present some simple properties of the Helly independent set of a general graph, which will be utilized in the subsequent results.

Lemma 2. Let G be a graph. Then:

- (a) The Helly number, $h_\Delta(G)$ satisfies $h_\Delta(G) \geq \alpha(G)$.
- (b) If S is a Helly-independent set in G , then no three vertices of S forms a K_3 in G .
- (c) If every pair of adjacent vertices forms a hull set in G , then $h_\Delta(G) = \max\{2, \alpha(G)\}$.

Proof. (a) Let S be a maximum independent vertex set of the graph G , with $|S| = \alpha(G)$. Clearly, $\langle S \rangle = S$ and $\langle S \setminus \{a\} \rangle = S \setminus \{a\}$ for all $a \in S$. This implies that S is Helly-independent. Therefore, $h_\Delta(G) \geq \alpha(G)$.
(b) Let S be a Helly-independent set of G . Suppose $\{a, b, c\} \subseteq S$ forms an induced K_3 in G . Then $a \in \langle S \setminus \{a\} \rangle$, which, according to the Remark 2, implies that S is Helly-dependent, a contradiction. Therefore, no three vertices of S can form a K_3 in G .
(c) We proceed by considering the general case where G may be either a complete or a non-complete graph.

First, consider the case where G is a complete graph. In this case, the Helly number, $h_\Delta(G) = 2$. Since $\alpha(G) = 1$ for a complete graph, we observe that $\max\{2, \alpha(G)\} = 2$, so the lemma holds in this case.

Now, suppose G is not a complete graph. Let S be the maximum independent vertex set of G . Thus $|S| = \alpha(G)$. Since every independent vertex set is H-independent, it follows that S is H-independent. Now, consider any $S' \subseteq V(G)$ with $|S'| > \alpha(G)$. Since G is not a complete graph, $\alpha(G) \geq 2$. Also since $|S'| > \alpha(G)$, there must exist $u, v \in S'$ with $uv \in E(G)$ and some $x \in S' \setminus \{u, v\}$. By assumption, $\langle u, v \rangle = V(G)$, implying that $x \in \bigcap_{a \in S'} \langle S' \setminus \{a\} \rangle$. Therefore, S' is H-dependent, which implies that the Helly number, $h_\Delta(G) = \alpha(G)$. \square

From Lemma 1 and 2, we immediately obtain the following corollary:

Corollary 1. If G is a 2-connected chordal graph, then $h_\Delta(G) = \max\{2, \alpha(G)\}$.

The following theorem provides an upper bound on the Helly number of a graph in terms of the number of triangles and the number of vertices not lying on any triangle:

Theorem 2. Let G be a graph with k triangles, and let m denote the number of vertices that do not lie on any triangles. Then the Helly number, $h_\Delta(G)$ satisfies $h_\Delta(G) \leq m + 2k$.

Proof. Let $S \subseteq V(G)$ be a Helly independent set in G . By Lemma 2, S can include at most two vertices from each triangle in G . With k triangles in G , the maximum number of vertices from triangles that can be in S is $2k$. The m vertices not lying on any triangle are Helly independent by default and all can be included in S . Therefore, the size of S is bounded by $m + 2k$, implying that $h_\Delta(G) \leq m + 2k$. \square

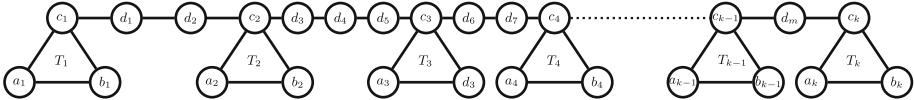


Fig. 1. Graph G with $d = h = r = m + 2k$

Remark 3. There exists a graph G with k triangles such that the Helly number, $h_\Delta(G) = m + 2k$, where m denotes the number of vertices not lying on any triangle. For example, in Fig. 1, $S = \{a_1, a_2, \dots, a_k, b_1, b_2, \dots, b_k, d_1, d_2, \dots, d_m\}$, where $|S| = m + 2k$. Then the following holds: $\langle S \setminus \{a_i\} \rangle = V(G) \setminus \{a_i, c_i\}$, $\langle S \setminus \{b_i\} \rangle = V(G) \setminus \{b_i, c_i\}$ and $\langle S \setminus \{d_j\} \rangle = V(G) \setminus \{d_j\}$ for $i = 1, 2, \dots, k$ and $j = 1, 2, 3, \dots, m$. That is, $\bigcap_{a \in S} \langle S \setminus \{a\} \rangle = \emptyset$ implying that S is Helly independent. For any $S' \subseteq V(G)$ with $|S'| > m + 2k$, S' contains all vertices of at least one triangle. Clearly S' is Helly dependent. Therefore, $h_\Delta = m + 2k$.

Furthermore, the Helly number can be arbitrarily large, in the sense that for any natural number n , one can construct a graph G with $h_\Delta(G) = n$, as stated in the following proposition.

Proposition 1. *For any natural number $n > 1$, there exists a graph G with $h_\Delta(G) = n$.*

We conclude this section by presenting the exact value for the Helly number for chordal graphs. For the following result, we define a block of a graph G as a maximal 2-connected subgraph.

Theorem 3. *Let G be a chordal graph. Then:*

- (a) *If G is a block graph with ℓ blocks, then $h_\Delta(G) = \ell + 1$.*
- (b) *If G is not a block graph, i.e., G contains non-complete blocks. Let B_1, B_2, \dots, B_ℓ be the complete blocks and B'_1, B'_2, \dots, B'_k be the non-complete blocks. Let G' be the induced subgraph of G formed by $\bigcup_{i=1}^k V(B'_j)$. Then $h_\Delta(G) = \alpha(G') + \ell$.*

Proof. (a) Let G be a chordal graph where all blocks are complete. i.e., G is a block graph. Let B_1, B_2, \dots, B_ℓ be the blocks of G , with each B_i being complete for $i = 1, 2, \dots, \ell$. Define the set $S = \{u_1, u_1', u_2, u_3, \dots, u_\ell\}$, where

$u_1, u'_1 \in B_1$, $u_i \in B_i$ for $i = 2, 3, \dots, \ell$, and no three vertices of S forms a K_3 (a complete graph on three vertices) in G . Then we have the following properties: $V(B_1) \setminus \{u'_1\} \not\subseteq \langle S \setminus \{u_1\} \rangle$, $V(B_1) \setminus \{u_1\} \not\subseteq \langle S \setminus \{u'_1\} \rangle$ and $V(B_i) \not\subseteq \langle S \setminus \{u_i\} \rangle$ for $i = 2, 3, \dots, \ell$. This implies that $\bigcap_{a \in S} \langle S \setminus \{a\} \rangle = \emptyset$. Hence, S is

H-independent.

Next, we must prove that there is no H-independent set of size greater than $\ell + 1$. Suppose, for contradiction, that there exists an H-independent set S' with $|S'| \geq \ell + 2$. By Lemma 2, S' contains two vertices from at least two blocks and not more than two vertices from a single block. Consider the following possible cases:

Case 1: S' contains at least one vertex from each block. Since $|S'| \geq \ell + 2$, S' contains two vertices from at least two blocks, say B_i and B_j with $i, j \in \{1, 2, \dots, \ell\}$. Let $u, u' \in V(B_i)$ and $v, v' \in V(B_j)$, where $u, u', v, v' \in S'$. Then, $\langle S' \setminus \{u\} \rangle = \langle S' \setminus \{u'\} \rangle = \langle S' \setminus \{v\} \rangle = \langle S' \setminus \{v'\} \rangle = V(G)$. Thus, $\{u, u', v, v'\} \subseteq \langle S' \setminus \{a\} \rangle$ for all $a \in S'$, implying that S' is H-dependent.

Case 2: S' does not contain vertices from some blocks.

Since $|S'| \geq \ell + 2$ and S' does not contain more than two vertices from any single block, let p be the number of blocks from which exactly two vertices are in S' and q be the number of blocks from which no vertices are in S' . Clearly, $p > q$. Thus, there exist blocks B_i and B_j such that $u, u', v, v' \in S'$ and $u, u' \in V(B_i)$ and $v, v' \in V(B_j)$, with a chain of blocks between B_i and B_j (denoted as $B_{i+1}, B_{i+2}, \dots, B_{j-1}$) such that $u_k \in S' \cap V(B_k)$ for $k = i+1, i+2, \dots, j-1$.

Hence $\cup_{h=i}^j V(B_h) \subset \langle S' \setminus \{w\} \rangle$, or simply we can say that $w \in \langle S' \setminus \{w\} \rangle$ for $w \in \{u, u', v, v'\}$. This implies that S' is H-dependent. In both cases, we conclude that S' cannot be H-independent, contradicting the assumption that $|S'| \geq \ell + 2$. Therefore, the largest H-independent set in G has size $\ell + 1$, proving that $h_\Delta(G) = \ell + 1$.

- (b) Let G be a chordal graph containing blocks $B_1, B_2, B_3, \dots, B_\ell$ and $B'_1, B'_2, B'_3, \dots, B'_k$, where the blocks B_i are complete for $i = 1, 2, \dots, \ell$ and the blocks B'_j are non-complete for $j = 1, 2, \dots, k$. Let G' be the induced subgraph of G formed by $\bigcup_{i=1}^k V(B'_j)$.

Let $S \subseteq V(G)$ be the set $S = \{\text{maximum independent vertex set of } G'\} \cup \{u_i \in V(B_i) \mid i = 1, 2, \dots, \ell\}$. Then $|S| = \alpha(G') + \ell$. We can easily see that $a \notin \langle S \setminus \{a\} \rangle$ for any $a \in S$, and $V(B_i) \setminus S \not\subseteq \langle S \setminus \{b\} \rangle$ for any $b \in V(B_i) \cap S$. Therefore, S is H-independent.

Next, we need to prove that there is no H-independent set of size greater than $\alpha(G') + \ell$. Suppose, for the sake of contradiction, that there exists an H-independent set S' with $|S'| > \alpha(G') + \ell$. Consider the following cases:

Case 1: S' contains more than $\alpha(G')$ vertices from the non-complete blocks. Then, S' must contain at least two vertices each from at least two distinct non-complete blocks, implying that a pair of adjacent vertices exists in S' . Let B'_j be a non-complete block such that $S' \cap V(B'_j)$ contains adjacent vertices.

Subcase 1.1: If $S' \cap V(B'_j)$ contains at least three vertices, say $u, v, w \in V(B'_j)$, with either $uv \in E(G)$ or $vw \in E(G)$, assume $uv \in E(G)$. Then,

$V(B'_j) \subseteq \langle u, v \rangle$, implying $w \in \langle S' \setminus \{w\} \rangle$. Thus, $w \in \bigcap_{a \in S'} \langle S' \setminus \{a\} \rangle$, and consequently, S' is H-dependent.

Subcase 1.2: If S' contains at most two vertices from each of the non complete blocks of G . Assume $S' \cap V(B'_j) = \{u, v\}$ with $uv \in E(G)$, then $V(B'_j) \subseteq \langle u, v \rangle$. Let $B'_j, B'_{j+1}, B'_{j+2}, \dots, B'_r$ be a sequence of chain of non complete blocks in G . If there exists $w', w'' \in V(B'_k) \cap S'$ such that $w_k w' \in E(G)$ and $w_k \in V(B'_{k-1}) \cap V(B'_k)$ for some $k = j+1, j+2, \dots, r$, then $w'' \in \langle S' \setminus \{w'\} \rangle$, implying $w'' \in \bigcap_{a \in S'} \langle S' \setminus \{a\} \rangle$, and S' is H-dependent. Similar things happen when there is an intermediate block B_i with $v \in V(B_i) \cap S'$.

Subcase 1.3: If there is no such B'_k as in **Subcase 1.2**, then the blocks $B'_j, B'_{j+1}, \dots, B'_k$ contributes additional vertices to the independence set, which contradicts the assumption that $|S'| > \alpha(G')$. Thus, there must exist another non-complete block with adjacent vertices in S' , and we can apply the same argument as that of **Subcase 1.2**. If there exists a block B'_i such that $V(B'_i) \cap S' = \emptyset$, for $i = j+1, \dots, k$ then, since $|S'| > \alpha(G')$, there must exist another non-complete block containing adjacent vertices in S' , and we can apply the same procedure as in **Subcase 1.2**. Thus, in this case, S' is H-dependent.

Case 2: S' contains more than ℓ vertices from the complete blocks.

If S' contains three vertices from any complete block, then S' is H-dependent. Similarly, if S' contains at least three vertices of a non-complete block, including a pair of adjacent vertices, then S' is H-dependent.

Subcase 2.1: If S' contains at most two vertices from each complete block. Assume $u, u' \in V(B_i) \cap S'$. Let B_i, B_{i+1}, \dots, B_k be a chain of complete blocks with $|V(B_j) \cap S'| = 1$ for $j = i+1, \dots, k$. If there is $v, v' \in V(B_k) \cap S'$, then $v \in \langle S' \setminus \{v\} \rangle$, which implies S' is H-dependent. Similar things happen when B_k is replaced by B'_k with $v, v' \in V(B'_k) \cap S'$ and $vw \in E(G)$ for $w \in V(B'_k) \cap V(B_{k-1})$ or there is an intermediate block B'_j such that $v' \in V(B'_j) \cap S'$ and $v'w' \in E(G)$ for some $w' \in V(B'_j) \cap V(B_{j-1})$.

Subcase 2.2: If there is no complete block B_k or non-complete block B'_k as in **Subcase 2.1**, then either there exist another complete block B with $|V(B) \cap S'| = 2$ (since G is not a block graph and $|S'| \geq \ell + 3$) or there is a non-complete block containing at least three vertices, including one pair of adjacent vertices. In this case, we can apply Case 2.1 and Case 2 respectively. Thus in both Cases 1 and 2, S' is H-dependent. Therefore, the largest H-independent set in G has size $\alpha(G') + \ell$, proving that $h_\Delta(G) = \alpha(G') + \ell$.

Hence the theorem. \square

3 Radon Number

In this section, we investigate the Radon number within the framework of Δ -convexity. We establish an upper bound for the Radon number that depends upon the number of triangles present in the graph and shows that this bound is the least upper bound. Furthermore, we determine the Radon number specifically for chordal graphs and note that it coincides with the Helly number for chordal graphs.

We start with the following observations, which arise directly from the definitions of Δ -convexity and R-independent sets:

Observation 3. *If G is a triangle-free graph, then the Radon number, $r_\Delta(G) = |V(G)|$.*

Observation 4. *If G is a complete graph K_n for $n > 2$, then $r_\Delta(G) = 2$.*

The following property follows directly from the definition of an R-dependent set:

Remark 4. If S is a set and $w \in S$ be such that $w \in \langle S \setminus \{w\} \rangle$, then S is R-dependent.

Next, we present some simple properties of the R-independent set of a general graph, which will be utilized in subsequent results.

Lemma 3. *Let G be a graph. Then:*

- (a) *The Radon number, $r_\Delta(G)$ satisfies $r_\Delta(G) \geq \alpha(G)$.*
- (b) *If S is an R-independent set in G , then no three vertices of S forms a K_3 in G .*
- (c) *If every pair of adjacent vertices forms a hull set in G , then $r_\Delta(G) = \max\{2, \alpha(G)\}$.*

Proof. (a) This follows directly from Levi's inequality and Lemma 2.

(b) Let S be an R-independent set of G . Suppose $\{a, b, c\} \subseteq S$ forms an induced K_3 in G . Then $a \in \langle S \setminus \{a\} \rangle$ and the Remark 4, implies that S is R-dependent, a contradiction. Therefore, no three vertices of S can form a K_3 in G .

(c) We proceed by considering the general case where G may be either a complete or a non-complete graph. First, consider the case where G is a complete graph. In this case, the Radon number, $r_\Delta(G) = 2$. Since $\alpha(G) = 1$ for a complete graph, we observe that $\max\{2, \alpha(G)\} = 2$, so the lemma holds in this case. Now, suppose G is not a complete graph. Let S be the maximum independent vertex set of G . Thus $|S| = \alpha(G)$. Since every independent vertex set is R-independent, it follows that S is R-independent. Now, consider any $S' \subseteq V(G)$ with $|S'| > \alpha(G)$. Then there must exist $u, v \in S'$ such that $uv \in E(G)$. By assumption, $\langle u, v \rangle = V(G)$.

Now consider the partition of $S' = S'_1 \cup S'_2$ where $S'_1 = \{u, v\}$ and $S'_2 = S' \setminus S'_1$. Since $\langle S'_1 \rangle \cap \langle S'_2 \rangle = \langle S'_2 \rangle$, this implies that S' is R-dependent. Hence, the Radon number, $r_\Delta(G) = \alpha(G)$. \square

From Lemmas 1 and 3, we immediately obtain the following corollary:

Corollary 2. *If G is a 2-connected chordal graph, then $r_\Delta(G) = \max\{2, \alpha(G)\}$.*

The following theorem provides an upper bound on the Radon number of a graph in Δ -convexity:

Theorem 4. Let G be a graph with k triangles, and let m denote the number of vertices not lying on any triangles. Then the Radon number, $r_\Delta(G)$ satisfies $r_\Delta(G) \leq m + 2k$.

Proof. Let $S \subseteq V(G)$ be an R-independent set in G . By Lemma 3, no three vertices in S can form a triangle in G . Therefore, S can contain at most two vertices from any triangle in G . Since G contains k triangles, the maximum number of vertices in S that can be chosen from the triangles is $2k$. Next, consider the m vertices of G that do not lie on any triangles. Since these vertices are not part of any triangle, they are R-independent by default. Therefore, all of these m vertices can be included in S . Therefore, the size of S is bounded by $m + 2k$, implying that $r_\Delta(G) \leq m + 2k$. \square

Remark 5. There exists a graph G with k triangles such that the Radon number, $r_\Delta(G) = m + 2k$, where m denotes the number of vertices not lying on any triangle. For example, in Fig. 1, consider the set $S = \{a_1, a_2, \dots, a_k, b_1, b_2, \dots, b_k, d_1, d_2, \dots, d_m\}$, where $|S| = m + 2k$. Then the following holds: $\langle S \setminus \{a_i\} \rangle = V(G) \setminus \{a_i, c_i\}$, $\langle S \setminus \{b_i\} \rangle = V(G) \setminus \{b_i, c_i\}$, and $\langle S \setminus \{d_j\} \rangle = V(G) \setminus \{d_j\}$, for $i = 1, 2, \dots, k$ and $j = 1, 2, \dots, m$. For any partition $\{S_1, S_2\}$ of S , we have $\langle S_1 \rangle \cap \langle S_2 \rangle = \emptyset$, meaning S is Radon independent. Hence by Theorem 4, $r_\Delta(G) = m + 2k$ follows.

Furthermore, the Radon number can be made arbitrarily large. Specifically, for any natural number n , it is possible to construct a graph G such that $r_\Delta(G) = n$, as stated in the following proposition.

Proposition 2. For any natural number $n > 1$, there exists a graph G with $r_\Delta(G) = n$.

The following theorem provides the exact values for the Radon number of a chordal graph in Δ -convexity.

Theorem 5. Let G be a chordal graph. Then:

- (a) If G is a block graphs with ℓ blocks, then $r_\Delta(G) = \ell + 1$.
- (b) If G is not a block graph, i.e., G contains non-complete blocks. Let B_1, B_2, \dots, B_ℓ be the complete blocks and B'_1, B'_2, \dots, B'_k be the non-complete blocks. Let G' be the induced subgraph of G formed by $\bigcup_{i=1}^k V(B'_j)$. Then $r_\Delta(G) = \alpha(G') + \ell$.

Proof. (a) Let G be a chordal graph where all the blocks are complete. That is, G is a block graph. Let B_1, B_2, \dots, B_ℓ be the blocks in G , where each block B_i is complete for $i = \{1, 2, \dots, \ell\}$. Consider the set $S = \{u_1, u'_1, u_2, \dots, u_\ell\}$ such that $\{u_1, u'_1\} \in V(B_1)$ and $u_i \in V(B_i)$ for $i \in \{2, 3, \dots, \ell\}$, and no three vertices of S forms a K_3 in G .

For any partition $\{S_1, S_2\}$ of S , if $u_1 \in S_1$ and $u'_1 \in S_2$, then $\langle S_1 \rangle = S_1$ and $\langle S_2 \rangle = S_2$, implying $\langle S_1 \rangle \cap \langle S_2 \rangle = \emptyset$. Thus, S is R-independent.

Now, we need to show that there does not exist an R-independent set in G that contains more than $\ell + 1$ elements. Assume, to the contrary, that there

exists an R-independent set $S' \subseteq V(G)$ such that $|S'| \geq \ell + 2$. Clearly, S' cannot contain three vertices that form a K_3 in G .

Consider the following cases:

Case 1: Suppose S' contains at least one vertex from each block.

Since $|S'| \geq \ell + 2$, S' contains two vertices from at least two blocks, say B_i and B_j with $i, j \in \{1, 2, \dots, \ell\}$. Let $u, u' \in V(B_i)$ and $v, v' \in V(B_j)$, where $u, u', v, v' \in S'$. Then, $\langle S' \setminus \{u\} \rangle = \langle S' \setminus \{u'\} \rangle = \langle S' \setminus \{v\} \rangle = \langle S' \setminus \{v'\} \rangle = V(G)$. By Remark 4, S' is R-dependent.

Case 2: S' does not contain vertices from some blocks in G .

Since $|S'| \geq \ell + 2$ and S' does not contain more than two vertices from any single block, let p be the number of blocks from which exactly two vertices are in S' , and q be the number of blocks from which no vertices are in S' . Clearly, $p > q$. Thus, there exist blocks B_i and B_j such that $u, u', v, v' \in S'$ and $u, u' \in V(B_i)$ and $v, v' \in V(B_j)$, and a chain of blocks between B_i and B_j (denoted as $B_{i+1}, B_{i+2}, \dots, B_{j-1}$) such that $u_k \in S' \cap V(B_k)$ for $k = i+1, i+2, \dots, j-1$.

Hence $\cup_{h=i}^j V(B_h) \subset \langle S' \setminus \{w\} \rangle$, or simply, $w \in \langle S' \setminus \{w\} \rangle$ for $w \in \{u, u', v, v'\}$. By remark 4, S' is R-dependent.

In both cases, we conclude that S' cannot be R-independent, contradicting the assumption that $|S'| \geq \ell + 2$. Therefore, the largest R-independent set in G has size $\ell + 1$, proving that $r_\Delta(G) = \ell + 1$.

- (b) Let G be a chordal graph containing blocks B_1, B_2, \dots, B_ℓ and B'_1, B'_2, \dots, B'_k , where the blocks B_i are complete for $i = 1, 2, \dots, \ell$ and the blocks B'_j are non-complete for $j = 1, 2, \dots, k$. Let G' be the induced subgraph of G formed by $\bigcup_{j=i}^k V(B'_j)$.

Let $S \subseteq V(G)$ be the set $S = \{\text{maximum independent vertex set of } G'\} \cup \{u_i \in V(B_i) \mid i = 1, 2, \dots, \ell\}$. Then $|S| = \alpha(G') + \ell$.

Next, we need to prove that there is no R-independent set of size greater than $\alpha(G') + \ell$. Suppose, for the sake of contradiction, that there exists an R-independent set S' with $|S'| > \alpha(G') + \ell$. Consider the following cases:

Case 1: S' contains more than $\alpha(G')$ vertices from the non-complete blocks. Then, S' must contain at least two vertices each from at least two distinct non-complete blocks, implying that a pair of adjacent vertices exists in S' . Let B'_j be a non-complete block such that $S' \cap V(B'_j)$ contains adjacent vertices.

Subcase 1.1: If $S' \cap V(B'_j)$ contains at least three vertices, say $u, v, w \in V(B'_j)$, with either $uv \in E(G)$ or $vw \in E(G)$, assume $uv \in E(G)$. Then, $V(B'_j) \subseteq \langle u, v \rangle$, implying $w \in \langle S' \setminus \{w\} \rangle$. Thus, S' is R-dependent.

Subcase 1.2: If S' contains at most two vertices from each of the non complete blocks of G . Assume $S' \cap V(B'_j) = \{u, v\}$ with $uv \in E(G)$, then $V(B'_j) \subseteq \langle u, v \rangle$. Let $B'_j, B'_{j+1}, \dots, B'_r$ be a sequence of chains of non complete blocks in G . If there exists $w', w'' \in V(B'_k) \cap S'$ such that $w_k w' \in E(G)$ and $w_k \in V(B'_{k-1}) \cap V(B'_k)$ for some $k = j+1, j+2, \dots, r$, then $w'' \in \langle S' \setminus \{w''\} \rangle$, implying S' is R-dependent. Similar thing happens when there is an intermediate block B_i with $v \in V(B_i) \cap S'$.

Subcase 1.3: If there is no such B'_k as in **Subcase 1.2**, then the blocks $B'_j, B'_{j+1}, \dots, B'_k$ contributes additional vertices to the independence set,

which contradicts the assumption that $|S'| > \alpha(G')$. Thus, there must exist another non-complete block with adjacent vertices in S' , and we can apply the same argument as that of **Subcase 1.2**. If there exists a block B'_i such that $V(B'_i) \cap S' = \emptyset$, for $i = j + 1, \dots, k$ then, since $|S'| > \alpha(G')$, there must exist another non-complete block containing adjacent vertices in S' , and we can apply the same procedure as in **Subcase 1.2**. Thus, in this case, S' is R-dependent.

Case 2: S' contains more than ℓ vertices from the complete blocks.

If S' contains three vertices from any complete block, then S' is R-dependent. Similarly, if S' contains at least three vertices of a non-complete block, including a pair of adjacent vertices, then S' is R-dependent.

Subcase 2.1: If S' contains at most two vertices from each complete block. Assume $u, u' \in V(B_i) \cap S'$. Let B_i, B_{i+1}, \dots, B_k be a chain of complete blocks with $|V(B_j) \cap S'| = 1$ for $j = i + 1, \dots, k$. If there is $v, v' \in V(B_k) \cap S'$, then $v \in \langle S' \setminus \{v\} \rangle$, which implies S' is R-dependent. Similar things happens when B_k is replaced by B'_k with $v, v' \in V(B'_k) \cap S'$ and $vw \in E(G)$ for $w \in V(B'_k) \cap V(B_{k-1})$ or there is an intermediate block B'_j such that $v' \in V(B'_j) \cap S'$ and $v'w' \in E(G)$ for some $w' \in V(B'_j) \cap V(B_{j-1})$.

Subcase 2.2 If there is no complete block B_k or non-complete block B'_k as in **Subcase 2.1**, then either there exist another complete block B with $|V(B) \cap S'| = 2$ (since G is not a block graph and $|S'| \geq \ell + 3$) or there is a non-complete block containing at least three vertices, including one pair of adjacent vertices. In this case, we can apply Case 2.1 and Case 2 respectively. Thus in both Cases 1 and 2, S' is R-dependent. Therefore, the largest R-independent set in G has size $\alpha(G') + \ell$, proving that $r_\Delta(G) = \alpha(G') + \ell$.

Hence the theorem. \square

4 Rank

In this section, we delve into the concept of rank within the framework of Δ -convexity. Similar to the previous sections, we establish an upper bound for the rank based on the number of triangles in the graph and demonstrate that this bound is the least possible upper bound. Moreover, we determine the rank of block graphs. Interestingly, for block graphs, we observe that the rank is identical to both the Helly number and the Radon number in the context of Δ -convexity.

Observation 5. *If G is a triangle free graph, then the rank, $d_\Delta(G) = |V(G)|$*

Observation 6. *If G is a complete graph K_n for $n > 2$, then $d_\Delta(G) = 2$.*

The following property follows directly from the definition of a convexly dependent set:

Remark 6. If S is a set and $w \in S$ such that $w \in \langle S \setminus \{w\} \rangle$, then S is convexly dependent.

Next, we present some simple properties of the convexly independent set of a general graph, which will be used in subsequent results.

Lemma 4. *For a graph G , the following properties holds.*

- (i) *The rank, $d_\Delta(G)$ satisfies $d_\Delta(G) \geq \alpha(G)$.*
- (ii) *If S is a convexly independent set, then no three vertices of S forms a K_3 in G .*
- (iii) *If every two adjacent vertices of a graph G form a hull set of G , then $d_\Delta(G) = \max\{2, \alpha(G)\}$.*

Proof. (i) Let S be a maximum independent vertex set of the graph G , with $|S| = \alpha(G)$. Clearly, $\langle S \rangle = S$ and $a \notin \langle S \setminus \{a\} \rangle$ for each $a \in S$. This implies that S is convexly independent. Therefore, $d_\Delta(G) \geq \alpha(G)$.

(ii) Suppose $S \subseteq V(G)$ is a convexly independent set of G . Then by definition, $a \notin \langle S \setminus \{a\} \rangle$ for each $a \in S$. Let $S = \{a_1, a_2, a_3, \dots, a_n\}; n \geq 3$ such that S contains three vertices which forms a triangle K_3 in G . Then the removal of each vertex $a_i \in K_3$ is contained in $\langle S \setminus \{a_i\} \rangle$, contradicting our assumption that S is convexly independent. Hence no three vertices of S form a K_3 in G .

(iii) Let G be a graph such that for every $uv \in V(G), \langle u, v \rangle = V(G)$. Suppose G is complete. Consider a subset $S \subseteq V(G)$ of cardinality 2. Let $S = \{u_1, u_2\}$. Clearly $u_1 \notin \langle S \setminus \{u_1\} \rangle, u_2 \notin \langle S \setminus \{u_2\} \rangle$ and S is convexly independent. If possible, let $|S| \geq 3$. Suppose $S = \{u_1, u_2, u_3\}$. If we consider $\langle S \setminus \{u_1\} \rangle$, the remaining vertices of S can generate $V(G)$ and definitely $u_1 \in \langle S \setminus \{u_1\} \rangle$ implies that $S = \{u_1, u_2, u_3\}$ is convexly dependent. Hence $|S| = 2$. Suppose G is a non complete graph such that for any $uv \in E(G), \langle u, v \rangle = V(G)$. A maximum independent set S is convexly independent and $d_\Delta(G) \geq \alpha(G)$ holds. If $|S| > \alpha(G) \geq 2$, then there exists different $x, y \in S$ such that $xy \in E(G)$. For $a \in S, a \notin \{x, y\}$, we have $a \in \langle S \setminus \{a\} \rangle$ and $d_\Delta(G) = \alpha(G)$.

From Lemmas 1 and 4, we immediately obtain the following:

Corollary 3. *If G is a 2-connected chordal graph, then $d_\Delta(G) = \max\{2, \alpha(G)\}$.*

Theorem 6. *Let G be a graph with k triangles, and let m denote the number of vertices not lying on any triangles. Then $d_\Delta(G) \leq m + 2k$.*

Proof. A convexly independent set $S \subseteq V(G)$ cannot contain more than two vertices from any triangle in G , limiting the number of vertices in S that can be chosen from the k triangles to at most $2k$. The remaining m vertices, which do not lie on any triangles, are also convexly independent. Therefore, S can include all of these m vertices. Consequently, $d_\Delta(G) \leq m + 2k$. \square

Remark 7. There exists a graph G with k triangles such that the rank, $d_\Delta(G) = m + 2k$; where m is the number of vertices not lying on any triangle. For example, in Fig. 1, consider the set $S = \{a_1, a_2, \dots, a_k, b_1, b_2, \dots, b_k, d_1, d_2, \dots, d_m\}$, where $|S| = m + 2k$. Then the following holds: $\langle S \setminus \{a_i\} \rangle = V(G) \setminus \{a_i, c_i\}, \langle S \setminus \{b_i\} \rangle = V(G) \setminus \{b_i, c_i\}$, and $\langle S \setminus \{d_j\} \rangle = V(G) \setminus \{d_j\}$. So S is convexly independent. If any subset $S' \subseteq V(G)$ has $|S'| > m + 2k$, it must include all vertices of at least one triangle, making S' convexly dependent. Thus, $d_\Delta(G) = m + 2k$.

Moreover, the rank can be made arbitrarily large. For any natural number n , there exists a graph G with $d_\Delta(G) = n$, as stated in the following proposition.

Proposition 3. *For any natural number $n > 1$, there exists a graph G with $d_\Delta(G) = n$.*

For block graphs, the rank depends on the number of blocks in the graph. We present the following results for block graphs:

Theorem 7. *If G is a block graph with blocks B_1, B_2, \dots, B_ℓ , then $d_\Delta(G) = \ell + 1$.*

Proof. Let G be a block graph with blocks B_1, B_2, \dots, B_ℓ . Consider a set $S \subseteq V(G)$ where $S = \{u_1, u'_1, u_2, \dots, u_\ell\}$, with $\{u_1, u'_1\} \in V(B_1)$ and $u_i \in V(B_i)$ for $i = 2, 3, 4, \dots, \ell$, such that no three vertices of S forms a K_3 in G . Then, for each $a \in S$, we have $a \notin \langle S \setminus \{a\} \rangle$, making S convexly independent. Next, we need to show that there does not exist a convexly independent set in G that contains more than $\ell + 1$ vertices. Assume, to the contrary, that there exists a convexly independent set $S' \subseteq V(G)$ such that $|S'| \geq \ell + 2$. By Lemma 4, S' includes two vertices from at least two blocks and not more than two vertices from any single block.

Consider the following cases:

Case 1: S' contains at least one vertex from each block.

Since $|S'| \geq \ell + 2$, S' must contain two vertices from at least two blocks, say B_i and B_j , where $i, j \in \{1, 2, \dots, \ell\}$. Let $u, u' \in V(B_i)$ and $v, v' \in V(B_j)$, where $u, u', v, v' \in S'$. Then, $\langle S' \setminus \{u\} \rangle = \langle S' \setminus \{u'\} \rangle = \langle S' \setminus \{v\} \rangle = \langle S' \setminus \{v'\} \rangle = V(G)$, implying S' is convexly dependent.

Case 2: S' does not contain vertices from some blocks of G .

Since $|S'| \geq \ell + 2$ and S' does not contain more than two vertices from any single block, let p be the number of blocks in G from which exactly two vertices are in S' , and q be the number of blocks in G from which no vertices are in S' . Clearly, $p > q$. Thus, there exists blocks B_i and B_j such that $u, u', v, v' \in S'$ with $u, u' \in V(B_i)$ and $v, v' \in V(B_j)$, and a chain of blocks between B_i and B_j (denoted as $B_{i+1}, B_{i+2}, \dots, B_{j-1}$) such that $u_k \in S' \cap V(B_k)$ for $k = i + 1, i + 2, \dots, j - 1$. Hence $\cup_{h=i}^j V(B_h) \subset \langle S' \setminus \{w\} \rangle$, or simply, $w \in \langle S' \setminus \{w\} \rangle$ for $w \in \{u, u', v, v'\}$, making S' convexly dependent.

In both cases, we conclude that S' cannot be convexly independent, violating our assumption that $|S'| \geq \ell + 2$. Therefore, the largest convexly independent set in G has size $\ell + 1$, establishing that $d_\Delta(G) = \ell + 1$. \square

Concluding Remarks

It may be noted that, for most of the convexity in discrete structures, especially in graphs, the Radon number is one more than the Helly number. In this paper, we have shown that the Helly and Radon numbers are the same for chordal graphs with respect to Δ -convexity. We also observe that for all the small-sized graphs that we have examined so far, the Helly and Radon numbers are equal.

This observation makes the Δ -convexity rather special compared to most of the other graph convexities. Hence, we pose the following as a conjecture.

Conjecture 1. If G is a non-trivial connected graph, then $h_{\Delta}(G) = r_{\Delta}(G)$.

Due to restriction of the page size, we couldn't give the proofs of the Propositions 1, 2 and 3. However, one can refer the arxiv (<https://arxiv.org/abs/2411.10816>) for the proofs.

Acknowledgment. Arun Anil acknowledges the financial support from the University of Kerala, for providing the University Post Doctoral Fellowship (Ac EVII 5911/2024/UOK dated 18/07/2024). The authors thank the anonymous referees for their valuable suggestions, which helped in improving this article.

References

1. Anand, B.S., Anil, A., Changat, M., Dourado, M.C., Ramla, S.S.: Computing the hull number in Δ -convexity. *Theor. Comput. Sci.* **844**, 217–226 (2020)
2. Anand, B.S., Dourado, M.C., Narasimha-Shenoi, P.G., Ramla, S.S.: On the Δ -interval and the Δ -convexity numbers of graphs and graph products. *Disc. Appl. Math.* **319**, 487–498 (2022)
3. Anand, B.S., Narasimha-Shenoi, P.G., Ramla, S.S.: Δ -convexity number and Δ -number of graphs and graph products. In: Changat, M., Das, S. (eds.) CALDAM 2020. LNCS, vol. 12016, pp. 209–218. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-39219-2_18
4. Centeno, C.C., Penso, L.D., Rautenbach, D., Pereira de Sa, V.G.: Geodetic number versus hull number in P_3 -convexity. *SIAM J. Disc. Math.* **27**(2), 717–731 (2013)
5. Changat, M., Mathew, J.: Induced path transit function, monotone and peano axioms. *Disc. Math.* **286**(3), 185–194 (2004)
6. Changat, M., Mulder, H.M., Sierksma, G.: Convexities related to path properties on graphs. *Disc. Math.* **290**(2–3), 117–131 (2005)
7. Doignon, J.-P., Reay, J.R., Sierksma, G.: A Tverberg-type generalization of the Helly number of a convexity space. *J. Geom.* **16**, 117–125 (1981)
8. Dourado, M.C., Penso, L.D., Rautenbach, D.: On the geodetic hull number of P_k -free graphs. *Theor. Comput. Sci.* **640**, 52–60 (2016)
9. Dourado, M.C., Sampaio, R.M.: Complexity aspects of the triangle path convexity. *Disc. Appl. Math.* **206**, 39–47 (2016)
10. Duchet, P.: Convex sets in graphs, II. Minimal path convexity. *J. Comb. Theory Ser. B* **44**(3), 307–316 (1988)
11. Farber, M.: Bridged graphs and geodesic convexity. *Disc. Math.* **66**(3), 249–257 (1987)
12. Jamison, R.E.: A general theory of convexity. University of Washington (1974)
13. Maria Aurora Morgana and Henry Martyn Mulder: The induced path convexity, betweenness, and svelte graphs. *Disc. Math.* **254**(1–3), 349–370 (2002)
14. Pelayo, I.M.: Geodesic Convexity in Graphs. Springer, Heidelberg (2013)
15. Rockefellar, R.T.: Convex Analysis. Princeton University Press, Princeton (1970)
16. Sierksma, G.: Carathéodory and Helly-numbers of convex-product-structures. *Pac. J. Math.* **61**(1), 275–282 (1975)

17. Siersma, G.: Relationships between Carathéodory, Helly, Radon and exchange numbers of convexity spaces, Rijksuniversiteit (1976)
18. van De Vel, M.L.J.: Theory of Convex Structures. Elsevier, Amsterdam (1993)
19. West, D.B., et al.: Introduction to Graph Theory, vol. 2. Prentice hall, Upper Saddle River (2001)



Forbidden Induced Subgraphs in Iterative Higher Order Line Graphs

Aryan Sanghi¹, Devsi Bantva², and Sudebkumar Prasant Pal¹(✉)

¹ Indian Institute of Technology, Kharagpur 721302, West Bengal, India
sanghi@kgpian.iitkgp.ac.in, sudebkumar@gmail.com

² Lukhdhirji Engineering College, Morvi 363642, Gujarat, India
devsi.bantva@lecollege.ac.in

Abstract. Let G be a simple finite connected graph. The line graph $L(G)$ of graph G is the graph whose vertices are the edges of G , where $ef \in E(L(G))$ when $e \cap f \neq \emptyset$. Iteratively, the higher order line graphs are defined inductively as $L^1(G) = L(G)$ and $L^n(G) = L(L^{n-1}(G))$ for $n \geq 2$. In [1,2], Beineke characterize line graphs in terms of nine forbidden subgraphs. Inspired by this result, in this paper, we characterize second order line graphs in terms of pure forbidden induced line subgraphs. We also give a sufficient list of forbidden subgraphs for a graph G such that G is a higher order line graph. We characterize all order line graphs of graph G with $\Delta(G) = 3$ and 4.

Keywords: Line graph · induced graphs · forbidden graphs

1 Introduction

Let G be a simple, finite, connected graph without loops and multiple edges. The line graph of G , denoted by $L(G)$, is the graph whose vertices are the edges of G , where $ef \in E(L(G))$ when $e \cap f \neq \emptyset$. Iteratively, the higher order line graph is defined as $L^1(G) = L(G)$ and $L^n(G) = L(L^{n-1}(G))$ for $n \geq 2$. It is known that a connected graph G is isomorphic to its line graph if and only if G is a cycle. Thus, $L^n(C_m) = C_m$ for all n while for $K_{1,3}$, $L^n(K_{1,3}) = L(K_{1,3})$ for all n , but $L(K_{1,3}) \neq K_{1,3}$. For a path P_n , $L(P_n) = P_{n-1}$, $L^{n-1}(P_n)$ is a vertex, and $L^m(P_n)$ does not exist if $m \geq n$. In this work, we often use the statement that G is a line graph that means G is a line graph of some graph H . In other words, G is a line graph that means $G = L(H)$ for some graph H . If $L(H) = G$ then we say that graph H is a preimage of a line graph G , written as $L^{-1}(G) = H$. A clique is a graph in which every two vertices are adjacent. A subgraph H of graph G is called an induced subgraph if two vertices of H are adjacent in G they are also adjacent in H .

In [7], Whitney gave the following result about graphs and their line graphs, popularly known as the Whitney graph isomorphism theorem.

Theorem 1 [7]. $L(G^1) \cong L(G^2)$ and if G^1 and G^2 are not a complete graph of three nodes and a complete bipartite graph $K_{1,3}$, respectively, then $G^1 \cong G^2$.

It is clear by the Whitney isomorphism theorem that there are only two connected graphs, namely $K_{1,3}$ and K_3 , that have the same line graph which is K_3 . Moreover, there is a one-to-one correspondence between the set of graphs excluding K_3 and $K_{1,3}$ and, its line graphs by line graph operation.

Observation 1. Observe that if graph G is a K_3 only then it is difficult to say which one from $K_{1,3}$ or K_3 is a preimage, but if $G \neq K_3$ is a connected line graph containing K_3 then its preimage can be easily determined as other vertex v adjacent to a vertex of $K_{1,3}$ or K_3 makes different line graphs (see the following Fig. 1). This property plays a key role in finding the preimage of any line graph.

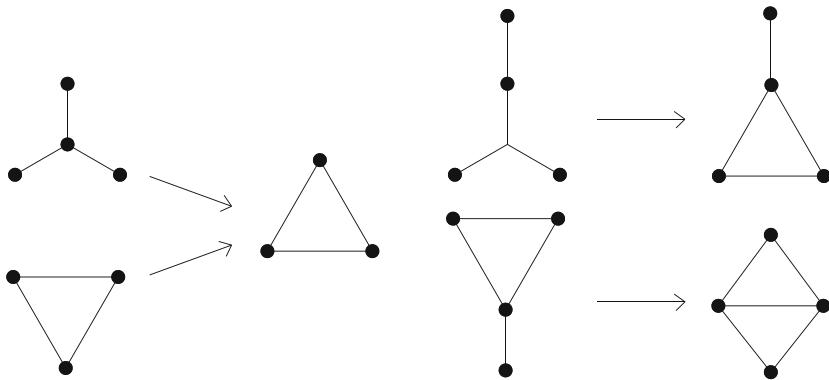


Fig. 1. Graphs $K_{1,3}$, K_3 and its line graphs.

The following characterization of line graphs was given by Krausz in [3] and Rooij in [4], respectively.

Theorem 2 [3]. *A graph is a line graph if and only if its edges can be partitioned into complete subgraphs in such a way that no vertex lies in more than two of these subgraphs.*

The partition of edges of a line graph into complete subgraphs as in Theorem 2 is called ‘line partition’ and the partition sets (complete subgraphs of a line partition), denoted by C^1, C^2, \dots, C^k , are known as ‘cells’ of line partition. A triangle of a graph G is called an *odd triangle* if there is a vertex of G adjacent to an odd number of its vertices (that is, adjacent to either one or all three vertices), and an *even triangle*, otherwise (that is, any vertex of graph G that is adjacent to vertices of the triangle is adjacent to exactly two vertices of the triangle).

The following succinct characterization for line graphs states such graphs as having a K_4 whenever there are two distinct odd triangles sharing a common edge.

Theorem 3 [4]. *A graph is a line graph if and only if it does not have the star $K_{1,3}$ as an induced subgraph and whenever a, b, c and b, c, d are distinct odd triangles, then a and d are adjacent.*

Theorem 4 [4]. *If G contains two even triangles with a common edge then it is one of the graphs $L(K_{1,3} + x) = E_1$, $L(E_1) = E_2$ or $L(K_4) = E_3$ (shown in following Figure 2).*

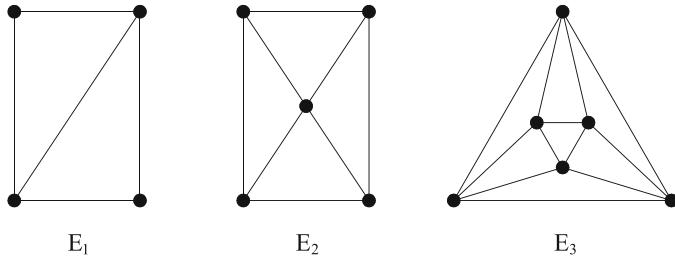


Fig. 2. Graphs E_1 , E_2 and E_3 .

Corollary 1. *For a line graph G not isomorphic to K_3 , E_1 , E_2 or E_3 , whenever two triangles in G share an edge, at least one of them is odd.*

Beineke gave the characterization in terms of forbidden subgraphs for line graphs in [1,2] as follows:

Theorem 5 [1,2]. *A graph is a line graph if and only if it does not contain any of the nine graphs shown in Figure 3 as an induced subgraph.*

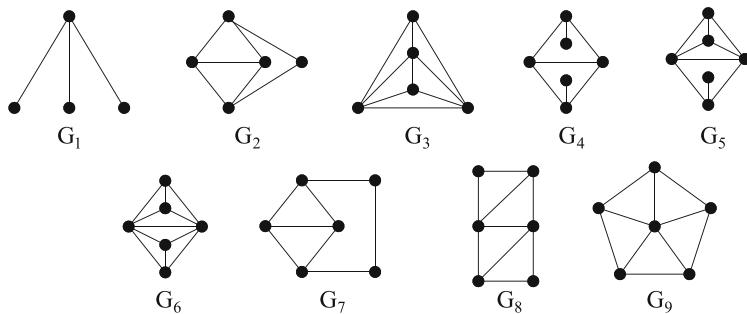


Fig. 3. The nine forbidden induced subgraphs in line graphs.

Šoltés [6] decreased the number of forbidden subgraphs from nine to seven and proved the following result.

Theorem 6 [6]. *The following five statements are equivalent for a connected graph G :*

- (a) G is a line graph.
- (b) G does not contain any of the graphs $G_1 - G_9$ (see Fig. 3) as an induced subgraph.
- (c) G does not contain any of the graphs $G_1 - G_8$ as an induced subgraph and G is not G_9 .
- (d) G does not contain any of the graphs $G_1 - G_7$ and G_9 as an induced subgraph and G is neither G_8 nor H_1 (see Fig. 4).
- (e) G does not contain any of the graphs $G_1 - G_7$ as an induced subgraph and G is not isomorphic to any of the graphs G_8, G_9, H_1, H_2 and H_3 (see Fig. 4).

Moreover, for every integer $i < 7$, there are infinitely many connected non-line graphs containing only G_i as an induced subgraph among $G_1 - G_9$.

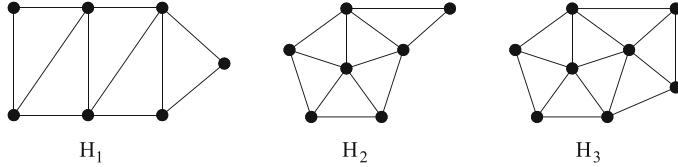


Fig. 4. Three forbidden induced subgraphs in line graphs.

Observation 2. *Let G be a line graph such that $G = L(H)$. Then the following hold.*

- (a) *Every induced subgraph of a line graph is itself a line graph.*
- (b) *If G is a line graph of H and G_1 is an induced subgraph of G then $L^{-1}(G_1) = H_1$ is a subgraph of H not necessarily induced.*
- (c) *For an odd induced triangle T in G , the edges corresponding to vertices of T in H constitute a star.*

Theorem 7. *For an even induced triangle T in $G \notin \{K_3, E_1, E_2, E_3\}$, the edges corresponding to vertices of T in $L^{-1}(G)$ constitute a triangle.*

Proof. Assume for the sake of contradiction that the edges in $L^{-1}(G)$, corresponding to the vertices of the even triangle T in G , form a star $K_{1,3}$.

Now, as the triangle T is even, any edge connected to the star in $L^{-1}(G)$ is between two of the vertices of the star; otherwise there is an edge from one of the vertices of star to a vertex outside star in $L^{-1}(G)$, the corresponding vertex in G will be connected to exactly one or three vertices of the triangle T which makes it odd - a contradiction. This implies there can not be more than 4 vertices in $L^{-1}(G)$; otherwise the star will be disconnected from those vertices. Hence, the only possible cases are $K_{1,3}, K_{1,3} + e, K_4 - e$ and K_4 . The $L(G)$ for these are K_3, E_1, E_2 and E_3 , which leads to contradiction. This completes the proof.

Corollary 2. *Let G be a line graph and $G \neq K_3$. Then every even triangle in G shares an edge with at least one other triangle.*

ROUSSOPOULOS [5] gave a $\max\{m, n\}$ algorithm for determining the graph H from its line graph G , where m and n are the number of edges and vertices, respectively. This algorithm also determines whether the given graph G is a line graph or not. For a given line graph G , $H = L^{-1}(G)$ is constructed as follows: The vertices of H correspond to the sets of P together with the set of W of vertices of G belonging to only one of the cells of P , where P is the set of vertices corresponding to cells of line partition of G . Two vertices in H are adjacent whenever the corresponding sets have a non empty intersection.

2 Main Results

In this section, we continue to use the terms and notations defined in the previous section. We first prove the following result, which we use to prove the next result about forbidden subgraphs in second order line graphs.

Theorem 8. *Let $G \neq K_3$ be a line graph of H and G' be an induced subgraph of G such that $G' = L(H')$. Then H' is an induced subgraph of H if and only if there is no $v \in V(G) \setminus V(G')$ such that v is adjacent to two vertices of G' and both edges which join v to two vertices of G' are in different cells of the line partition of subgraph induced by $G' \cup \{v\}$.*

Proof. Necessity: Suppose H' is an induced subgraph of H . If possible, assume that there is $v \in V(G) \setminus V(G')$ such that v is adjacent to two vertices of G' such that both edges which joins v to G' are in different cells of the line partition of subgraph induced by $G' \cup \{v\}$. Let $L(H'') = G' \cup \{v\}$ and v is adjacent to v_1 and v_2 . Using Algorithm given in [5], it is clear that in H' , we have $v_1, v_2 \in W$ and hence the vertices corresponding to v_1 and v_2 are pendant vertices while in H'' , we have $\{v, v_1\}, \{v, v_2\} \in P$ and hence the vertices corresponding to $\{v, v_1\}$ and $\{v, v_2\}$ are adjacent in H . Hence, we obtain H' is not an induced subgraph of H which is a contradiction. Therefore, our assumption is not true which completes the proof of necessity part.

Sufficiency: Suppose $G' = L(H')$ is an induced subgraph of G and there is no $v \in V(G) \setminus V(G')$ such that v is adjacent to two vertices of G' and both edges which join v to two vertices of G' are in different cells of the line partition of subgraph induced by $G' \cup \{v\}$. If possible, assume that H' is not an induced subgraph of H . Since H' is not an induced subgraph of H , there exist two vertices v_1 and v_2 in H' such that v_1 and v_2 are adjacent in H but not in H' . Let e be the edge in H which joins v_1 and v_2 . Since G is connected graph, note that these two vertices v_1 and v_2 are incident to x and y , respectively (possibly $x = y$). Let $v_1x = e_1$ and $v_2y = e_2$. Note that $e \in V(L(H))$ but $e \notin V(L(H')) = V(G')$. Hence, $e \in V(G) \setminus V(G')$ such that e is adjacent to two vertices $e_1 \in V(G')$ and $e_2 \in V(G')$ of $L(H) = G$ and note that they are in different cells as the vertices $\{v_1\}$ and $\{v_2\}$ corresponding to e_1 and e_2 are different, which is a contradiction. This completes the proof.

A subgraph G' of a line graph $G = L(H)$ is called a *pure induced line subgraph* if $G' = L(H')$ then H' is an induced subgraph of H .

Theorem 9. *Let G be any connected graph and G is not isomorphic to any of the graphs G_8, G_9, H_1, H_2 and H_3 . Then $G = L^2(H)$ for some graph H if and only if the following holds:*

- (a) *Every triangle in G is a part of some induced subgraph isomorphic to $K_4 - e$.*
- (b) *G does not contain any of the graphs $G_1 - G_7$ as an induced subgraph and $L(G_2) - L(G_7)$ as a pure induced line subgraph.*

Proof. Necessity: Suppose $G = L^2(H)$ for some graph H . Since $G = L(L(H))$, G and $L^{-1}(G)$ both does not contain any of the graph $G_1 - G_7$ as an induced subgraph by Theorem 6. Observe that, by Theorem 8, G does not contain any of the graphs $L(G_2) - L(G_7)$ as a pure induced line subgraph. Hence, the condition (b) is satisfied. We now prove (a). Since $L^{-1}(G)$ is claw free, it is clear that G has no triangle alone. Moreover, if $L^{-1}(G)$ contains triangle and as G is connected, we have every triangle is a part of $K_4 - e$.

Sufficiency: Suppose (a) and (b) hold for G . Since G does not contain $G_1 - G_7$ as an induced subgraph, by Theorem 6, we obtain $G = L(H')$ for some graph H' . Since G does not contain $L(G_1) - L(G_7)$ as pure induced line subgraphs, $L^{-1}(G) = H'$ does not contain $G_1 - G_7$ as induced subgraphs and hence $H' = L(H)$ by Theorem 6. Hence, we obtain, $G = L(H') = L(L(H)) = L^2(H)$ for some graph H .

Theorem 10. *Let G be any connected graph and G is not isomorphic to any of the graphs K_3, G_8, G_9, H_1, H_2 and H_3 . Then $G = L^2(H)$ for some graph H if the following holds:*

- (a) *Every triangle in G is a part of some induced subgraph isomorphic to $K_4 - e$.*
- (b) *G does not contain any of the graphs $G_1 - G_7$ (given in Fig. 3) and $F_1 - F_2$ (given below in Fig. 5) as an induced subgraph.*

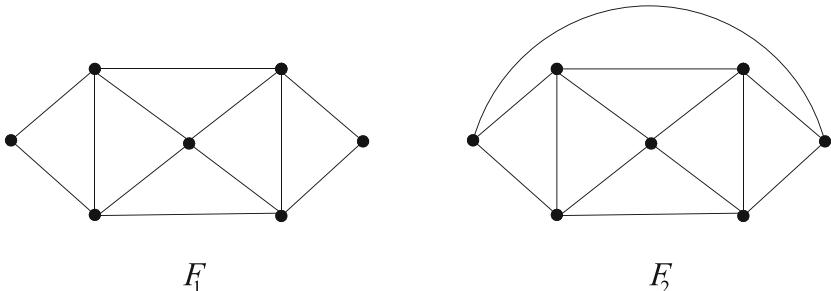


Fig. 5. Graphs $F_1 = L(G_4)$ (left) and $F_2 = L(G_2)$ (right).

Proof. Let G be a connected graph that is not isomorphic to any of the graphs $K_3, G_8, G_9, H_1, H_2, H_3$, and G satisfy the conditions (a) and (b) of the hypotheses. We prove that $G = L^2(H)$ for some graph H .

Note that, by Theorem 6 (e), it suffices to prove that $L^{-1}(G)$ does not contain any of the graphs $G_1 - G_7$ as an induced subgraph as this ensures the existence of a graph H such that $L^{-1}(G) = L(H)$. Since G does not contain $F_1 = L(G_4)$ and $F_2 = L(G_2)$ as an induced subgraph, $L^{-1}(G)$ does not contain G_4 and G_2 as a subgraph. Since G does not contain F_1 as an induced subgraph, $L^{-1}(G)$ does not contain G_4 as a subgraph of G and hence G does not contain G_5, G_6 and G_7 as an induced subgraph as G_4 is a subgraph of G_5, G_6 and G_7 . Similarly, G does not contain F_2 as an induced subgraph implies $L^{-1}(G)$ does not contain G_2 as a subgraph. Hence G does not contain G_3 as an induced subgraph because G_3 contains G_2 as a subgraph. Now it remains to show that $L^{-1}(G)$ does not contain $G_1 = K_{1,3}$.

Suppose $L^{-1}(G)$ contains $G_1 = K_{1,3}$ as an induced subgraph. Let $V(K_{1,3}) = \{a, b, c, d\}$ with $E(K_{1,3}) = \{ab, ac, ad\}$. Since $K_{1,3}$ is an induced subgraph of $L^{-1}(G)$, it is clear that no two vertices of $\{b, c, d\}$ are mutually adjacent. Now, as there is no edge that is connected to exactly two of the edges of $K_{1,3}$ in $L(H)$, there is no vertex connected to exactly two of the vertices of triangle formed by vertices corresponding to the edges ab, ac and ad . This implies the triangle is not a part of $K_4 - e$ in G , which is a contradiction with (a). Hence, $L^{-1}(G)$ does not contain $K_{1,3}$ which completes the proof.

Theorem 11. *Let G be any connected graph and G is not isomorphic to any of the graphs K_3, G_8, G_9, H_1, H_2 and H_3 . If $G = L^3(H)$ for some graph H , then every odd triangle in G is a part of some induced subgraph isomorphic to $K_4 - e$ and every even triangle in G is a part of some induced subgraph isomorphic to $L(K_4 - e)$.*

Proof. Since $G = L^3(H) = L^2(L(H))$ for some graph H , the graph follows the condition stated in Theorem 9 (a), implying every triangle, including odd triangle, is part of an induced subgraph isomorphic to $K_4 - e$. Now, consider an even triangle in graph G . Let the vertices of the even triangle be a, b, c . Now, by Theorem 7, in $L^{-1}(G)$, the corresponding edges to a, b, c will be a part of triangle. Now, in $L^{-1}(G) = L^2(H)$, by Theorem 9 (a), the triangle is a part of an induced subgraph isomorphic to $K_4 - e$, say A . Hence, $L(A) = L(K_4 - e)$ is an induced subgraph of $L^3(H)$. Therefore, every even triangle in G is a part of some induced subgraph isomorphic to $L(K_4 - e)$.

Theorem 12. *Let G be any connected graph and G is not isomorphic to any of the graphs K_3, G_8, G_9, H_1, H_2 and H_3 . Then $G = L^n(H), n \geq 2$ for some graph H if the following holds:*

- (a) $L^{-i}(G)$ is claw free for all $i \in \{1, 2, \dots, n-1\}$.
- (b) G does not contain any of the graphs $G_1 - G_7$ (given in Fig. 3) and $F_1 - F_2$ (given in Fig. 5) as an induced subgraph.

Proof. We give proof using induction. The statement holds for $n = 2$ as proved in Theorem 10.

For any graph G , let the statement hold for all $k \in \{3, \dots, n - 1\}$. We now prove that the statement holds for $k = n$. Let a graph G satisfy conditions (a) and (b) stated in the hypothesis. Note that G is a line graph by Theorem 6 as graph G does not contain $G_1 - G_7$. Since G does not contain $F_1 - F_2$ and either F_1 or F_2 is a subgraph of $L(G_2) - L(G_7)$, G does not contain $L(G_2) - L(G_7)$ as induced subgraphs. Again, as G does not contain $L(G_2) - L(G_7)$ as induced subgraphs, $L^{-1}(G)$ does not contain $G_2 - G_7$ as subgraphs by observation 2 and hence $L^{-1}(G)$ does not contain $G_2 - G_7$ as an induced subgraph. Since $L(F_1)$ and $L(F_2)$ contain F_1 as an induced subgraph and G does not contain F_1 and F_2 , G does not contain $L(F_1)$ and $L(F_2)$ as an induced subgraph. Hence, $L^{-1}(G)$ does not contain F_1 and F_2 as subgraphs by observation 2. Therefore, $L^{-1}(G)$ does not contain F_1 and F_2 as induced subgraphs. Moreover, $L^{-i}(G)$ is claw free for all $i \in \{1, 2, \dots, k - 1\}$ by (a). Hence, by the induction hypothesis, $L^{-1}(G) = L^{n-1}(H)$ for some graph H . Hence, we obtain $G = L^n(H)$. Therefore, by induction, the theorem is true for all n .

Theorem 13. *Let G be a connected graph with $\Delta(G) = 3$ and $G \neq G_2$. Then*

- (a) *$G = L(H)$ for some graph H if and only if G does not contain any of G_1, G_4, G_7 (shown in Fig. 3) as an induced subgraph.*
- (b) *$G = L^2(H)$ for some graph H if and only if G is isomorphic to $L_{k,n}$, where $k \in \{1, 2\}$ and $n \geq k$ as shown in Fig. 6.*

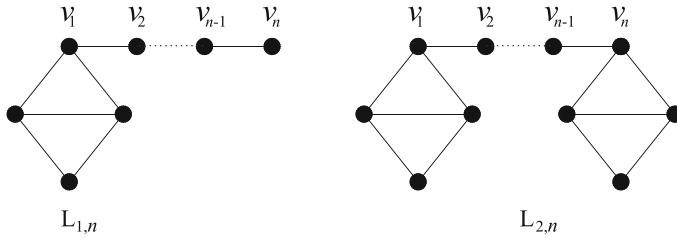


Fig. 6. Two classes of line graphs $G = L^2(H)$ with $\Delta(G) = 3$.

- (c) *For $n \geq 3$, $G \neq L^n(H)$ for any graph H .*

Proof. (a) **Necessity:** Suppose G is a line graph with $\Delta(G) = 3$ and $G \neq G_2$. By Theorem 5, it is clear that G does not contain any of the G_1, G_4, G_7 as an induced subgraph.

Sufficiency: Suppose G is not G_2 and G does not contain any of G_1, G_4, G_7 as an induced subgraph. Since $\Delta(G_3) = \Delta(G_5) = \Delta(G_8) = 4$ and $\Delta(G_6) = \Delta(G_9) = 5$, G does not contain any of G_3, G_5, G_6, G_8, G_9 as an induced subgraph. We now prove that G does not contain G_2 as an induced subgraph. Suppose G contains G_2 as an induced subgraph. Since $G \neq G_2$, there is a vertex in G

which is adjacent to degree 2 vertex of G_2 and not adjacent to any other vertex of G_2 . Then observe that G contains $K_{1,3}$ as an induced subgraph which is a contradiction. Hence G does not contain G_2 as an induced subgraph. By Theorem 5, G is a line graph.

(b) **Necessity:** Since G is simple and $\Delta(G) = 3$, it is clear that $|V(G)| \geq 4$ and hence $G \neq K_3$. Moreover, observe that G does not contain K_n for $n \geq 5$ as an induced subgraph; otherwise $\Delta(G) \geq 4$. Also, G does not contain K_4 ; otherwise G is K_4 only as $\Delta(G) = 3$ and $L^{-1}(G)$ contains $K_{1,3}$ which is a contradiction. Since $\Delta(G) = 3$, let u be the vertex with $d(u) = 3$. Assume that u is adjacent to v_1, w and x . Since G does not contain $K_{1,3}$ as an induced subgraph, u forms a triangle with two of $\{v_1, w, x\}$. Assume u, w and x forms a triangle. By Theorem 9, note that every triangle is a part of $K_4 - e$. Now we have the following two cases.

Case-1: v_1 is adjacent to w . Observe that u and w are not adjacent to any other vertex of G as $\Delta(G) = 3$ and $d(u) = d(w) = 3$. Moreover, note that v_1 and x are not adjacent; otherwise G is isomorphic to K_4 (outer connections are not possible as $\Delta(G) = 3$) which is a contradiction.

Claim-1: At least one of $\{v_1, x\}$ has degree 2.

If possible, then assume that $d(v_1) = d(x) = 3$. In this case, we have the following possibilities: (i) Let v_2 be the vertex adjacent to both v_1 and x then note that G contains G_2 - a contradiction with G being a line graph, (ii) Let v_2 and y be two vertices adjacent to v_1 and x , respectively. If v_2 and y are adjacent then G contains G_7 - a contradiction with G being a line graph. If v_2 and y are not adjacent then G contains G_4 as an induced subgraph - a contradiction with G being a line graph. Hence, our assumption $d(v_1) = d(x) = 3$ is not true. Therefore, at least one of v_1 and x has degree 2.

Without loss of generality, assume that $d(x) = 2$. If $d(v_1) = 2$ then G is $K_4 - e = L_{1,1}$. If $d(v_1) = 3$ then let v_2 is adjacent to v_1 . If $d(v_2) = 1$ then G is $L_{1,2}$. Continuing in this way, let v_2, v_3, \dots, v_n be the sequence of vertices such that v_i is adjacent to v_{i-1} for $2 \leq i \leq n$ and $d(v_n) = 1$, $d(v_i) = 2$ for $2 \leq i \leq n-1$ then G is $L_{1,n}$. Let v_n be the first vertex such that $d(v_n) = 3$. Since v_n is the first vertex in the sequence, v_n is not adjacent to any of $\{v_1, v_2, \dots, v_{n-2}, u, w, x\}$. Let v_n is adjacent to a and b . Again, note that a and b are not adjacent to $\{v_1, v_2, \dots, v_{n-1}, u, w, x\}$. Since G does not contain $K_{1,3}$ as an induced subgraph, a and b are adjacent and hence v_n, a and b form a triangle. By Theorem 9, note that v_n, ab is a part of $K_4 - e$. Let c is adjacent to a and b but observe that it is not adjacent to v_n as G does not contain K_4 . By claim-1, it is clear that $d(c) = 2$ and hence G is $L_{2,n}$.

Case-2: v_1 is not adjacent to w . Then let y be the vertex adjacent to both w and x . By claim-1, note that $d(y) = 2$. Now, the subgraph induced by v_1, w, x, y, u is isomorphic to $L_{1,2}$. Now continuing similar to case-1, it is easy to show that G is either $L_{1,n}$ or $L_{2,n}$ for $n \geq 1$.

Sufficiency: Suppose G is isomorphic to $L_{k,n}$, where $k \in \{1, 2\}$ and $n \geq k$, then note that $G = L^2(H)$, where H is the graph obtained by identifying a pendant vertex of a claw $K_{1,3}$ with one end vertex of path $P_{n+1}, n \geq 1$.

(c) Suppose there exists a graph G with $\Delta(G) = 3$ such that $G = L^n(H)$ for some $n \geq 3$. In this case, observe that G has an induced subgraph K such that $L^{-(n-2)}(K) = K_4 - e$ by Theorem 9 (a). Hence, $L^2(H)$ contains $K_4 - e$ as an induced subgraph. Then note that $\Delta(L(K_4 - e)) = 4$ which is a contradiction with $\Delta(G) = 3$. Therefore, we obtain that for $G \neq L^3(H)$ for any graph H . Since every $L^n(H) = L^3(L^{n-3}(H))$, we obtain that $G \neq L^n(H)$ for $n \geq 3$.

Let G be a simple finite connected graph. A subgraph H of a graph G is called a pendant subgraph if H is connected to $G - H$ by an edge only.

Theorem 14. *Let G be a connected graph with $\Delta(G) = 4$ and $G \neq G_3, G_8, G_9, H_1, H_2, H_3$. Then*

- (a) *$G = L(H)$ for some graph H if and only if G does not contain any of $G_1, G_2, G_4, \dots, G_7$ as an induced subgraph and G does not contain G_3 as a pendant subgraph.*
- (b) *$G = L^3(H)$ for some graph H if and only if G is isomorphic to $L(L_{k,n})$, where $k \in \{1, 2\}$ and $n \geq k$.*
- (c) *For $n \geq 4$, $G \neq L^n(H)$ for any graph H .*

Proof. (a) Necessity: Suppose G is a line graph with $\Delta(G) = 4$ and $G \neq G_3, G_8, G_9, H_1, H_2, H_3$. By Theorem 6, G does not contain any of the $G_1, G_2, G_3, \dots, G_7$. Since G does not contain G_3 , G does not contain G_3 as a pendant subgraph.

Sufficiency: Suppose G is a connected graph with $\Delta(G) = 4$ and $G \notin \{G_3, G_8, G_9, H_1, H_2, H_3\}$ and, G does not contain any of $G_1, G_2, G_4, \dots, G_7$ and G does not contain G_3 as a pendant subgraph. It is enough to prove that G does not contain G_3 with two edge connections in $G - G_3$. If possible, assume that G contains G_3 with two edges in $G - G_3$. Let $\{a, b, c, d, x\}$ induces G_3 in G such that abc and bcd are two triangles and x is adjacent to a, b, c and d . Observe that $d(b) = d(c) = d(x) = 4$ and hence b, c and x are not adjacent to any other vertex of G as $\Delta(G) = 4$. Moreover, observe that a and d are not adjacent; otherwise, the subgraph induced by $\{a, b, c, d, x\}$ is K_4 and in this case, G is K_4 only and hence $\Delta(G) = 3$ which is a contradiction. Since $G \neq G_3$, $d(a) = d(d) = 3$ and $\Delta(G) = 4$, there exists either one vertex or two distinct vertices that are adjacent to a and d . Then we have the following possible cases and in each case, we obtain that G contains one of graph $\{G_2, G_4, G_7\}$ as an induced subgraph which is a contradiction with G being a line graph:

- If a vertex u is adjacent to both a and d then the subgraph induced by $\{a, b, d, x, u\}$ is G_2 and hence G contains G_2 as an induced subgraph.
- If two vertices u and v are adjacent to a and d , respectively, then
 - if u and v are adjacent then the subgraph induced by $\{a, b, d, x, u, v\}$ is G_7 and hence G contains G_7 as an induced subgraph.
 - if u and v are not adjacent then the subgraph induced by $\{a, b, d, x, u, v\}$ is G_4 and hence G contains G_4 as an induced subgraph.

Therefore, in all above cases, our assumption that G contains G_3 with at least two edge connections in G leads to a contradiction. Hence, G does not contain G_3 as a non-pendant subgraph.

(b) Necessity: Let G be a connected graph with $\Delta(G) = 4$ and $G = L^3(H)$ for some connected graph H . Note that $L^{-i}(G)$, $i = 1, 2$ does not contain $K_{1,3}$ as an induced subgraph as $G = L^3(H) = L(L^2(H)) = L(L(L(H)))$. Since $\Delta(G) = 4$, there exists a vertex u such that $d(u) = 4$. Let u_1, u_2, u_3 and u_4 be the vertices adjacent to u . As $G = L(L^2(H))$, u will form a triangle with two of u_1, u_2, u_3, u_4 as it does not contain $K_{1,3}$ as an induced subgraph by Theorem 5. By Theorem 9, every triangle is a part of $K_4 - e$ and by Theorem 3, observe that one of those triangles is even. Hence, by Theorem 11, even triangle is a part of $E_2 = L(K_4 - e)$, which is an induced subgraph of G . Let $u_1 - u_2 - u_3 - u_4 - u_1$ be a cycle and u be the central vertex of degree 4 in the induced graph E_2 .

Claim-1: If $G \neq E_2$ then there is a vertex $v_1 \in V(G) \setminus V(E_2)$ such that v_1 is adjacent to two consecutive vertices of cycle form by $\{u_1, u_2, u_3, u_4\}$.

Let $v_1 \in V(G) \setminus V(E_2)$. It is clear that v_1 is not adjacent to u as $d(u) = 4$ and $\Delta(G) = 4$. We consider the following possibilities:

- If v_1 is adjacent to one vertex only, say u_1 , then the subgraph induced by $\{u_1, v_1, u_2, u_4\}$ is $K_{1,3}$ centered at u_1 which is a contradiction with G being a line graph.
- If v_1 is adjacent to two non-consecutive vertices of the cycle of E_2 , then the subgraph induced by $\{v_1, u, u_1, u_2, u_3\}$ is G_2 which is a contradiction with G being a line graph.
- If v_1 is adjacent to three vertices, say u_1, u_2, u_3 , of the cycle of E_2 , then the subgraph induced by $\{v_1, u, u_1, u_3, u_4\}$ is G_2 which is a contradiction with G being a line graph.
- If v_1 is adjacent to all four vertices u_1, u_2, u_3, u_4 then $V(G) = \{v_1, u, u_1, \dots, u_4\}$ and $G = L(K_4)$ and hence $L^{-2}(G) = K_{1,4}$ which contains $K_{1,3}$ which is a contradiction with $L^{-2}(G) = L(H)$.

Hence, $v_1 \in V(G) \setminus V(E_2)$ is adjacent to two consecutive vertices, say u_1 and u_2 , of cycle induced by $\{u_1, u_2, u_3, u_4\}$ which completes the proof of Claim-1.

Claim-2: There is no $x \in V(G) \setminus \{v_1, u, u_1, \dots, u_4\}$ such that x is adjacent to any of $\{u, u_1, \dots, u_4\}$.

If possible, assume that x is adjacent to some vertex from $\{u, u_1, \dots, u_4\}$. Since $d(u) = d(u_1) = d(u_2) = 4$, x is not adjacent to u , u_1 and u_2 . If x is adjacent to u_3 only, then the subgraph induced by $\{v_1, x, u_1, u_3, u_4\}$ is G_4 which is a contradiction with G being a line graph. If x is adjacent to both u_3 and u_4 then $L^{-1}(G)$ contains

- G_4 if x and v_1 are not adjacent and no vertex adjacent to both x and v_1 ,
- G_2 if x and v_1 are adjacent and no vertex adjacent to both x and v_1 ,
- G_7 if another vertex y is adjacent to both non-adjacent vertices v_1 and x ,
- if a vertex y is adjacent to both adjacent vertices v_1 and x , then it also has to be adjacent to all of u_1, u_2, u_3 and u_4 .

Therefore, in all above cases we obtain a contradiction as $L^{-1}(G) = L(L(H))$ is a line graph and $\Delta(G) = 4$. The proof of claim-2 is complete.

Claim-3: $v_1 - v_2 - \dots - v_n (n \geq 2)$ is a path then either $d(v_n) = 1$ or v_n is adjacent to two consecutive vertices of the 4-cycle of E_2 .

Assume that $d(v_n) \neq 1$. We prove that v_n is a part of E_2 . It is clear that $d(v_n) \leq 4$ as $\Delta(G) = 4$. Note that v_n is the first vertex on the path joined to v_1 such that $d(v_n) \geq 3$. If possible, assume that $d(v_n) = 4$ then as mentioned in the construction earlier, there is a vertex x such that $d(x) = 3$ and it appears prior to v_n on a path joining v_1 and v_n which is a contradiction. Hence, we obtain $d(v_n) \neq 4$. Assume that $d(v_n) = 3$. Let v_n be adjacent to two other vertices w_1 and w_2 . Since v_n is the first vertex with $d(v_n) = 3$ and G does not contain $K_{1,3}$, w_1 and w_2 are adjacent. Hence, v_n, w_1 and w_2 form a triangle. By Theorem 11, triangle v_n, w_1, w_2 is a part of $K_4 - e$. Now repeating the argument as earlier again, observe that the graph attached to v_n is same as one at v_1 .

It is clear from claims 1 to 3 that G is one of the graphs $L(L_{1,n})$ or $L(L_{2,n})$, where $n \geq 1$.

Sufficiency: Suppose G is isomorphic to $L(L_{k,n})$, where $k \in \{1, 2\}$ and $n \geq k$, then note that $G = L^3(H)$, where H is the graph obtained by identifying a pendant vertex of a claw $K_{1,3}$ with one end vertex of path P_{n+1} , $n \geq 1$.

(c) It is clear from (a) that if $G = L^3(H)$ and $\Delta(G) = 4$, then either G is $L(L_{1,n})$ or $L(L_{2,n})$. Since $\Delta(L^{n-1}(L(L_{i,n}))) \geq 5$ for $i \in \{1, 2\}$ and $n \geq 4$, $G \neq L^n(H)$ for any graph H .

Acknowledgement. The authors are grateful to the three anonymous referees for their careful reading of the manuscript and helpful comments.

References

1. Beineke, L.W.: Derived graphs and digraphs, Beitrage zur Graphentheorie, pp. 17–33. Teubner, Leipzig (1968)
2. Beineke, L.W.: Characterization of derived graphs. J. Comb. Theory **9**, 129–135 (1970)
3. Krausz, J.: Demonstration nouvelle d'une théorème de Whitney sur les réseaux. Mat. Fiz. Lapok **50**, 75–85 (1953)
4. van Rooij, A.C.M., Wilf, H.S.: The interchange graph of a finite graph. Acta Math. Acad. Sci. Hung. **16**, 263–269 (1965)
5. Roussopoulos, N.D.: A $\max\{m, n\}$ algorithm for determining the graph H from its line graph G . Inf. Process. Lett. **2**, 108–112 (1973)
6. Šoltés, L.: Forbidden induced subgraphs for line graphs. Disc. Math. **132**, 391–394 (1994)
7. Whitney, H.: Congruent graphs and the connectivity of graphs. Am. J. Math. **54**(1), 150–168 (1932)



Total Domination and Open Packing in Two Subclasses of Triangle-Free Graphs

M. A. Shalu and V. K. Kirubakaran

Indian Institute of Information Technology, Design and Manufacturing,
Kancheepuram, Chennai, India
shalu_mat19d002@iitdm.ac.in

Abstract. A vertex subset D of a graph $G(V, E)$ is called a total dominating set if every vertex in G has a neighbor in D , and a vertex subset S of G is called an open packing in G if no pair of distinct vertices in S have a common neighbor in G . The size of a minimum total dominating set (resp. maximum open packing) in G is called the total domination number (resp. open packing number) of G and is denoted by $\gamma_t(G)$ (resp. $\rho^o(G)$). Notably, the open packing number is a lower bound for the total domination number in graphs with no isolated vertices [Henning and Slater, 1999]. Given a graph G and a positive integer k , TOTAL DOMINATING SET problem decides whether $\gamma_t(G) \leq k$ and OPEN PACKING problem tests whether $\rho^o(G) \geq k$.

In this work, we study the complexity of the problems TOTAL DOMINATING SET and OPEN PACKING in two subclasses of K_3 -free graphs: $\{P_6, C_6, K_3\}$ -free graphs and triangle-free cubic planar graphs. We prove that in a connected $\{P_6, C_6, K_3\}$ -free graph, every minimum total dominating set is a biclique. We use this result to prove that (i) for a connected $\{P_6, C_6, K_3\}$ -free graph G , $\gamma_t(G) \leq \rho^o(G) + 1$ and (ii) a minimum total dominating set and a maximum open packing in a $\{P_6, C_6, K_3\}$ -free graph can be found in $O(n^4)$ time. We also show that for a connected $\{P_6, C_6, K_3\}$ -free graph G , $\gamma_t(G) = \rho^o(G)$ if and only if at least one vertex of G is not contained in any C_5 of G . In contrast to the above set of polynomial-time algorithm and structural characterization, we prove that TOTAL DOMINATING SET and OPEN PACKING are NP-complete on triangle-free cubic planar graphs.

Keywords: Total dominating set · Open packing · P_6 -free graphs · Cubic planar graphs · Triangle-free graphs

1 Introduction

Given a graph $G(V, E)$, a vertex subset D of G is called a *dominating set* if every vertex in $V(G) \setminus D$ has a neighbor in D . The dominating set problem is one of the classical problems in graph theory, and it finds applications in facility location problems, communication networks, and several other real-world problems [4, 8].

In addition to the dominating set problem, several variants of the dominating set problem have also been studied in the literature [2, 3, 6, 7]. In this work, we focus on a variant of dominating set problem, called total dominating set, and its dual problem, called open packing in graphs. The study of dual problems will always help us to design efficient algorithms for primal problems [25]. Some notable primal-dual graph problems are (i) max-flow and min-cut, (ii) vertex cover and matching, and (iii) edge cover and independent set.

A vertex subset D of a graph G is called a *total dominating set* if every vertex in G has a neighbor in D . The cardinality of a minimum total dominating set in G is called the *total domination number* of G and is denoted by $\gamma_t(G)$. A vertex subset S of a graph G is called an *open packing* in G if no pair of vertices in S have a common neighbor in G . The size of a maximum open packing in G is called the *open packing number*, $\rho^o(G)$, of G . Note that (i) if D is a total dominating set in G , then $|D \cap N_G(x)| \geq 1$ for every $x \in V(G)$, where $N_G(x)$ denotes the set of neighbors of x in G , and (ii) if S is an open packing in G , then $|S \cap N_G(x)| \leq 1$ for every vertex $x \in V(G)$. This validates the primal-dual relation between the problems total dominating set and open packing. Hence, for graphs without isolated vertices, $\gamma_t(G) \geq \rho^o(G)$ (graph with isolated vertices does not admit a total dominating set). The decision version of the problems total dominating set and open packing are as follows.

TOTAL DOMINATING SET
Instance : A graph G and a positive integer $k \leq V(G) $.
Question: Does G has a total dominating set of size at most k ?

OPEN PACKING
Instance : A graph G and a positive integer $k \leq V(G) $.
Question: Is there an open packing of size at least k in G ?

Total dominating set is one of the well-studied problems in the literature and an extensive list of results can be found in [8, 10, 11, 18]. Several bounds and complexity results of the open packing problem can be found in [9, 16, 19, 22]. In the context of our research, it is interesting to note that $\gamma_t(G) - \rho^o(G)$ can be arbitrarily large even for triangle-free graphs (see [1, 23]). Rall [17] proved that for a tree T with at least two vertices, $\gamma_t(T) = \rho^o(T)$. A recent work by Antony et al. [1] implies that this equality can be extended to chordal bipartite graphs without isolated vertices (also see [20]). Also, the complexity (P vs. NPC) of TOTAL DOMINATING SET and OPEN PACKING on H -free graphs are similar (i.e., either both the problems are in P or both are NPC) for every graph H with at least three vertices [21, 22]. Moreover, it is known that TOTAL DOMINATING SET and OPEN PACKING are NP-complete for split graphs (i.e., $\{2K_2, C_4, C_5\}$ -free graphs, a subclass of $\{P_6, C_6\}$ -free graphs) [5, 19] and 2-degenerate planar perfect elimination bipartite graphs of maximum degree three (a subclass of K_3 -free graphs) [20]. In this work, we study the complexity of TOTAL DOMINATING SET and OPEN PACKING in two subclasses of K_3 -free graphs, namely, (a) $\{P_6, C_6, K_3\}$ -free graphs and (b) triangle-free cubic planar graphs. The results

in this work are organized as follows. In Sect. 3, we show that for a connected $\{P_6, C_6, K_3\}$ -free graph G , (i) every optimal total dominating set is a biclique, (ii) $\gamma_t(G) \leq \rho^o(G) + 1$, (iii) $\gamma_t(G) = \rho^o(G)$ if and only if at least one vertex in G that is not contained in any C_5 in G , and (iv) a minimum total dominating set and a maximum open packing of G can be found in $O(n^4)$ time. In contrast to the polynomial-time algorithm for $\{P_6, C_6, K_3\}$ -free graphs, in Sect. 4, we show that TOTAL DOMINATING SET and OPEN PACKING are NP-complete for cubic planar bipartite graphs.

Proofs of Observations, Lemmas, Construction, Algorithm descriptions and Theorems in this work are provided in [23].

2 Preliminaries

We follow West [26] for terminology and notation. The graphs considered in this work are simple, unweighted, and undirected. Given a graph $G(V, E)$, let n and m denote the number of vertices and the number of edges in G , respectively. Every graph considered in this work consists of at least two vertices (i.e., $n \geq 2$). Given a vertex $x \in V(G)$, the (open) neighborhood of x in G is defined as $N_G(x) = \{y \in V(G) : xy \in E(G)\}$, and let the degree of a vertex x in G be $\deg_G(x) = |N_G(x)|$. We use $N(x)$ instead of $N_G(x)$ when there is no ambiguity on G . Given $U \subseteq V(G)$, the subgraph of G induced by U is denoted as $G[U]$. Given a graph H , G is said to be H -free if no induced subgraph of G is isomorphic to H . For a vertex $x \in V(G)$, let $E_G(x)$ denote the set of all edges incident on x , and for an edge $e \in E(G)$, let $V_G(e)$ denote the end vertices of e in G . Note that for $u \in V(G)$ and $e \in E(G)$, the edge $e \in E_G(u)$ if and only if $u \in V_G(e)$.

Given a total dominating set D in G , we say that a vertex $u \in V(G)$ is a *private neighbor* of some vertex $x \in D$ with respect to D if $N_G(u) \cap D = \{x\}$, i.e., x is the only neighbor of u in D . Given a vertex x and a vertex subset W of G , we say that x dominates W if $(W \setminus \{x\}) \subseteq N_G(x)$.

An edge subset M of a graph G is called a *matching* in G if $V_G(e) \cap V_G(e') = \emptyset$ for every pair of distinct $e, e' \in M$. The cardinality of a largest matching in G is called the *matching number*, $\alpha'(G)$, of G . An edge subset F of a graph G is called an *edge cover* in G if every vertex in G is an end vertex of some edge in F . The cardinality of a minimum edge cover in G is called the *edge cover number*, $\beta'(G)$, of G . A vertex subset I of G is called an *independent set* in G if no pair of vertices in I are adjacent in G . A vertex subset V_1 of G is called a *vertex cover* in G if every edge in the graph is incident on some vertex in V_1 in G .

Let P_n , C_n , and K_n denote the path, cycle, and complete graph on n vertices, respectively. Given two disjoint vertex subsets X and Y of a graph G , let $[X, Y]$ denote the set of edges in G with one end vertex in X and the other end vertex in Y , and we say that $[X, Y]$ is complete if each vertex in X is adjacent to every vertex in Y . A vertex subset W of G is called a *biclique* in G if there exists a partition of W into two non-empty independent sets X and Y such that $[X, Y]$ is complete.

A graph G is *bipartite* if there exists a partition of $V(G)$ into two independent sets X and Y , where $V(G) = X \cup Y$ is called a *bipartition* of G . A bipartite graph G is said to be a *complete bipartite graph* if $V(G)$ is a biclique.

Given a graph G and a positive integer $k \leq |V(G)|$, (i) the problem INDEPENDENT SET decides whether G has an independent set of size at least k and (ii) the VERTEX COVER problem checks whether G has a vertex cover of size at most k in G .

3 $\{P_6, C_6, K_3\}$ -free Graphs

This section presents an $O(n^4)$ -time algorithm to find an optimal total dominating set and a maximum open packing in $\{P_6, C_6, K_3\}$ -free graphs. The interest in the complexity of TOTAL DOMINATING SET and OPEN PACKING in the class of $\{P_6, C_6, K_3\}$ -free graphs is motivated by the following theorem by Liu and Zhou [13], which connects the structure of a dominating set in a subclass of triangle-free graphs with $\{P_6, C_6\}$ -free graphs.

Theorem 1 ([13]). *Let G be a triangle-free graph. Then, G is $\{P_6, C_6\}$ -free if and only if every connected induced subgraph of G has a dominating complete bipartite subgraph.*

The above theorem also forms a basic block in our study on the class of $\{P_6, C_6, K_3\}$ -free graphs. In addition to the $O(n^4)$ -time algorithm to find the total domination number and the open packing number of $\{P_6, C_6, K_3\}$ -free graphs, we also show that for a connected $\{P_6, C_6, K_3\}$ -free graph G , $\gamma_t(G) - \rho^o(G) \leq 1$. Note that C_5 is the only possible odd cycle in $\{P_6, C_6, K_3\}$ -free graphs. We prove that the total domination number and the open packing number are equal in a connected $\{P_6, C_6, K_3\}$ -free graph if and only if there exists a vertex in G that is not contained in any C_5 of G . The following observation about the set of private neighbors of the vertices in a total dominating set plays a prominent role in our proofs.

Observation 1. *Let D be a total dominating set of a graph G . Then, D is a minimal total dominating set if and only if every vertex in D has a private neighbor with respect to D in G .*

Next, we characterize the structure of minimum total dominating sets in $\{P_6, C_6, K_3\}$ -free graphs using the following lemma.

Lemma 1. *Let G be a connected $\{P_6, C_6, K_3\}$ -free graph. Then, every minimum total dominating set in G is a biclique.*

Corollary 1. *Let G be a connected $\{P_6, C_6, K_3\}$ -free graph. Then, there exists a dominating biclique $X_1 \cup Y_1$ with $X_1 \neq \emptyset$ and $Y_1 \neq \emptyset$.*

Corollary 1 follows from Lemma 1.

Throughout this section, if we say that a graph G is a connected $\{P_6, C_6, K_3\}$ -free graph, then we assume that G has a dominating biclique $D = X_1 \cup Y_1$ with

$X_1 \neq \emptyset$ and $Y_1 \neq \emptyset$. Also, we assume that $X = \bigcup_{y \in Y_1} N(y)$ and $Y = \bigcup_{x \in X_1} N(x)$. Since $X_1 \cup Y_1$ is a dominating set of G , $V(G) = X \cup Y$. The following observation shows that $X \cup Y$ is a partition of $V(G)$.

Observation 2. Let G be a connected $\{P_6, C_6, K_3\}$ -free graph, and let $X = \bigcup_{y \in Y_1} N(y)$ and $Y = \bigcup_{x \in X_1} N(x)$. Then, $X \cap Y$ is empty.

The following observation proves the disconnectedness between the vertices in X_1 (resp. Y_1) and $X \setminus X_1$ (resp. $Y \setminus Y_1$). This observation motivates our study about the structure of $X \setminus X_1$ and $Y \setminus Y_1$, presented as Lemma 2.

Observation 3. Let G be a connected $\{P_6, C_6, K_3\}$ -free graph. Then, $[X_1, X \setminus X_1] = \emptyset$ and $[Y_1, Y \setminus Y_1] = \emptyset$.

The lemma below provides a structural characterization for $G[X \setminus X_1]$ and $G[Y \setminus Y_1]$.

Lemma 2. Let G be a connected $\{P_6, C_6, K_3\}$ -free graph. Then, $G[X \setminus X_1]$ and $G[Y \setminus Y_1]$ are $2K_2$ -free bipartite graphs.

The following definition about the partition of $X \setminus X_1$ and $Y \setminus Y_1$ into independent sets, based on Lemma 2, helps us in proving the bound between the total domination number and the open packing number in $\{P_6, C_6, K_3\}$ -free graphs.

Definition 1. Let G be a connected $\{P_6, C_6, K_3\}$ -free graph. We define I_1 (resp. J_1) to be the set of all isolated vertices in $G[X \setminus X_1]$ (resp. $G[Y \setminus Y_1]$).

Let X_2 be the set of all vertices in $X \setminus X_1$ but not in I_1 , i.e., $X_2 = ((X \setminus X_1) \setminus I_1)$. Then, since $G[X \setminus X_1]$ is $2K_2$ -free bipartite and every isolated vertex in $X \setminus X_1$ is in I_1 , the graph induced by X_2 is connected and bipartite. Let $X_2 = I_2 \cup I_3$ be the bipartition of $((X \setminus X_1) \setminus I_1)$.

Similarly, there exists a partition of $((Y \setminus Y_1) \setminus J_1)$ into two independent sets, J_2 and J_3 . Also, by the choice of J_1 and the fact that $G[Y \setminus Y_1]$ is $2K_2$ -free bipartite, the graph $G[J_2 \cup J_3]$ is connected (see Fig. 1).

Next, we define a property based on the partition of $X \setminus X_1$ and $Y \setminus Y_1$ in Definition 1. We use this property as the bridge to prove that the following classes are equivalent: (i) $\{P_6, C_6, K_3\}$ -free graphs with total domination number and open packing number equal and (ii) $\{P_6, C_6, K_3\}$ -free graphs in which every graph G consists of a vertex that is not contained in any C_5 in G .

Property 1: Let G be a connected $\{P_6, C_6, K_3\}$ -free graph. We say that G satisfies Property 1 if the following are true.

- (a) There exists a biclique $D \subseteq V(G)$ with $|D| = 3$, i.e., D has a bipartition $X_1 \cup Y_1$ such that $X_1 = \{x\}$ and $Y_1 = \{y_1, y_2\}$, and D dominates $V(G)$,
- (b) for the partition of $V(G) = X_1 \cup Y_1 \cup I_1 \cup I_2 \cup I_3 \cup J_1 \cup J_2 \cup J_3$ as in Definition 1 with respect to the dominating biclique $X_1 \cup Y_1$, the set $I_2 \cup I_3$ is non-empty, and

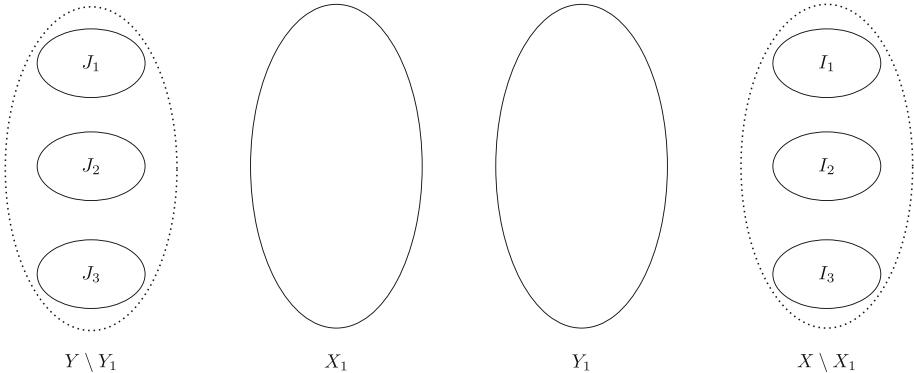


Fig. 1. Schematic representation of a $\{P_6, C_6, K_3\}$ -free graph with a dominating biclique $X \cup Y$. For further details refer to Definition 1.

(c) every vertex in $V(G) \setminus (N(y_1) \cap N(y_2))$ is adjacent to some vertex in $I_2 \cup I_3$.

Theorem 2. Let G be a connected $\{P_6, C_6, K_3\}$ -free graph. Then, $\gamma_t(G) \leq \rho^o(G) + 1$. Also, the following are equivalent.

- (1) G satisfies Property 1.
- (2) For every $u \in V(G)$, there exists $U \subseteq V(G)$ such that $u \in U$ and $G[U] \cong C_5$.
- (3) $\gamma_t(G) = \rho^o(G) + 1$.

Note that Theorem 2 characterizes the class of $\{P_6, C_6, K_3\}$ -free graphs whose total domination number and open packing number are equal and proves that for a connected $\{P_6, C_6, K_3\}$ -free graph G , $\gamma_t(G) \leq \rho^o(G) + 1$.

Remark 1. The total domination number and the open packing number of a connected $\{P_6, C_6, K_3\}$ -free graphs can be arbitrarily large (see Fig. 2).

Next, we study the complexity of finding $\gamma_t(G)$ and $\rho^o(G)$.

Given a graph G with a dominating set D , and vertices $x, y \in D$ such that $xy \in E(G)$ and $\{x, y\}$ dominates D , van 't Hof and Paulusma [24] gave an $O(n^2)$ -time algorithm to find a minimal dominating set D' of G such that $D' \subseteq D$ and $x, y \in D'$. We present a modified version of van 't Hof and Paulusma's algorithm in Lemma 3, which, when given a graph G with a total dominating set D as input, will output a minimal total dominating set D' of G such that $D' \subseteq D$.

Lemma 3. Let D be a total dominating set in graph G . Then, a minimal total dominating set $D' \subseteq D$ in G and the set of private neighbors for every vertex $x \in D'$ with respect to D' can be found in $O(n^2)$ time.

The following theorem by van 't Hof and Paulusma [24] helps us initialize our algorithm to find an optimal total dominating set and a maximum open packing of a connected $\{P_6, C_6, K_3\}$ -free graph.

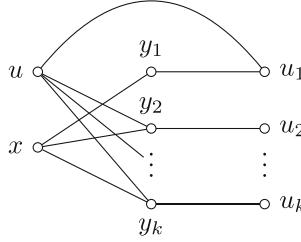


Fig. 2. A connected $\{P_6, C_6, K_3\}$ -free graph G with $X_1 = \{x\}$, $Y_1 = \{y_1, y_2, \dots, y_k\}$, $I_1 = \{u_2, u_3, \dots, u_k\}$, $I_2 = \{u_1\}$, $I_3 = \{u\}$, and $Y \setminus Y_1 = \emptyset$. Note that $\gamma_t(G) = \rho^o(G) = k + 1$, where $D = \{x, y_1, y_2, \dots, y_k\}$ is a total dominating set in G (for $k \geq 2$) and $S = \{y_1, u_1, u_2, \dots, u_k\}$ is an open packing in G .

Theorem 3 ([24]). *A connected graph G is P_6 -free if and only if each connected induced subgraph of G contains a dominating induced C_6 or a dominating (need not be induced) complete bipartite graph. Moreover, such a dominating subgraph of G can be found in $O(n^3)$ time.*

Note that for a connected $\{P_6, C_6, K_3\}$ -free graph G , the dominating set output by the above theorem will eventually be a biclique.

Next, we give the outline of the proposed $O(n^4)$ -time algorithm to find an optimal total dominating set and a maximum open packing in a connected $\{P_6, C_6, K_3\}$ -free graph. The following lemma and observation are useful in understanding the course and the complexity of the algorithm.

Lemma 4 ([20]). *Given a graph G and a vertex subset S of G , it can be tested in $O(n)$ time whether S is an open packing in G or there exist $u_1, u_2 \in S$ and $v \in V(G)$ such that $v \in N_G(u_1) \cap N_G(u_2)$.*

Observation 4. *Let G be a connected $\{P_6, C_6, K_3\}$ -free graph, and let $D = X_1 \cup Y_1$ be a minimal dominating biclique in G such that $I_2 \cup I_3 \neq \emptyset$. Then, there exists $i \in \{2, 3\}$ and exactly one vertex $y \in Y_1$ such that I_i is the set of all private neighbors of y with respect to D .*

The vertex y_1 in Fig. 2 is an example for Observation 4.

Proof of Theorem 2, given in [23], will help us understand the structure of the following outline.

Outline of the Proposed Algorithm:

- Set $i = 0$, and find a dominating biclique D'_1 of G using Theorem 3.
- Set $i := i + 1$, and apply Lemma 3 to find a minimal dominating biclique $D_i \subseteq D'_i$ of G and the set of private neighbours P_z for every vertex $z \in D_i$ with respect to D_i .
- Find a bipartition $X_1 \cup Y_1$ of D_i . Construct sets I_1, I_2, I_3, J_1, J_2 , and J_3 as in Definition 1 with respect to D_i .

- (d) If $I_2 \cup I_3 \neq \emptyset$ and $J_2 \cup J_3 \neq \emptyset$, then find a set S consisting of exactly one private neighbor u_z of every vertex in $z \in D_i$ such that $u_z \in I_1 \cup J_1$ whenever possible. If S is an open packing in G , then output (D_i, S) (i.e., D_i is a minimum total dominating set and S is maximum open packing) and terminate. Otherwise, do Step (e).
- (e) Choose $u_{z_1}, u_{z_2} \in S$ and a vertex $v \in N(u_{z_1}) \cap N(u_{z_2})$, set $D'_{i+1} = (D_i \setminus \{z_1, z_2\}) \cup \{v\}$ and go to Step (b) (Lemma 4 can be used to do this part).
- (f) If $I_2 \cup I_3$ and $J_2 \cup J_3$ are empty. Create a set S consisting of exactly one private neighbor u_z of every vertex $z \in D_i$. If S is an open packing in G , then output (D_i, S) and terminate. Otherwise, do Step (e).
- (g) If $I_2 \cup I_3 \neq \emptyset$ and $J_2 \cup J_3 = \emptyset$, then
 1. If $|X_1| = 1$ and $|Y_1| = 2$, then test every three-sized subset of S , to find whether an open packing of size three exists in G . If an open packing S of size three exists, then output (D_i, S) . Otherwise, choose an edge $uv \in E(G)$, set $S_1 = \{u, v\}$, output (D_i, S_1) , and terminate.
 2. If $|X_1| = 1$ and $|Y_1| > 2$, then construct a set S_Y consisting of exactly one private neighbor u_z of every vertex $z \in Y_1$. If S_Y is an open packing in G , then find a vertex $y \in Y_1$ such that I_j is the set of private neighbors of y for $j = 2$ or 3 (see Observation 4), set $S = \{y\} \cup S_Y$, output (D_i, S) , and terminate. Otherwise, do Step (e) on S_Y .
 3. If $|X_1| > 1$ and $|Y_1| = 2$, then construct a set S_X consisting of exactly one private neighbor u_z of every vertex $z \in X_1$. If S_X is an open packing in G , choose an edge uv in $G[I_2 \cup I_3]$, set $S = \{u, v\} \cup S_X$, output (D_i, S) , and terminate. Otherwise, do Step (e) on S_X .
 4. If $|X_1| > 1$ and $|Y_1| > 2$, then construct two sets S_X and S_Y in G such that S_X (resp. S_Y) consists of exactly one private neighbor u_z of every vertex z in X_1 (resp. Y_1). If S_X and S_Y are open packing in G , then set $S = S_X \cup S_Y$, output (D_i, S) , and terminate. Otherwise, if S_X (resp. S_Y) is not an open packing in G , do Step (e) on S_X (resp. S_Y).
- (h) If $I_2 \cup I_3 = \emptyset$ and $J_2 \cup J_3 \neq \emptyset$, then swap X_1 and Y_1 , and do Step (g).

The complete description of the algorithm is given in [23]. Based on the algorithm proposed, we have the following theorem.

Theorem 4. *An optimal total dominating set and a maximum open packing of a connected $\{P_6, C_6, K_3\}$ -free graph can be found in $O(n^4)$ time.*

The above theorem completes our study on the class of $\{P_6, C_6, K_3\}$ -free graphs. Next, we prove the hardness results for triangle-free cubic planar graphs.

4 Triangle-Free Cubic Planar Graphs

In this section, we prove that TOTAL DOMINATING SET and OPEN PACKING are NP-complete for triangle-free cubic planar graphs. Since VERTEX COVER and INDEPENDENT SET are NP-complete for cubic planar graphs [15], the following construction helps us prove the desired results.

Construction 1.

Input: A cubic planar graph $G(V, E)$, where $V = \{u_1, u_2, \dots, u_n\}$ and $E = \{e_1, e_2, \dots, e_m\}$.

Output: A triangle-free cubic planar graph G' .

Guarantee 1: G has a vertex cover of size k if and only if G' has a total dominating set of size $(k + \beta'(G) + 6m)$.

Guarantee 2: G has an independent set of size k if and only if G' has an open packing of size $(k + \alpha'(G) + 5m)$.

Procedure:

Step 1: Subdivide each edge $e = uu'$ of G into a three-vertex path ueu' in G' .

Step 2: For each edge e_i in G , introduce a subgraph H_i isomorphic to the graph H shown in Fig. 3 with $V(H_i) = \{x_{i1}, x_{i2}, x_{i3}, x_{i4}, y_{i1}, y_{i2}, y_{i3}, y_{i4}, w_{i1}, w_{i2}, w_{i3}, w_{i4}, w_{i5}, w_{i6}, w_{i7}, w_{i8}, z_i\}$.

Step 3: For every $i \in \{1, 2, \dots, m\}$, add an edge $e_i z_i$ in G' .

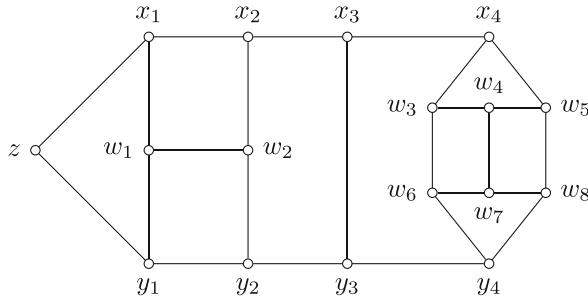


Fig. 3. The edge gadget H used in Construction 1.

An example of Construction 1 is given in Fig. 4. Note that $V(G') = V(G) \cup E(G) \cup (\bigcup_{i=1}^m V(H_i))$ and $E(G') = \{ue : u \in V(G), e \in E(G), \text{ and } u \in V_G(e)\} \cup \{e_i z_i : 1 \leq i \leq m\} \cup (\bigcup_{i=1}^m E(H_i))$. Hence, $|V(G')| = n + m + 17m = n + 18m$ and $|E(G')| = 2m + m + 25m = 28m$. Also, since $\beta'(G)$ and $\alpha'(G)$ can be found in $O(m\sqrt{n})$ time [12, 14], the instances $(G', k + \beta'(G) + 6m)$ and $(G', k + \alpha'(G) + 5m)$ can be constructed in $O(m\sqrt{n})$ time. Hence, the reduction given in Construction 1 is polynomial-time computable.

Next, we prove that the output graph G' of Construction 1 is a triangle-free cubic planar graph. The graph G' is cubic because (i) $\deg_{G'}(u) = \deg_G(u) = 3$ for every $u \in V(G)$, (ii) $\deg_{G'}(e_i) = |V_G(e_i) \cup \{z_i\}| = 3$ for every $e_i \in E(G)$, (iii) for $1 \leq i \leq m$, $\deg_{G'}(v) = \deg_{H_i}(v) = 3$ for every $v \in V(H_i) \setminus \{z_i\}$, (iv) $\deg_{G'}(z_i) = \deg_{H_i}(z_i) + |\{e_i\}| = 3$ for every $i \in \{1, 2, \dots, m\}$, and (v) $V(G') = V(G) \cup E(G) \cup (\bigcup_{i=1}^m V(H_i))$. The graph G' is planar as it is obtained by subdividing each edge of a planar graph and then attaching an edge gadget H_i (isomorphic to the planar graph H in Fig. 3) to each newly formed vertex e_i . The following claim proves that G' is triangle-free.

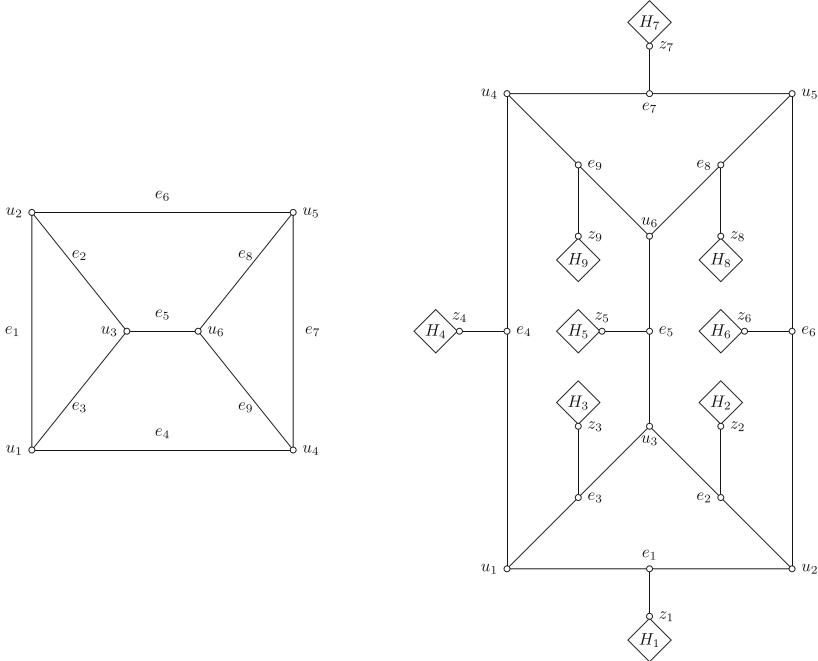


Fig. 4. (a) A cubic planar graph G and (b) a triangle-free cubic planar graph G' constructed from G using Construction 1, where A_i denotes the graph isomorphic to A in Fig. 3 for $1 \leq i \leq m$.

Claim 1. G' is a triangle-free graph.

Proof of Claim 1. Note that $[V(H_i), V(H_j)] = \emptyset$ for every $1 \leq i, j \leq m$ with $i \neq j$, and the edge $e_i z_i$ that connects $V(G) \cup E(G)$ with $V(H_i)$ is a cut edge in G' for every $i \in \{1, 2, \dots, m\}$. So, if there exists a vertex subset U in G' such that $G'[U] \cong K_3$, then either $U \subseteq V(G) \cup E(G)$ or $U \subseteq V(H_i)$ for some $i \in \{1, 2, \dots, m\}$. Note that $V(G) \cup E(G)$ induces a bipartite subgraph of G' . So for any vertex subset U that induces a triangle in G' , $U \subseteq V(H_i)$ for some $1 \leq i \leq m$. However, it is evident from Fig. 3 that H_i is triangle-free for $1 \leq i \leq m$. Hence, G' is triangle-free.

Since TOTAL DOMINATING SET and OPEN PACKING are in NP for every subclass of simple graphs, we have the following theorems by Construction 1 and the facts that VERTEX COVER and INDEPENDENT SET are NP-complete for cubic planar graphs [15].

Theorem 5. TOTAL DOMINATING SET is NP-complete on triangle-free cubic planar graphs.

Theorem 6. OPEN PACKING is NP-complete on triangle-free cubic planar graphs.

5 Conclusion

In this work, we have shown that TOTAL DOMINATING SET and OPEN PACKING are $O(n^4)$ -time solvable in $\{P_6, C_6, K_3\}$ -free graphs. To prove the polynomial-time result, we proved that in a connected $\{P_6, C_6, K_3\}$ -free graph G , (i) every optimal total dominating set is a biclique and (ii) $\gamma_t(G) \leq \rho^o(G) + 1$. Based on (i) and (ii), we propose the following questions: (I) characterize the hereditary class of graphs, say \mathcal{H} , such that for every connected graph G in \mathcal{H} , every optimal total dominating set in G is a biclique and (II) given a graph class \mathcal{G} , analogous to χ -binding functions, does there exist a function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that $\gamma_t(G) \leq f(\rho^o(G))$?

It is known that, for $r \geq 3$, TOTAL DOMINATING SET is NP-complete for r -regular triangle-free graphs [1]. We proved that TOTAL DOMINATING SET and OPEN PACKING are NP-complete on triangle-free cubic planar graphs. So, the question of interest would be the complexity of OPEN PACKING on r -regular triangle-free graphs for $r \geq 4$.

References

1. Antony, D., Chandran, L.S., Gayen, A., Gosavi, S., Jacob, D.: Total domination, separated-cluster, cd-coloring: algorithms and hardness. In: Soto, J.A., Wiese, A. (eds.) LATIN 2024: Theoretical Informatics, pp. 97–113. Springer, Heidelberg (2024). https://doi.org/10.1007/978-3-031-55598-5_7
2. Camby, E., Schaudt, O.: A new characterization of P_k -free graphs. Algorithmica **75**, 2015–217 (2016). <https://doi.org/10.1007/s00453-015-9989-6>
3. Cockayne, E.J., Dawes, R.M., Hedetniemi, S.T.: Total domination in graphs. Networks **10**(3), 211–219 (1980). <https://doi.org/10.1002/net.3230100304>
4. Corcoran, P., Gagarin, A.: Heuristics for k-domination models of facility location problems in street networks. Comput. Oper. Res. **133**, 105368 (2021). <https://doi.org/10.1016/j.cor.2021.105368>
5. Corneil, D.G., Perl, Y.: Clustering and domination in perfect graphs. Disc. Appl. Math. **9**(1), 27–39 (1984). [https://doi.org/10.1016/0166-218X\(84\)90088-X](https://doi.org/10.1016/0166-218X(84)90088-X)
6. Farber, M.: Domination, independent domination, and duality in strongly chordal graphs. Disc. Appl. Math. **7**(2), 115–130 (1984). [https://doi.org/10.1016/0166-218X\(84\)90061-1](https://doi.org/10.1016/0166-218X(84)90061-1)
7. Fitzpatrick, S., Hartnell, B.: Paired-domination. In: Discussiones Mathematicae, pp. 63–72 (1998)
8. Haynes, T.W., Hedetniemi, S., Slater, P.: Fundamentals of Domination in Graphs, 1st edn. CRC Press, Boca Raton (1998). <https://doi.org/10.1201/9781482246582>
9. Henning, M.A.: Packing in trees. Disc. Math. **186**(1), 145–155 (1998). [https://doi.org/10.1016/S0012-365X\(97\)00228-8](https://doi.org/10.1016/S0012-365X(97)00228-8)
10. Henning, M.A., Yeo, A.: Total Domination in Graphs. Springer, Heidelberg (2015). <https://doi.org/10.1007/978-1-4614-6525-6>
11. Laskar, R., Pfaff, J., Hedetniemi, S.M., Hedetniemi, S.T.: On the algorithmic complexity of total domination. SIAM J. Algeb. Disc. Methods **5**(3), 420–425 (1984). <https://doi.org/10.1137/0605040>
12. Lawler, E.: Combinatorial Optimization: Networks and Matroids. Dover Publications, Mineola (2001)

13. Liu, J., Zhou, H.: Dominating subgraphs in graphs with some forbidden structures. *Disc. Math.* **135**, 163–168 (1994). [https://doi.org/10.1016/0012-365X\(93\)E0111-G](https://doi.org/10.1016/0012-365X(93)E0111-G)
14. 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 (1980). <https://doi.org/10.1109/SFCS.1980.12>
15. Mohar, B.: Face covers and the genus problem for apex graphs. *J. Comb. Theory, Ser. B* **82**(1), 102–117 (2001). <https://doi.org/10.1006/jctb.2000.2026>
16. Mojdeh, D.A., Peterin, I., Samadi, B., Yero, I.G.: (Open) packing number of some graph products. *Disc. Math. Theor. Comput. Sci.* **22** (2020). <https://doi.org/10.23638/DMTCS-22-4-1>
17. Rall, D.F.: Total domination in categorical products of graphs. *Discussiones Mathematicae Graph Theory* **25**, 35–44 (2005). <https://doi.org/10.7151/dmgt.1257>
18. Ramalingam, G., Rangan, C.P.: A unified approach to domination problems on interval graphs. *Inf. Process. Lett.* **27**(5), 271–274 (1988). [https://doi.org/10.1016/0020-0190\(88\)90091-9](https://doi.org/10.1016/0020-0190(88)90091-9)
19. Ramos, I., Santos, V.F., Szwarcfiter, J.L.: Complexity aspects of the computation of the rank of a graph. *Disc. Math. Theor. Comput. Sci.* **16** (2014). <https://doi.org/10.46298/dmtcs.2075>
20. Shalu, M.A., Kirubakaran, V.K.: Total domination and open packing in some subclasses of bipartite graphs. *Manuscript-UnderReview* (2023)
21. Shalu, M.A., Kirubakaran, V.K.: Open packing in graphs: bounds and complexity (2024). <https://arxiv.org/abs/2406.06982>
22. Shalu, M.A., Kirubakaran, V.K.: Open packing in H-free graphs and subclasses of split graphs. In: Kalyanasundaram, S., Maheshwari, A. (eds.) *Algorithms and Discrete Applied Mathematics*, pp. 239–251. Springer, Heidelberg (2024). https://doi.org/10.1007/978-3-031-52213-0_17
23. Shalu, M.A., Kirubakaran, V.K.: Supplementary material of total domination and open packing in two subclasses of triangle-free graphs. *Researchgate* (2024). <https://doi.org/10.13140/RG.2.2.16301.09448>
24. van 't Hof, P., Paulusma, D.: A new characterization of P_6 -free graphs. *Disc. Appl. Math.* **158**(7), 731–740 (2010). <https://doi.org/10.1016/j.dam.2008.08.025>
25. Vazirani, V.V.: *Approximation Algorithms*. Springer, Heidelberg (2013). <https://doi.org/10.1007/978-3-662-04565-7>
26. West, D.B.: *Introduction to Graph Theory*, 2nd edn. Pearson, Boston (2018)



The MASEMPR Problem and Its Applications in Logistics

A. Subramani¹, K. Subramani^{2(✉)}, Piotr Wojciechowski²,
and Sangram K. Jena³

¹ Candorcare, Morgantown, WV, USA
anand.subramani@candorcare.org

² LDCSEE, West Virginia University, Morgantown, WV, USA
{k.subramani,pwojciec}@mail.wvu.edu

³ DCSSE, Miami University, Oxford, OH, USA
jenas2@miamioh.edu

Abstract. In this paper, we propose a new problem called the Maximum cardinality Acyclic Subset of Edges that Meets the Potential Requirements (MASEMPR). This problem abstracts several problems in logistics. Our focus is on a particular logistics problem called the Maximum Path Set (MPS) problem. We show that the MASEMPR problem is **NP-hard**. We then propose an exact exponential algorithm for the MASEMPR problem in subcubic graphs (graphs in which each vertex has degree at most 3). Finally, we design a constant factor approximation algorithm for the MASEMPR problem in general graphs.

1 Introduction

In this paper, we introduce the MASEMPR problem in graphs and discuss exact exponential algorithms for this problem specialized to various graph classes. As argued in Sect. 3, the MASEMPR problem has applications in several domains, such as logistics, network reliability, and traffic management. The problem is a generalization of the Maximum Path Set (MPS) problem. The problem of finding the edge complement of a Maximum Path Set is called the Minimum Co-Path Set (MCPS) problem. The MCPS problem has been studied extensively within the research community. It holds significant importance in biology, particularly in the domain of RH mapping [3, 4]. It has been established that the decision version of the MCPS problem ($MCPS_D$) is **NP-complete** in general graphs [3]. Additionally, it is known that the $MCPS_D$ problem remains **NP-complete** even when restricted to cubic graphs [5]. A $\frac{10}{7}$ -approximation algorithm for the MCPS problem, running in time $O(n^{1.5})$, was also presented in [2]. Furthermore, they demonstrated that the MCPS problem is **APX-complete** [2].

The rest of this paper is organized as follows: In Sect. 2, we formally describe the problems under consideration in this paper. The motivation for our work is discussed in Sect. 3. In Sect. 4, we discuss an exact exponential algorithm for the MASEMPR problem in subcubic graphs when the input potentials are all 2.

An approximation algorithm for the MASEMPR problem is detailed in Sect. 5. We conclude in Sect. 6, by summarizing our work in this paper and identifying avenues for future research.

2 Statement of Problems

In this section, we formally define the problems under consideration in this paper.

Consider the graph $G = (V, E)$. Associated with G is a potential function $p : V \rightarrow \{0, 1, 2\}$. Let n denote the number of vertices in V and m denote the number of edges in E . Consider a subset of edges S . The induced subgraph (V, S) of S is denoted as G_S . We also have a function $A : (2^E, V) \rightarrow \mathbb{Z}^+$, which denotes the degree of a specific vertex in G_S . A subset of edges S meets the potential requirements if, for all vertices $v_i \in V$, $A(S, v_i) \leq p(v_i)$. For instance, the subset of edges highlighted in red in Fig. 1 does not meet the potential requirements.

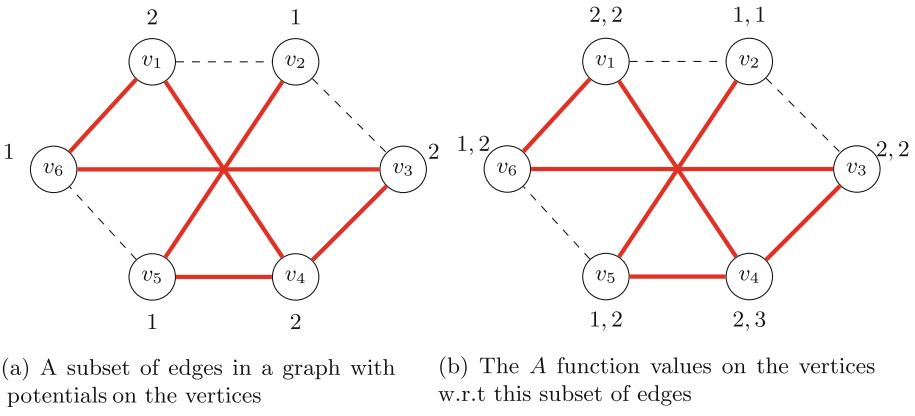


Fig. 1. A subset of edges that does not meet the potential requirements

Definition 1 (The MASEMPR problem). *Given a graph $G = (V, E)$ and a potential function $p : V \rightarrow \{0, 1, 2\}$, find a Maximum cardinality Acyclic Subset of Edges that Meets the Potential Requirements, i.e., a MASEMPR.*

For our approximation algorithm, we also need to define the following problem.

Definition 2 (The MEMPR problem). *Given a graph $G = (V, E)$ and a potential function $p : V \rightarrow \{0, 1, 2\}$, find a Maximum cardinality subset of Edges that Meets the Potential Requirements, i.e., a MEMPR.*

An arbitrary acyclic subset of edges that meets the potential requirements is called an ASEMPR. Similarly, an arbitrary subset of edges that meets the potential requirements is called an EMPR. The decision version of the MASEMPR problem is defined below.

Definition 3 (The MASEMPR_D problem). *Given a graph $G = (V, E)$, a potential function $p : V \rightarrow \{0, 1, 2\}$, and a number k , is there an ASEMPR of size $\geq k$?*

Our exact exponential algorithm solves the MASEMPR problem in subcubic graphs.

A path set in a graph is a subset of edges whose induced subgraph consists of a collection of vertex disjoint simple paths. We now describe the Maximum Path Set (MPS) problem.

Definition 4 (The MPS problem). *Given a graph $G = (V, E)$, find a Maximum cardinality Path Set, i.e., an MPS.*

We also consider the decision version of the problem.

Definition 5 (The MPS_D problem). *Given a graph $G = (V, E)$ and a number k , is there a path set of size $\geq k$?*

The Minimum Co-Path Set problem is based on finding the edge complement of the maximum path set. A co-path set is a set of edges whose removal results in a path set.

Definition 6 (The MCPS problem). *Given a graph $G = (V, E)$, find a Minimum cardinality Co-Path Set, i.e., an MCPS.*

We also consider the decision version of the problem.

Definition 7 (The MCPS_D problem). *Given a graph $G = (V, E)$ and a number k , is there a co-path set of size $\leq k$?*

The MCPS_D problem is known to be **NP-complete** [3]. It follows that the MPS_D problem is also **NP-complete**, as it is trivially in **NP** and there is a co-path set of size $\leq k$ in a graph, if and only if there is a path set of size $\geq m - k$. However, proving that the MASEMPR_D problem is **NP-complete** is not as trivial.

Theorem 1. *The MASEMPR_D problem is **NP-complete** when all potentials are 2.*

Proof. Consider an instance of the MPS_D problem. This instance consists of a graph $G = (V, E)$ and a number k . From the MPS_D instance we construct an instance of the MASEMPR_D problem. In the constructed instance we keep G and k the same and set the potential of every vertex to 2.

($\Rightarrow:$) Consider a path set of size $\geq k$. To make an ASEMPR of size $\geq k$, we simply use the same set of edges. By definition, a path set is acyclic. Furthermore, there can clearly exist no degree three vertices, as those could not be part of a path. Thus, if there exists a path set of size $\geq k$, then there exists an ASEMPR of size $\geq k$ in the new instance.

(\Leftarrow) Consider an ASEMPR of size $\geq k$. We show that this is a path set of size $\geq k$. Assume that the ASEMPR is not a path set. Thus, there must exist a vertex of degree > 2 or a cycle. By definition, ASEMPRs are acyclic. Furthermore, since all of the potentials are set to 2, the ASEMPR must have no vertex of degree > 2 . Hence, this contradicts the assumption that the ASEMPR is not a path set.. Thus, if there exists an ASEMPR of size $\geq k$ in the new instance, then there exists a path set of size $\geq k$.

This completes the reduction from the MPS_D problem to the MASEMPR_D problem and, since the MPS_D problem is **NP-complete**, the MASEMPR_D problem is **NP-hard**. The MASEMPR_D problem is clearly in **NP**. Thus, the MASEMPR_D problem is **NP-complete**. \square

Since this was a reduction from the MCPS_D problem that preserved the graph, it follows that the MASEMPR_D problem is **NP-complete** in any graph class where the MCPS_D problem is **NP-complete**. It is known that the MCPS_D problem is **NP-complete** in cubic graphs [5], and thus so is the MASEMPR_D problem.

The principal contributions of this paper are as follows:

1. An $O^*(1.4656^n)$ algorithm for the MASEMPR problem in subcubic graphs with all input potentials being 2, i.e., an algorithm for MPS.
2. A bounded error approximation algorithm for the MASEMPR problem in arbitrary graphs.

3 Motivation

In this section, we discuss the motivation behind considering the MASEMPR problem. The MASEMPR problem has significant applications in network reliability and traffic management [1,8]. One such application involves ensuring efficient and safe routing in communication networks, particularly in scenarios where network stability and load balancing are crucial. For example, consider a large-scale distributed system where data packets need to be transmitted across multiple nodes without overloading any single node beyond its capacity.

In this context, each node represents a vertex in a graph, and the edges denote communication links. The potential function p maps each vertex to its maximum allowable degree, reflecting its capacity to handle concurrent connections. The function A represents the degree of a vertex in a subgraph induced by a subset of edges, illustrating the current load on that node.

To prevent network overloads, it is essential to ensure that the degree of each vertex in any selected subgraph does not exceed its potential as defined by p . This constraint ensures that no node is overwhelmed, maintaining network reliability, and preventing communication failures.

The MASEMPR problem, which seeks to find the maximum cardinality acyclic subset of edges that meets these potential requirements, is crucial for optimizing network performance. By identifying the largest set of non-overlapping,

feasible communication paths, we can maximize data flow efficiency while adhering to node capacity constraints. This optimization directly translates to more robust and reliable network operations.

4 An Exact Exponential Algorithm for the MASEMPR Problem in Subcubic Graphs

In this section, we design and analyze an exact exponential algorithm for the MASEMPR problem in subcubic graphs with all input potentials 2 that runs in time $O^*(1.4656^n)$. Note that, as per the discussion in Sect. 2, this algorithm solves MPS.

4.1 Algorithm for Maximum Degree Two Graphs

In this section, we describe the sub-procedure necessary for our algorithm when there exist no vertices of degree three.

Note that the MEMPR problem is also known in the literature as the Degree Constrained Subgraph (DCS) problem. This problem has upper and lower bounds on the degree of each vertex, but we can set the lower bounds to zero and the upper bounds to the potentials. Furthermore, the DCS problem has known polynomial time algorithms. One example of this is in [7], in which an $O(\sqrt{\sum_{v_i \in V} p(v_i)} \cdot m)$ algorithm is proposed. Since our potentials are bounded by two, this would be an $O(m \cdot \sqrt{n})$ time algorithm to solve the MEMPR problem.

Observe that a maximum degree two graph consists of a collection of paths and cycles. This means every connected component is either a path or a cycle. First, we find the MEMPR R in the graph using the algorithm in [7]. Then, we remove an edge from every cycle. This obviously forms a MASEMPR in every component that is a path. This algorithm is summarized in Algorithm 4.1. Consider a component that is a set of edges C . Observe that a MASEMPR is of at most size $|C - 1|$, because all $|C|$ edges would form a cycle. Therefore, if $C \cap R = C$, then we can remove an edge to get a MASEMPR in the subgraph containing just C . Thus, this algorithm returns a MASEMPR in every connected component and, therefore, a MASEMPR in the whole graph. Furthermore, this whole algorithm takes time $O(m \cdot \sqrt{n})$.

Algorithm 4.1. Algorithm for MASEMPR in maximum degree two graphs

Input: The maximum degree two graph G with potential function p

Output: A MASEMPR

procedure MAXDEGTWO(**graph** G , **potential function** p)

 Find the MEMPR R in G

 Remove an edge from every cycle to make R'

return (R')

4.2 Algorithm

The algorithm begins by removing all vertices of potential < 1 . Our algorithm takes as input the graph and our current path set, which is initialized to \emptyset . Our algorithm is a branching algorithm that branches on a given edge (v_i, v_j) into whether there exists at least one MASEMPR with it or whether no MASEMPR contains it. If the total potential of the graph is ≤ 7 , we simply solve the MASEMPR problem by brute force. If an edge creates a cycle in our ASEMPR, we remove it from the graph.

First, we need the following sub-procedure, which removes an edge, if we think it is in at least one MASEMPR. If an edge is in the ASEMPR the algorithm is considering, it is removed from the graph. Then, the algorithm can find a MASEMPR in the remaining graph and combine the two. However, to respect the potential requirements, both of the vertices incident on the edge (v_i, v_j) on cannot maintain the same potentials. To fix this, the algorithm decreases both of the potentials by one. Consequently, when we find a MASEMPR in the remaining graph and combine it with (v_i, v_j) , the potential requirements are met. This is summarized in Algorithm 4.2.

Algorithm 4.2. Edge inclusion algorithm

Input: The subcubic graph G with potential function p , our current ASEMPR M , and the edge we are branching on (v_i, v_j)

Output: A new subcubic graph G' with an updated potential function p' and an updated ASEMPR M'

procedure INMASEMPR(**graph** G , **potential function** p , **ASEMPR** M , **edge** (v_i, v_j))

Create a new graph G' and set $G' = G \triangleright$ Also create the potential function p' of G' to have the same potential values as p

Create a new ASEMPR M' and set $M' = M$

Remove (v_i, v_j) from G' and add it to M'

Decrease $p'(v_i)$ and $p'(v_j)$ in G' by one each

return $(G', p', \text{ and } M')$

We also need the following sub-procedure for when the algorithm branches into an edge in no MASEMPRs. If an edge is in no MASEMPRs, we remove the edge and call Algorithm 4.2 on all other edges incident on v_i and v_j . This discussion is summarized in Algorithm 4.3.

Finally, if a vertex v_i has degree one, we add the edge incident on v_i to the current ASEMPR, decrease the potential of v_i 's neighbor, remove v_i , and recurse. Our procedure is summarized in Algorithm 4.4.

4.3 Proof of Correctness

In this section, we prove that Algorithm 4.4 returns a MASEMPR. First, we prove the following lemma.

Algorithm 4.3. Edge removal algorithm

Input: The subcubic graph G with potential function p , our current ASEMPR M , and the edge we are branching on (v_i, v_j)

Output: A new subcubic graph G' with an updated potential function p' and an updated ASEMPR M'

procedure OUTASEMPR(**graph** G , **potential function** p , **ASEMPR** M , **edge** (v_i, v_j))

Set G_0 , p_0 , and M_0 to G , p , and M , respectively.

for (all vertices v_k in V) **do**

if (v_k is adjacent to v_i or v_j) **then**

 Update G_0 , p_0 , and M_0 using INASEMPR($G_0, p_0, M_0, (v_i, v_k)$)

 Remove (v_i, v_j) from G_0

return (G_0 , p_0 , and M_0)

Lemma 1. Consider two neighboring vertices v_i and v_j , with the degree of v_i being one. There exists a MASEMPR with (v_i, v_j) .

Proof. Assume there exists a MASEMPR A without (v_i, v_j) . Consider adding (v_i, v_j) to A to create an ASEMPR A' . v_i 's potential requirement is met as there only exist vertices of potential ≥ 1 . If v_j 's potential is violated, we can remove one of the preexisting edges incident on v_j . This shifting procedure is shown in Fig. 2. Since we have added one edge and removed at most one edge from A , we have that $|A'| \geq |A|$, i.e., A' is a MASEMPR. Furthermore, A' contains (v_i, v_j) . \square

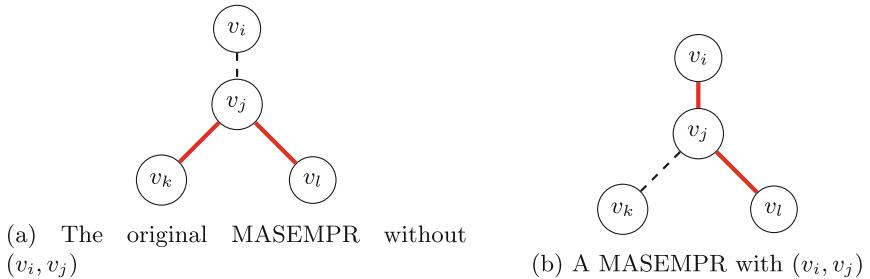


Fig. 2. The shifting of an edge to create a MASEMPR with (v_i, v_j)

Lemma 2. Consider two neighboring vertices v_i and v_j , at least one of which is of degree three. Without loss of generality, let us assume the degree three vertex is v_i . Let v_i 's two other neighbors be v_k and v_l . Let v_j 's other neighbor(s) be v_o (and if it has two, v_q). If there does not exist a MASEMPR with (v_i, v_j) , then all MASEMPRs contain all other edges incident on v_i and v_j .

Algorithm 4.4. Algorithm for the MASEMPR problem in subcubic graphs

Input: The subcubic graph G with potential function p and our current ASEMPR M

Output: A MASEMPR

```

procedure EXACTEXPO(graph  $G$ , potential function  $p$ , ASEMPR  $M$ )
  for (all vertices  $v_k \in V$ ) do
    if ( $d(v_k) = 1$ ) then
      Create  $G'$  by removing  $v_k$  from  $G$ 
      Decrease the potential of  $v_k$ 's neighbor by one in  $G'$ 
      Add the only edge incident on  $v_k$  to  $M$ 
      return (EXACTEXPO( $G'$ ,  $p$ ,  $M$ ))
    for (all vertices  $v_k \in V$ ) do
      if ( $p(v_k) \leq 0$ ) then
        Create  $G'$  by removing  $v_k$  from  $G$ 
        return (EXACTEXPO( $G'$ ,  $p$ ,  $M$ ))
    if (all vertices have degree two or less) then
      Use Algorithm 4.1
    if (the total potential of vertices in  $G$  is  $\leq 7$ ) then
      Solve MASEMPR by enumerating all subsets of edges, narrowing that down
      to ASEMPRs, and then returning the one of maximum size.
    Pick an arbitrary edge  $e = (v_i, v_j)$  with  $v_i$  of degree 3
    if (adding  $(v_i, v_j)$  to  $M$  creates a cycle) then
      Create  $G'$  by removing  $(v_i, v_j)$  from  $G$ 
      return (EXACTEXPO( $G'$ ,  $p$ ,  $M$ ))

    Create  $G_1$ ,  $p_1$ , and  $M_1$  using INASEMPR( $G$ ,  $p$ ,  $M$ ,  $(v_i, v_j)$ )
    Create  $G_2$ ,  $p_2$ , and  $M_2$  using OUTASEMPR( $G$ ,  $p$ ,  $M$ ,  $(v_i, v_j)$ )
    Let  $R_1$  denote the EXACTEXPO( $G_1$ ,  $p_1$ ,  $M_1$ )
    Let  $R_2$  denote the EXACTEXPO( $G_2$ ,  $p_2$ ,  $M_2$ )
    if ( $|R_1| > |R_2|$ ) then
      return ( $R_1$ )
    else
      return ( $R_2$ )
  
```

Proof. Let it be the case that no MASEMPR contains (v_i, v_j) . Consider a MASEMPR B that does not contain an edge e incident on v_i or v_j (not including (v_i, v_j)). Without loss of generality, if e is incident on v_i , we can assume $e = (v_i, v_k)$ (symmetric arguments can be made for (v_i, v_l)). Observe that the potential of v_i must be two, as Algorithm 4.4 only drops the potentials of vertices when an edge incident on them is being removed.

Consider adding (v_i, v_j) to B to make an ASEMPR B' . Observe that v_i has at most one edge incident on it in B , that edge being (v_i, v_l) . This is because we are assuming (v_i, v_k) is not in B by contradiction and the hypothesis is that (v_i, v_j) is in no MASEMPRs (including B itself). Since v_i is the endpoint of at most one edge in B and v_i 's potential is two, adding (v_i, v_j) to B does not violate v_i 's potential requirement.

On the other hand, v_j can be the endpoint of two edges in B (namely (v_j, v_o) and (v_j, v_q)). In this case, we remove one of these two edges from B' . Now, we can add (v_i, v_j) to B' and v_j 's potential requirement will not be violated either (this shifting is shown in Fig. 2). Thus, we have removed at most one edge from B and added exactly one edge to create B' , meaning $|B'| \geq |B|$. Observe that B is a MASEMPR, and therefore so is B' . However, B' contains (v_i, v_j) , contradicting the fact that no MASEMPR contains (v_i, v_j) . Thus, if there does not exist a MASEMPR with (v_i, v_j) , then all MASEMPRs contain all other edges incident on v_i and v_j . \square

We also need the following lemma.

Lemma 3. *Consider two neighboring vertices v_i and v_j . Let v_i have degree 3. Consider constructing a graph G' , which consists of G without the edge (v_i, v_j) and with the potentials of v_i and v_j lowered by one. Let M' be a MASEMPR in G' . If (v_i, v_j) is in at least one MASEMPR A in G , $M = M' \cup \{(v_i, v_j)\}$ is a MASEMPR in G .*

Proof. By contradiction, let $|A| > |M|$. Consider $A' = A \setminus \{(v_i, v_j)\}$. Observe that A' is an ASEMPR in G' . Because M' is a MASEMPR in G' , we have $|M'| \geq |A'| \Rightarrow |M'| + 1 \geq |A'| + 1$. However, we can add one edge to M' to create M and one edge to A' to create A , so $|M'| + 1 = |M|$ and $|A'| + 1 = |A|$. Note that, for contradiction, A is not a MASEMPR in these figures. So, we have $|M| \geq |A|$. However, this contradicts our assumption that $|A| > |M|$, and thus M is a MASEMPR. \square

Now, we conclude the following.

Theorem 2. *Algorithm 4.4 returns a MASEMPR.*

Proof. We prove this using the second principle of induction. First note that, for graphs of potential ≤ 7 , Algorithm 4.4 is correct because it simply enumerates all ASEMPRs and chooses the one of maximum size. Let it be that Algorithm 4.4 returns a MASEMPR for all graphs with potential $\leq k$.

Consider a graph with a total potential $(k + 1)$. Without loss of generality, we assume there exists a vertex v_i of degree three. If v_i does not exist, then the algorithm uses Algorithm 4.1, which we have already proven correct. Let one neighbor of v_i be v_j . Observe that (v_i, v_j) must be in at least one MASEMPR or no MASEMPRs.

If v_j has degree one, we can apply Lemma 1 and lower the potential of the graph. By the inductive hypothesis Algorithm 4.4 finds a MASEMPR in the remaining graph. By Lemma 3, this MASEMPR combined with (v_i, v_j) (the ASEMPR Algorithm 4.4 creates) forms a MASEMPR in G .

Let it be the case that (v_i, v_j) is in at least one MASEMPR. Consider constructing a graph G' which consists of G without the edge (v_i, v_j) and with the potentials of v_i and v_j lowered by one. This is precisely what Algorithm 4.4 does when branching into this case. According to the inductive hypothesis (observe that the potential is reduced by two), Algorithm 4.4 finds a MASEMPR in G' .

Furthermore, according to Lemma 3, we have that the union of a MASEMPR in G' and (v_i, v_j) (i.e., the ASEMPR Algorithm 4.4 creates) is a MASEMPR in G .

Let it be the case that (v_i, v_j) is in no MASEMPRs. Let (v_i, v_k) be an edge incident on v_i . According to Lemma 2, (v_i, v_k) must be in all (and therefore at least one) MASEMPRs. Observe that Algorithm 4.4 still uses Algorithm 4.2 on (v_i, v_k) (i.e., assumes it is in at least one MASEMPR). After calling Algorithm 4.2 on all edges incident on v_i and v_i except (v_i, v_j) and (v_i, v_k) , we can then use the above argument because (v_i, v_k) is in at least one MASEMPR (note that we must replace v_j with v_k).

Thus, we have that if Algorithm 4.4 works for all graphs with total potential $\leq k$, then Algorithm 4.4 works for all graphs of total potential $(k + 1)$. Thus, Algorithm 4.4 is correct. \square

4.4 Time Analysis

In this subsection, we analyze the time taken by Algorithm 4.4. Checking for vertices of potential zero or less takes linear time. Algorithm 4.2 takes linear time. Algorithm 4.3 also takes linear time. We simply go through every vertex and apply Algorithm 4.2 if necessary (there are at most four of these vertices). For the edge (v_i, v_j) , let v_j have degree two. In this case, we either call Algorithm 4.2 on (v_i, v_j) , dropping the total potential by two, or call Algorithm 4.2 on three edges total (using Algorithm 4.3), dropping the total potential by six. Let C denote the total potential. This yields the following recurrence.

$$\begin{aligned} T(C) &= O(1), \text{ if } C \leq 7 \\ T(C) &= T(C - 2) + T(C - 6) + O(n), \text{ otherwise} \end{aligned}$$

Solving this recurrence gives a running time of $O^*(1.2106^C)$ using techniques in [6]. Now, let v_j had degree three. In this case, we either call Algorithm 4.2 on (v_i, v_j) , dropping the total potential by two, or call Algorithm 4.2 on four edges total (using Algorithm 4.3), dropping the total potential by eight. Let C denote the total potential. Since the potential of every vertex is 2, $C = 2 \cdot n$. This yields the following recurrence.

$$\begin{aligned} T(C) &= O(1), \text{ if } C \leq 7 \\ T(C) &= T(C - 2) + T(C - 8) + O(n), \text{ otherwise} \end{aligned}$$

Solving this recurrence gives a running time of $O^*(1.1749^C)$ using techniques in [6]. Therefore, in the worst case, our algorithm runs in time $O^*(1.2106^C)$. Observe that $C = 2 \cdot n$. Thus, the total running time is $O^*(1.2106^{2 \cdot n}) \subseteq O^*(1.4656^n)$.

4.5 Space Analysis

Observe that we can reuse space from previous recursions. In other words, the maximum number of graphs we need to store is the longest branch of the recursion ending at an empty graph. This happens when we go along the $T(C - 2)$

branch on every graph that we get. This means we have to store $\frac{C}{2}$ graphs. Since $C = 2 \cdot n$, our algorithm uses $O(n^2)$ space.

Corollary 1. *The MPS problem in subcubic graphs can be solved in $O^*(1.4656^n)$ time using $O(n^2)$ space.*

Proof. Follows from the discussion above. \square

5 An Approximation Algorithm for the MASEMPR Problem

In this section, we design and analyze a $\frac{\lambda-1}{\lambda}$ -approximation for the MASEMPR problem where λ is the girth of the input graph. Note that the girth of an undirected graph is the minimum number of edges in any cycle in that graph.

5.1 Algorithm and Proof of Approximation Factor

Our algorithm first computes a MEMPR using the algorithm in [7]. Then, we remove one edge from every cycle in this MEMPR and return the remaining ASEMPR. These steps are summarized in Algorithm 5.1.

Algorithm 5.1. Approximation algorithm for MASEMPR in general graphs

Input: The graph G with potential function p

Output: An ASEMPR

procedure MASEMPRAPROX(**graph** G , **potential function** p)

 Find the MEMPR R in G using the algorithm in [7]

 Remove an edge from every cycle in R to create R'

return (R')

For a given set of edges S in G , let $C[S]$ denote the number of cycles in S . The size of the path set we return is $(OPT_{MEMPR}(G) - C[OPT_{MEMPR}(G)])$. Let λ be the girth of G . Observe that,

$$OPT_{MEMPR}(G) - C[OPT_{MEMPR}(G)] \geq OPT_{MEMPR}(G) - \frac{OPT_{MEMPR}(G)}{\lambda}$$

This is because a cycle contains at least λ edges, so the number of cycles in a graph with h edges is at most $\frac{h}{\lambda}$. In this case, the number of cycles is $\leq \frac{OPT_{MEMPR}(G)}{\lambda}$. Observe that

$$\begin{aligned} OPT_{MEMPR}(G) - \frac{OPT_{MEMPR}(G)}{\lambda} &= \frac{(\lambda-1) \cdot OPT_{MEMPR}(G)}{\lambda} \\ &\geq \frac{\lambda-1}{\lambda} \cdot OPT_{MASEMPR}(G). \end{aligned}$$

The reason that this step is correct is that, since a MASEMPR is technically an EMPR, $OPT_{MEMPR}(G) \geq OPT_{MASEMPR}(G)$.

Thus, $OPT_{MEMPR}(G) - C[OPT_{MEMPR}(G)] \geq \frac{\lambda-1}{\lambda} \cdot OPT_{MASEMPR}(G)$. We therefore conclude that this algorithm has approximation ratio $\frac{\lambda-1}{\lambda}$. As a corollary, we note that this algorithm is also an approximation algorithm for the MPS problem.

5.2 Running Time

Applying the algorithm from [7] takes $O(m \cdot \sqrt{n})$ time (see Sect. 4.1). Furthermore, removing an edge from every cycle takes linear time. Thus, our algorithm runs in time $O(m \cdot \sqrt{n})$.

Corollary 2. *There exists a polynomial time approximation algorithm with approximation factor $\frac{\lambda-1}{\lambda}$ to find the Maximum path set in a graph of girth λ .*

Proof. Follows from the discussion above. \square

6 Conclusion

In this paper, we studied the MASEMPR problem. We investigated the problem along two directions, viz., finding exact solutions and finding approximate solutions. On the exact front, we designed an $O^*(1.47^n)$ (quadratic space) algorithm for subcubic graphs. On the approximation front, we designed a polynomial time, constant factor approximation algorithm.

References

- Ball, M.O., Colbourn, C.J., Provan, J.S.: Network reliability. *Handb. Oper. Res. Manag. Sci.* **7**, 673–762 (1995)
- Chen, Z.-Z., Lin, G., Wang, L.: An approximation algorithm for the minimum co-path set problem. *Algorithmica* **60**(4), 969–986 (2011)
- Cheng, Y., Cai, Z., Goebel, R., Lin, G., Zhu, B.: The radiation hybrid map construction problem: recognition, hardness, and approximation algorithms (2008)
- Cox, D.R., Burmeister, M., Price, E.R., Kim, S., Myers, R.M.: Radiation hybrid mapping: a somatic cell genetic method for constructing high-resolution maps of mammalian chromosomes. *Science* **250**(4978), 245–250 (1990)
- Feng, Q., Zhou, Q., Li, S.: Randomized parameterized algorithms for co-path set problem. In: International Workshop on Frontiers in Algorithmics, pp. 82–93 (2014)
- Fomin, F.V., Kratsch, D.: *Exact Exponential Algorithms*, 1st edn. Springer, Heidelberg (2010)
- Gabow, H.N.: An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In: Proceedings of the 15th Annual ACM Symposium on Theory of Computing, Boston, Massachusetts, USA, 25–27 April 1983, pp. 448–456. ACM (1983)
- Taylor, M.A.P.: Dense network traffic models, travel time reliability and traffic management. II: application to network reliability. *J. Adv. Transport.* **33**(2), 235–251 (1999)



Broadcast Graph is NP-Complete

Jinghan Xu^(✉) and Zhiyuan Li^(✉)

Guangdong Provincial Key Laboratory IRADS, BNU-HKBU United International College, Zhuhai 519087, China
{xujinghan,goliathli}@uic.edu.cn

Abstract. The broadcast model is widely used to describe the process of information dissemination from a single node to all nodes within an interconnected network. In this model, a graph represents the network, where vertices correspond to nodes and edges to communication links. The efficiency of this broadcasting process is evaluated by the broadcast time, the minimum discrete time units required to broadcast from a given vertex. Determining the broadcast time is referred to as the problem BROADCAST TIME. The set of vertices with the minimum broadcast time among the graph is called the broadcast center. Identifying this center or determining its size are both proven to be NP-hard. For a graph with n vertices, the minimum broadcast time is at least $\lceil \log_2 n \rceil$. The BROADCAST GRAPH problem asks in a graph of n vertices, whether the broadcast time from any vertex equals $\lceil \log_2 n \rceil$. Extensive research over the past 50 years has focused on constructing broadcast graphs, which are optimal network topologies for one-to-all communication efficiency. However, the computational complexity of the BROADCAST GRAPH problem has rarely been the subject of study. We believe that the difficulty lies in the mapping reduction for an NP-completeness proof. Consequently, we must construct broadcast graphs for yes-instances and non-broadcast graphs for no-instances. The most closely related result is the NP-completeness of BROADCAST TIME proved by Slater et al. in 1981. More recently, Fomin et al. has proved that BROADCAST TIME is fixed-parameter tractable. In this paper, we prove that BROADCAST GRAPH is NP-complete by proving a reduction from BROADCAST TIME. We also improve the results on the complexity of the broadcast center problem. We show BROADCAST CENTER SIZE is in Δ_p^2 , and is D^P -hard, implying a complexity upper bound of Δ_p^2 -complete and a lower bound of D^P -hard ($NP \subseteq D^P \subseteq \Delta_p^2$).

Keywords: Computational complexity · Broadcast graphs · Broadcast center

1 Introduction

In distributed file systems, in order to maintain data consistency, modified files from one node are synchronized with their copies stored in all other nodes in

Our work was supported in part by the Guangdong Provincial Key Laboratory IRADS (2022B1212010006, R0400001-22) and in part by Guangdong Higher Education Upgrading Plan (2021–2025) with UIC research grant UICR0400025-21.

the network. The synchronization is a one-to-all dissemination procedure, called *broadcast*. The broadcast process starts with one node named *originator* holding the information, and each informed node making parallel calls to adjacent nodes. A *call* refers to the transmission of information from one *sender* to one adjacent *receiver* in one discrete time unit. This paper is based on the classic broadcast model with the following assumptions.

- A broadcast is divided into discrete time units.
- Only one node, the originator, has the information at time unit 0.
- Each informed node can inform at most one uninformed neighbor per time unit.
- The broadcast ends when all the nodes in the network are informed.

A network is modeled as a connected graph $G = (V, E)$, where V is the vertex set representing the nodes, and E is the edge set representing the communication links.

Other dissemination models, such as unicast (one-to-one), multicast (one-to-many), and gossip (all-to-all), are also discussed in the literature [4, 11, 21].

A call through an edge is represented by a directed edge from the sender (an informed vertex) to the receiver (an uninformed vertex). The complete set of calls in the graph forms a *broadcast tree* of G , a directed spanning tree rooted at the originator. The tree visually represents the *broadcast scheme*, sequencing calls over time units to illustrate the entire broadcast process. In a broadcast scheme, if an informed vertex v does not make a call at time t , then v is *idle* at that time unit; otherwise, v is *busy*.

The distance between 2 vertices u and v , denoted by $d(u, v)$, is the number of edges in the shortest path between them. During the broadcast process, the information passing from an informed vertex u to a vertex v costs at least $d(u, v)$ time units.

The *broadcast time*, $b(G, v)$, denotes the minimum time units required to broadcast from vertex v in graph G . The broadcast time of the graph, $b(G)$, is the maximum broadcast time from any originator $u \in V$, formally $b(G) = \max_{u \in V}(b(G, u))$.

The problem of finding the broadcast time $b(G, v)$ is NP-complete [31]. More recently, it has been shown to be fixed-parameter tractable (FPT) using three different kernelizations: the vertex cover, the feedback edge set, or the input deadline of broadcast time [12]. A follow-up study on parameterized complexity of broadcast time [32] indicates that the problem remains NP-complete with a feedback vertex set of size 1 (and therefore of tree width 2). Concurrently, a double exponential lower bound is established when parameterized by the solution size.

In the classic broadcast model, each sender can inform at most one receiver per time unit. Thus, the number of informed vertices can at most double in one time unit. For arbitrary graph G on n vertices, $b(G) \geq \lceil \log n \rceil$. Throughout this paper, the base of the logarithm is assumed to be 2 and is therefore omitted.

A graph G on n vertices is called a *broadcast graph* if $b(G) = \lceil \log n \rceil$, the theoretically lowest broadcast time. The value of n is always partitioned into

discrete ranges between two consecutive power of 2, $2^{t-1} + 1 \leq n \leq 2^t$, because broadcast graphs within this range have the same broadcast time t . The broadcast graph structure ensures the most efficient one-to-all data transmission in the network. A broadcast graph with the minimum possible number of edges is called a *minimum broadcast graph*. However, minimum broadcast graphs are only constructed for some special values of n . Therefore, more studies focus on broadcast graph construction (not necessarily minimum) after broadcast graphs were introduced by Knödel [23] in 1975. Most constructions are classified into the following methods: the compound method [8, 9, 14, 17], the ad-hoc method [2, 10, 13, 20], and the vertex addition or deletion method [2, 16, 20]. Certain broadcast graph constructions with specific number of vertices are also studied [2, 24, 25, 34].

Although numerous results have focused on the broadcast time problem and broadcast graph constructions, the complexity of verifying whether a graph is a broadcast graph has not yet been discussed. This task is challenging because a mapping reduce from a known problem X to the broadcast graph problem is needed. This reduction requires transforming yes-instances of X to broadcast graphs and no-instances of X to non-broadcast graphs, which is more difficult than constructing certain broadcast graphs. This paper proves that the broadcast graph problem is NP-complete. The broadcast graph problem is formally defined as follows.

Problem 1. BROADCAST GRAPH

- Input: A graph G .
- Question: Is G a broadcast graph?

Currently, the most related result is given by Slater, Cockayne, and Hedetniemi in [31]. They proved the NP-completeness of BROADCAST TIME by a reduction from 3-DIMENSIONAL MATCHING. Other results on NP-completeness of broadcast time problem include work on 3-regular planar graphs [27], bounded degree graphs [7] and chordal graphs [22].

Problem 2. BROADCAST TIME

- Input: A three-tuple (G, V_0, t) , where $G = (V, E)$ is a graph, $V_0 \subseteq V$ is the set of originators, and $t \in \mathbb{N}$ is the given broadcast time bound.
- Question: Is $b(G, v) \leq t$ for all $v \in V_0$?

We prove BROADCAST GRAPH is NP-complete by a reduction from ST-BROADCAST TIME, a specialization of BROADCAST TIME, where V_0 is a singleton set of one originator, and $t = \lceil \log |V| \rceil$.

Problem 3. Single Origin Time-bounded Broadcast Time (ST-BROADCAST TIME)

- Input: A two-tuple (G, v) , where $G = (V, E)$ is a graph, and $v \in V$ is the originator.
- Question: Is $b(G, v) = \lceil \log |V| \rceil$?

Papadimitriou and Yannakakis [29] proved a similar problem to be NP-complete, which is given a graph G , whether one can find a binomial spanning tree of G . This problem is equivalent to asking if there exists a vertex v such that $b(G, v) = \lceil \log n \rceil$, while v is specified in ST-BROADCAST TIME. Therefore, we provide an independent proof for the NP-completeness of ST-BROADCAST TIME by a reduction from 3-DIMENSIONAL MATCHING. This proof, which is omitted due to the space restriction, can be found in the full version [33].

In addition to the ST-BROADCAST TIME problem, people are also interested in determining the most efficient vertex from which to broadcast in the network. Focusing on these critical nodes leads to the study of the *broadcast center*, the set of vertices having the minimum broadcast time in a graph G , denoted by BC_G . It is demonstrated that both decision problems *Broadcast Center Deciding Problem* and *Broadcast Center Size Problem* of determining the broadcast center or its size, are NP-hard [18].

Problem 4. Broadcast Center Size (BC-SIZE)

- Input: A two-tuple (G, x) , where $G = (V, E)$ is a graph, and $x \in \mathbb{N}$ is the given broadcast center size bound.
- Question: Is $|BC_G| = x$?

In the present paper we demonstrate that BC-SIZE is in Δ_p^2 and is D^P -hard, which allows us to locate the complexity of BC-SIZE within an interval, in which the lower bound is improved from NP-hard to D^P -hard and the upper bound is Δ_p^2 -complete. Notably, it is not our concern whether BC-SIZE is in D^P , given that D^P is currently below the lower bound.

Δ_p^2 is a complexity class on the second level of the polynomial hierarchy (PH). A problem is in Δ_p^2 if it can be solved by a polynomial-time algorithm with the access to an NP oracle. Thus, Δ_p^2 is also denoted as P^{NP} . The difference polynomial time D^P is the complexity class for languages that are in the intersection of a language in NP and a language in $co\text{-}NP$. D^P is equivalent to BH_2 , the second level on the Boolean hierarchy, and the entire BH is contained in Δ_p^2 . Our proof of the D^P -hardness is established by a reduction from UNIQUE-SAT, which is known to be D^P -complete [3]. For more details about PH and BH , readers are referred to [5, 6, 30].

Problem 5. Unique Satisfiability (UNIQUE-SAT)

- Input: A CNF boolean formula $\phi(x_0, \dots, x_{n-1})$ of n variables and c clauses.
- Question: Is there a unique assignment to satisfy ϕ ?

This paper is organized as follows. Section 2 reviews some important graph structures used in our proofs. Section 3 proves the NP-completeness of BROADCAST GRAPH. Section 4 studies the complexity of BC-SIZE.

2 Preliminaries

Our proof of NP-completeness for the BROADCAST GRAPH problem involves constructing broadcast graphs (as problem instances) under specific conditions. Therefore, this section reviews some useful graph structures [15] benefitting the proof.

To construct broadcast graphs, we use the compounding method introduced by Averbuch, Shabtai and Roditty [1], and later used by Harutyunyan and Li [17], which combines hypercubes or Knödel graphs with binomial trees.

Definition 1. A binomial tree $BT_k = (V, E)$ of degree $k \geq 0$ is defined recursively:

- (base case) when $k = 0$, $BT_0 = (\{\varepsilon\}, \{\})$;
- (recursion) for $k \geq 1$, BT_k rooted at r is consisted of 3 parts, BT_{k-1} rooted at r , its replica BT'_{k-1} rooted at r' , and an additional edge (r, r') . Each vertex is represented by a binary string of k bits in BT_k , either ‘1 suffixed by its binary in BT_{k-1} ’, or ‘0 suffixed by its binary in BT'_{k-1} ’ (Fig. 1).

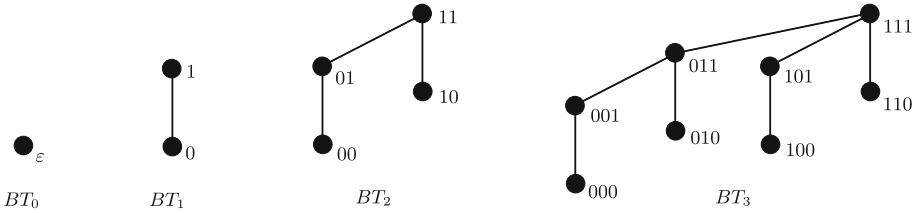


Fig. 1. binomial trees of degree 0 to 3. Each vertex is labeled by its binary representation. This notation allows us to sort the leaves (vertices ends with ‘0’) by their distances to the root in descending order, which is useful in Sect. 4. In BT_i , the first $i - 1$ bit of the leaves are considered. For example, in BT_3 , the first two bits of the leaves are $bin(00) = dec(0)$, $bin(01) = dec(1)$, $bin(10) = dec(2)$, $bin(11) = dec(3)$.

We introduce binary notation for vertices in binomial trees. Note ε is the empty string. A leaf’s binary ends with 0. We define $l_i = \lambda_0 \lambda_1 \dots \lambda_{d-2} 0$ as the i^{th} leaf if $\lambda_0 \lambda_1 \dots \lambda_{d-2}$ is the binary representation of i .

To broadcast from the root v_0 in BT_k , the broadcast scheme follows the recursive definition of BT_k that, each root of a BT_{k-i} calls its replica’s root at time i . Thus, $b(BT_k, v_0) = k$. Since BT_k has 2^k vertices, $k = \log 2^k$ is the minimum time for broadcast in BT_k , with each vertex busy in this broadcast scheme.

Since 2^k is the maximum number of vertices that can be informed in k time units, any broadcast tree with broadcast time k is a subtree of the binomial tree BT_k .

Observation 1. A multicast from the root v_0 in BT_k to its first x leaves $\{l_0, \dots, l_{x-1}\}$ requires time k .

This is because $\{l_0, \dots, l_{x-1}\}$ contains l_0 , and the distance $d(v_0, l_0) = k$, the multicast time cannot be smaller than k .

Definition 2. A Knödel graph $KG_n = (V, E)$ is defined on $n = 2k$ vertices for $k \geq 0$, where

- $V = \{v_0, v_1, \dots, v_{n-1}\}$;
- $E = \{(v_x, v_y) | x + y \equiv 2^d - 1 \pmod{n}, 1 \leq d \leq \lfloor \log n \rfloor\}$. Each edge (v_x, v_y) corresponds to dimension d .

Knödel graphs are regular and edge-transitive. Hence, a dimensional broadcast scheme is applicable, except that in the final time unit $\lceil \log n \rceil$, all vertices call their 1st-dimensional neighbor.

Observation 2. In a Knödel graph, both the originator and its 1st-dimensional neighbor are idle in the last time unit.

This can be easily demonstrated through the dimensional broadcast scheme, a property that makes Knödel graphs valuable in broadcast graph constructions [17, 19, 20].

3 Broadcast Graph Problem is NP-Complete

In this section, we prove the NP-completeness of the BROADCAST GRAPH problem by performing a polynomial time many-one reduction (ST-BROADCAST TIME \leq_p BROADCAST GRAPH) from the known NP-complete problem ST-BROADCAST TIME. We demonstrate that BROADCAST GRAPH is both in NP and NP-hard.

Lemma 1. BROADCAST GRAPH is in NP.

Proof. We prove this lemma by presenting a polynomial-time verifier. Let $G = (V, E)$ be an instance of BROADCAST GRAPH with n vertices, and $V = \{v_1, v_2, \dots, v_n\}$. The certificate is a set of spanning trees $S = \{T_1, T_2, \dots, T_n\}$ of G , such that for $1 \leq i \leq n$, each T_i is rooted at v_i , and represents a valid broadcast scheme with originator v_i . To verify whether T_i is a subtree of a degree n binomial tree $BT_{\lceil \log n \rceil}$, the time complexity is $O(n) \times (O(n^{5/2}) + O(\log n))$, which is $O(n^{7/2})$ [26]. \square

For the reduction, we construct an instance of BROADCAST GRAPH from an ST-BROADCAST TIME instance and prove that this construction is a polynomial-time reduction.

Definition 3. Given an instance of STBT, (G_s, v_s) on n_s vertices, a graph G_u is constructed by the following algorithm.

1. Create a Knödel graph KG_6 on 6 vertices labeled v_0, v_1, \dots, v_5 .
2. Create 5 copies of binomial trees T_1, \dots, T_5 of degree $\lceil \log n_s \rceil + 1$ rooted at r_1, \dots, r_5 .
3. Create 1 binomial tree T_6 of degree $\lceil \log n_s \rceil$ rooted at r_6 .
4. Merge each pair of vertices (v_i, r_i) for $1 \leq i \leq 5$ and merge (v_s, v_0) .
5. Add edges (u, r_1) and (u, r_5) for every vertex u in G_s , T_1 , T_5 , and T_6 (if not already adjacent). Also add an edge (v_s, r_6) .
6. Add edges (u, r_2) and (u, r_4) for every vertex u in T_2 , T_3 , and T_4 (if not already adjacent).

From the construction, we calculate the number of vertices.

$$\begin{aligned}
|V_u| &= 5(2^{\lceil \log n_s \rceil + 1}) && \text{vertices in } BT_1, \dots, BT_5 \\
&\quad + 2^{\lceil \log n_s \rceil} && \text{vertices in } BT_6 \\
&\quad + n_s && \text{the vertices from } G_s \\
&= 11(2^{\lceil \log n_s \rceil}) + n_s
\end{aligned}$$

Let $\lceil \log n_s \rceil = t$ for some $t \in \mathbb{N}$.

$$\begin{aligned}
2^{t-1} + 1 &\leq |V_s| \leq 2^t \\
2^t \times 11 + 2^{t-1} + 1 &\leq |V_u| \leq 2^t \times 11 + 2^t \\
2^{t+3} = 2^t \times 8 &< |V_u| < 2^t \times 16 = 2^{t+4}
\end{aligned}$$

Thus, if G_u is a broadcast graph (yes-instance of BROADCAST GRAPH), then the broadcast time $b(G_u) = \lceil \log |V_u| \rceil = t + 4 = \lceil \log n_s \rceil + 4$. The number of edges in G_u is calculated as follows.

$$\begin{aligned}
|E_u| &= |E_s| && \text{the edges from } G_s \\
&\quad + 5(2^{\lceil \log n_s \rceil + 1}) && \text{edges in } BT_1, \dots, BT_5 \\
&\quad + 2^{\lceil \log n_s \rceil} && \text{edges in } BT_6 \\
&\quad + 2(n_s) + 4(2^{\lceil \log n_s \rceil + 1}) + 2(2^{\lceil \log n_s \rceil}) + 1 && \text{step 5} \\
&\quad + 6(2^{\lceil \log n_s \rceil + 1}) && \text{step 6} \\
&= 33(2^{\lceil \log n_s \rceil}) + 2n_s + |E_s| + 1
\end{aligned}$$

G_u is polynomial-time constructible on G_s . Next, we prove from both directions (yes and no instances) that the construction is a reduction from $b(G_s, v_s)$ to G_u .

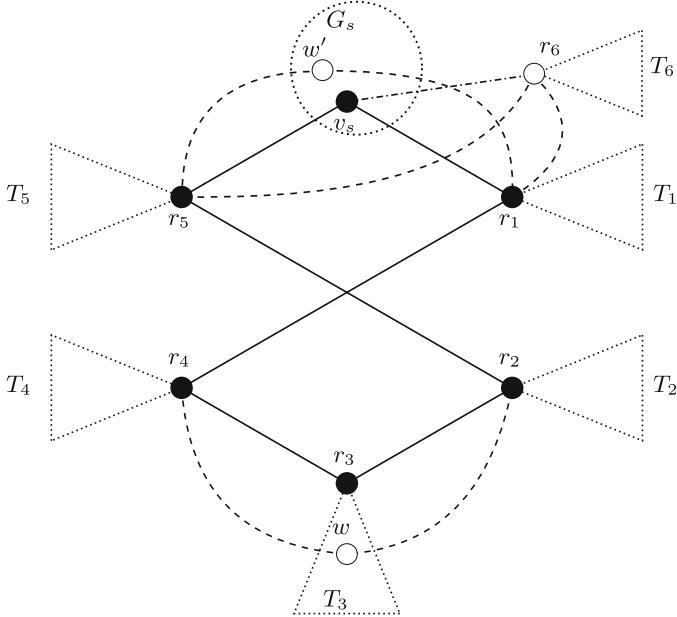


Fig. 2. Example construction of G_u on G_s . The dotted triangles represent the binomial trees T_1, \dots, T_5 of degree $\lceil \log n_s \rceil + 1$, and T_6 of degree $\lceil \log n_s \rceil$. A Knödel graph KG_6 is formed by r_1, \dots, r_5 , and v_s . The dashed lines represent edges added in step 5 and 6 in Definition 6. The dot-dashed line is (v_s, r_6) .

Lemma 2. (G_s, v_s) is a yes-instance of ST-BROADCAST TIME if and only if G_u is a broadcast graph.

Proof. (\Rightarrow) Assume (G_s, v_s) is a yes-instance of ST-BROADCAST TIME, that is, $b(G_s, v_s) = \lceil \log n_s \rceil = t$. We apply a broadcast scheme from [20] for each originator position in G_u . There are 3 cases for originators.

1. If the originator is an arbitrary vertex in KG_6 , it informs all vertices in KG_6 in the first 3 time units using the dimensional broadcast scheme of Knödel graphs. From time 4, r_1, \dots, r_5 broadcast in T_1, \dots, T_5 separately, completing in $3 + t + 1 = t + 4$. Simultaneously, v_s informs r_6 by time 4. From time 5 to $t + 4$, v_s completes the broadcast in G_s , and r_6 in T_6 .
2. If the originator (not in KG_6) is an arbitrary vertex in T_1, T_5, T_6, G_s , for example, w' in Fig. 2, then w' mimics v_s to broadcast in KG_6 in the first 3 time units. This is similar to the dimensional broadcast, except that v_s is informed by r_5 or r_1 at time 3. By Observation 2, either r_5 or r_1 is idle in the last time unit. The rest follows case 1.
3. If the originator (not in KG_6) is an arbitrary vertex in T_2, T_3, T_4 , for example w in Fig. 2, the broadcast follows case 2 except w mimics r_3 in KG_6 , informed by r_2 or r_4 in time unit 3.

(\Leftarrow) If (G_s, v_s) is a no-instance of ST-BROADCAST TIME, which means $b(G_s, v_s) > \lceil \log n_s \rceil = t$, then exists $w \in G_u$ such that the broadcast from w in G_u cannot finish in $t + 4$ time units. Assume w is an arbitrary non-root vertex in T_3 , the proof enumerates all possible broadcast schemes from w in G_u .

First, note $\{(r_1, r_4), (r_2, r_5)\}$ is a cut in G_u , with $d(w, r_1) = d(w, r_5) = 2$. Thus, originating from w , r_1 and r_5 cannot both be informed by time 2. One of r_1 and r_5 can be informed by time 3, the earliest possible time. With r_1 and r_5 being symmetric in G_u (there is an automorphism $f : G_u \rightarrow G_u$ such that $f(r_1) = r_5$), we assume r_1 and r_5 are informed in time 2 and 3. At time 3, the broadcast from w splits into two parallel independent multi-originator broadcast sub-schemes: one in $T_1 \cup T_5 \cup T_6 \cup G_s$ from $\{r_1, r_5, x_1\}$, where x_1 is a neighbor of r_1 ; another in $T_2 \cup T_3 \cup T_4$ from $\{w, r_2, r_4, x_2, x_3\}$, with x_2 as a neighbor of w and x_3 of r_4 . Since no vertex in one sub-scheme can inform the other (if not, the broadcast time trivially exceed $t + 4$), if the first sub-scheme cannot finish in $t + 4 - 3 = t + 1$, then G_u is not a broadcast graph.

Next, for the first sub-scheme, consider T_1 and T_5 . Broadcasting from r_5 in T_5 needs $t + 1$ time units - the exact remaining time for r_5 . Similarly, r_1 needs $t + 1$ units in T_1 . Vertices in T_1 and T_5 are kept busy in time 4 to $t + 4$. Thus, the question is which neighbor x_1 is informed by r_1 at time 3? Besides in binomial trees T_1 or T_5 , there are four choices:

1. If r_1 informs v_s at time 3, then
 - if v_s informs r_6 in time 4, broadcasting in G_s cannot finish in $t + 4$ due to $b(G_s, v_s) > t$;
 - if v_s does not inform r_6 in time 4, broadcasting in T_6 cannot finish in $t + 4$, as T_6 is a degree t binomial tree.
2. If r_1 informs r_6 at time 3, it is similar to case 1.
3. If the neighbor x_1 is in T_6 (not r_6), then $d(v_s, x_1) \geq 2$, v_s cannot be informed by time 4, and broadcasting in G_s cannot finish in $t + 4$.
4. If the neighbor x_1 is in G_s (not v_s), then $d(r_6, x_1) \geq 2$, r_6 cannot be informed by time 4, and broadcasting in T_6 cannot finish in $t + 4$.

Thus, if (G_s, v_s) is a no-instance of ST-BROADCAST TIME, G_u is not a broadcast graph. \square

Summarizing lemmas 1 and 2, and definition 3 is a polynomial-time construction,

Theorem 1. BROADCAST GRAPH is NP-complete.

4 The Complexity of BC-SIZE Problem

In this section, we examine the complexity of BC-SIZE. We demonstrate that BC-SIZE is in Δ_2^P and D^P -hard by reducing from UNIQUE-SAT, which is D^P -complete [28].

Lemma 3. BC-SIZE is in Δ_2^P .

Algorithm 1: Polynomial algorithm for deciding BC-SIZE with access to the oracle BROADCAST TIME.

```

Input:  $G = (V, E)$ ,  $c \in \mathbb{N}$ 
Output: 1 if  $|BC_G| = c$ ; 0 otherwise.
1 for  $i = \log\lceil|V|\rceil$  to  $n$  do
2   size = 0
3   for  $v \in V$  do
4     if  $(G, \{v\}, i) \in$  BROADCAST TIME then
5       | size++
6   if size! = 0 then
7     if size ==  $c$  then
8       | return 1
9     else
10    | return 0
11 return 0

```

Proof. The algorithm uses an NP-complete oracle for BROADCAST TIME on line 4 and runs in $O(n^2)$. \square

For the D^P -hardness, we construct a BC-SIZE instance based on a graph representation of a UNIQUE-SAT instance.

Definition 4. Consider a CNF formula $\phi(x_0, \dots, x_{n-1})$, an instance of UNIQUE-SAT with c clauses, $\delta_1, \delta_2, \dots, \delta_c$. A graph G is constructed by following algorithm.

1. Create $2n$ copies of binomial trees $T^{x_0}, T^{\overline{x_0}}, \dots, T^{x_{n-1}}, T^{\overline{x_{n-1}}}$ of degree $d_1 = \lceil \log c \rceil + 1$, rooted at literals $x_0, \overline{x_0}, \dots, x_{n-1}, \overline{x_{n-1}}$ respectively.
2. Create $2n$ copies of binomial trees $H^{x_0}, H^{\overline{x_0}}, \dots, H^{x_{n-1}}, H^{\overline{x_{n-1}}}$ of degree $d_2 = \lceil \log n \rceil + 1$, rooted at $r^{x_0}, r^{\overline{x_0}}, \dots, r^{x_{n-1}}, r^{\overline{x_{n-1}}}$ respectively.
3. Create a binomial tree T^r of degree $d = d_1 + d_2 + 1 = \lceil \log c \rceil + \lceil \log n \rceil + 3$, rooted at r .
4. Construct a star with $\{r^{x_0}, r^{\overline{x_0}}, \dots, r^{x_{n-1}}, r^{\overline{x_{n-1}}}, r, s\}$, where s is the center.
5. Assume that the literal \hat{x} of variable x is contained in j clauses $\delta'_1, \delta'_2, \dots, \delta'_j$. Construct edges $(\lambda_i^{\hat{x}}, \delta'_j)$ for $0 \leq i < j$ where $\lambda_i^{\hat{x}}$ is the i^{th} leaf of $T^{\hat{x}}$. And do the same for every literal.
6. For each $H^{\hat{x}_i}$, connect each k^{th} leaf to both x_k and $\overline{x_k}$, except the i^{th} leaf to only x_i . Formally, $\{(l_k^{\hat{x}_i}, x_k) | 0 \leq k \leq n-1\} \cup \{(l_k^{\hat{x}_i}, \overline{x_k}) | 0 \leq k \leq n-1, x_k \neq \hat{x}_i\}$.
7. Connect each of the first c leaves of T^r to one distinct clause. Formally, $\{(l_{i-1}, \delta_i) | 1 \leq i \leq c\}$.
8. Attach $2n - \lceil \log c \rceil - 2$ vertices to each δ_i to form c stars of size $2n - \lceil \log c \rceil - 1$ (equivalent to $2n - d_1$) with center δ_i .
9. Add a path $P = (p_0, p_2, \dots, p_{2n+\lceil \log n \rceil+1})$. Connect the star center s to the path P with an edge (s, p_0) .

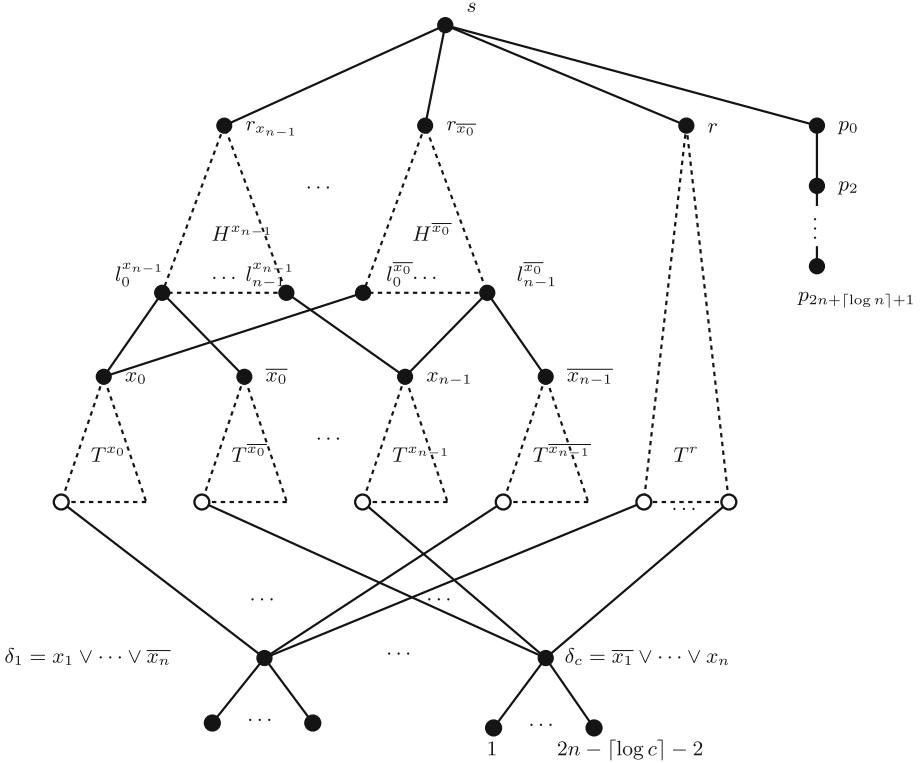


Fig. 3. An example of the construction of G . The binomial tree $H^{\hat{x}_i}$ represents the assignment $\hat{x}_i = 1$. Edges between H 's and T 's show the two possible assignments for each variable. Each binomial tree $T^{\hat{x}_i}$ presents a literal \hat{x}_i . Each δ_i is a clause. The binomial tree T^r ensures s and r are in the broadcast center. The path p_0, \dots forces all vertices other than s, r, r_{x_0}, \dots are out of the broadcast center.

Figure 3 shows an example of the construction. The construction is polynomial-time.

The variable gadget \hat{x}_i is a single vertex adjacent to two literals x_i and \bar{x}_i . The clause gadget δ_k is a vertex adjacent to x_i (or \bar{x}_i) if x_i (or \bar{x}_i) is a literal in δ_k . Assignments on \hat{x}_i are represented by choosing either x_i or \bar{x}_i for broadcast. For example, if $x_i = 0$, then \hat{x}_i calls \bar{x}_i , which calls its clauses (see Fig. 4 for an example), and vice versa. A literal can be in multiple clauses, so each T^{x_i} (or $T^{\bar{x}_i}$) of degree d_1 connects to each literal x_i (or \bar{x}_i) adjacent to theirs leaves. Broadcasting from the set of all variables (as multi-originator) to all clauses completes in $d_1 + 2$ time units.

The construction also ensures a “unique” assignment on ϕ , achieved by selecting each essential literal among the $2n$ literals. In G , this is represented by a binomial tree $H^{\hat{x}_i}$ rooted at $r_{\hat{x}_i}$. Selecting x_i forces it to be *True*, so $H^{\hat{x}_i}$'s leaves connect only to x_i , not \bar{x}_i . Broadcasting from $r_{\hat{x}_i}$ completes in $d_1 + d_2 + 3$ time

units if and only if ϕ is satisfiable with $\hat{x}_i = \text{True}$ (see Fig. 4 for example). The rest steps of the construction create the following.

- A star centered at s to originate a broadcast.
- A binomial tree T^r ensures s and r are in the broadcast center even if ϕ is not satisfiable.
- A path P simplifies the proof.
- Stars centered at clauses are to fill the broadcast time.

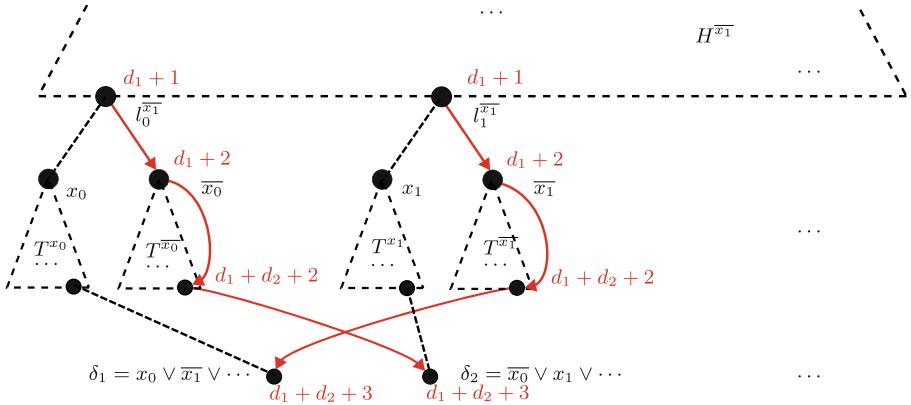


Fig. 4. An example for variable, literal, and clause gadgets. Assume $\delta_1 = x_0 \vee \bar{x}_1 \vee \dots$ and $\delta_2 = \bar{x}_0 \vee x_1 \vee \dots$ are two clauses in ϕ . Vertex δ_1 is adjacent to one leaf in T^{x_0} and another leaf in $T^{\bar{x}_1}$ (same for δ_2). If all leaves of $H^{\bar{x}_1}$ are informed at time unit $d_1 + 1$, then directed edges and time units (in red) show the calls from $l_0^{\bar{x}_1}$ and $l_1^{\bar{x}_1}$ to δ_1 and δ_2 respectively at $d_1 + d_2 + 3$, mimicking the assignment $x_0 = \text{False}$ and $x_1 = \text{False}$ to satisfy both δ_1 and δ_2 .

Lemma 4. *There exists a unique assignment to $\phi(x_0, \dots, x_{n-1})$ if and only if the graph G constructed above has a broadcast center of size $n + 2$.*

Proof. Let $t = 2n + \lceil \log n \rceil + 3$. The proof consists of four steps.

1. We first show that

$$b(G, s) = b(G, r) = t,$$

using the following broadcast scheme. In time unit 1, s calls r (or vice versa). From time unit 2 to $d_1 + d_2 + 1$, r broadcasts within T^r . All clauses are informed by time $d_1 + d_2 + 2$, and all vertices in stars are informed by time t ($(\lceil \log c \rceil + 1) + (\lceil \log n \rceil + 1) + 3 + (2n - \lceil \log c \rceil - 2)$).

Simultaneously, s calls p_0 at time 2, and the broadcast along the path completes by time t . Starting at time 3, s calls all other r^x 's. The last r^x is informed by time $n + 2$ and its binomial tree H^x completes by time t .

2. Next, we prove that,

$$\text{If } \phi \text{ is satisfiable when } x_i = 1, \text{ then } b(G, r_{x_i}) = t.$$

This case is similar to the first, with the difference occurring at time d_1 , when the broadcast in H^{x_i} finishes. If there is a unique satisfying assignment, the leaves of H^{x_i} select $n - 1$ literals of the assignment (including x_i). The n T -trees, rooted at these literals, are informed by time $d_1 + 1$. Consequently, all clauses are informed by time $d_1 + d_2 + 2$. The rest steps follow case 1.

3. We also need to show that,

$$\text{If } \phi \text{ is not satisfiable when } x_i = 1, \text{ then } b(G, r_{x_i}) > t.$$

Consider the broadcast from r_{x_i} , it must inform s in the first time unit because the distance $d(r_{x_i}, p_{2n+\lceil \log n \rceil+1}) = t$. If the satisfying assignment is unique, since $r_{\bar{x}_i}$ is unreachable in $d_1 + 1$, there exists some clause δ_k cannot be informed by time $d_1 + d_2 + 2$. Thus, completing the star at δ_k requires at least $t + 1$ time.

4. Finally,

$$\text{For all vertices } v \text{ not in previous cases, } b(G, v) > t.$$

If v is not on the path, then $b(G, v) \geq d(v, p_0) + b(P, p_0) = d(v, r) + d(r, p_0) + b(P, p_0) \geq 1 + 2 + b(P, p_0) = 2n + \lceil \log n \rceil + 4 = t + 1$.

If v is on the path, then $b(G, v) \geq d(v, \delta_1) + b(\text{Star}, \delta_1) = d(v, s) + d(s, \delta_1) + b(\text{Star}, \delta_1) \geq 1 + 1 + d_1 + d_2 + 3 + 1 + (2n - d_1) = 2n + \lceil \log n \rceil + 5 = t + 2$.

According to the above, $b(G) = t$. If ϕ is uniquely satisfiable when $x_i = 1$, then $b(G, r_{x_i}) = b(G) = b(G, s) = b(G, r) = t$, so $|BC_G| = n + 3$. If there are multiple satisfying assignments, some vertices in case 3 are also in the broadcast center, making $|BC_G| > n + 3$. If there is no satisfying assignment, $|BC_G| = |\{s, r\}| = 2$. In conclusion, s and r are always in the broadcast center, and r_{x_i} is included if and only if ϕ is satisfiable when $x_i = 1$. \square

Summarizing lemmas 3 and 4, and definition 4 is a polynomial-time construction,

Theorem 2. BC-SIZE is Δ_2^p and D^P -hard.

5 Conclusion

In this paper, we demonstrated that the BROADCAST GRAPH problem is NP-complete by reducing it from ST-BROADCAST TIME. The NP-completeness proof of ST-BROADCAST TIME is also provided in the appendix. Moreover, we further investigated the complexity of the NP-hard problem BC-SIZE. We show that BC-SIZE is D^P -hard by reducing it from UNIQUE-SAT. Present work provides a finer complexity interval for the problem, which is between D^P -hard and Δ_2^p -complete. In future work, we aim to precisely locate the complexity of BC-SIZE within the boolean hierarchy (BH) or prove its Δ_2^p -completeness.

References

1. Averbuch, A., Shabtai, R.H., Roditty, Y.: Efficient construction of broadcast graphs. *Disc. Appl. Math.* **171**, 9–14 (2014)
2. Bermond, J.C., Hell, P., Liestman, A.L., Peters, J.G.: Sparse broadcast graphs. *Disc. Appl. Math.* **36**(2), 97–130 (1992)
3. Blass, A., Gurevich, Y.: On the unique satisfiability problem. *Inf. Control* **55**(1–3), 80–88 (1982)
4. Boyd, S., Ghosh, A., Prabhakar, B., Shah, D.: Randomized gossip algorithms. *IEEE Trans. Inf. Theory* **52**(6), 2508–2530 (2006)
5. Cai, J.Y., Gundermann, T., Hartmanis, J., Hemachandra, L.A., Sewelson, V., Wagner, K., Wechsung, G.: The boolean hierarchy i: structural properties. *SIAM J. Comput.* **17**(6), 1232–1252 (1988)
6. Cai, J.Y., Gundermann, T., Hartmanis, J., Hemachandra, L.A., Sewelson, V., Wagner, K., Wechsung, G.: The boolean hierarchy ii: applications. *SIAM J. Comput.* **18**(1), 95–111 (1989)
7. Dinneen, M.J.: The complexity of broadcasting in bounded-degree networks (1994)
8. Dinneen, M.J., Ventura, J.A., Wilson, M.C., Zakeri, G.: Compound constructions of broadcast networks. *Disc. Appl. Math.* **93**(2–3), 205–232 (1999)
9. Farley, A.M.: Minimal broadcast networks. *Networks* **9**(4), 313–332 (1979)
10. Farley, A., Hedetniemi, S., Mitchell, S., Proskurowski, A.: Minimum broadcast graphs. *Disc. Math.* **25**(2), 189–193 (1979)
11. Fertin, G., Peters, J.G., Raabe, L., Xu, C.: Odd gossiping. *Disc. Appl. Math.* **216**(4), 319–349 (2017)
12. Fomin, F.V., Fraigniaud, P., Golovach, P.A.: Parameterized complexity of broadcasting in graphs. *Theor. Comput. Sci.* **997**, 114508 (2024)
13. Grigni, M., Peleg, D.: Tight Bounds on Minimum Broadcast Networks. *SIAM J. Disc. Math.* **4**(2), 207–222 (1991)
14. Harutyunyan, H.A., Li, Z.: A simple construction of broadcast graphs. In: Du, D.-Z., Duan, Z., Tian, C. (eds.) COCOON 2019. LNCS, vol. 11653, pp. 240–253. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26176-4_20
15. Harutyunyan, H., Liestman, A., Peters, J., Richards, D.: Broadcasting and gossiping. In: Handbook of Graph Theory, pp. 1477–1494. Chapman and Hall (2013)
16. Harutyunyan, H.A.: An efficient vertex addition method for broadcast networks. *Internet Math.* **5**(3), 211–225 (2008)
17. Harutyunyan, H.A., Li, Z.: A new construction of broadcast graphs. *Disc. Appl. Math.* **280**, 144–155 (2020)
18. Harutyunyan, H.A., Li, Z.: The complexity of finding a broadcast center. In: Wu, W., Du, H. (eds.) AAIM 2021. LNCS, vol. 13153, pp. 57–70. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-93176-6_6
19. Harutyunyan, H.A., Li, Z.: Broadcast graphs using new dimensional broadcast schemes for knödel graphs. *Disc. Appl. Math.* **336**, 56–67 (2023)
20. Harutyunyan, H.A., Liestman, A.L.: More broadcast graphs. *Disc. Appl. Math.* **98**(1–2), 81–102 (1999)
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)
22. Jansen, K., Müller, H.: The minimum broadcast time problem for several processor networks. *Theor. Comput. Sci.* **147**(1), 69–85 (1995)
23. Knödel, W.: New gossips and telephones. *Disc. Math.* **13**(1), 95 (1975)

24. Labahn, R.: A minimum broadcast graph on 63 vertices. *Disc. Appl. Math.* **53**(1), 247–250 (1994)
25. Mahéo, M., Saclé, J.F.: Some minimum broadcast graphs. *Disc. Appl. Math.* **53**(1–3), 275–285 (1994)
26. Matula, D.W.: Subtree isomorphism in $\mathcal{O}(n^5/2)$. *Ann. Disc. Math.* **2**, 91–106 (1978)
27. Middendorf, M.: Minimum broadcast time is np-complete for 3-regular planar graphs and deadline 2. *Inf. Process. Lett.* **46**(6), 281–287 (1993)
28. Papadimitriou, C., Yannakakis, M.: The complexity of facets (and some facets of complexity). *J. Comput. Syst. Sci.* **28**(2), 244–259 (1984)
29. Papadimitriou, C.H., Yannakakis, M.: The complexity of restricted spanning tree problems. *J. ACM* **29**, 285–309 (1982)
30. Schaefer, M., Avenue, S.W., Umans, C.: Completeness in the polynomial-time hierarchy a compendium (2008)
31. Slater, P.J., Cockayne, E.J., Hedetniemi, S.T.: Information dissemination in trees. *SIAM J. Comput.* **10**(4), 692–701 (1981)
32. Tale, P.: Double exponential lower bound for telephone broadcast (2024)
33. Xu, J., Li, Z.: Broadcast graph is np-complete (2024). <https://arxiv.org/abs/2412.03024>
34. Zhou, J., Zhang, K.: A minimum broadcast graph on 26 vertices. *Appl. Math. Lett.* **14**(8), 1023–1026 (2001)



Maximizing the Maximum Degree in Ordered Nearest Neighbor Graphs

Péter Ágoston^{1(✉)}, Adrian Dumitrescu^{2(✉)}, Arsenii Sagdeev^{1,3(✉)}, Karamjeet Singh^{4(✉)}, and Ji Zeng^{1,5(✉)}

¹ HUN-REN Alfréd Rényi Institute of Mathematics, Budapest, Hungary
agostonp@renyi.hu

² Algoresearch L.L.C., Milwaukee, WI, USA
ad.dumitrescu@algoreresearch.org

³ Karlsruhe Institute of Technology, Karlsruhe, Germany
sagdeevarsenii@gmail.com

⁴ Indraprastha Institute of Information Technology, Delhi, India
karamjeets@iiitd.ac.in

⁵ University of California San Diego, La Jolla, CA, USA
jzeng@ucsd.edu

Abstract. For an ordered point set in a Euclidean space or, more generally, in an abstract metric space, the *ordered Nearest Neighbor Graph* is obtained by connecting each of the points to its closest predecessor by a directed edge. We show that for every set of n points in \mathbb{R}^d , there exists an order such that the corresponding ordered Nearest Neighbor Graph has maximum degree at least $\log n/(4d)$. Apart from the $1/(4d)$ factor, this bound is the best possible. As for the abstract setting, we show that for every n -element metric space, there exists an order such that the corresponding ordered Nearest Neighbor Graph has maximum degree $\Omega(\sqrt{\log n / \log \log n})$.

1 Introduction

For a given point set P in the plane, in d -dimensional Euclidean space, or, even more generally, in an arbitrary metric space, its *Nearest Neighbor Graph* is a directed graph based on minimum distances. More precisely, these points are the vertices of the Nearest Neighbor Graph, and each vertex has precisely one outgoing edge towards its closest neighbor. To make this notion well-defined, we assume that our point set is in *general position*, which here means that no point triple determines an isosceles triangle.

These graphs play an important role in modern Computational Geometry due to their relevance in the context of computing geometric shortest paths [15, 16]. Further applications of Nearest Neighbor Graphs for computing spanners, well-separated pairs, and approximate minimum spanning trees in \mathbb{R}^d can be found in the survey [20], books [9, 18], or monograph [17], see also [13]. A systematic study of the basic combinatorial properties of Nearest Neighbor Graphs dates back at least to the classical paper [12] by Eppstein, Paterson, and Yao in which,

among many other results, they made the following simple observation: two edges with the same endpoint meet at an angle of at least $\pi/3$. Hence, the maximum indegree of the Nearest Neighbor Graph of $P \subset \mathbb{R}^d$ is bounded from above by the *kissing number* of \mathbb{R}^d regardless of the size of P , see [7].

Here we extend the latter result and study the maximum indegree in a closely related class of *ordered Nearest Neighbor Graphs* introduced in [1, 10] in the context of dynamic algorithms. In this case, the vertices appear one by one, and each new vertex has precisely one outgoing edge towards its closest predecessor, see Fig. 1.

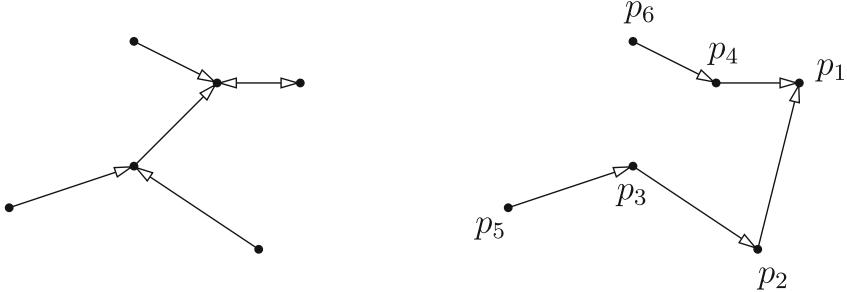


Fig. 1. Unordered (left) and ordered (right) Nearest Neighbor Graphs on the same set of six points.

It is not hard to see that every point set admits an order such that the corresponding ordered Nearest Neighbor Graph is a path, and thus its maximum indegree is 1 (Proposition 1). We therefore focus on a ‘dual’ problem, where the goal is to construct an order that *maximizes* the maximum indegree. We have the following results, partially addressing this question in three different settings.

For n points on the line we can obtain the maximum indegree about $\log n$, which is optimal. (Unless specified otherwise, all logarithms in this paper are in base 2.)

Theorem 1. *For every set of n points on the line, there exists an order such that the corresponding ordered Nearest Neighbor Graph has maximum indegree at least $\lceil \log n \rceil$. On the other hand, there is a set of n points on the line such that for every order, the indegree of each point is at most $\lceil \log n \rceil$.*

A similar maximum indegree (with a loss factor $1/(4d)$) can be obtained for n points in \mathbb{R}^d , and in view of Theorem 1, this bound is best possible apart from the loss factor.

Theorem 2. *For every set of n points in \mathbb{R}^d , there exists an order such that the corresponding ordered Nearest Neighbor Graph has maximum indegree at least $\log n/(4d)$.*

The maximum indegree we can obtain in a metric space on n points is only about $\sqrt{\log n}$, and closing the gap between this lower bound and the $\lceil \log n \rceil$ upper bound remains as a challenging open problem.

Theorem 3. *For every n -element metric space, there exists an order such that the corresponding ordered Nearest Neighbor Graph has maximum indegree $\Omega(\sqrt{\log(n)}/\log \log(n))$.*

Related Work. For a point set in the plane, the notion of (ordered or unordered) Nearest Neighbor Graph admits the following generalization. The space around each point is divided into k cones of equal angle similarly positioned around each point, and each point is connected to a ‘nearest’ neighbor in each cone. Depending on the definition of ‘nearest’ (either in a sense of the Euclidean distance or in terms of the projection on the cones’ bisectors), the resulting graph is called either a *Yao* or a *Theta Graph*. These notions were introduced by Yao [21] and Clarkson [8], respectively, and their ordered variants are due to Bose, Gudmundsson, and Morin [4]. These graphs are useful in constructing spanners with nice additional properties, such as logarithmic maximum degree and logarithmic diameter that are obtained for suitable insertion orders, see also [11]. Sparse graphs with a small dilation have been studied in [3–6], among others.

Note that the special case $k = 1$ retrieves the original definition of the (ordered) Nearest Neighbor Graph. However, for larger k , it is an open problem in the field to determine whether every point set P admits an order such that the maximum indegree of the corresponding ordered Yao or Theta Graph is bounded from above by some constant c_k , independent of the size of P , see [4]. To get a better understanding of how degrees behave in these graphs, here we attempt to maximize the maximum indegree. The general case $k \geq 2$ will be addressed in a subsequent paper.

In contrast, constructing an order minimizing the maximum indegree of the ordered Nearest Neighbor Graph is rather straightforward.

Proposition 1. *For every point set P , there is an order such that the corresponding ordered Nearest Neighbor Graph is a path. Moreover, any point in P can be chosen to be the tail of this path.*

Proof. Write $n = |P|$ and fix an arbitrary point $p_n \in P$. For $1 \leq i \leq n - 1$, let p_{n-i} be the point closest to p_{n-i+1} among the points $P \setminus \{p_n, p_{n-1}, \dots, p_{n-i+1}\}$. Now consider the order $\pi = (p_1, p_2, \dots, p_n)$. It is easy to check that the ordered Nearest Neighbor Graph G corresponding to π is just the path $p_n \rightarrow p_{n-1} \rightarrow \dots \rightarrow p_1$. Indeed, at the moment when p_i is introduced, we connect p_i to the point among $\{p_1, p_2, \dots, p_{i-1}\}$ that is closest to p_i , which is exactly p_{i-1} according to our construction of π . \square

2 Points on the Line: Proof of Theorem 1

Upper bound. We construct the desired set of points on the line recursively. For $n = 2^1$, we put $P = P_1 = \{0, 1\}$. For $k \geq 1$ and $n = 2^{k+1}$, we put $P = P_{k+1} = P_k \cup (3^k + P_k)$. It is straightforward to verify by induction that $\text{diam}(P_k) = (3^k - 1)/2$, which is smaller than the distance between the two ‘halves’ of P_{k+1} . Therefore, in every order of P_{k+1} , each point can have at most one edge incoming

from the other ‘half’. Hence by induction, the indegree of each point does not exceed $k+1 = \log n$. Finally, if $2^k < n \leq 2^{k+1}$, then an arbitrary subset $P \subset P_{k+1}$ of size n inherits the latter property.

Lower Bound. Choose k such that $\lceil \log n \rceil = k+1$, i.e., $2^k < n \leq 2^{k+1}$. The proof is by induction on k . Note that the statement is trivial for $k=0$, so let us now assume that $k \geq 1$. We identify a suitable subset $P' \subset P$ such that its reveal already forces the indegree of one of the points to reach the required threshold. The remaining points in $P \setminus P'$ are introduced afterwards and clearly do not decrease any degree. We may assume that P lies on the x -axis. We outline a recursive procedure Order for generating a suitable order that employs the following subroutines and a distinguished element of P :

- $\text{Center}(X)$: refers to a specified element of X that has a high indegree (to be determined). If $X = \{x\}$, then $\text{Center}(X) = x$.
- $\text{Order}(X)$: lists the elements of X in a suitable order that ensures the existence of $\text{Center}(X)$. Moreover, $\text{Center}(X)$ appears first in this order.
- $\text{AnyOrder}(X)$: lists the elements of X in an arbitrary order; for instance from left to right.
- $\text{Delete}(L, x)$: deletes x from list L and returns the resulting list.

Algorithm Order(P)

Step 1: Let a and b denote the leftmost and rightmost points of P .

Step 2: Partition $P = A \cup B$ around the midpoint of ab : let A be the set of points of P that are closer to a than to b (including a itself), while B is the set of remaining points; assume without loss of generality that $|A| \geq |B|$; if $|A| < |B|$ proceed analogously by switching A with B and a with b .

Step 3: List P as follows: $\text{Center}(A)$, b , $\text{Delete}(\text{Order}(A), \text{Center}(A))$, $\text{AnyOrder}(B \setminus \{b\})$.

Step 4: $\text{Center}(P) \leftarrow \text{Center}(A)$.

Analysis. Let $\Delta^-(P)$ denote the indegree of $\text{Center}(P)$. Note that $\text{Order}(A)$ is a list obtained by a recursive call and that the distance from b to A is strictly larger than the diameter of A by construction. Therefore, the indegree of $\text{Center}(A)$ is increased by 1 due to the directed edge $b \rightarrow \text{Center}(A)$ that is added when b is revealed to the algorithm as the second point. As a result,

$$\Delta^-(P) \geq \Delta^-(A) + 1 \geq k + 1,$$

where the last inequality holds by induction hypothesis since $|A| \geq 2^{k-1} + 1$. \square

3 Points in \mathbb{R}^d : Proof of Theorem 2

Here we further develop the ideas from the previous subsection to translate them from the line to \mathbb{R}^d , for any fixed d . As the main supplementary tool, we use the following standard result in discrete geometry, that can be deduced, e.g., from [19, Ineq. (4) and (6)].

Theorem 4. Let K be a convex set in \mathbb{R}^d , K_0 be its interior, and $r < 1$ be a positive number. Then K can be covered by at most $(1+1/r)^d(d \ln d + d \ln \ln d + 5d)$ translates of $-rK_0$, reflected scaled copies of K_0 .

For a finite point set $P \subset \mathbb{R}^d$, we use $\text{diam}(P)$ to denote its diameter, i.e. the maximum distance between any pair in P , and use $\text{conv}(P)$ to denote its convex hull.

Corollary 1. Let P be a finite set of points in \mathbb{R}^d such that $\text{diam}(P) \leq 1$. Then P can be partitioned into at most $16^d/2$ subsets of diameter less than $1/2$.

Proof. We apply Theorem 4 with $K = \text{conv}(P)$ and $r = 1/2$. Note that $\text{diam}(K) = \text{diam}(P) \leq 1$, and thus the distance between any two points of $-(1/2)K_0$ is strictly less than $1/2$. It remains only to note that the number of translates given by Theorem 4 is smaller than $16^d/2$ for all $d \geq 2$. \square

Now we construct the desired order of P with large maximum indegree of the corresponding ordered Nearest Neighbor Graph by the following algorithm, which is a higher-dimensional extension of the procedure we used in Sect. 2.

Algorithm Order(P)

Step 1: Compute a diameter pair ab , where we may assume that $|ab| = 1$.

Step 2: Let $A = \{p \in P : |pa| \leq |pb|\}$, and $B = \{p \in P : |pb| \leq |pa|\}$. Assume w.l.o.g. that $|A| \geq |B|$, thus $|A| \geq n/2$.

Step 3: By Corollary 1, partition A into at most $16^d/2$ subsets where each subset has diameter less than $1/2$. One of these subsets $C \subset A$ contains at least $n/16^d$ points.

Step 4: List P in the following order:

Center(C), b , Delete(Order(C), Center(C)), AnyOrder($P \setminus (C \cup \{b\})$).

Step 5: Center(P) \leftarrow Center(C)

Analysis. Order(P) is a list obtained by a recursive call. Observe that in Order(P), the element Center(P) is listed first, as required. As earlier, let $\Delta^-(P)$ denote its indegree. By construction, the indegree of Center(C) is increased by 1 due to the directed edge $b \rightarrow \text{Center}(C)$ that is added when b is revealed to the algorithm as the second point. Moreover, for each subsequent step that reveals an element $c \in C$, since $\text{diam}(C) < 1/2$ and $|cb| \geq 1/2$, C is processed in a manner independent of the presence of b . As a result,

$$\Delta^-(P) \geq \Delta^-(C) + 1.$$

We now show that $\Delta^-(P) \geq \log_{16^d} n = \log n/(4d)$. This inequality is clearly satisfied for $n = 1, 2$. By induction, it is verified that

$$\Delta^-(P) \geq \Delta^-(C) + 1 \geq \log_{16^d} (n/16^d) + 1 = \log n/(4d) - 1 + 1 = \log n/(4d).$$

\square

4 Abstract Metric Spaces: Proof of Theorem 3

Let ρ be a metric on a finite set $P = \{p_1, \dots, p_n\}$, namely a positive and symmetric function $\rho : P \times P \rightarrow \mathbb{R}_+$ satisfying the triangle inequality. So far we labeled the points of P arbitrarily just for definiteness, the desired order on them will be constructed later. We shall assume the notion of hypergraph, and the unfamiliar readers are referred to, say [14], for related definitions.

We define a 3-coloring of a complete 3-uniform hypergraph $K_n^{(3)}$ on the vertex set $[n]$ as follows: for all $1 \leq i_1 < i_2 < i_3 \leq n$, we color the triple $\{i_1, i_2, i_3\}$ by red (resp. green or blue) whenever $v_{i_2}v_{i_3}$ (resp. $v_{i_1}v_{i_3}$ or $v_{i_1}v_{i_2}$) is the ‘shortest side’ of a triangle $v_{i_1}v_{i_2}v_{i_3}$, that is $\rho(v_{i_2}, v_{i_3}) < \rho(v_{i_1}, v_{i_2})$ and $\rho(v_{i_2}, v_{i_3}) < \rho(v_{i_1}, v_{i_3})$. Recall that our points are in general position, meaning that no point triple determines an isosceles triangle, and so each triple gets exactly one color. A subhypergraph of $K_n^{(3)}$ is *monochromatic* if all its triples are of the same color. A *forward star* $S_k^{(3)}$ is a 3-uniform hypergraph on an ordered vertex set $\{i_1, \dots, i_k\} \subset [n]$, labeled in the ascending order, consisting of edges $\{i_1, i_j, i_{j'}\}$ for all $i_j < i_{j'}$. We claim that if there exists a red clique $K_k^{(3)}$, or a green forward star $S_k^{(3)}$, or a blue forward star $S_k^{(3)}$, then there exists an order such that the corresponding Nearest Neighbor Graph has maximum indegree at least $k - 1$. To verify this claim, suppose that such a structure exists on the vertex set $I = \{i_1, \dots, i_k\}$, labeled in the ascending order, and consider the following three cases.

First, suppose that these vertices form a red clique. We claim that under the order

$$v_{i_k}, v_{i_1}, v_{i_2}, \dots, v_{i_{k-1}}, \text{AnyOrder}(P \setminus I),$$

the indegree of v_{i_k} is at least $k - 1$. Indeed, note that as each new vertex v_{i_j} is revealed, $\rho(v_{i_j}, v_{i_k})$ is the shortest distance among all currently visible points. Hence, $v_{i_j} \rightarrow v_{i_k}$ is an edge of the ordered Nearest Neighbor Graph for all $1 \leq j \leq k - 1$.

Similarly, if these vertices form a blue forward star, then the indegree of v_{i_1} under the order

$$v_{i_1}, v_{i_k}, v_{i_{k-1}}, \dots, v_{i_2}, \text{AnyOrder}(P \setminus I),$$

is at least $k - 1$. Indeed, when v_{i_j} is revealed, the distance $\rho(v_{i_1}, v_{i_j})$ will be the shortest among all current points as $\{i_1, i_j, i_\ell\}$ is always blue.

Finally, if these vertices form a green forward star, then the indegree of v_{i_1} under the order

$$v_{i_1}, v_{i_2}, v_{i_3}, \dots, v_{i_k}, \text{AnyOrder}(P \setminus I),$$

is at least $k - 1$. Indeed, when v_{i_j} is revealed, an arbitrary currently visible point v_{i_ℓ} satisfies $\{i_1, i_\ell, i_j\}$ being green. So the distance $\rho(v_{i_1}, v_{i_j})$ will be the shortest.

The last ingredient of the proof is the following Ramsey type result due to He and Fox; more precisely, their Theorem 1.2 and Lemma 5.2 in [14]. We rewrite this proof to make it explicit that the argument in [14] produces forward stars and also extends to more than two colors.

Theorem 5. Let $K_n^{(3)}$ be a complete 3-uniform hypergraph on an ordered vertex set $[n]$, and its edges be colored by either red, green, or blue. If $K_n^{(3)}$ contains neither a red clique $K_k^{(3)}$, nor a green forward star $S_k^{(3)}$, nor a blue forward star $S_k^{(3)}$, then $n \leq \exp(O(k^2 \log(k)))$.

Applying this theorem with $k = c' \sqrt{\log(n) / \log \log(n)}$ for a sufficiently small $c' > 0$, we ensure the existence of a monochromatic special structure, and thus the existence of an order such that the corresponding ordered Nearest Neighbor Graph has maximum indegree at least $k - 1$, which completes the proof of Theorem 3.

Proof (of Theorem 5). We consider a process of picking and deleting vertices from $V(K_n^{(3)})$, and constructing an auxiliary graph G . We keep track of two sets U , the vertex set of G , and W , the set of vertices waiting to be picked. At the beginning of the process, we have $U = \emptyset$ and $W = V(K_n^{(3)}) = [n]$. In each iteration, we pick the smallest vertex $v \in W$, and delete it from W . Then, following a rule we shall describe, we create edges from v to vertices in U , assign v a color (red/green/blue), and add it into U . During this step, immediately after we created an edge $\{u, v\}$, we look at every vertex $w \in W$: if at least $|W|/k$ vertices w satisfy that $\{u, v, w\}$ is green, then we color the edge $\{u, v\}$ green (and vertex v green, see further below), and delete all $w \in W$ with $\{u, v, w\}$ not green; otherwise, we perform the same checking, coloring, and deleting with “blue” in place of “green”; if neither of these cases happens, we color the edge $\{u, v\}$ by red, and delete all $w \in W$ such that $\{u, v, w\}$ is not red.

Now we describe how we create edges: once a new vertex v is picked, we keep creating edges from v to the red vertices in U following the order until we encounter a green or blue edge; if the last edge we created is green (resp. blue), we color the vertex green (resp. blue); otherwise we color this vertex red. The new vertex and the created edges are considered as added into G . The next iteration begins as long as there are still vertices left in W .

Notice that if there are k red vertices in G , then by construction all the 2-edges between them are red. And again by the way we color the edges in G , these vertices correspond to a 3-uniform red clique $K_k^{(3)}$ inside $K_n^{(3)}$. Hence, without loss of generality, we assume that there are fewer than k red vertices in G . Given this assumption, if there are $(k - 1)^2$ green vertices in G , then by our construction and the pigeonhole principle, there must be one red vertex v_1 and $k - 1$ green vertices v_2, \dots, v_k such that $\{v_1, v_i\}$ is green for $2 \leq i \leq k$. Then by the way we color the edges in G , these vertices correspond to a 3-uniform green forward star $S_k^{(3)}$ inside $K_n^{(3)}$. Hence, without loss of generality, there are fewer than $(k - 1)^2$ green vertices in G , and a similar claim holds for blue vertices as well. Therefore, G has fewer than $k + 2(k - 1)^2$ vertices in total.

Next, we upper bound the number of edges in G . Since each green vertex gives exactly one green edge and there is no other source of green edges, there are fewer than $(k - 1)^2$ green edges. Similarly, there are fewer than $(k - 1)^2$ blue edges. There are at most $(k - 1)^2$ red edges between red vertices. To count other red edges, notice that the i -th red vertex has fewer than $k - 1$ green (resp. blue)

edges adjacent to it, and each endpoint of these edges is adjacent to $i - 1$ red edges. In summary, the number of red edges in G is fewer than

$$(k-1)^2 + \sum_{i=1}^{k-1} (i-1)(k-1) = k(k-1)^2/2.$$

Finally, notice that each vertex in G decreases the size of W by 1, each green or blue edge shrinks the size of W by $1/k$, and each red edge shrinks the size of W by $(1 - 2/k)$. Since by the end of our construction of G when W is depleted, G has fewer than $k + 2(k-1)^2$ vertices, fewer than $2(k-1)^2$ green or blue edges, and fewer than $k(k-1)^2/2$ red edges, the size of W in the beginning, namely n , is at most

$$(k + 2(k-1)^2)(1/k)^{-2(k-1)^2}(1 - 2/k)^{-k(k-1)^2/2} \leq \exp(O(k^2 \log(k))).$$

Here, we used an elementary estimate that $(1 - 2/k)^{-k/2} < e^2$. \square

5 Concluding Remarks

Recall from the second half of Theorem 1 that there exists a set of n points on the line such that for every order, the indegree of each point in the corresponding Nearest Neighbor Graph is at most $\lceil \log n \rceil$. Unlike the linear case, we do not know if this construction is close to being tight for larger dimensions or abstract metric spaces.

Note that both in the definition of Nearest Neighbor Graph corresponding to a metric space and in our proof of Theorem 3, we didn't really use the triangle inequality, and thus we could argue in a more general setting of *semimetric spaces*. Moreover, the *exact* distances between the points are also not important, since only their *relative order* determines the Nearest Neighbor Graph. Furthermore, we know from [2] that any order of the distances on an n -element set can be realized in the $(n-1)$ -dimensional Euclidean space. Hence, a sufficiently strong improvement upon the result of Theorem 2 could strengthen the lower bound from Theorem 3 as well.

Observe that our proof of Theorem 3 works without any changes if instead of a red clique, we could guarantee only a red ‘backward star’, which is a much sparser hypergraph. (A *backward star* is a 3-uniform hypergraph similar to a forward star, where the only difference is that the common vertex of all the hyperedges has not the smallest, but the largest index.) We are unaware of any variant of Theorem 5 that guarantees a backward star in one color or a forward star in (one of) the other color(s) for a much smaller n . Note that the proof of Theorem 5 works for *any* coloring of the complete hypergraph, while colorings that come from genuine metric spaces satisfy additional constraints due to transitivity¹.

¹ For instance, in the notation of Sect. 4, it is not possible that two triples $\{1, 2, 3\}$ and $\{1, 3, 4\}$ are blue, while $\{1, 2, 4\}$ is green, because otherwise we would have $v_1v_2 < v_1v_3 < v_1v_4 < v_1v_2$, a contradiction.

A closely related problem is the following:

Problem 1. For an n -element metric space P and $p \in P$, let $d(p)$ be the maximum indegree of p in the ordered Nearest Neighbor Graph over all $n!$ possible orders. Can $\sum_p 2^{-d(p)}$ be larger than 1?

On the one hand, note that if the sum $\sum_p 2^{-d(p)}$ is at most one, then $d(p) \geq \lceil \log n \rceil$ for at least one $p \in P$, which would strongly improve upon the result of Theorem 3. On the other hand, if $\sum_p 2^{-d(p)} > 1$ for some P , then all the points of an appropriate ‘iterated blow-up’ Q of P satisfy $d(q) < c \log |Q|$ for some $c < 1$ and all $q \in Q$, and this would improve the aforementioned upper bound from the second half of Theorem 1.

We managed to verify the inequality $\sum_p 2^{-d(p)} \leq 1$ for all $n \leq 5$ by computer search. Moreover, it is not hard to check that this sum equals exactly 1 for the metric spaces constructed in the second half of the proof of Theorem 1, and that these are not the only examples. However, computer experiments suggest that such metric spaces become rare as n grows. This can be tentatively considered as evidence towards a negative resolution of Problem 1.

Acknowledgments. The authors would like to thank the organizers of the Focused Week on Geometric Spanners (Oct 23–Oct 29, 2023) at the Erdős Center, Budapest, where this joint work began. Research partially supported by ERC Advanced Grant ‘GeoScape’ No. 882971, by the Erdős Center and by the Ministry of Innovation and Technology NRD Office within the framework of the Artificial Intelligence National Laboratory (RRF-2.3.1-21-2022-00004). Research was also partially supported by the Overseas Research Fellowship of IIIT-Delhi, India.

References

1. Agarwal, P., Eppstein, D., Matoušek, J.: Dynamic half-space reporting, geometric optimization, and minimum spanning trees. In: Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, pp. 80–90 (1992)
2. Almendra-Hernández, V.H., Martínez-Sandoval, L.: On prescribing total orders and preorders to pairwise distances of points in Euclidean space. Comput. Geom. **107**, 101898 (2022)
3. Aronov, B., de Berg, M., Cheong, O., Gudmundsson, J., Haverkort, H., Vigneron, A.: Sparse geometric graphs with small dilation. Comput. Geom. **40**(3), 207–219 (2008)
4. Bose, P., Gudmundsson, J., Morin, P.: Ordered theta graphs. Comput. Geom. **28**(1), 11–18 (2004)
5. Bose, P., Gudmundsson, J., Smid, M.: Constructing plane spanners of bounded degree and low weight. Algorithmica **42**, 249–264 (2005)
6. Bose, P., Maheshwari, A., Narasimhan, G., Smid, M., Zeh, N.: Approximating geometric bottleneck shortest paths. Comput. Geom. **29**(3), 233–249 (2004)
7. Boyvalenkov, P., Dodunekov, S., Musin, O.R.: A survey on the kissing numbers. Serdica Math. J. **38**, 507–522 (2012)

8. Clarkson, K.: Approximation algorithms for shortest path motion planning. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, pp. 56–65 (1987)
9. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry, 3rd edn. Springer, Heidelberg (2008)
10. Eppstein, D.: Fully dynamic maintenance of Euclidean minimum spanning trees and maxima of decomposable functions. Technical Report 92-88, University of California, Irvine (1992)
11. Eppstein, D.: Spanning trees and spanners. In: Sack, J.-R., Urrutia, J. (eds.) Handbook of Computational Geometry, pp. 425–461. Elsevier Science, Amsterdam (2000)
12. Eppstein, D., Paterson, M.S., Yao, F.F.: On nearest-neighbor graphs. *Discret. Comput. Geom.* **17**, 263–282 (1997)
13. Harary, F., Jacobson, M.S., Lipman, M.J., McMorris, F.R.: Abstract sphere-of-influence graphs. *Math. Comput. Model.* **17**(11), 77–83 (1993)
14. He, X., Fox, J.: Independent sets in hypergraphs with a forbidden link. *Proc. Lond. Math. Soc.* **123**(4), 384–409 (2021)
15. Mitchell, J.S.B.: Geometric shortest paths and network optimization. In: Sack, J.-R., Urrutia, J. (eds.) Handbook of Computational Geometry, pp. 633–701. Elsevier Science, Amsterdam (2000)
16. Mitchell, J.S.B., Mulzer, W.: Proximity algorithms. In: Goodman, J.E., O'Rourke, J., Tóth, C.D. (eds.) Handbook of Discrete and Computational Geometry, 3rd edn. CRC Press (2017)
17. Narasimhan, G., Smid, M.: Geometric Spanner Networks. Cambridge University Press, Cambridge (2007)
18. Preparata, F.P., Shamos, M.I.: Computational Geometry. Springer, New York (1985)
19. Rogers, C.A., Zong, C.: Covering convex bodies by translates of convex bodies. *Mathematika* **44**(1), 215–218 (1997)
20. Smid, M.: Closest-point problems in computational geometry. In: Sack, J.-R., Urrutia, J. (eds.) Handbook of Computational Geometry, pp. 877–936. Elsevier Science, Amsterdam (2000)
21. Yao, A.C.C.: On constructing minimum spanning trees in k -dimensional spaces and related problems. *SIAM J. Comput.* **11**(4), 721–736 (1982)

Author Index

A

- Ágoston, Péter 358
Akhtar, Sheikh Shakil 1
Anand, Bijo S. 292
Anil, Arun 292
Antony, Cyriac 13
Antony, Jacob 13
Ashok, Pradeesha 25

B

- Balamoorthy, S. 38
Bantva, Devsi 307
Benkoczi, Robert 210
Bhattacharya, Bhaswar B. 50
Bhyravarapu, Sriram 60, 85

C

- Chakraborty, Dipayan 73
Changat, Manoj 292

D

- Das, Arun Kumar 97, 109
Das, Sandip 50, 109
Devarakonda, Pradyun 25
Devi Yamini, S. 13
Dumitrescu, Adrian 358
Dutta, Madhura 121

F

- Feng, Zhidan 134
Fernau, Henning 134, 197
Foucaud, Florent 147

G

- Ghasemi Nezhad, Sina 160
Ghosh, Aakash 222

H

- Hakanen, Anni 173

I

- Islam, Sk Samim 109
Islam, Sk. Samim 50

J

- Jacob, Dalu 234
Jena, Sangram K. 331
Joshi, Subhasmita 234
Junnila, Ville 173

K

- Kare, Anjeneya Swami 258
Karthika, D. 85
Kavaskar, T. 38
Kirubakaran, V. K. 319
Kumar, Pritesh 85
Kumari, Swati 60

L

- Laavanya, D. 13
Laihonen, Tero 173
Li, Zhiyuan 343

M

- Madani, Amirali 185
Madathil, Jayakrishnan 1
Maheshwari, Anil 121, 185
Majumder, Atrayee 147
Mann, Kevin 134, 197
Maowa, Jannatul 210
Miikonen, Havu 173
Miraftab, Babak 185
Misra, Pranabendu 1
Mitra, Ritam Manna 109
Miyazawa, Flávio K. 247
Moghaddas, Maryam 160
Mömke, Tobias 147
Muthucumaraswamy, R. 85

N

- Nair, Revathy S. 292
Nandy, Subhas C. 121
Nanoti, Saraswati Girish 222
Narasimha-Shenoi, Prasanth G. 292

P

- Pal, Sudebkumar Prasant 307
Panda, Bhawani Sankar 234
Panolan, Fahad 160
Philip, Geevarghese 1
Phogat, Shiven 25

R

- Raman, Indhumathi 134
Rayala, Swaroop A. Ram 25
Reddy, Sangam Balchandar 258
Romero, Pablo 270, 280
Roshany-Tabrizi, Aida 147
Roy, Bodhayan 109, 185

S

- Sacher, Silas Cato 134
Sagdeev, Arsenii 358

Sanghi, Aryan 307

- Sen, Saumya 50
Shalu, M. A. 319
Sherin, J. A. 25
Singh, Karamjeet 358
Subramani, A. 331
Subramani, K. 331

V

- Valdés Ravelo, Santiago 247
Vinod Reddy, I. 60

W

- Wagler, Annegret K. 73
Wojciechowski, Piotr 331

X

- Xu, Jinghan 343

Y

- Yero, Ismael G. 173

Z

- Zeng, Ji 358