HAL open science

# 13th International Symposium on Parameterized and Exact Computation. IPEC 2018, August 22-24, 2018, Helsinki, Finland

Christophe Paul, Michal Pilipczuk

## To cite this version:

Christophe Paul, Michal Pilipczuk. 13th International Symposium on Parameterized and Exact Computation. IPEC 2018, August 22-24, 2018, Helsinki, Finland. 13th International Symposium on Parameterized and Exact Computation (IPEC 2018 ), Leibniz International Proceedings in Informatics , 115, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 978-3-95977-084-2. 10.4230/LIPIcs.IPEC.2018.0 . hal-02019803

**HAL Id: hal-02019803**

**https://hal.science/hal-02019803v1**

Submitted on 12 Apr 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

# 13th International Symposium on Parameterized and Exact Computation

**IPEC 2018, August 22-24, 2018, Helsinki, Finland**

Edited by

Christophe Paul
Michał Pilipczuk

LIPICS

LIPIcs – Vol. 115 – IPEC 2018 www.dagstuhl.de/lipics

*Editors*

Christophe Paul
LIRMM
CNRS, Université de Montpellier
France
christophe.paul@lirmm.fr

Michał Pilipczuk
Institute of Informatics
Faculty of Mathematics, Informatics, and Mechanics
of the University of Warsaw, Poland
michal.pilipczuk@mimuw.edu.pl

*ACM Classification 2012*
Theory of computation → Complexity classes, Theory of computation → Design and analysis of algorithms, Mathematics of computing → Discrete mathematics

**ISBN 978-3-95977-084-2**

*Published online and open access by*
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at http://www.dagstuhl.de/dagpub/978-3-95977-084-2.

*Publication date*
January, 2019

*Bibliographic information published by the Deutsche Nationalbibliothek*
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

*License*
This work is licensed under a Creative Commons Attribution 3.0 Unported license (CC-BY 3.0): http://creativecommons.org/licenses/by/3.0/legalcode.
In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:
- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/LIPIcs.IPEC.2018.0

## LIPIcs – Leibniz International Proceedings in Informatics

LIPIcs is a series of high-quality conference proceedings across all fields in informatics. LIPIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

*Editorial Board*

- Luca Aceto (*Chair*, Gran Sasso Science Institute and Reykjavik University)
- Susanne Albers (TU München)
- Christel Baier (TU Dresden)
- Javier Esparza (TU München)
- Michael Mitzenmacher (Harvard University)
- Madhavan Mukund (Chennai Mathematical Institute)
- Anca Muscholl (University Bordeaux)
- Catuscia Palamidessi (INRIA)
- Raimund Seidel (Saarland University and Schloss Dagstuhl – Leibniz-Zentrum für Informatik)
- Thomas Schwentick (TU Dortmund)
- Reinhard Wilhelm (Saarland University)

**ISSN 1868-8969**

**http://www.dagstuhl.de/lipics**

# Contents

Preface
*Christophe Paul and Michał Pilipczuk* ..... 0:vii

Counting Problems in Parameterized Complexity
*Radu Curticapean* ..... 1:1–1:18

A Complexity Dichotomy for Hitting Small Planar Minors Parameterized by Treewidth
*Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos* ..... 2:1–2:13

Lower Bounds for Dynamic Programming on Planar Graphs of Bounded Cutwidth
*Bas A. M. van Geffen, Bart M. P. Jansen, Arnoud A. W. M. de Kroon, and Rolf Morel* ..... 3:1–3:14

Multivariate Analysis of Orthogonal Range Searching and Graph Distances
*Karl Bringmann, Thore Husfeldt, and Måns Magnusson* ..... 4:1–4:13

A Faster Tree-Decomposition Based Algorithm for Counting Linear Extensions
*Kustaa Kangas, Mikko Koivisto, and Sami Salonen* ..... 5:1–5:13

Generalized Distance Domination Problems and Their Complexity on Graphs of Bounded mim-width
*Lars Jaffke, O-joung Kwon, Torstein J. F. Strømme, and Jan Arne Telle* ..... 6:1–6:14

A Parameterized Complexity View on Collapsing $k$-Cores
*Junjie Luo, Hendrik Molter, and Ondřej Suchý* ..... 7:1–7:14

Parameterized Complexity of Multi-Node Hubs
*Saket Saurabh and Meirav Zehavi* ..... 8:1–8:14

The Parameterised Complexity of Computing the Maximum Modularity of a Graph
*Kitty Meeks and Fiona Skerman* ..... 9:1–9:14

On the Distance Identifying Set Meta-Problem and Applications to the Complexity of Identifying Problems on Graphs
*Florian Barbero, Lucas Isenmann, and Jocelyn Thiebaut* ..... 10:1–10:14

The Parameterized Complexity of Finding Point Sets with Hereditary Properties
*David Eppstein and Daniel Lokshtanov* ..... 11:1–11:14

Dual Parameterization of Weighted Coloring
*Júlio Araújo, Victor A. Campos, Carlos Vinícius G. C. Lima, Vinícius Fernandes dos Santos, Ignasi Sau, and Ana Silva* ..... 12:1–12:14

Computing Kernels in Parallel: Lower and Upper Bounds
*Max Bannach and Till Tantau* ..... 13:1–13:14

Exploring the Kernelization Borders for Hitting Cycles
*Akanksha Agrawal, Pallavi Jain, Lawqueen Kanesh, Pranabendu Misra, and Saket Saurabh* ..... 14:1–14:14

13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Best-Case and Worst-Case Sparsifiability of Boolean CSPs
*Hubie Chen, Bart M. P. Jansen, and Astrid Pieterse* ..... 15:1–15:13

Parameterized Leaf Power Recognition via Embedding into Graph Products
*David Eppstein and Elham Havvaei* ..... 16:1–16:14

Parameterized Complexity of Independent Set in H-Free Graphs
*Édouard Bonnet, Nicolas Bousquet, Pierre Charbit, Stéphan Thomassé, and Rémi Watrigant* ..... 17:1–17:13

Multi-Budgeted Directed Cuts
*Stefan Kratsch, Shaohua Li, Dániel Marx, Marcin Pilipczuk, and Magnus Wahlström* ..... 18:1–18:14

Matching Cut: Kernelization, Single-Exponential Time FPT, and Exact Exponential Algorithms
*Christian Komusiewicz, Dieter Kratsch, and Van Bang Le* ..... 19:1–19:13

Subset Feedback Vertex Set on Graphs of Bounded Independent Set Size
*Charis Papadopoulos and Spyridon Tzimas* ..... 20:1–20:14

Integer Programming in Parameterized Complexity: Three Miniatures
*Tomáš Gavenčiak, Dušan Knop, and Martin Koutecký* ..... 21:1–21:16

Solving Target Set Selection with Bounded Thresholds Faster than $2^n$
*Ivan Bliznets and Danil Sagunov* ..... 22:1–22:14

Resolving Conflicts for Lower-Bounded Clustering
*Katrin Casel* ..... 23:1–23:14

Counting Induced Subgraphs: A Topological Approach to #W[1]-hardness
*Marc Roth and Johannes Schmitt* ..... 24:1–24:14

A Strongly-Uniform Slicewise Polynomial-Time Algorithm for the Embedded Planar Diameter Improvement Problem
*Daniel Lokshtanov, Mateus de Oliveira Oliveira, and Saket Saurabh* ..... 25:1–25:13

The PACE 2018 Parameterized Algorithms and Computational Experiments Challenge: The Third Iteration
*Édouard Bonnet and Florian Sikora* ..... 26:1–26:15

# Preface

This volume contains the papers presented at IPEC 2018: the 13th International Symposium on Parameterized and Exact Computation, which took place on August 22-24 in Helsinki, Finland. IPEC was collocated with five other algorithms conferences as a part of the annual ALGO congress.

The International Symposium on Parameterized and Exact Computation (IPEC, formerly IWPEC) is a series of international symposia covering research in all aspects of parameterized and exact algorithms and complexity. Started in 2004 as a biennial workshop, it became an annual event in 2009.

| | Previous IPEC |
|---|---|
| 2004 | Bergen, Norway |
| 2006 | Zürich, Switzerland |
| 2008 | Victoria, Canada |
| 2009 | Copenhagen, Denmark |
| 2010 | Chennai, India |
| 2011 | Saarbrücken, Germany |
| 2012 | Ljubljana, Slovenia |
| 2013 | Sophia Antipolis, France |
| 2014 | Wrocław, Poland |
| 2015 | Patras, Greece |
| 2016 | Aarhus, Denmark |
| 2017 | Vienna, Austria |

In response to the call for papers, 48 papers were submitted. Each submission was reviewed by at least 3 reviewers. The reviews came from the 15 members of the program committee, and from 50 external reviewers contributing 145 external reviews. The program committee held electronic meetings through the EasyChair platform. The program committee felt that the median submission quality was very high, and in the end selected 24 of the submissions for presentation at the symposium and for inclusion in this proceedings volume. This year, both the Best Paper Award and the Excellent Student Paper Award distinguish the paper *Counting Induced Subgraphs: A Topological Approach to* #W[1]*-hardness* co-authored by Marc Roth and Johannes Schmitt.

IPEC invited one plenary speaker to the ALGO meeting, Magnus Wahlström, as part of the award ceremony for the 2017 EATCS-IPEC Nerode Prize for outstanding papers in the area of multvariate algorithmics. The award was given by a committee consisting of Dániel Marx, Jianer Chen and Hans L. Bodlaender to Stefan Kratsch and Magnus Wahlström for their paper *Compression via Matroids: A Randomized Polynomial Kernel for Odd Cycle Transversal* (ACM Transactions on Algorithms 10(4): 20:1-20:15 (2014)). IPEC also invited Radu Curtipean to present a tutorial on *Counting Problems in Parameterized Complexity*. Finally, IPEC hosted the ceremony of the 2018 PACE awards. The Parameterized Algorithms and Computational Experiments Challenge (PACE) was conceived in Fall 2015 to deepen the relationship between parameterized algorithms and practice. This volume contains a report by Édouard Bonnet and Florian Sikora on the 2018 PACE challenge.

We would like to thank the program committee, together with the external reviewers, for their commitment in the difficult paper selection process. We also thank all the authors who submitted their work for consideration. Finally, we are grateful to the local organizers of ALGO, chaired by Parinya Chalermsook, Petteri Kaski and Jukka Suomela, for their efforts, which made chairing IPEC an enjoyable experience.

Christophe Paul and Michał Pilipczuk
Montpellier and Warsaw, November 2018

13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

# Programme Committee

| | | |
|---|---|---|
| David Coudert | Université Côte d'Azur, Inria, CNRS, I3S | France |
| Serge Gaspers | UNSW Sydney and Data61, CSIRO | Australia |
| Fedor Fomin | University of Bergen | Norway |
| Christian Knauer | Universität Bayreuth | Germany |
| Stephan Kreutzer | TU Berlin | Germany |
| Michael Lampis | Lamsade, Université Paris Dauphine | France |
| Sebastian Ordyniak | TU Wien | Austria |
| Sang-il Oum | KAIST | South Korea |
| Christophe Paul (co-chair) | LIRMM, CNRS, Université de Montpellier | France |
| Daniël Paulusma | Durham University | United Kingdom |
| Geevarghese Philip | Chennai Mathematical Institute | India |
| Michał Pilipczuk (co-chair) | University of Warsaw | Poland |
| Hisao Tamaki | Meiji University | Japan |
| Till Tantau | Universität zu Lübeck | Germany |
| Meirav Zehavi | Ben-Gurion University | Israel |

13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

# List of External Reviewers

N.R. Aravind
Julien Baste
Max Bannach
Rémy Belmonte
Édouard Bonnet
Marin Bougeret
Nicolas Bousquet
Yijia Chen
Rajesh Chitnis
Radu Curticapean
Konrad Kazimierz Dabrowski
Eduard Eiben
Bruno Escoffier
Jakub Gajarský
Archontia Giannopoulou
Petr Golovach
Daniel Gonçalves
Lars Jaffke
Peter Jonsson
Ioannis Katsikarelis
Eun Jung Kim
Dušan Knop
Van Bang Le
Euiwoong Lee
Erik Jan van Leeuwen
Kitty Meeks
Tillmann Miltzow
Pranabendu Misra
Valia Mitsou
Kamran Najeebullah
Rolf Niedermeier
Yota Otachi
Fahad Panolan
Marcin Pilipczuk
Ashutosh Rai
Felix Reidl
Ignasi Sau
Saket Saurabh
Sebastian Siebertz
Malte Skambath
Ondřej Suchý
Prafullkumar Tale
Dimitrios Thilikos
Mathias Weller
Stanislav Živný

13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

# List of Authors


Akanksha Agrawal, Hungarian Academy of Sciences, Hungary

Ana Silva, Universidade Federal do Ceará, Fortaleza, Brazil

Arnoud A. W. M. de Kroon, University of Oxford, United Kingdom

Astrid Pieterse, Eindhoven University of Technology, Netherlands

Bart M. P. Jansen, University of Eindhoven, Netherlands

Bas A. M. van Geffen, University of Oxford, United Kingdom

Carlos Vinícius G. C. Lima, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

Charis Papadopoulos, University of Ioannina, Greece

Christian Komusiewicz, Philipps-Universität Marburg, Germany

Daniel Lokshtanov, University of Bergen, Norway

Dániel Marx, Hungarian Academy of Sciences, Hungary

Danil Sagunov, Saint Petersburg Academic University of the Russian Academy of Sciences, Russia

David Eppstein, University of California, Irvine, United States

Dieter Kratsch, Université de Lorraine - Metz, France

Dimitrios M. Thilikos, CNRS, Université de Montpellier, France

Dušan Knop, University of Bergen, Norway

Édouard Bonnet, Université de Lyon, France

Elham Havvaei, University of California, Irvine, United States

Fiona Skerman, Uppsala University, Sweden

Florian Barbero, Université de Montpellier, France

Hendrik Molter, TU Berlin, Germany

Hubie Chen, University of London, United Kingdom

Ignasi Sau, CNRS, Université de Montpellier, France

Ivan Bliznets, St. Petersburg Department of Steklov Institute of Mathematics of the Russian Academy of Sciences, Russia

Jan Arne Telle, University of Bergen, Norway

Jocelyn Thiebaut, Université de Montpellier, France

Johannes Schmitt, ETH Zurich, Switzerland

Juho-Kustaa Kangas, Aalto University, Finland

Julien Baste, Sorbonne Université, France

Júlio Araújo, Universidade Federal do Ceará, Fortaleza Brazil

Junjie Luo, TU Berlin, China

Karl Bringmann, Max-Planck-Institut für Informatik, Germany

Katrin Casel, University of Potsdam, Germany

Kitty Meeks, University of Glasgow, United Kingdom

Lars Jaffke, University of Bergen, Norway

Lawqueen Kanesh, The Institute of Mathematical Sciences, Chennai, India

Lucas Isenmann, Université de Montpellier, France

Magnus Wahlström, Royal Holloway, University of London, United Kingdom

Måns Magnusson, Lund University, Sweden

Marc Roth, Saarland University, Germany



13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Marcin Pilipczuk, Institute of Informatics, University of Warsaw, Poland

Martin Koutecký, Technion, Israel

Mateus De Oliveira Oliveira, University of Bergen, Norway

Max Bannach, Universität zu Lübeck, Germany

Meirav Zehavi, Ben-Gurion University, Israel

Mikko Koivisto, University of Helsinki, Finland

Nicolas Bousquet, Grenoble INP, France

O-joung Kwon, Incheon National University, South Korea

Ondřej Suchý, Czech Technical University in Prague, Czechia

Pallavi Jain, The Institute of Mathematical Sciences, Chennai, India

Pierre Charbit, Université Paris Diderot, France

Pranabendu Misra, University of Bergen, India

Rémi Watrigant, Université de Lyon, France

Rolf Morel, University of Oxford, United Kingdom

Saket Saurabh, The Institute of Mathematical Sciences, Chennai, India

Sami Salonen, University of Helsinki, Finland

Shaohua Li, Institute of Informatics, University of Warsaw, Poland

Spyridon Tzimas, University of Ioannina, Greece

Stefan Kratsch, HU Berlin, Germany

Stéphan Thomassé, Université de Lyon, France

Thore Husfeldt, IT University of Copenhagen and Lund University, Denmark

Till Tantau, University Lübeck, Germany

Tomáš Gavenčiak, Charles University Prague, Czechia

Torstein J. F. Strømme, University of Bergen, Norway

Van Bang Le, Universität Rostock, Germany

Victor A. Campos, Universidade Federal do Ceará, Fortaleza Brazil

Vinícius Fernandes Dos Santos, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

# Counting Problems in Parameterized Complexity

**Radu Curticapean**
Basic Algorithms Research Copenhagen (BARC) and IT University of Copenhagen, Copenhagen, Denmark
radu.curticapean@gmail.com
https://orcid.org/0000-0001-7201-9905

**Abstract**

This survey is an invitation to parameterized counting problems for readers with a background in parameterized algorithms and complexity. After an introduction to the peculiarities of counting complexity, we survey the parameterized approach to counting problems, with a focus on two topics of recent interest: Counting small patterns in large graphs, and counting perfect matchings and Hamiltonian cycles in well-structured graphs.

While this survey presupposes familiarity with parameterized algorithms and complexity, we aim at explaining all relevant notions from counting complexity in a self-contained way.

**2012 ACM Subject Classification** Theory of computation → Parameterized complexity and exact algorithms, Theory of computation → Problems, reductions and completeness

**Keywords and phrases** counting complexity, parameterized complexity, graph motifs, perfect matchings, graph minor theory, Hamiltonian cycles

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.1

**Category** Invited Tutorial

**Funding** BARC is supported by Villum Foundation Grant No. 16582.

**Acknowledgements** I wish to thank all my colleagues for the discussions on this topic and the collaborations over the last years.

## 1 Introduction

Many problems in computer science ask about the *existence* of solutions, such as satisfying assignments or graph structures with certain properties. However, in both practice and theory, it may be equally important to *count* solutions.

In network science, counting problems occur in the context of *graph motifs* [79], which are small patterns in graphs that occur unexpectedly often. For instance, social networks exhibit lower numbers of induced 3-vertex paths than one would expect in comparable random networks. This is a consequence of the *triadic closure* [56], the fact that two people with a common friend are likely to be friends themselves. Such phenomena cannot be observed by merely testing existence of patterns.

In statistical physics, thermodynamic properties of discrete systems are determined by *partition functions* [59]; such functions essentially count the admissible states of systems. As an example, the *dimer model* is a graph-based system that models how atoms can pair up to two-atom molecules [90, 65, 66]. In the 1960s, while attempting to find formulas for the partition function of the dimer model, physicists invented a polynomial-time algorithm for counting perfect matchings in planar graphs.

© Radu Curticapean;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 1; pp. 1:1–1:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Counting can also help for problems that do not explicitly ask about numbers: Some of the best known algorithms for Hamiltonicity or coloring problems rely on intermediate counting problems [57, 8, 7].

## 1.1 Counting complexity

Most interesting counting problems are not known to admit polynomial-time algorithms. Sometimes, this can be explained through NP-hardness: For instance, counting the satisfying assignments of Boolean formulas is clearly NP-hard, since counting is certainly not easier than deciding existence. Such arguments however do *not* show hardness for counting problems that admit polynomial-time solvable decision versions, such as counting perfect matchings in graphs, even though these problems are presumed to be hard.

This called for a complexity theory that is specific to counting problems—such a theory was provided in 1979 by Valiant [93] in a seminal paper that introduced the complexity class #P, which captures most natural counting problems. The problems in #P are the counting versions of problems in NP: Rather than deciding the existence of a witness on a given input, the task is to count the witnesses. Counting problems are #P-hard if they admit polynomial-time Turing reductions from counting satisfying assignments of CNF formulas. As desired, this includes natural counting problems with easy decision versions, such as the problem of counting perfect matchings, as shown in Valiant's paper [93]. The #P-hardness proof for this problem required the introduction of several new reduction techniques, which we survey in Section 1.3, and showed that counting complexity amounts to more than just checking that NP-hardness reductions carry over to the counting setting.

Since the initial #P-hardness results, the study of counting problems advanced to a rich sub-area of complexity theory. In particular, large territories of counting problems were classified successfully by means of *dichotomy theorems* that sort into polynomial-time solvable and #P-hard problems [46, 16, 45, 21, 19]. For instance, a dichotomy theorem for the counting versions of *constraint satisfaction problems* was first shown a decade ago [17, 47], while the analogous result for the decision version was a long-standing open problem that was resolved only last year [18, 99]. The known dichotomy theorems for counting problems show that most interesting problems are indeed #P-hard, with only few polynomial-time solvable exceptions, such as counting perfect matchings in planar graphs [90, 65, 66], counting spanning trees [67], and problems solved by holographic algorithms [94, 20].

Thus, even more so than for NP-hard decision problems, relaxations were needed to cope with #P-hard counting problems. The most popular such coping technique is *approximate counting*, which allows outputs within a multiplicative error of the actual count [38, 24, 54, 55, 64]. Important #P-hard counting problems admit randomized polynomial-time approximation schemes; these famously include counting perfect matchings in bipartite graphs [60]. (The same result for *general* graphs remains a major open problem in approximate counting.) Counting problems were also investigated through *parameterized*, *exponential-time*, and *fine-grained* complexity [48, 77, 40]. In this survey, we focus on these paradigms.

## 1.2 Parameterized counting complexity

The multivariate paradigm has proven to be very successful for decision and optimization problems [44, 49, 82, 36], and it also enables a deeper understanding of counting problems. One of the first results on parameterized counting actually lies in the three-fold intersection of parameterization, approximation, and counting: Arvind and Raman [6] showed in 2002 how to approximately count $H$-subgraph copies in a graph $G$ to within a multiplicative error of $1 \pm \epsilon$ in randomized time $f(k, \epsilon) \cdot n^{\mathbf{tw}(H)+O(1)}$. This gives an FPT-algorithm for approximately counting $k$-paths, $k$-cycles, and $k$-matchings.

However, no FPT-algorithm is known for counting such subgraph patterns *exactly*, creating the need for a parameterized analogue of #P: While the problem of counting $k$-cliques (even approximately) is W[1]-hard through its decision version, we cannot say the same about counting $k$-paths, $k$-cycles, or $k$-matchings, as the decision versions of these problems are FPT [4]. To deal with such issues, Flum and Grohe [48] and independently McCartin [77] introduced the class #W[1], a natural parameterized version of #P: In analogy to W[1], the canonical #W[1]-hard problem is that of counting $k$-cliques in a graph, and a problem is #W[1]-hard if it admits a parameterized Turing reduction from counting $k$-cliques.

Apart from introducing the class #W[1], Flum and Grohe also proved an impressive parameterized reduction from counting $k$-cliques to counting $k$-cycles and $k$-paths, thus proving the #W[1]-hardness of these problems and paving the way for a rich complexity theory of parameterized counting problems. Techniques in parameterized counting were later used to classify the complexity of general subgraph counting problems parameterized by pattern size [10, 27, 33, 78, 63, 61, 62, 15, 85, 86, 31], to obtain FPT-approximation results for #P-hard problems [62, 63, 78, 1, 13], and to investigate classical #P-hard problems like counting perfect matchings under structural parameters [28, 35, 32, 34].

As for decision and optimization problems, more precise running time bounds for parameterized counting problems can be obtained through the exponential-time hypothesis ETH [58]. This hypothesis also admits a counting version #ETH, introduced by Dell et al. [40], which asserts that counting satisfying assignments to Boolean 3-CNF-formulas requires time $2^{\Omega(n)}$. A sparsification lemma [40] and counting-specific reduction techniques [29] are known for #ETH, enabling tight lower bounds for a variety of problems [14, 29]. Note that #ETH may be a weaker assumption than ETH.

## 1.3 Techniques for counting problems

To gain some intuition for the algorithmic boundary of parameterized counting, let us examine the fates of some techniques for parameterized algorithms as we go from decision to counting:

- To start on a positive note, *dynamic programming* often carries over to counting problems. For instance, in treewidth-based DPs, *join* nodes correspond very roughly to multiplication (more realistically, *convolution* [51]) and *forget* nodes correspond roughly to summation over the possible states of the vertex to be forgotten. Most importantly, Courcelle's theorem admits analogous counting versions [5, 26, 74]. There also is an FPT-algorithm for counting the models of FOL formulas on structures of locally bounded treewidth [52].
- On the other hand, *win-win* approaches like *bidimensionality* [42, 50] fail for counting problems when the large-treewidth case "wins" via a guaranteed solution: As an example, knowing that $k$-paths exist due to the presence of a $\sqrt{k} \times \sqrt{k}$ grid minor does a priori not facilitate counting. However, if the large-treewidth case guarantees the *absence* of the solutions sought for, then we only need to address the low-treewidth case: As mentioned above, treewidth-based DPs for decision problems often support the extension to counting.
- *Color coding* [4] fails, strictly speaking: While every $k$-set will be colorful under at least one coloring, there is otherwise no guarantee on the number of such colorings. In fact, it is known [2] that a perfectly balanced collection of colorings needs size $\Omega(n^{k/2})$ . There are however approximately balanced collections that enable approximate counting [1, 3].
- Most *kernelization* techniques preserve only the existence of solutions. However, under proper definitions of a counting kernel, some techniques like the sunflower kernel for bounded-cardinality $k$-hitting sets do carry over [91]. In Section 3.1, we will also see an application of protrusion replacement [12] that works for counting.

What we lose in algorithmic techniques when going into the counting world, we gain in new reduction techniques for proving hardness. In the following, we survey some reduction techniques for #P- and #W[1]-hardness that are unavailable for NP- and W[1]-hardness.

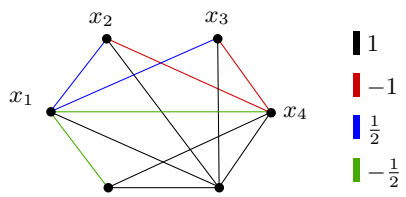
### Cancellations in gadgets

Gadgets in NP- or W[1]-hardness proofs typically serve the purpose of testing local states: They can be extended by an internal solution iff their external state is *admissible* in a problem-specific sense. In counting problems with easy decision versions, gadgets with this property may not necessarily exist, as they might imply hardness of the decision version. We must actually use the power of counting: As first done by Valiant [93], we can design gadgets that do admit unwanted internal extensions for non-admissible external states, but such that the (weighted) contributions of unwanted internal extensions *cancel out to zero*.

As an example, let us consider a weighted version of counting perfect matchings: Given an edge-weighted graph $G = (V, E, w)$, we define

$$\mathrm{PerfMatch}(G) = \sum_{M} \prod_{e \in M} w(e),$$

where $M$ ranges over the perfect matchings of $G$. The following edge-weighted gadget $\Gamma$ with special vertices $X = \{x_1, \ldots, x_4\}$ acts as an "equality gadget" for PerfMatch: For $S = \emptyset$ and $S = X$, we have $\mathrm{PerfMatch}(\Gamma - S) = 1$, and we also have $\mathrm{PerfMatch}(\Gamma - S) = 0$ for all other $S \subseteq X$. (The gadget is from [34] and was found with a computer algebra system.)




Summarizing, if the gadget $\Gamma$ is used in a larger construction, then *effectively* either all or none of $X$ must be matched by edges outside of $\Gamma$. Even though $\Gamma$ does admit internal matchings in some of the other cases, the weighted number of such matchings cancels to zero.

### Interpolation

In the counting world, Turing reductions become very powerful, as the results of oracle queries can be added, multiplied, subtracted—a luxury we do not have with Boolean values. This enables linear algebra on oracle outputs, in particular, *polynomial interpolation* [93, 92, 98, 29]. As an example of this prevalent technique, we show a classical reduction that computes $\mathrm{PerfMatch}(G)$ for an $n$-vertex graph $G$ with edge-weights $\pm 1$ when given an oracle for counting perfect matchings in unweighted graphs.

Let $x$ be an indeterminate and define a graph $G_x$ by replacing each edge of weight $-1$ in $G$ by an edge of weight $x$. Then $p := \mathrm{PerfMatch}(G_x)$ is a polynomial in $x$ of degree at most $\frac{n}{2}$ such that $p(-1) = \mathrm{PerfMatch}(G)$. If we knew the $\frac{n}{2} + 1$ values $p(0), \ldots, p(\frac{n}{2})$, we could recover all coefficients of $p(G)$ via Lagrange interpolation and could thus evaluate $p(-1)$. But the evaluation $p(i)$ for $0 \leq i \leq \frac{n}{2}$ can be obtained as $\mathrm{PerfMatch}(G_i)$ for the graph $G_i$ in which each edge of weight $x$ is replaced by an edge of weight $i$. Each such edge in turn can be simulated by a bundle of $i$ parallel unweighted edges.

Overall, we can recover the value of $\mathrm{PerfMatch}(G)$ for the $\pm 1$-weighted graph $G$ by counting perfect matchings in the unweighted graphs $G_0, \ldots, G_{n/2}$ via oracle calls and performing linear algebra on the oracle results to interpolate the polynomial $p$.

### Inclusion-exclusion

Also taking linear combinations of oracle outputs, the inclusion-exclusion principle is a very useful tool for *parameterized* hardness results [78, 39, 25, 33]. Given a universe $\Omega$ and "bad" subsets $A_1, \ldots, A_k \subseteq \Omega$, inclusion-exclusion allows us to count those elements of $\Omega$ that avoid all bad subsets, provided that we know the sizes of intersections of bad subsets:

$$\left|\Omega \setminus \bigcup_{i=1}^{k} A_i\right| = \sum_{S \subseteq \{1,\ldots,k\}} (-1)^{|S|} \, |A_S| \tag{1}$$

with $A_\emptyset := \Omega$ and $A_S := \bigcap_{i \in S} A_i$ for $\emptyset \subset S \subseteq [k]$. We can thus determine the left-hand side with $2^k$ oracle calls to numbers $|A_S|$ for $S \subseteq [k]$.

We use (1) for a parameterized reduction from the colorful subgraph isomorphism problem to the uncolored version [78, 33]. Let $H$ and $G$ be graphs such that $G$ is (not necessarily properly) vertex-colored with colors from $\{1, \ldots, k\}$ for $k = |V(H)|$. A subgraph copy of $H$ in $G$ is *colorful* if it does not avoid any color. Thus, writing $\Omega$ for the set of all $H$-copies in $G$, and defining $A_i$ for $i \in [k]$ to be the set of $H$-copies that *do avoid* color $i$, we can invoke (1) to obtain the number of colorful $H$-copies if we knew the cardinalities $|A_S|$ for $S \subseteq [k]$. But $|A_S|$ is just the number of $H$-copies avoiding all colors in $S$. These are precisely the $H$-copies in the graph $G_S$ obtained from $G$ by removing all vertices with a color among $S$ and ignoring colors. The numbers of $H$-copies in $G_S$ can then be obtained by oracle calls to counting uncolored $H$-copies, and we only need $2^k$ such numbers to evaluate (1).

Somewhat peculiarly, this reduction goes the opposite way as in the decision problem: In the decision version of subgraph isomorphism, *color-coding* reduces from the uncolored to the colorful setting, while the reverse direction is generally not known. In the counting version however, we have just seen a reduction from the uncolored to the colorful setting. Furthermore, the converse reduction is generally false: Counting $k$-paths is #W[1]-hard [48], whereas a simple FPT-algorithm counts colorful $k$-paths. See also an article by Meeks [78] for a nice overview of the relationship between colored and uncolored subgraph problems.

## Organization of this survey

After this brief introduction, we apply the multivariate perspective on counting to two specific problems: In Section 2, we count small patterns in large graphs. Complementary to that, in Section 3, we then count large patterns like perfect matchings and Hamiltonian cycles in graphs excluding fixed minors or bounded treewidth.

# 2 Counting small patterns

Large networks can often be approached by counting small patterns [79, 73]. This naturally leads to parameterized pattern counting problems: Given as input a pattern graph $H$ and a host graph $G$, determine the number of occurrences of $H$ in $G$, parameterized by the size of $H$. Such problems have been intensively studied in different communities; we survey some results in Section 2.1 and present recent developments in Section 2.2.

## 2.1 Results for individual pattern types

There are different ways of formalizing an *occurrence* of a pattern in a graph, and they can lead to stark differences in complexity. Below are some of the most common notions:

- Let us write $\mathrm{sub}(H, G)$ for the number of (not necessarily induced) *subgraphs* of $G$ that are isomorphic to $H$. Closely related is $\mathrm{emb}(H, G)$, the number of *embeddings* from $H$ to $G$: An embedding from $H$ to $G$ is an injective function $f : V(H) \to V(G)$ such that $uv \in E(H)$ implies $f(u)f(v) \in E(G)$. We have $\mathrm{emb}(H, G) = \mathrm{aut}(H) \cdot \mathrm{sub}(H, G)$, where $\mathrm{aut}(H)$ is the number of automorphisms of $H$.
- Depending on the application, we may also ask for the number $\mathrm{ind}(H, G)$ of *induced subgraphs* isomorphic to $H$. Extending this, Jerrum and Meeks introduced a more general model [62, 61]: For a fixed property $\Phi$ of $k$-vertex graphs, we can ask to determine the number $\mathrm{ind}(\Phi, G)$ of $k$-vertex subsets inducing a subgraph with property $\Phi$.
- We could also ask for the number $\hom(H, G)$ of *homomorphisms* from $H$ to $G$. These are essentially embeddings that need not be injective, that is, functions $f : V(H) \to V(G)$ such that $uv \in E(H)$ implies $f(u)f(v) \in E(G)$. Homomorphism counts appear in the setting of database queries [22], they are a fundamental notion in the theory of graph limits and graph algebras [73], and we will see in Section 2.2 that they enable a depeer understanding of subgraph and induced subgraph counting problems.

### Improved algorithms

All of the above numbers can be computed in time $O(f(k) \cdot n^k)$ by brute force; we write $k = |V(H)|$ and $n = |V(G)|$ throughout. As even a moderate value of $k$ can be prohibitively large for large host graphs $G$, the possibility of improved algorithms was investigated:

- Fast matrix multiplication solves the above problems in time $f(k) \cdot n^{\omega k/3+O(1)}$, where $\omega < 2.373$ is the exponent of matrix multiplication [81].
- For computing $\mathrm{sub}(H, G)$ and thin patterns $H$, an additional improvement to $f(k) \cdot n^{0.46k+2\mathbf{pw}(H)+O(1)}$ time is possible, where $\mathbf{pw}(H)$ denotes the pathwidth of $H$ [9].
- Again for $\mathrm{sub}(H, G)$, substantial improvements can be made when $H$ has bounded vertex-cover number $\mathbf{vc}(H)$: As an example, if $H = K_{1,k}$ is a star with $k \geq 2$ rays, then $\mathrm{sub}(H, G) = \sum_{v \in V(G)} \binom{d(v)}{k}$, where $d(v)$ denotes the degree of $v$. This formula can be evaluated in linear time. More generally, if $H$ has a vertex-cover $C$, then we can iterate over all $n^{|C|}$ maps from $C$ into $G$ and determine, for each map, the number of ways to extend the image of $C$ to a full copy of $H$. This last step can be performed in polynomial time, enabling an overall running time of $f(k) \cdot n^{\mathbf{vc}(H)+O(1)}$ [69, 97].
- We will show in Section 2.2 that $\mathrm{sub}(H, G)$ can be computed in $f(k) \cdot n^{0.174\ell+O(1)}$ time when $H$ has $\ell$ edges; this is useful for counting matchings, paths, or cycles.

It is natural to ask how far such improvements can be pushed. For fixed graphs $H$, let us write $\mathrm{sub}(H, \cdot)$, $\mathrm{ind}(H, \cdot)$ and $\hom(H, \cdot)$ for the graph parameters[1] that map input graphs $G$ to the numbers $\mathrm{sub}(H, G)$, $\mathrm{ind}(H, G)$ and $\hom(H, G)$. As it is interesting to quantify running time improvements even for algorithms that are not FPT, we define the *complexity exponent* of graph parameters like $\mathrm{sub}(H, \cdot)$ to be the infimum over all $C \in \mathbb{R}$ such that $\mathrm{sub}(H, G)$ can be evlauated in time $O(n^C)$ on input $G$ [30]. Note that the complexity exponent of $\mathrm{sub}(H, \cdot)$, $\mathrm{ind}(H, \cdot)$ and $\hom(H, \cdot)$ for fixed $H$ is always bounded from above by $k$, but it may be well below $k$, say, for $\mathrm{sub}(H, \cdot)$ where $H$ has low vertex-cover number.

[1] A graph parameter $f$ is a function from graphs into $\mathbb{Q}$. Please note that *graph parameters* are not (necessarily) *parameters* in the sense of parameterized complexity—it is a bit awkward to talk about the "parameterized complexity of graph parameters", but graph parameters are a well-established notion.

### Hardness results

We want to understand the optimal complexity exponents of graph parameters that count small patterns. Parameterized complexity allows us to aggregate graph parameters into *classes* $\mathcal{H}$ and prove statements on the asymptotic behaviour of the complexity exponents: Given a graph class $\mathcal{H}$, we define the problem $\mathrm{sub}(\mathcal{H}, \cdot)$, where the task is to compute $\mathrm{sub}(H, G)$ on input a pattern graph $H \in \mathcal{H}$ and an arbitrary host graph $G$. An analogous definition gives the problems $\mathrm{ind}(\mathcal{H}, \cdot)$ and $\mathrm{hom}(\mathcal{H}, \cdot)$.

If a problem $\mathrm{sub}(\mathcal{H}, \cdot)$ is FPT or even polynomial-time solvable, then there is a fixed constant $C$ that bounds all complexity exponents of $\mathrm{sub}(H, \cdot)$ for $H \in \mathcal{H}$. If $\mathrm{sub}(\mathcal{H}, \cdot)$ is #W[1]-hard, then we believe that no such constant exists. Furthermore, conditional lower bounds under #ETH enable lower bounds on the asymptotic growth of complexity exponents. Indeed, such lower bounds are known for counting subgraphs, induced subgraphs, and homomorphisms for restricted pattern classes $\mathcal{H}$ that are recursively enumerable.

- If $\mathcal{H}$ is finite, then $\mathrm{ind}(\mathcal{H}, \cdot)$ is polynomial-time solvable, otherwise it is #W[1]-hard [25]. Furthermore, if $\mathcal{H}$ is infinite, then there is no $f(k) \cdot n^{o(k)}$ time algorithm for $\mathrm{ind}(\mathcal{H}, \cdot)$ unless ETH fails. Note that this result holds for *every* fixed recursively enumerable class $\mathcal{H}$; it shows that no pattern structure whatsoever can be exploited for counting induced subgraphs. Jerrum and Meeks generalized this result and proved that counting $k$-vertex subgraphs with fixed properties $\Phi$ is #W[1]-hard for certain well-behaved properties $\Phi$ like connectedness or having even/odd number of edges [61, 62, 63].
- For counting not necessarily induced subgraphs, we have seen above that pattern structure *can* be exploited: If the maximum vertex-cover number of graphs in $\mathcal{H}$ is finite, then $\mathrm{sub}(\mathcal{H}, \cdot)$ is polynomial-time solvable. Otherwise the problem is known to be #W[1]-hard [33]. We will see in the next section that an $f(k) \cdot n^{o(\mathbf{vc}(H)/\log \mathbf{vc}(H))}$ time algorithm would refute #ETH, even for patterns from fixed classes $\mathcal{H}$.
- For counting homomorphisms, we may even allow bounded-treewidth patterns: If the treewidth of $\mathcal{H}$ is finite, then $\mathrm{hom}(\mathcal{H}, \cdot)$ is polynomial-time solvable; otherwise the problem is #W[1]-hard [39]. As a consequence of Marx's cornerstone "Can you beat treewidth" paper, #ETH rules out an $f(k) \cdot n^{o(\mathbf{tw}(H)/\log \mathbf{tw}(H))}$ time algorithm [76].

## 2.2 Graph motif parameters

Despite the similar appearance of the problems $\mathrm{ind}(\mathcal{H}, \cdot)$, $\mathrm{sub}(\mathcal{H}, \cdot)$, and $\mathrm{hom}(\mathcal{H}, \cdot)$, their dichotomy theorems were each proven about five years apart, by different people, using somewhat different techniques. Recently, it was shown that these problems *can* be studied in a uniform way: From the perspective of parameterized complexity, these problems become *the same problem* when "extended linearly".

More formally, graph parameters like $\mathrm{sub}(H, \cdot)$ or $\mathrm{ind}(H, \cdot)$ for fixed $H$ are special cases of so-called *graph motif parameters*, a notion introduced in [30] that adapts early works by Lovász [71, 73] to study counting small patterns from a parameterized perspective. Apart from $\mathrm{sub}(H, \cdot)$ and $\mathrm{ind}(H, \cdot)$, graph motif parameters include the graph parameters $\mathrm{ind}(\Phi, \cdot)$ that count induced $k$-vertex subgraphs with a fixed property $\Phi$, and generally, every graph parameter that depends only on the numbers of constant-sized subgraphs.

▶ **Definition 1.** A *graph motif parameter* is any graph parameter $f$ that can be written as

$$f(\cdot) = \sum_{F \in \mathcal{F}} \alpha_F \cdot \mathrm{ind}(F, \cdot) \tag{2}$$

for a finite set of graphs $\mathcal{F}$ and coefficients $\alpha_F$ for $F \in \mathcal{F}$. In other words, $f$ is a (point-wise) linear combination of a finite set of functions $\mathrm{ind}(F, \cdot)$ for fixed patterns $F$. Recall that the function $\mathrm{ind}(F, \cdot)$ takes as input a graph $G$ and outputs $\mathrm{ind}(F, G)$.

The graph motif parameter evaluation problem asks to evaluate $f(G)$ when given as input a graph $G$ and a representation of $f$ via $\mathcal{F} = \{F_1, \ldots, F_t\}$ and coefficients $\alpha_1, \ldots, \alpha_t$. We parameterize the problem by the description length of $f$.

Note that the functions $\mathrm{ind}(H, \cdot)$ for fixed $H$ are graph motif parameters themselves, but using linear combinations of such functions allows us to express much more counting functions as graph motif parameters. As an example, for any $H$, we have

$$\mathrm{sub}(H, \cdot) = \sum_{F \text{ extends } H} \beta_F \cdot \mathrm{ind}(F, \cdot), \tag{3}$$

where an unlabeled graph $F$ extends $H$ if $F$ is a supergraph of $H$ on $|V(H)|$ vertices, and the coefficient $\beta_F$ equals $\mathrm{sub}(F, H)$. Since only finitely many graphs $F$ occur in (3), the graph parameter $\mathrm{sub}(H, \cdot)$ for fixed $H$ is a graph motif parameter. Moreover, the graphs and coefficients can be computed in time depending only on $H$. It follows that the problem of computing $\mathrm{sub}(H, G)$ on input $H$ and $G$, parameterized by $H$, admits a parameterized reduction to the graph motif evaluation problem.

Interestingly, (3) admits a converse form that expresses the number of induced subgraphs $\mathrm{ind}(H, \cdot)$ as a finite linear combination of subgraph counts $\mathrm{sub}(F, \cdot)$, see [73, (5.17)]. Thus, subgraph counts are finite linear combinations of induced subgraph counts, *and vice versa.* We could thus replace ind with sub in Definition 1 and still end up defining the same objects: Any graph motif parameter $f = \sum_{F \in \mathcal{F}} \alpha_F \cdot \mathrm{ind}(F, \cdot)$ can also be written as

$$f(\cdot) = \sum_{F \in \mathcal{F}'} \beta_F \cdot \mathrm{sub}(F, \cdot). \tag{4}$$

for a finite set of graphs $\mathcal{F}'$ and coefficients $\beta_F$ for $F \in \mathcal{F}'$ that can be computed from $\mathcal{F}$ and the coefficients $\alpha_F$ for $F \in \mathcal{F}$. We call (4) the *sub-expansion* of $f$ and (2) its *ind-expansion.*[2]

### Homomorphism counts count

There are actually many equivalent ways of expanding graph motif parameters into finite linear combinations of basis functions, with explicit formulas for changing bases between such expansions [73, Chapter 5.2]. As it turns out, such basis changes enable a much better understanding of the *complexity* of graph motif parameters. In particular, expanding graph motif parameters into linear combinations of the previously neglected *homomorphism* counts will prove to be extremely useful for algorithmic purposes.

Let us have a more detailed look at the relation between embeddings[3] and homomorphisms: We can easily express the number of homomorphisms $\mathrm{hom}(H, \cdot)$ as a finite linear combination of embeddings $\mathrm{emb}(F, \cdot)$. A graph $F$ is a *quotient* of $H$ if $F$ can be obtained from $H$ by repeated identifications of vertex pairs. These identifications can be performed all at once: If $\pi$ is a partition of $V(H)$, then the quotient $H/\pi$ is obtained from $H$ by identifying, for each

[2] We could have chosen to call the parameterized counting problem from Definition 1 the "evaluation problem for graph motif parameters in the ind-expansion" and could have defined an analogous version for the sub-expansion, but the fact that the sub-expansion and the ind-expansion can be transformed into each other algorithmically implies that these problems are FPT-interreducible.

[3] Recall that an embedding is an injective homomorphism, and that $\mathrm{emb}(H, \cdot) = \mathrm{aut}(H) \cdot \mathrm{sub}(H, \cdot)$.

block $B$ of $\pi$, the vertices in $B$ to a single vertex. Quotients of graphs $H$ correspond to the images that $H$ can attain under homomorphisms, and we obtain

$$\hom(H,\cdot) = \sum_{\pi} \mathrm{emb}(H/\pi,\cdot), \tag{5}$$

where $\pi$ ranges over all partitions of $V(H)$. Different partitions can yield isomorphic quotients, and after collecting for isomorphic copies in (5) and rewriting emb = aut · sub, we obtain a formula for the sub-expansion of $\hom(H,\cdot)$.

More interestingly, as shown in [73, (5.18)], the formula (5) can be inverted[4] to read

$$\mathrm{emb}(H,\cdot) = \sum_{\pi} \underbrace{(-1)^{k-|\pi|} \prod_{B\in\pi} (|B|-1)!}_{=:\mu(\pi)} \cdot \hom(H/\pi,\cdot), \tag{6}$$

where we write $|\pi|$ for the number of blocks in $\pi$. The precise form of (6) and $\mu(\pi)$ will become important later on, but for now let us just use it to conclude that every graph motif parameter $f$ can be expressed as a linear combination of homomorphism counts

$$f(\cdot) = \sum_{F\in\mathcal{F}''} \gamma_F \cdot \hom(F,\cdot) \tag{7}$$

for a finite set of graphs $\mathcal{F}''$ and coefficients $\gamma_F$ for $F \in \mathcal{F}''$. We call (7) the *hom-expansion* of $f$. This works because every graph motif parameter has a sub-expansion via (4), and because subgraph counts are rescaled embedding counts, which in turn are linear combinations of homomorphism counts by (6).

### Complexity monotonicity of the hom-expansion

Lovász showed that homomorphism counts enjoy a wealth of mathematical properties that make them more well-behaved than subgraph or induced subgraph counts [73], and some of these properties translate almost directly into complexity-theoretic properties. Most importantly for us, the hom-expansion of graph motif parameters enjoys a monotonicity property with respect to complexity exponents: Any linear combination of homomorphism counts, as in (7), is unconditionally at least as hard to evaluate as its hardest terms.

▶ **Theorem 2** ([30, 22]). *Let $f$ be a graph motif parameter with $f = \sum_{F\in\mathcal{F}} \alpha_F \cdot \hom(F,\cdot)$ for a finite set of pairwise non-isomorphic graphs $\mathcal{F}$ with $\alpha_F \neq 0$ for all $F \in \mathcal{F}$. Then the complexity exponent of $f$ is exactly the maximum over all complexity exponents of the functions $\hom(F,\cdot)$ for $F \in \mathcal{F}$.*

▶ Remark. A result similar to Theorem 2 cannot be obtained for the sub-expansion or the ind-expansion. In these cases, we can "hide" cliques in easy linear combinations: For any $k \in \mathbb{N}$, summing over the induced subgraph counts from all $k$-vertex graphs $H$ gives $\sum_H \mathrm{ind}(H,G) = \binom{|V(G)|}{k}$, which can be evaluated in linear time for any graph $G$, even though the left-hand side contains a $k$-clique. Theorem 2 rules this out for homomorphisms.

Via Theorem 2, we obtain a good grip on the complexity of a graph motif parameter $f$ if we manage to understand (i) what patterns $F$ occur in the hom-expansion of $f$, and (ii) how hard counting homomorphisms $\hom(F,\cdot)$ is for these patterns. For the first item, we can use enumerative combinatorics to obtain formulas like (6). For the second item, we can use the known lower bounds under ETH for cliques [23] or patterns of large treewidth [76].

[4] This is done via Möbius inversion on the partition lattice, a generalization of the inclusion-exclusion principle, which can be viewed as Möbius inversion on the subset lattice [88].

### Understanding subgraph counting via homomorphisms

In the following, we use the two-step approach to give upper and lower bounds on the complexity of counting subgraphs $\mathrm{sub}(H, \cdot)$ for fixed $H$. Let us start with an upper bound.

▶ **Theorem 3** ([30]). *On input $H$ and $G$, the number $\mathrm{sub}(H, G)$ can be determined in time $k^{O(k)} \cdot n^{0.174\ell + O(1)}$ with $\ell = |E(H)|$.*

**Proof.** Consider the formula (6) for transforming embedding counts into homomorphism counts. The right-hand side sums over all partitions $\pi$ of $V(H)$, of which there are at most $k^{O(k)}$, and requires homomorphism numbers from quotients $H/\pi$ into $G$. Any quotient $H/\pi$ has at most $\ell$ edges, and the treewidth of $\ell$-edge graphs $F$ is known to be at most $\mathbf{tw}(F) \leq 0.174\ell + o(\ell)$ [87]. By a standard DP approach, we can determine $\hom(H/\pi, G)$ in time $2^{O(k)} \cdot n^{0.174\ell + O(1)}$ for every fixed $\pi$, and the overall running time bound follows. ◀

It is not much harder to prove hardness results for subgraph counting via Theorem 2:

▶ **Theorem 4.** *For any fixed recursively enumerable class $\mathcal{H}$, the problem $\mathrm{sub}(\mathcal{H}, \cdot)$ cannot be solved in time $f(k) \cdot n^{o(\mathbf{vc}(H)/\log \mathbf{vc}(H))}$ for patterns $H \in \mathcal{H}$ unless* ETH *fails.*

**Proof.** Consider $\mathrm{sub}(H, \cdot)$ for $H \in \mathcal{H}$. If $H$ has vertex-cover number $b \in \mathbb{N}$, then $H$ contains a matching $M$ with $t = b/2$ edges. We show that for every $t$-edge graph $S$, the hom-expansion of $\mathrm{sub}(H, \cdot)$ contains a supergraph of $S$ with non-zero coefficient. In particular, this holds for bounded-degree expanders $S$ of treewidth $\Omega(t)$. Supergraphs have only larger treewidth.

Every $t$-edge graph $S$ is some quotient of the $t$-matching $M$ in $H$. It follows that some supergraph of $S$ is a quotient of $H$. Using (6), we can see that every quotient $H/\pi$ appears with non-zero coefficient in the hom-expansion of $\mathrm{sub}(H, \cdot)$ even after collecting for isomorphic graphs: While different partitions $\pi, \pi'$ may lead to isomorphic quotients, this requires $|\pi| = |\pi'|$. But since the sign of $\mu(\pi)$ in (6) depends only on $|\pi|$, the contributions of $\pi$ and $\pi'$ cannot cancel. ◀

The hom-expansion of graph motif parameters is a fascinating object of study on its own that connects various mathematical areas: After the results for subgraph counting, Theorem 2 was used by Roth [85] to classify the complexity of counting homomorphism variants such as locally injective homomorphisms. This was obtained by finding large-treewidth patterns in the relevant hom-expansions through matroid theory. Very recently, an almost exhaustive #W[1]-hardness proof for counting induced $k$-vertex subgraphs with a fixed monotone property $\Phi$ was given by Roth and Schmitt [86] by using topological properties of abstract simplicial complexes. In unpublished work by the author, connections to finite model theory and chromatic polynomials were also uncovered.

## 3 Counting large patterns in "simple" graphs

In this last and shorter part of the survey, we count large objects in graphs of simple structure: In Section 3.1, we attempt to count perfect matchings in graphs that exclude fixed minors, and in Section 3.2, we count Hamiltonian cycles in graphs of bounded pathwidth.

### 3.1 Counting perfect matchings in minor-free graphs

Counting perfect matchings was the first problem shown to be #P-hard for interesting reasons, and it admits a beautiful polynomial-time algorithm on planar graphs, the *Fisher-Kasteleyn-Temperley (FKT) method* [90, 65, 66], which we already mentioned earlier. Other

than planar graphs, there are further tractable graph classes $\mathcal{C}$: For instance, while planar graphs exclude both $K_{3,3}$ and $K_5$ as minors, it was shown that excluding *either* of these two minors is sufficient [70, 96, 89]. The FKT method was also extended to an FPT-algorithm with running time $4^\gamma n^{O(1)}$ for graphs that are embedded on a surface of genus $\gamma$ [53]; together with an FPT-algorithm for finding such embeddings [80], this shows that counting perfect matchings is FPT when parameterized by genus. Furthermore, dynamic programming with subset convolution yields an $O(2^{\mathbf{tw}(G)}n)$ time algorithm when $G$ is given with an optimal tree-decomposition [95]. It is also possible to count perfect matchings in time $O(n^{\mathbf{cw}(G)+1})$ on graphs of cliquewidth $\mathbf{cw}(G)$ given with an optimal parse-tree, but unlike the previous results, this holds only for unweighted graphs [75].

The above list shows that, apart from cliquewidth, all tractability results for counting perfect matchings address graphs that exclude some fixed minor. In fact, the algorithms for $K_{3,3}$-free and $K_5$-free graphs make use of classical precursors to Robertson and Seymour's graph structure theorem for $H$-minor free graphs [84]: By Wagner's theorem [43], the $K_{3,3}$-free and $K_5$-free graphs admit tree-decompositions of adhesion 3 in which every torso is planar or has bounded size. Prior to proving the full graph structure theorem, Robertson and Seymour showed very similar decompositions for all graphs excluding minors $H$ that can be drawn in the plane with at most one crossing [83], such as $H = K_{3,3}$ and $H = K_5$. Such decompositions can be used algorithmically:

▶ **Theorem 5** ([28, 89]). *For any fixed graph $H$ that can be drawn in the plane with at most one crossing, there is an $O(n^4)$ time algorithm for counting perfect matchings in $H$-minor free graphs $G$.*

**Proof.** For single-crossing minors $H$, a tree-decomposition of $H$-free graphs $G$ into planar and bounded-treewidth torsos of adhesion 3 can be obtained in $O(f(H) \cdot n^4)$ time [41]. Given such a decomposition, the algorithm now uses a counting version of protrusion replacement [12]: When counting perfect matchings, any 3-boundaried graph $S$ can be simulated by an equivalent planar 3-boundaried graph $S'$ with edge-weights [94], and the graph $S'$ can be computed in FPT-time for planar and bounded-treewidth graphs $S$. Traversing the decomposition of $G$ bottom-up, we can successively replace torsos $S$ by planar replacement graphs $S'$ until all of $G$ is processed. ◀

How far can we push this result? In particular, is it possible to obtain polynomial-time algorithms for counting perfect matchings for *any* graph class that excludes some fixed graph $H$? More strongly, is the problem FPT when parameterized by the *Hadwiger number*, which is the maximum size of a clique minor in $G$? The answer to the second question is negative, as we can show #W[1]-hardness on $k$-apex graphs. These are the graphs that become planar after deleting at most $k$ vertices; this upper-bounds their Hadwiger number by $k + 5$.

▶ **Theorem 6** ([35]). *Counting perfect matchings in $k$-apex graphs is #W[1]-hard and an $f(k)n^{o(k/\log k)}$ time algorithm would refute #ETH.*

Thus, counting perfect matchings is also #W[1]-hard when parameterized by the Hadwiger number. However, an XP-algorithm might still be possible, e.g., because there is a simple $O(n^{k+3})$ time algorithm for counting perfect matchings in $k$-apex graphs: Simply iterate over all $n^k$ possible choices for matching the $k$ apices to vertices in the planar base graph, delete the apices together with their matching partners, and count perfect matchings in the remaining *planar* graph via the FKT method.

To recapitulate, we know that counting perfect matchings is FPT parameterized by genus, in XP parameterized by apex number, and we have seen some techniques for dealing

with clique-sums in Theorem 5. By the graph structure theorem [84], these are almost all the building blocks of $H$-minor free graphs, and thus an XP algorithm for counting perfect matchings on such graphs almost seems within reach.

Alas, we blocked out *vortices*: In general, $H$-minor free graphs allow for graphs of bounded pathwidth to be inserted at the faces of the bounded-genus parts; these structures are called vortices. It turns out that counting perfect matchings is #P-hard even in planar graphs with one single vortex, and such graphs can be seen to exclude a constant-sized minor.

▶ **Theorem 7** (Unpublished work by the author with Mingji Xia). *There is a fixed graph $H$ such that counting perfect matchings is #P-hard in graphs excluding $H$.*

Thus, polynomial-time algorithms cannot be obtained for each excluded minor $H$, but it is still open to find out *which* fixed graphs $H$ can be excluded for polynomial-time algorithms.

## 3.2 Counting Hamiltonian cycles in low-pathwidth graphs

To conclude, we briefly consider counting Hamiltonian cycles parameterized by pathwidth. The standard DP for this problem gives a running time of $O^*(\mathbf{pw}^{O(\mathbf{pw})})$ for counting and decision, but celebrated improvements [37] enabled $O^*((2+\sqrt{2})^{\mathbf{pw}})$ time for the decision problem and for counting modulo 2, and $O^*(6^{\mathbf{pw}})$ time for counting [11]. The bases $2+\sqrt{2}$ and 6 are optimal under the strong exponential-time hypothesis SETH [37, 32].

Curiously, the concrete numbers $2+\sqrt{2}$ and 6 can be explained through *connection matrices*. These are particularly nice objects in the intersection of linear algebra and combinatorics [72, 73]. Given a graph parameter $f$ such as the number of Hamiltonian cycles, and $k \in \mathbb{N}$, the $k$-th connection matrix of $f$ is an infinite matrix $C_{f,k}$, indexed by $k$-boundaried graphs, that describes the behavior of $f$ under graph separations of size $k$. The entry $C_{f,k}(G,H)$ for $k$-boundaried graphs $G, H$ is defined to be $f(G \oplus H)$, where $G \oplus H$ is the union of $k$-boundaried graphs $G$ and $H$ with matching vertices identified. Even though the matrices $C_{f,k}$ are infinite-sized, they have finite rank for many interesting graph parameters $f$, including all MSOL-definable $f$ [68]. It can be shown furthermore that evaluating graph parameters $f$ with finite-rank connection matrices is nonuniformly FPT when parameterized by treewidth [73]; this gives an alternative proof of Courcelle's theorem.

Let us focus again on the connection matrix $C_{\mathrm{HC},k}$ for the number of Hamiltonian cycles: Lovász [72] upper-bounded the rank of $C_{\mathrm{HC},k}$ over $\mathbb{Q}$ by $k^{O(k)}$. This bound was recently improved to $\Theta(6^k)$ up to polynomial factors, with a matching lower bound [32]. It was also known before that the rank of $C_{\mathrm{HC},k}$ drops to $\Theta((2+\sqrt{2})^k)$ over the ring $\mathbb{Z}_2$ with a matching lower bound [37]. Note that the constants 6 and $2+\sqrt{2}$ appearing in the rank of $C_{\mathrm{HC},k}$ over $\mathbb{Q}$ and $\mathbb{Z}_2$ are the running time bases for counting Hamiltonian cycles over these rings parameterized by pathwidth. This is a consequence of a more general statement:

▶ **Theorem 8** ([32]). *Let $\mathfrak{R}$ be any of the rings $\mathbb{Q}$ or $\mathbb{Z}_p$ for prime $p$. If the connection matrix $C_{\mathrm{HC},k}$ of the number of Hamiltonian cycles has rank $\Omega(c^k)$ over $\mathfrak{R}$, then an algorithm with running time $O^*((c-\epsilon)^{\mathbf{pw}})$ for counting Hamiltonian cycles over $\mathfrak{R}$ refutes SETH.*

While we do know that the rank of $C_{\mathrm{HC},k}$ is $\Omega(3.97^k)$ over $\mathbb{Z}_p$ when $p \neq 2$, resulting in a corresponding complexity lower bound [32], algorithms faster than $O^*(6^{\mathbf{pw}})$ are not known. It is also interesting to close this gap and to investigate to what extent the proof technique in Theorem 8 can be applied to study problems other than Hamiltonian cycles through their connection matrices.

## References


1 Noga Alon, Phuong Dao, Iman Hajirasouliha, Fereydoun Hormozdiari, and S Cenk Sahinalp. Biomolecular network motif counting and discovery by color coding. *Bioinformatics*, 24(13):i241–i249, 2008. doi:10.1093/bioinformatics/btn163.

2 Noga Alon and Shai Gutner. Balanced Hashing, Color Coding and Approximate Counting. In *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, pages 1–16, 2009. doi:10.1007/978-3-642-11269-0_1.

3 Noga Alon and Shai Gutner. Balanced families of perfect hash functions and their applications. *ACM Trans. Algorithms*, 6(3):54:1–54:12, 2010. doi:10.1145/1798596.1798607.

4 Noga Alon, Raphael Yuster, and Uri Zwick. Color-Coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.

5 Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy Problems for Tree-Decomposable Graphs. *J. Algorithms*, 12(2):308–340, 1991. doi:10.1016/0196-6774(91)90006-K.

6 Vikraman Arvind and Venkatesh Raman. Approximation Algorithms for Some Parameterized Counting Problems. In *Algorithms and Computation, 13th International Symposium, ISAAC 2002 Vancouver, BC, Canada, November 21-23, 2002, Proceedings*, pages 453–464, 2002. doi:10.1007/3-540-36136-7_40.

7 Andreas Björklund. Determinant Sums for Undirected Hamiltonicity. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 173–182, 2010. doi:10.1109/FOCS.2010.24.

8 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set Partitioning via Inclusion-Exclusion. *SIAM Journal on Computing*, 39(2):546–563, 2009. doi:10.1137/070683933.

9 Andreas Björklund, Petteri Kaski, and Lukasz Kowalik. Counting Thin Subgraphs via Packings Faster than Meet-in-the-Middle Time. *ACM Trans. Algorithms*, 13(4):48:1–48:26, 2017. doi:10.1145/3125500.

10 Markus Bläser and Radu Curticapean. Weighted Counting of $k$-Matchings Is #W[1]-Hard. In *Parameterized and Exact Computation - 7th International Symposium, IPEC 2012, Ljubljana, Slovenia, September 12-14, 2012. Proceedings*, pages 171–181, 2012. doi:10.1007/978-3-642-33293-7_17.

11 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.

12 Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) Kernelization. *J. ACM*, 63(5):44:1–44:69, 2016. doi:10.1145/2973749.

13 Cornelius Brand, Holger Dell, and Thore Husfeldt. Extensor-coding. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 151–164, 2018. doi:10.1145/3188745.3188902.

14 Cornelius Brand, Holger Dell, and Marc Roth. Fine-Grained Dichotomies for the Tutte Plane and Boolean #CSP. In *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, pages 9:1–9:14, 2016. doi:10.4230/LIPIcs.IPEC.2016.9.

15 Cornelius Brand and Marc Roth. Parameterized Counting of Trees, Forests and Matroid Bases. In *Computer Science - Theory and Applications - 12th International Computer Science Symposium in Russia, CSR 2017, Kazan, Russia, June 8-12, 2017, Proceedings*, pages 85–98, 2017. doi:10.1007/978-3-319-58747-9_10.

16 Andrei Bulatov and Martin Grohe. The complexity of partition functions. *Theoretical Computer Science*, 348(2):148–186, 2005. doi:10.1016/j.tcs.2005.09.011.

17 Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. *J. ACM*, 60(5):34, 2013. doi:10.1145/2528400.

18 Andrei A. Bulatov. A Dichotomy Theorem for Nonuniform CSPs. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 319–330, 2017. doi:10.1109/FOCS.2017.37.

19 Jin-Yi Cai and Xi Chen. Complexity of Counting CSP with Complex Weights. *J. ACM*, 64(3):19:1–19:39, 2017. doi:10.1145/2822891.

20 Jin-yi Cai and Pinyan Lu. Holographic algorithms: From art to science. *J. Comput. Syst. Sci.*, 77(1):41–61, 2011. doi:10.1016/j.jcss.2010.06.005.

21 Jin-yi Cai, Pinyan Lu, and Mingji Xia. Computational Complexity of Holant Problems. *SIAM J. Comput.*, 40(4):1101–1132, 2011. doi:10.1137/100814585.

22 Hubie Chen and Stefan Mengel. Counting Answers to Existential Positive Queries: A Complexity Classification. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 315–326, 2016. doi:10.1145/2902251.2902279.

23 Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David W. Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized NP-hard problems. *Information and Computation*, 201(2):216–231, 2005. doi:10.1016/j.ic.2005.05.001.

24 Xi Chen, Martin E. Dyer, Leslie Ann Goldberg, Mark Jerrum, Pinyan Lu, Colin McQuillan, and David Richerby. The complexity of approximating conservative counting CSPs. *J. Comput. Syst. Sci.*, 81(1):311–329, 2015. doi:10.1016/j.jcss.2014.06.006.

25 Yijia Chen, Marc Thurley, and Mark Weyer. Understanding the Complexity of Induced Subgraph Isomorphisms. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, pages 587–596, 2008. doi:10.1007/978-3-540-70575-8_48.

26 Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discrete Applied Mathematics*, 108(1-2):23–52, 2001. doi:10.1016/S0166-218X(00)00221-3.

27 Radu Curticapean. Counting Matchings of Size $k$ Is #W[1]-Hard. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 352–363, 2013. doi:10.1007/978-3-642-39206-1_30.

28 Radu Curticapean. Counting perfect matchings in graphs that exclude a single-crossing minor. *CoRR*, abs/1406.4056, 2014. URL: http://arxiv.org/abs/1406.4056, arXiv:1406.4056.

29 Radu Curticapean. Block interpolation: A framework for tight exponential-time counting complexity. *Inf. Comput.*, 261(Part):265–280, 2018. doi:10.1016/j.ic.2018.02.008.

30 Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 210–223, 2017. doi:10.1145/3055399.3055502.

31 Radu Curticapean, Holger Dell, and Marc Roth. Counting Edge-Injective Homomorphisms and Matchings on Restricted Graph Classes. In *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, pages 25:1–25:15, 2017. doi:10.4230/LIPIcs.STACS.2017.25.

32 Radu Curticapean, Nathan Lindzey, and Jesper Nederlof. A Tight Lower Bound for Counting Hamiltonian Cycles via Matrix Rank. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1080–1099, 2018. doi:10.1137/1.9781611975031.70.

**33** Radu Curticapean and Dániel Marx. Complexity of Counting Subgraphs: Only the Boundedness of the Vertex-Cover Number Counts. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 130–139, 2014. doi:10.1109/FOCS.2014.22.

**34** Radu Curticapean and Dániel Marx. Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1650–1669, 2016. doi:10.1137/1.9781611974331.ch113.

**35** Radu Curticapean and Mingji Xia. Parameterizing the Permanent: Genus, Apices, Minors, Evaluation Mod 2k. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 994–1009, 2015. doi:10.1109/FOCS.2015.65.

**36** Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

**37** Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast Hamiltonicity Checking Via Bases of Perfect Matchings. *J. ACM*, 65(3):12:1–12:46, 2018. doi:10.1145/3148227.

**38** Paul Dagum and Michael Luby. Approximating the Permanent of Graphs with Large Factors. *Theoretical Computer Science*, 102(2):283–305, 1992. doi:10.1016/0304-3975(92)90234-7.

**39** Víctor Dalmau and Peter Jonsson. The complexity of counting homomorphisms seen from the other side. *Theoretical Computer Science*, 329(1-3):315–323, 2004. doi:10.1016/j.tcs.2004.08.008.

**40** Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlen. Exponential Time Complexity of the Permanent and the Tutte Polynomial. *ACM Transactions on Algorithms*, 10(4):21, 2014. doi:10.1145/2635812.

**41** E. Demaine, M. Hajiaghayi, and D. Thilikos. Exponential Speedup of Fixed-Parameter Algorithms for Classes of Graphs Excluding Single-Crossing Graphs as Minors. *Algorithmica*, 41(4):245–267, 2005. doi:10.1007/s00453-004-1125-y.

**42** Erik D. Demaine and MohammadTaghi Hajiaghayi. The Bidimensionality Theory and Its Algorithmic Applications. *Comput. J.*, 51(3):292–302, 2008. doi:10.1093/comjnl/bxm033.

**43** Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

**44** Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. doi:10.1007/978-1-4612-0515-9.

**45** Martin Dyer, Leslie Ann Goldberg, and Mike Paterson. On counting homomorphisms to directed acyclic graphs. *J. ACM*, 54, December 2007. doi:10.1145/1314690.1314691.

**46** Martin E. Dyer and Catherine S. Greenhill. The complexity of counting graph homomorphisms. *Random Struct. Algorithms*, 17(3-4):260–289, 2000. doi:10.1002/1098-2418(200010/12)17:3/4\%3C260::AID-RSA5\%3E3.0.CO;2-W.

**47** Martin E. Dyer and David Richerby. An Effective Dichotomy for the Counting Constraint Satisfaction Problem. *SIAM J. Comput.*, 42(3):1245–1274, 2013. doi:10.1137/100811258.

**48** Jörg Flum and Martin Grohe. The parameterized complexity of counting problems. *SIAM Journal on Computing*, 33(4):892–922, 2004. doi:10.1137/S0097539703427203.

**49** Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.

**50** Fedor V. Fomin, Erik D. Demaine, and Mohammad Taghi Hajiaghayi. Bidimensionality. In *Encyclopedia of Algorithms*. Springer, 2015. doi:10.1007/978-3-642-27848-8_47-2.

**51** Fedor V. Fomin and Dieter Kratsch. *Subset Convolution*, pages 125–139. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. doi:10.1007/978-3-642-16533-7_7.

52 Markus Frick. Generalized Model-Checking over Locally Tree-Decomposable Classes. *Theory Comput. Syst.*, 37(1):157–191, 2004. doi:10.1007/s00224-003-1111-9.

53 Anna Galluccio and Martin Loebl. On the Theory of Pfaffian Orientations. I. Perfect Matchings and Permanents. *Electronic Journal of Combinatorics*, 6, 1998.

54 Leslie Ann Goldberg and Mark Jerrum. Approximating the Tutte polynomial of a binary matroid and other related combinatorial polynomials. *J. Comput. Syst. Sci.*, 79(1):68–78, 2013. doi:10.1016/j.jcss.2012.04.005.

55 Leslie Ann Goldberg and Mark Jerrum. The Complexity of Approximately Counting Tree Homomorphisms. *TOCT*, 6(2):8, 2014. doi:10.1145/2600917.

56 Mark S. Granovetter. The Strength of Weak Ties. *American Journal of Sociology*, 78(6):1360–1380, 1973.

57 Thore Husfeldt. Invitation to Algorithmic Uses of Inclusion-Exclusion. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, pages 42–59, 2011. doi:10.1007/978-3-642-22012-8_3.

58 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. System Sci.*, 63(4):512–530, 2001.

59 A. Isihara. *Statistical physics.* Academic Press, 1971.

60 M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM*, 51(4):671–697, 2004. doi:10.1145/1008731.1008738.

61 Mark Jerrum and Kitty Meeks. Some hard families of parameterized counting problems. *ACM Transactions on Computation Theory*, 7(3):11, 2015. doi:10.1145/2786017.

62 Mark Jerrum and Kitty Meeks. The parameterised complexity of counting connected subgraphs and graph motifs. *J. Comput. Syst. Sci.*, 81(4):702–716, 2015. doi:10.1016/j.jcss.2014.11.015.

63 Mark Jerrum and Kitty Meeks. The parameterised complexity of counting even and odd induced subgraphs. *Combinatorica*, pages 1–26, 2016. doi:10.1007/s00493-016-3338-5.

64 Mark Jerrum and Alistair Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing*, 22(5):1087–1116, 1993. doi:10.1137/0222066.

65 Pieter W. Kasteleyn. The statistics of dimers on a lattice: I. The number of dimer arrangements on a quadratic lattice. *Physica*, 27(12):1209–1225, 1961. doi:10.1016/0031-8914(61)90063-5.

66 Pieter W. Kasteleyn. Graph Theory and Crystal Physics. In *Graph Theory and Theoretical Physics*, pages 43–110. Academic Press, 1967.

67 Gustav Kirchhoff. Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird. *Annalen der Physik und Chemie*, LXXIL(12), 1847.

68 Tomer Kotek and Johann A. Makowsky. Connection Matrices and the Definability of Graph Parameters. *Logical Methods in Computer Science*, 10(4), 2014. doi:10.2168/LMCS-10(4:1)2014.

69 Miroslaw Kowaluk, Andrzej Lingas, and Eva-Marta Lundell. Counting and Detecting Small Subgraphs via Equations. *SIAM J. Discrete Math.*, 27(2):892–909, 2013. doi:10.1137/110859798.

70 Charles Little. An Extension of Kasteleyn's method of enumerating the 1-factors of planar graphs. In *Combinatorial Mathematics*, LNCS, pages 63–72. Springer, 1974. doi:10.1007/BFb0057377.

71 László Lovász. Operations with structures. *Acta Mathematica Hungarica*, 18(3-4):321–328, 1967.

**72** László Lovász. The rank of connection matrices and the dimension of graph algebras. *Eur. J. Comb.*, 27(6):962–970, 2006. doi:10.1016/j.ejc.2005.04.012.

**73** László Lovász. *Large networks and graph limits*, volume 60. American Mathematical Society Providence, 2012.

**74** Johann A. Makowsky. Algorithmic uses of the Feferman-Vaught Theorem. *Annals of Pure and Applied Logic*, 126(1-3):159–213, 2004. Provinces of logic determined. Essays in the memory of Alfred Tarski. Parts I, II and III. doi:10.1016/j.apal.2003.11.002.

**75** Johann A. Makowsky, Udi Rotics, Ilya Averbouch, and Benny Godlin. Computing Graph Polynomials on Graphs of Bounded Clique-Width. In *Graph-Theoretic Concepts in Computer Science, 32nd International Workshop, WG 2006, Bergen, Norway, June 22-24, 2006, Revised Papers*, pages 191–204, 2006. doi:10.1007/11917496_18.

**76** Dániel Marx. Can You Beat Treewidth? *Theory of Computing*, 6(1):85–112, 2010. doi:10.4086/toc.2010.v006a005.

**77** Catherine McCartin. Parameterized counting problems. *Ann. Pure Appl. Logic*, 138(1-3):147–182, 2006. doi:10.1016/j.apal.2005.06.010.

**78** Kitty Meeks. The challenges of unbounded treewidth in parameterised subgraph counting problems. *Discrete Applied Mathematics*, 198:170–194, 2016. doi:10.1016/j.dam.2015.06.019.

**79** Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002. doi:10.1126/science.298.5594.824.

**80** Bojan Mohar. A Linear Time Algorithm for Embedding Graphs in an Arbitrary Surface. *SIAM J. Discrete Math.*, 12(1):6–26, 1999. URL: http://epubs.siam.org/sam-bin/dbq/article/29248.

**81** Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

**82** Rolf Niedermeier. *Invitation to fixed-parameter algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2006.

**83** Neil Robertson and Paul D. Seymour. Excluding a graph with one crossing. In *Proceedings of the AMS-IMS-SIAM Joint Summer Research Conference on Graph Minors, Held June 22 to July 5, 1991*, pages 669–675, 1993.

**84** Neil Robertson and Paul D. Seymour. Graph Minors. XVI. Excluding a non-planar graph. *J. Comb. Theory, Ser. B*, 89(1):43–76, 2003. doi:10.1016/S0095-8956(03)00042-X.

**85** Marc Roth. Counting Restricted Homomorphisms via Möbius Inversion over Matroid Lattices. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, pages 63:1–63:14, 2017. doi:10.4230/LIPIcs.ESA.2017.63.

**86** Marc Roth and Johannes Schmitt. Counting Induced Subgraphs: A Topological Approach to #W[1]-hardness. *CoRR*, abs/1807.01920, 2018. arXiv:1807.01920.

**87** Alexander D. Scott and Gregory B. Sorkin. Linear-programming design and analysis of fast algorithms for Max 2-CSP. *Discrete Optimization*, 4(3-4):260–287, 2007. doi:10.1016/j.disopt.2007.08.001.

**88** Richard P. Stanley. *Enumerative Combinatorics*, volume 1. Cambridge University Press, second edition, 2011.

**89** Simon Straub, Thomas Thierauf, and Fabian Wagner. Counting the Number of Perfect Matchings in $K_5$-Free Graphs. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 66–77, 2014. doi:10.1109/CCC.2014.15.

**90** H. N. V. Temperley and Michael E. Fisher. Dimer problem in statistical mechanics - an exact result. *Philosophical Magazine*, 6(68):1478–6435, 1961.

**91** Marc Thurley. Kernelizations for Parameterized Counting Problems. In *Theory and Applications of Models of Computation, 4th International Conference, TAMC 2007, Shanghai, China, May 22-25, 2007, Proceedings*, pages 703–714, 2007. doi:10.1007/978-3-540-72504-6_64.

**92** Salil P. Vadhan. The Complexity of Counting in Sparse, Regular, and Planar Graphs. *SIAM J. Comput.*, 31(2):398–427, 2001. doi:10.1137/S0097539797321602.

**93** Leslie G. Valiant. The complexity of computing the permanent. *Theoret. Comput. Sci.*, 8(2):189–201, 1979.

**94** Leslie G. Valiant. Holographic Algorithms. *SIAM J. Comput.*, 37(5):1565–1594, 2008. doi:10.1137/070682575.

**95** Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic Programming on Tree Decompositions Using Generalised Fast Subset Convolution. In *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, pages 566–577, 2009. doi:10.1007/978-3-642-04128-0_51.

**96** Vijay V. Vazirani. NC algorithms for computing the number of perfect matchings in $K_{3,3}$-free graphs and related problems. *Inf. Comput.*, 80(2):152–164, 1989.

**97** Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM Journal on Computing*, 42(3):831–854, 2013. doi:10.1137/09076619X.

**98** Mingji Xia, Peng Zhang, and Wenbo Zhao. Computational complexity of counting problems on 3-regular planar graphs. *Theoretical Computer Science*, 384(1):111–125, 2007. Theory and Applications of Models of Computation. doi:10.1016/j.tcs.2007.05.023.

**99** Dmitriy Zhuk. A Proof of CSP Dichotomy Conjecture. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 331–342, 2017. doi:10.1109/FOCS.2017.38.

# A Complexity Dichotomy for Hitting Small Planar Minors Parameterized by Treewidth

**Julien Baste**
Sorbonne Université, Laboratoire d'Informatique de Paris 6, LIP6, Paris, France
julien.baste@uni-ulm.de

**Ignasi Sau**
LIRMM, CNRS, Université de Montpellier, Montpellier, France
ignasi.sau@lirmm.fr
https://orcid.org/0000-0002-8981-9287

**Dimitrios M. Thilikos**
LIRMM, CNRS, Université de Montpellier, Montpellier, France, and
Department of Mathematics, National and Kapodistrian University of Athens, Greece
sedthilk@thilikos.info
https://orcid.org/0000-0003-0470-1800

## Abstract

For a fixed graph $H$, we are interested in the parameterized complexity of the following problem, called $\{H\}$-M-Deletion, parameterized by the treewidth $\mathsf{tw}$ of the input graph: given an $n$-vertex graph $G$ and an integer $k$, decide whether there exists $S \subseteq V(G)$ with $|S| \leq k$ such that $G \setminus S$ does not contain $H$ as a minor. In previous work [IPEC, 2017] we proved that if $H$ is planar and connected, then the problem cannot be solved in time $2^{o(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ under the $\mathsf{ETH}$, and can be solved in time $2^{\mathcal{O}(\mathsf{tw} \cdot \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$. In this article we manage to classify the optimal asymptotic complexity of $\{H\}$-M-Deletion when $H$ is a connected planar graph on at most 5 vertices. Out of the 29 possibilities (discarding the trivial case $H = K_1$), we prove that 9 of them are solvable in time $2^{\Theta(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$, and that the other 20 ones are solvable in time $2^{\Theta(\mathsf{tw} \cdot \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$. Namely, we prove that $K_4$ and the $\mathsf{diamond}$ are the only graphs on at most 4 vertices for which the problem is solvable in time $2^{\Theta(\mathsf{tw} \cdot \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$, and that the $\mathsf{chair}$ and the $\mathsf{banner}$ are the only graphs on 5 vertices for which the problem is solvable in time $2^{\Theta(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$. For the version of the problem where $H$ is forbidden as a *topological* minor, the case $H = K_{1,4}$ can be solved in time $2^{\Theta(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$. This exhibits, to the best of our knowledge, the first difference between the computational complexity of both problems.

**2012 ACM Subject Classification** Mathematics of computing → Graph algorithms, Theory of computation → Parameterized complexity and exact algorithms

**Keywords and phrases** parameterized complexity, graph minors, treewidth, hitting minors, topological minors, dynamic programming, Exponential Time Hypothesis

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.2

**Related Version** All the results and proofs of this article, as well as those of [1] and some other ones, are permanently available at `https://arxiv.org/abs/1704.07284`.

**Funding** Work supported by projects DEMOGRAPH (ANR-16-CE40-0028) and ESIGMA (ANR-17-CE23-0010).

**Acknowledgements** We would like to thank Eun Jung Kim and Édouard Bonnet for insightful discussions on the topic of this paper.

© Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 2; pp. 2:1–2:13
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

# 1 Introduction

Let $H$ be a fixed graph. In the $\{H\}$-M-Deletion (resp. $\{H\}$-TM-Deletion) problem, we are given an $n$-vertex graph $G$ and an integer $k$, and the objective is to decide whether there exists a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G \setminus S$ does not contain $H$ as a minor (resp. topological minor). These problems belongs to the more general category of *graph modification problems*. The cases where $H$ is planar and connected are already quite general, as the cases $H = K_2$ and $H = K_3$ correspond to Vertex Cover and Feedback Vertex Set, respectively. We are interested in the parameterized complexity of $\{H\}$-M-Deletion and $\{H\}$-TM-Deletion taking as the parameter the treewidth of $G$, denoted by $\mathsf{tw}$.

Determining the optimal asymptotic complexity of $\{H\}$-M-Deletion parameterized by treewidth has been an active area in the parameterized complexity community during the last years. As relevant examples, Vertex Cover is easily solvable in time $2^{\mathcal{O}(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$, called *single-exponential*, by standard dynamic-programming techniques, and no algorithm with running time $2^{o(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ exists unless the Exponential Time Hypothesis ($\mathsf{ETH}$)[1] fails [10]. For Feedback Vertex Set, the existence of a single-exponential algorithm remained open for a while, until Cygan et al. [6] presented the *Cut&Count* technique. See also [2, 9, 11, 15].

We recently studied these problems in [1] and proved[2], among other results, that if $H$ is planar and connected, then the problems cannot be solved in time $2^{o(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ under the $\mathsf{ETH}$, and can be solved in time $2^{\mathcal{O}(\mathsf{tw} \cdot \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ (for $\{H\}$-TM-Deletion, we additionally need $H$ to have maximum degree at most 3). We also presented a dichotomy when $H = C_i$ is a cycle on $i$ vertices, by proving that both problems (which are clearly equivalent for subcubic graphs) can be solved in single-exponential time if and only if $i \leq 4$. We aimed at a similar dichotomy when $H = P_i$ is a path on $i$ vertices, but we left open the case $H = P_5$.

In this article we obtain the following results, the lower bounds holding under the $\mathsf{ETH}$:

1. When $H = K_{1,i}$, the star with $i$ leaves, $\{K_{1,i}\}$-M-Deletion is solvable in time $2^{\Theta(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ for $i \leq 3$, and in time $2^{\Theta(\mathsf{tw} \cdot \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ for $i \geq 4$. On the other hand, $\{K_{1,i}\}$-TM-Deletion can be solved in time $2^{\Theta(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ for every $i \geq 1$. To the best of our knowledge, this is the first example of a graph $H$ for which the complexity of $\{H\}$-M-Deletion and $\{H\}$-TM-Deletion differ.
2. When $H = \theta_i$, the multigraph consisting of two vertices and $i \geq 1$ parallel edges, both problems can be solved in time $2^{\Theta(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ for $i \leq 2$, and in time $2^{\Theta(\mathsf{tw} \cdot \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ for $i \geq 3$. The same dichotomy occurs when $H = K_{2,i}$.
3. We classify the optimal asymptotic complexity of $\{H\}$-M-Deletion when $H$ is a connected planar graph on at most 5 vertices. Out of the 29 possibilities (discarding the trivial case $H = \{K_1\}$), we prove that 9 of them are solvable in time $2^{\Theta(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$, and that the other 20 ones are solvable in time $2^{\Theta(\mathsf{tw} \cdot \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$; a summary is shown in Figure 1. Note that $K_4$ and the $\mathsf{diamond}$ are the only graphs on at most 4 vertices for which the problem is solvable in time $2^{\Theta(\mathsf{tw} \cdot \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$, and that the $\mathsf{chair}$ and the $\mathsf{banner}$ are the only graphs on 5 vertices for which the problem is solvable in time $2^{\Theta(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$. In particular, this settles the complexity of $\{P_5\}$-M-Deletion, which we left open in [1]. All the lower bounds also hold for $\{H\}$-TM-Deletion, with the only difference that the case $H = K_{1,4}$ can be solved in time $2^{\Theta(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$, as mentioned in item **1** above.

[1] The $\mathsf{ETH}$ states that 3-SAT on $n$ variables cannot be solved in time $2^{o(n)}$; see [10] for more details.

[2] In [1] we considered the more general case where all the graphs in a fixed finite family $\mathcal{F}$ are forbidden as (topological) minors. For simplicity, we only consider here the case where $\mathcal{F}$ contains a single graph.

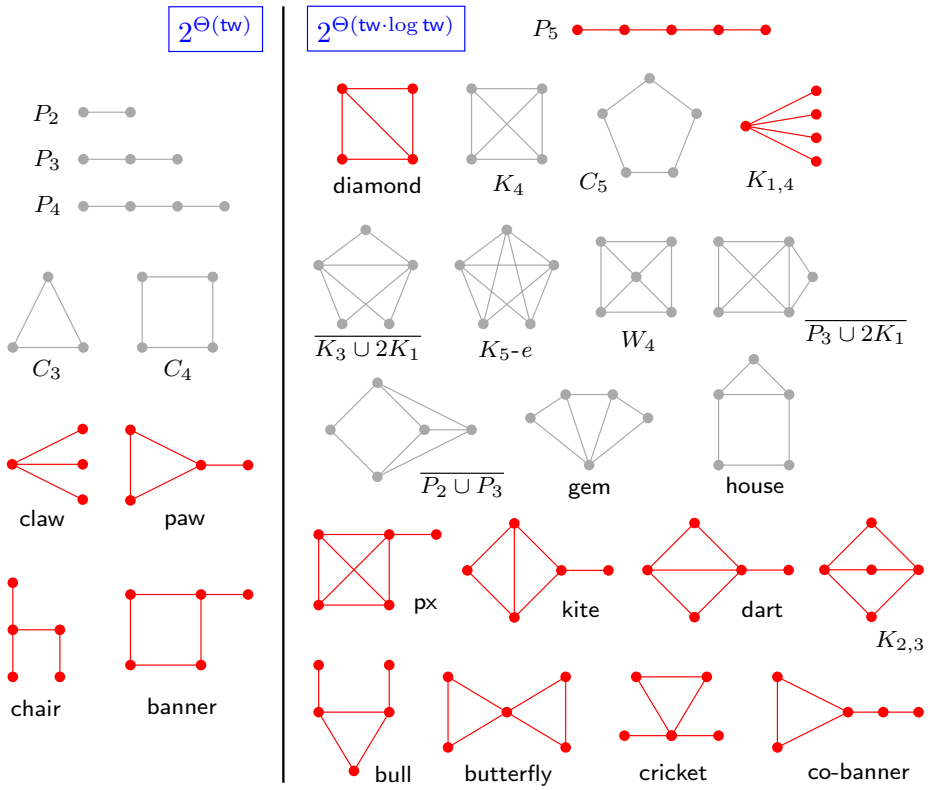



**Figure 1** Classification of the complexity of $\{H\}$-M-Deletion for all connected simple planar graphs $H$ with $|V(H)| \leq 5$ and $|E(H)| \geq 1$: for the 9 graphs on the left (resp. 20 graphs on the right), the problem is solvable in time $2^{\Theta(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ (resp. $2^{\Theta(\mathsf{tw}\cdot\log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$). For $\{H\}$-TM-Deletion, $K_{1,4}$ should be on the left. The results of this article correspond to the red (darker) graphs; the other ones were either already known (namely, $P_2$, $P_3$, and $C_3$) or obtained in [1].

Let us discuss about the techniques that we used to obtain the above results. The single-exponential algorithms are ad hoc, some being easier than others. All of them exploit a structural characterization of the graphs that exclude that particular graph as a (topological) minor; cf. for instance Lemma 1. Intuitively, the "complexity" of this characterization is what determines the difficulty of the corresponding dynamic programming algorithm, and is also what makes the difference between being solvable in single-exponential time or not.

More precisely, the algorithms for $\{K_{1,s}\}$-TM-Deletion are simple and use standard dynamic programming techniques on graphs of bounded treewidth. The algorithm for {paw}-TM-Deletion is more involved and uses the rank-based approach introduced by Bodlaender et al. [2], similarly to the algorithm for $\{C_4\}$-TM-Deletion that we presented in [1]. Finally, the algorithms for {chair}-TM-Deletion and {banner}-TM-Deletion are a combination of some of the algorithms given here and in [1], the latter one using the rank-based approach.

The superexponential lower bounds of this article are inspired by a reduction of Bonnet et al. [4]. Namely, we prove subexponential lower bounds for $P_5$, $K_{1,i}$ with $i \geq 4$ for the minor version, $K_{2,i}$ and $\theta_i$ for $i \geq 3$ (note that if $G$ is a simple graph, $\{\theta_3\}$-Deletion is equivalent to {diamond}-Deletion), and the following graphs depicted in Figure 1: the px, the kite, the dart, the bull, the butterfly, the cricket, and the co-banner. All these reductions are based on a general construction (cf. Section 4.1), and then we need particular small gadgets to deal with each of the graphs. On the other hand, one can easily check that all the other graphs

on the right hand side of Figure 1, namely $K_4$, $C_5$, $\overline{K_3 \cup 2K_1}$, $K_5$-$e$, $W_4$, $\overline{P_3 \cup 2K_1}$, $\overline{P_2 \cup P_3}$, the gem, and the house, satisfy the general condition given in [1, Theorem 20], and therefore the superexponential lower bound follows for both problems and each of these graphs. This completes the dichotomy for all connected (simple) planar graphs on at most 5 vertices.

The remainder of this article is organized as follows. In Section 2 we provide some preliminaries. The single-exponential algorithms are presented in Section 3 and the superexponential lower bounds in Section 4. We conclude the article in Section 5. Due to space constraints, the proofs of the results marked with '($\star$)' can be found in the full version of this article.

## 2 Preliminaries

We use standard graph-theoretic notation, and we refer to [7] for any undefined term and the notions of minor and topological minor. We also refer to [8, 5] for the basic definitions of parameterized complexity, tree decompositions, and treewidth. We need to introduce nice tree decompositions, which make the presentation of the algorithms much simpler.

Let $\mathcal{D} = (T, \mathcal{X})$ be a tree decomposition of $G$, $r$ be a vertex of $T$, and $\mathcal{G} = \{G_t \mid t \in V(T)\}$ be a collection of subgraphs of $G$, indexed by the vertices of $T$. We say that the triple $(\mathcal{D}, r, \mathcal{G})$ is a *nice tree decomposition* of $G$ if the following conditions hold:

- $X_r = \emptyset$ and $G_r = G$,
- each node of $\mathcal{D}$ has at most two children in $T$,
- for each leaf $t \in V(T)$, $X_t = \emptyset$ and $G_t = (\emptyset, \emptyset)$. Such $t$ is called a *leaf node*,
- if $t \in V(T)$ has exactly one child $t'$, then either
  - $X_t = X_{t'} \cup \{v_{\text{insert}}\}$ for some $v_{\text{insert}} \notin X_{t'}$ and $G_t = G[V(G_{t'}) \cup \{v_{\text{insert}}\}]$. The node $t$ is called *introduce vertex* node and the vertex $v_{\text{insert}}$ is the *insertion vertex* of $X_t$,
  - $X_t = X_{t'} \setminus \{v_{\text{forget}}\}$ for some $v_{\text{forget}} \in X_{t'}$ and $G_t = G_{t'}$. The node $t$ is called *forget vertex* node and $v_{\text{forget}}$ is the *forget vertex* of $X_t$.
- if $t \in V(T)$ has exactly two children $t'$ and $t''$, then $X_t = X_{t'} = X_{t''}$, and $E(G_{t'}) \cap E(G_{t''}) = \emptyset$. The node $t$ is called a *join* node.

For each $t \in V(T)$, we denote by $V_t$ the set $V(G_t)$. Given a tree decomposition, it is possible to transform it in polynomial time to a *nice* new one of the same width [12]. Moreover, by Bodlaender et al. [3] we can find in time $2^{\mathcal{O}(\mathsf{tw})} \cdot n$ a tree decomposition of width $\mathcal{O}(\mathsf{tw})$ of any graph $G$. Hence, since in this article we focus on single-exponential algorithms, we may assume that a nice tree decomposition of width $w = \mathcal{O}(\mathsf{tw})$ is given with the input.

If a graph $G$ contains a graph $H$ as a minor (resp. topological minor), we denote it by $H \preceq_{\mathsf{m}} G$ (resp. $H \preceq_{\mathsf{tm}} G$). For a fixed graph $H$ and a graph $G$, we define the parameter $\mathbf{m}_{\{H\}}(G)$ (resp. $\mathbf{tm}_{\{H\}}(G)$) as the minimum size of a set $S \subseteq V(G)$ such that $H \not\preceq_{\mathsf{m}} G \setminus S$ (resp. $H \not\preceq_{\mathsf{tm}} G \setminus S$).

## 3 Single-exponential algorithms

In this section we present single-exponential algorithms for hitting particular graphs. Since the algorithms when $H = K_{1,s}$ use standard dynamic programming techniques, they have been moved to the full version. In what follows we present a single-exponential algorithm for {paw}-TM-Deletion. For completeness, the basic ingredients and notations of the rank-based approach of Bodlaender et al. [2] are given in the full version. The algorithms for the chair and the banner are also given in the full version.

We start with a simple structural characterization of the simple graphs that exclude the paw as a topological minor; recall the paw graph in Figure 1.

▶ **Lemma 1.** *A simple graph $G$ satisfies $\mathsf{paw} \not\preceq_{\mathsf{tm}} G$ if and only if each connected component of $G$ is either a cycle or a tree.*

**Proof.** It is easy to see that neither a cycle nor a tree contain the $\mathsf{paw}$ as a topological minor. Let $G$ be a graph such that $\mathsf{paw} \not\preceq_{\mathsf{tm}} G$. Let us assume w.l.o.g. that $G$ is connected. If $G$ does not contain a cycle, then it is a tree. Otherwise, let $C$ be a chordless cycle in $G$. If $G$ contains a vertex $v$ that is not in $C$, then, as $G$ is connected, there exists a path from $v$ to $C$ containing at least 2 vertices. This is not possible, as it would imply that $G$ contains the $\mathsf{paw}$ as a topological minor. As $C$ is chordless and $G$ is simple, we obtain that $G$ is exactly the cycle $C$, and the lemma follows. ◀

We present an algorithm that solves the decision version of $\{\text{PAW}\}$-TM-DELETION. As the algorithm that we presented for $\{C_4\}$-TM-DELETION in [1], this algorithm is based on the one given in [2, Section 3.5] for FEEDBACK VERTEX SET. Let $G$ be a graph and $k$ be an integer. The idea of the following algorithm is to partition $V(G)$ into three sets. The first one will be the solution set $S$, the second one will be a set $F$ of vertices that induces a forest, and the third one will be a set $C$ of vertices that induces a collection of cycles. If we can partition our graph into three such sets $(S, F, C)$ such that there is no edge between a vertex of $F$ and a vertex of $C$ and such that $|S| \leq k$, then, using Lemma 1, we know that $\mathbf{tm}_{\{\mathsf{paw}\}}(G) \leq k$. On the other hand, if such a partition does not exist, we know that $\mathbf{tm}_{\{\mathsf{paw}\}}(G) > k$. The main idea of this algorithm is to combine classical dynamic programming techniques in order to verify that $C$ induces a collection of cycles, and the rank-based approach in order to verify that $F$ induces a forest.

As for $\{C_4\}$-TM-DELETION (see [1]), we define a new graph $G_0 = (V(G) \cup \{v_0\}, E(G) \cup E_0)$, where $v_0$ is a new vertex and $E_0 = \{\{v_0, v\} \mid v \in V(G)\}$. For each subgraph $H$ of $G_0$, for each $Z_1 \subseteq V(H)$, and for each $Y \subseteq E_0 \cap E(H[Z_1])$, we denote by $H\langle Z_1, Y\rangle$ the graph $\big(Z_1, Y \cup E\big(H[Z_1 \setminus \{v_0\}]\big)\big)$.

Given a nice tree decomposition of $G$ of width $w$, we define a nice tree decomposition $((T, \mathcal{X}), r, \mathcal{G})$ of $G_0$ of width $w+1$ such that the only empty bags are the root and the leaves and for each $t \in T$, if $X_t \neq \emptyset$, then $v_0 \in X_t$. Note that this can be done in linear time. For each bag $t$, each integers $i$, $j$, and $\ell$, each function $\mathbf{s} : X_t \to \{0, 1, 2_0, 2_1, 2_2\}$, each function $\mathbf{s}_0 : \{v_0\} \times \mathbf{s}^{-1}(1) \to \{0, 1\}$, and each partition $p \in \Pi(\mathbf{s}^{-1}(1))$, we define:

$$
\begin{aligned}
\mathcal{E}_t(p, \mathbf{s}, \mathbf{s}_0, i, j, \ell) \quad = \quad & \{(Z_1, Z_2, Y) \mid (Z_1, Z_2, Y) \in 2^{V_t} \times 2^{V_t} \times 2^{E_0 \cap E(G_t)},\ Z_1 \cap Z_2 = \emptyset, \\
& |Z_1| = i,\ |Z_2| = \ell, |E(G_t[Z_1 \setminus \{v_0\}]) \cup Y| = j, \\
& \forall e \in E_0 \cap E_t,\ \mathbf{s}_0(e) = 1 \Leftrightarrow e \in Y, \\
& \forall v \in Z_2 \cap X_t,\ \mathbf{s}(v) = 2_z \text{ with } z = \mathsf{deg}_{G_t[Z_2]}(v), \\
& \forall v \in Z_2 \setminus X_t,\ \mathsf{deg}_{G_t[Z_2]}(v) = 2, \\
& Z_1 \cap X_t = \mathbf{s}^{-1}(1),\ v_0 \in X_t \Rightarrow \mathbf{s}(v_0) = 1, \\
& \forall u \in Z_1 \setminus X_t :\ \text{either } t \text{ is the root or} \\
& \qquad \exists u' \in \mathbf{s}^{-1}(1) : u \text{ and } u' \text{ are connected in } G_t\langle Z_1, Y\rangle, \\
& \forall v_1, v_2 \in \mathbf{s}^{-1}(1) : p \sqsubseteq V_t[\{v_1, v_2\}] \Leftrightarrow v_1 \text{ and } v_2 \text{ are con-} \\
& \qquad\qquad \text{nected in } G_t\langle Z_1, Y\rangle, \\
& \forall (u, v) \in (Z_1 \setminus \{v_0\}) \times Z_2,\ \{u, v\} \notin E(G_t)\}
\end{aligned}
$$

$$
\mathcal{A}_t(\mathbf{s}, \mathbf{s}_0, i, j, \ell) \quad = \quad \{p \mid p \in \Pi(\mathbf{s}^{-1}(1)),\ \mathcal{E}_t(p, \mathbf{s}, \mathbf{s}_0, i, j, \ell) \neq \emptyset\}.
$$

In the definition of $\mathcal{E}_t$, the sets $Z_1$ (resp. $Z_2$) correspond to the set $F$ (resp. $C$) restricted to $G_t$. The vertex $v_0$ and the set $Y$ exist to ensure that $F$ will be connected.

By Lemma 1, we have that the given instance of {PAW}-TM-DELETION is a YES-instance if and only if for some $i$ and $\ell$, $i+\ell \geq |V(G) \cup \{v_0\}| - k$ and $\mathcal{A}_r(\varnothing, i, i-1, \ell) \neq \emptyset$. For each $t \in V(T)$, we assume that we have already computed $\mathcal{A}_{t'}$ for every children $t'$ of $t$, and we proceed to the computation of $\mathcal{A}_t$. As usual, we distinguish several cases depending on the type of node $t$.

**Leaf.** By definition of $\mathcal{A}_t$, we have $\mathcal{A}_t(\varnothing, \varnothing, 0, 0, 0) = \{\emptyset\}$.

**Introduce vertex.** Let $v$ be the insertion vertex of $X_t$, let $t'$ be the child of $t$, let $\mathbf{s} : X_t \to \{0, 1, 2_0, 2_1, 2_2\}$, $\mathbf{s}_0 : \{v_0\} \times \mathbf{s}^{-1}(1) \to \{0, 1\}$, and let $H = G_t\langle \mathbf{s}^{-1}(1), \mathbf{s}_0^{-1}(1)\rangle$.

- If $v = v_0$ and $\mathbf{s}(v_0) \in \{0, 2_0, 2_1, 2_2\}$, then by definition of $\mathcal{A}_t$ we have that $\mathcal{A}_t(\mathbf{s}, \mathbf{s}_0, i, j, \ell) = \emptyset$.
- Otherwise, if $v = v_0$, then by construction of the nice tree decomposition, we know that $t'$ is a leaf of $T$ and so $\mathbf{s} = \{(v_0, 1)\}$, $j = \ell = i - 1 = 0$ and $\mathcal{A}_t(\mathbf{s}, \mathbf{s}_0, i, j, \ell) = \mathsf{ins}(\{v_0\}, \mathcal{A}_{t'}(\varnothing, \varnothing, 0, 0, 0))$.
- Otherwise, if $\mathbf{s}(v) = 0$, then, by definition of $\mathcal{A}_t$, it holds that $\mathcal{A}_t(\mathbf{s}, \mathbf{s}_0, i, j, \ell) = \mathcal{A}_{t'}(\mathbf{s}|_{X_{t'}}, \mathbf{s}|_{E_{t'}}, i, j, \ell)$.
- Otherwise, if $\mathbf{s}(v) = 2_z$, $z \in \{0, 1, 2\}$, then let $Z_2' = N_{G_t[X_t]}(v) \setminus \mathbf{s}^{-1}(0)$. If $Z_2' \not\subseteq \mathbf{s}^{-1}(\{2_1, 2_2\})$ or $|Z_2'| \neq z$, then $\mathcal{A}_t(\mathbf{s}, \mathbf{s}_0, i, j, \ell) = \emptyset$. Otherwise $Z_2' \subseteq \mathbf{s}^{-1}(\{2_1, 2_2\})$ and $|Z_2'| = z$, and with $\mathbf{s}' : X_{t'} \to \{0, 1, 2_0, 2_1, 2_2\}$ defined such that $\forall v' \in X_{t'} \setminus Z_2'$, $\mathbf{s}'(v') = \mathbf{s}(v')$ and for each $v' \in Z_2'$ such that $\mathbf{s}(v') = 2_{z'}$, $z' \in \{1, 2\}$, $\mathbf{s}'(v') = 2_{z'-1}$. It holds that $\mathcal{A}_t(\mathbf{s}, \mathbf{s}_0, i, j, \ell) = \mathcal{A}_{t'}(\mathbf{s}', \mathbf{s}_0, i, j, \ell - 1)$.
- Otherwise, we know that $v \neq v_0$, $\mathbf{s}(v) = 1$, and $v_0 \in N_{G[\mathbf{s}^{-1}(1)]}(v)$. First, if $N_{G_t[X_t]}(v) \setminus \mathbf{s}^{-1}(0) \not\subseteq \mathbf{s}^{-1}(1)$, then $\mathcal{A}_t(\mathbf{s}, \mathbf{s}_0, i, j, \ell) = \emptyset$. Indeed, this implies that the cycle part and the forest part are connected. As $\mathbf{s}(v) = 1$, we have to insert $v$ in the forest part and we have to make sure that all vertices of $N_H[v]$ are in the same connected component of $H$. The only remaining choice is to insert the edge $\{v, v_0\}$ or not. Again, this is handled by the function $\mathbf{s}_0$. By adding $v$, we add one vertex and $|N_H(v)|$ edges in the forest part. Therefore, we have that
$$\begin{aligned}\mathcal{A}_t(\mathbf{s}, \mathbf{s}_0, \mathbf{r}, i, j, \ell) \quad &= \\ &\mathsf{glue}(N_H[v], \mathsf{ins}(\{v\}, \mathcal{A}_{t'}(\mathbf{s}|_{X_{t'}}, \mathbf{s}_0|_{E_{t'}}, i-1, j - |N_H(v)|, \ell))).\end{aligned}$$

**Forget vertex.** Let $v$ be the forget vertex of $X_t$, let $t'$ be the child of $t$, and let $\mathbf{s} : X_t \to \{0, 1, 2_0, 2_1, 2_2\}$. As a vertex from the collection of cycles can be removed only if it has exactly two neighbors, we obtain that
$$\begin{aligned}\mathcal{A}_t(\mathbf{s}, i, j, \ell) \quad = \quad & A_{t'}(\mathbf{s} \cup \{(v, 0)\}, \mathbf{s}_0, i, j, \ell) \\ & \uplus\, \mathsf{proj}(\{v\}, A_{t'}(\mathbf{s} \cup \{(v, 1)\}, \mathbf{s}_0 \cup \{(\{v_0, v\}, 0)\}, i, j, \ell)) \\ & \uplus\, \mathsf{proj}(\{v\}, A_{t'}(\mathbf{s} \cup \{(v, 1)\}, \mathbf{s}_0 \cup \{(\{v_0, v\}, 1)\}, i, j, \ell)) \\ & \uplus\, A_{t'}(\mathbf{s} \cup \{(v, 2_2)\}, \mathbf{s}_0, i, j, \ell).\end{aligned}$$

**Join.** Let $t'$ and $t''$ be the two children of $t$, let $\mathbf{s} : X_t \to \{0, 1, 2_0, 2_1, 2_2\}$, $\mathbf{s}_0 : \{v_0\} \times \mathbf{s}^{-1}(1) \to \{0, 1\}$, and let $H = G_t\langle \mathbf{s}^{-1}(1), \mathbf{s}_0^{-1}(1)\rangle$. Given three functions $\mathbf{s}^*, \mathbf{s}', \mathbf{s}'' : X_t \to \{0, 1, 2_0, 2_1, 2_2\}$, we say that $\mathbf{s}^* = \mathbf{s}' \oplus \mathbf{s}''$ if for each $v \in \mathbf{s}^{-1}(\{0, 1\})$, $\mathbf{s}^*(v) = \mathbf{s}'(v) = \mathbf{s}''(v)$, and for each $v \in X_t$ such that $\mathbf{s}^*(v) = 2_z$, $z \in \{0, 1, 2\}$, there exist $z', z'' \in \{0, 1, 2\}$ such that $\mathbf{s}'(v) = 2_{z'}$, $\mathbf{s}''(v) = 2_{z''}$, and $z = z' + z'' - \mathsf{deg}_{G_t[X_t \setminus \mathbf{s}^{-1}(0)]}(v)$.

We join every compatible entries $A_{t'}(\mathbf{s}', \mathbf{s}'_0, i', j', \ell')$ and $A_{t''}(\mathbf{s}'', \mathbf{s}''_0, i'', j'', \ell'')$. For two such entries being compatible, we need $\mathbf{s}' \oplus \mathbf{s}''$ to be defined and $\mathbf{s}'_0 = \mathbf{s}''_0$. We obtain that

$$\mathcal{A}_t(\mathbf{s}, \mathbf{s}_0, i, j, \ell) = \biguplus_{\substack{\mathbf{s}', \mathbf{s}'' : X_t \to \{0,1,2_0,2_1,2_2\}, \\ \mathbf{s} = \mathbf{s}_1 \oplus \mathbf{s}_2 \\ i'+i''=i+|V(H)| \\ j'+j''=j+|E(H)| \\ \ell'+\ell''=\ell+|\mathbf{s}^{-1}(\{2_0,2_1,2_2\})|}} \mathsf{join}(A_{t'}(\mathbf{s}', \mathbf{s}_0, i', j', \ell'), A_{t''}(\mathbf{s}'', \mathbf{s}_0, \mathbf{r}'', i'', j'', \ell'')).$$

▶ **Theorem 2.** $\{$PAW$\}$-TM-DELETION *can be solved in time* $2^{\mathcal{O}(\mathsf{tw})} \cdot n^7$.

**Proof.** The algorithm works in the following way. For each node $t \in V(T)$ and for each entry $M$ of its table, instead of storing $\mathcal{A}_t(M)$, we store $\mathcal{A}'_t(M) = \mathsf{reduce}(\mathcal{A}_t(M))$ by using [2, Theorem 3.7]. As each of the operations we use preserves representation by [2, Lemma 3.6], we obtain that for each node $t \in V(T)$ and for each possible entry $M$, $\mathcal{A}'_t(M)$ represents $\mathcal{A}_t(M)$. In particular, we have that $\mathcal{A}'_r(M) = \mathsf{reduce}(\mathcal{A}_r(M))$ for each possible entry $M$. Using the definition of $\mathcal{A}_r$ and Lemma 1, we have that $\mathbf{tm}_{\{\mathsf{paw}\}}(G) \leq k$ if and only if for some $i$ and $\ell$, $i + \ell \geq |V(G) \cup \{v_0\}| - k$ and $\mathcal{A}'_r(\varnothing, i, i-1, \ell) \neq \emptyset$.

We now focus on the running time of the algorithm. The size of the intermediate sets of weighted partitions for a leaf node and for an introduce vertex node, are upper-bounded by $2^{|\mathbf{s}^{-1}(1)|}$. For a forget vertex node, we take the union of four sets of size $2^{|\mathbf{s}^{-1}(1)|}$, so the intermediate sets of weighted partitions have size at most $4 \cdot 2^{|\mathbf{s}^{-1}(1)|}$. For a join node, as in the big union operation we take into consideration at most $5^{|X_t|}$ possible functions $\mathbf{s}'$, as many functions $\mathbf{s}''$, at most $n + |\mathbf{s}^{-1}(1)|$ choices for $i'$ and $i''$, at most $n + |\mathbf{s}^{-1}(1)|$ choices for $j'$ and $j''$ (as we can always assume, during the algorithm, that $H$ is a forest), and at most $n + |\mathbf{s}^{-1}(\{2_0, 2_1, 2_2\}|$ choices for $\ell'$ and $\ell''$, we obtain that the intermediate sets of weighted partitions have size at most $25^{|X_t|} \cdot (n + |\mathbf{s}^{-1}(1)|)^2 \cdot (n + |\mathbf{s}^{-1}(\{2_0, 2_1, 2_2\}|) \cdot 4^{|\mathbf{s}^{-1}(1)|}$. We obtain that the intermediate sets of weighted partitions have size at most $(n + |X_t|)^3 \cdot 100^{|X_t|}$. Moreover, for each node $t \in V(T)$, the function $\mathsf{reduce}$ will be called as many times as the number of possible entries, i.e., at most $2^{\mathcal{O}(w)} \cdot n^3$ times. Thus, using [2, Theorem 3.7], $\mathcal{A}'_t$ can be computed in time $2^{\mathcal{O}(w)} \cdot n^6$. The theorem follows by taking into account the linear number of nodes in a nice tree decomposition. ◀

## 4 Superexponential lower bounds

Let $\mathcal{Q} = \{P_5\} \cup \{K_{2,s} \mid s \geq 3\} \cup \{\theta_s \mid s \geq 3\}$ and let $\mathcal{R} = \{K_{1,s} \mid s \geq 4\}$. In this section, we provide superexponential lower bounds for $\{H\}$-M-DELETION when $H \in \mathcal{Q} \cup \mathcal{R}$, for $\{H\}$-TM-DELETION when $H \in \mathcal{Q}$, and for both problems for some particular graphs on 5 vertices depicted in Figure 1, namely the cricket, the px, the butterfly, the co-banner, the bull, the kite, and the dart. As mentioned before, the reductions of this section are strongly inspired by the ideas of Bonnet et al. [4]. More precisely, we will prove the following theorems.

▶ **Theorem 3.** *Let $H \in \mathcal{Q} \cup \mathcal{R}$. Unless the ETH fails, $\{H\}$-M-DELETION cannot be solved in time $2^{o(\mathsf{tw} \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$.*

▶ **Theorem 4.** *Let $H \in \mathcal{Q}$. Unless the ETH fails, $\{H\}$-TM-DELETION cannot be solved in time $2^{o(\mathsf{tw} \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$.*

▶ **Theorem 5.** *Let $H \in \{\{\mathsf{cricket}\}, \{\mathsf{px}\}, \{\mathsf{butterfly}\}, \{\mathsf{co\text{-}banner}\}, \{\mathsf{bull}\}, \{\mathsf{kite}\}, \{\mathsf{dart}\}\}$. Unless the ETH fails, neither $\{H\}$-M-DELETION nor $\{H\}$-TM-DELETION can be solved in time $2^{o(\mathsf{tw} \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$.*

In the following we focus on the proof of Theorem 3 and Theorem 5. Theorem 4 can be proved by using the same reductions as for Theorem 3 by just replacing "minor" by "topological minor" and "-M-Deletion" by "-TM-Deletion". Note that this holds when $H \in \mathcal{Q}$ but not when $H \in \mathcal{R}$.

We first provide in Section 4.1 a general framework that will be used for every $H \in \mathcal{Q} \cup \mathcal{R}$ and the graphs $H$ listed in Theorem 5, and then we explain how to modify this framework for each specific $H$. Namely, in Section 4.2 we deal with $P_5$ and in Section 4.3 with the stars. The other cases can be found in the full version of this article. All these proofs for particular graphs are quite similar and follow the same structure, but we need different gadgets and slight changes in the analysis to deal with each of the graphs $H$.

## 4.1 The general construction

In order to prove Theorems 3 and 5, we present several reductions from $k \times k$ Permutation Independent Set, introduced by Lokshtanov et al. [14].

| |
|---|
| $k \times k$ Permutation Independent Set<br>**Input:** An integer $k$ and a graph $G$ with vertex set $[1,k] \times [1,k]$.<br>**Parameter:** $k$.<br>**Output:** Is there an independent set of size $k$ in G with exactly one element from each row and one element from each column? |

▶ **Theorem 6** (Lokshtanov et al. [14]). *The $k \times k$ Permutation Independent Set problem cannot be solved in time $2^{o(k \log k)}$ unless the ETH fails.*

The general construction that we proceed to present only depends on the number of vertices of $H$. Let $H \in \mathcal{Q} \cup \mathcal{R} \cup \{\{\text{cricket}\}, \{\text{px}\}, \{\text{butterfly}\}, \{\text{co-banner}\}, \{\text{bull}\}, \{\text{kite}\}, \{\text{dart}\}\}$ and let $(G, k)$ be an instance of $k \times k$ Permutation Independent Set. As we are asking for an independent set that contains exactly one vertex in each row, we will assume w.l.o.g. that, for each $(i,j)$, $(i,j')$ in $V(G)$, $\{(i,j),(i,j')\} \in E(G)$. Let $n = |V(G)|$ and $m = |E(G)|$. We proceed to construct a graph $F$ that displays the encoding of the $m$ subgraphs of $G$ consisting of exactly one edge. This is done in such a way that each edge is encoded exactly once. Moreover, these encodings are arranged in a cyclic way separated by gadgets ensuring the consistency of the selected solution.

Namely, we first define the graph $K = K_{h-1}$ where $h = |V(H)|$. For each $e \in E(G)$, and each $(i,j) \in [1,k]^2$, we define the graph $B^e_{i,j}$ to be the disjoint union of two copies of $K$ and two new vertices $a^e_{i,j}$ and $b^e_{i,j}$. Informally, every graph $B^e_{i,j}$, $e \in E(G)$, plays the same role and corresponds to the vertex $(i,j) \in V(G)$. For each $e \in E(G)$ and each $j \in [1,k]$, we define the graph $C^e_j$ obtained from the disjoint union of every $B^e_{i,j}$, $i \in [1,k]$, such that two graphs $B^e_{i_1,j}$ and $B^e_{i_2,j}$, $i_1 \neq i_2$, are complete to each other, i.e., for every $i_1 \neq i_2$, if $v_1 \in V(B^e_{i_1,j})$ and $v_2 \in V(B^e_{i_2,j})$, then $\{v_1, v_2\} \in E(C^e_j)$. Informally, every graph $C^e_j$, $e \in E(G)$, plays the same role and corresponds to the column $j$ of $G$.

For every $e \in E(G)$, we also define the graph $D^e$ obtained from the disjoint union of every $C^e_j$, $j \in [1,k]$, by adding, if $e = \{(i,j),(i',j')\}$, every edge $\{v_1, v_2\}$ such that $v_1 \in V(B^e_{i,j})$ and $v_2 \in V(B^e_{i',j'})$. The graph $D^e$ is depicted in Figure 2. Informally, the graph $D^e$, $e \in E(G)$, encodes the edge $e$ of the graph $G$.

For every $e \in E(G)$, we also define $J^e$ such that $V(J^e) = \{c^e_j \mid j \in [1,k]\} \cup \{r^e_i \mid i \in [1,k]\}$ is a set of new vertices and $E(J^e) = \emptyset$. The graphs $J^e$, $e \in E(G)$, are the separators that will ensure the consistency of the selected solution.

Finally, the graph $F$ is obtained from the disjoint union of every $D^e$, $e \in E(G)$, and every $J^e$, $e \in E(G)$, by adding the following edges, for a given fixed cyclic permutation

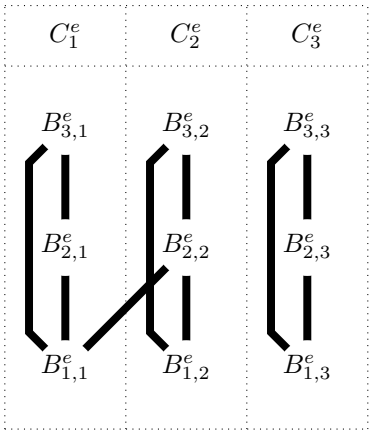



**Figure 2** The graph $D^e$ for $e = \{(1,1),(2,2)\} \in E(G)$ where $k = 3$. A bold edge means that two graphs $B$ are connected in a complete bipartite way.

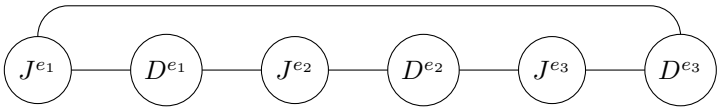



**Figure 3** The shape of the framework graph $F$ assuming that $k = 3$, $G$ contains only the three edges $e_1$, $e_2$, and $e_3$, and $\sigma$ is the cyclic permutation $(e_1, e_2, e_3)$.

$\sigma$ of the elements of $E(G)$: for each $e \in E(G)$ and each $i, j \in [1, k]$, we add the edges $\{a_{i,j}^e, r_i^e\}$, $\{r_i^e, b_{i,j}^{\sigma^{-1}(e)}\}$, $\{b_{i,j}^{\sigma^{-1}(e)}, c_j^e\}$, and $\{c_j^e, a_{i,j}\}$ to $E(F)$. This concludes the definition of the framework graph $F$, which is depicted in Figure 3 (a similar figure appears in [4]). The pair $(F, \ell := 2h(k-1)km)$ is called the *$H$-framework* of $(G, k)$. For convenience, we always assume that we know the permutation $\sigma$ linked to the graph $F$.

Let us now discuss about the treewidth of $F$. First note that for each $e \in E(G)$, the set $V(J^e) \cup V(J^{\sigma(e)})$ disconnects the vertex set $V(D^e)$ from the remaining part of $F$. Moreover, if $e = \{(i,j),(i',j')\}$, then the bags $V(C_1^e) \cup V(B_{i,j}^e) \cup V(B_{i',j'}^e)$, $V(C_2^e) \cup V(B_{i,j}^e) \cup V(B_{i',j'}^e), \ldots, V(C_k^e) \cup V(B_{i,j}^e) \cup V(B_{i',j'}^e)$ form a path decomposition of $D^e$ of width $2h(k+2) - 1$. Combining this decomposition with the circular shape of $F$ and the fact that $V(J^e) \cup V(J^{\sigma(e)})$ disconnects the vertex set $V(D^e)$ from the remaining part of $F$, we obtain that the treewidth (in fact, also the pathwidth) of $F$ is at most $6k + 2h(k+2) - 1$, and therefore $\mathsf{tw}(F) = \mathcal{O}(k)$.

For each graph $H$, we will consider $(F, \ell)$, the $H$-framework of $(G, k)$, and create another pair $(F_H, \ell)$, where $F_H$ is a graph obtained starting from $F$ by adding some vertices and edges. We will claim that there exists a solution of $k \times k$ Permutation Independent Set on $(G, k)$ if and only if there exists a solution of $\{H\}$-M-Deletion on $(F_H, \ell)$. In order to do this, we will prove the two following properties for each graph $H$.

▶ **Property 1.** *Let $S$ be a solution of $\{H\}$-M-Deletion on $(F_H, \ell)$. For every $e \in E(G)$ and $j \in [1, k]$ such that $|V(C_j^e) \setminus S|$ is maximized, there exists $i \in [1, k]$ such that $V(C_j^e) \setminus S \subseteq V(B_{i,j}^e)$.*

The above property states that for each column $C_j^e$, $j \in [1, k]$ and $e \in E(G)$, containing a minimum number of vertices of the solution, the remaining vertices all belong to the same row.

▶ **Property 2.** *Let $S$ be a solution of $\{H\}$-M-Deletion on $(F_H, \ell)$. For every $e \in E(G)$, and for every $i, j \in [1, k]$, if $b^e_{i,j} \notin S$, then for every $i' \in [1, k] \setminus \{i\}$, we have $a^{\sigma(e)}_{i',j} \in S$.*

The above property states that the choices of the vertices $a^e_{i,j}, b^e_{i,j}$ are consistent through the whole framework graph $F_H$.

If we assume that Property 1 holds, we have the following lemma.

▶ **Lemma 7.** *If Property 1 holds, then for every solution $S$ of $\{H\}$-M-Deletion on $(F_H, \ell = 2h(k-1)km)$, for every $e \in E(G)$, and for every $j \in [1, k]$, there exists $i \in [1, k]$ such that $V(C^e_j) \setminus S = V(B^e_{i,j})$. Moreover, for every $e \in E(G)$, $V(J^e) \cap S = \emptyset$.*

**Proof.** Assume that Property 1 holds and let $S$ be a solution of $\{H\}$-M-Deletion on $(F_H, 2h(k-1)km)$. By Property 1, we know that for every $e \in E(G)$, and for every $j \in [1, k]$, $|V(C^e_j) \cap S| \geq 2h(k-1)$. As there are exactly $m$ edges and $k$ columns, the budget is tight and we obtain that $|V(C^e_j) \cap S| = 2h(k-1)$. This implies that $|V(C^e_j) \setminus S| = 2h$, corresponding to the size of a set $B^e_{i,j}$ for some $i \in [1, k]$. The lemma follows. ◀

For each specific graph $H$, in order to prove Property 1 and Property 2, we will first prove Property 1 and then use Lemma 7 in order to prove Property 2.

▶ **Lemma 8 ($\star$).** *If Property 1 and Property 2 hold and there exists a solution $S$ of $\{H\}$-M-Deletion on $(F_H, \ell = 2h(k-1)km)$, then, for any $e \in E(G)$, the set $T^e = \{(i, j) \mid V(B^e_{i,j}) \cap S = \emptyset\}$ is a solution of $k \times k$ Permutation Independent Set on $(G, k)$.*

Using the same argumentation, we obtain the same results for the topological minor version.

▶ **Property 3.** *Let $S$ be a solution of $\{H\}$-TM-Deletion on $(F_H, \ell)$. For every $e \in E(G)$ and $j \in [1, k]$ such that $|V(C^e_j) \setminus S|$ is maximized, there exists $i \in [1, k]$ such that $V(C^e_j) \setminus S \subseteq V(B^e_{i,j})$.*

▶ **Property 4.** *Let $S$ be a solution of $\{H\}$-TM-Deletion on $(F_H, \ell)$. For every $e \in E(G)$, and for every $i, j \in [1, k]$, if $b^e_{i,j} \notin S$, then for every $i' \in [1, k] \setminus \{i\}$, we have $a^{\sigma(e)}_{i',j} \in S$.*

▶ **Lemma 9.** *If Property 3 holds, then for every solution $S$ of $\{H\}$-TM-Deletion on $(F_H, \ell = 2h(k-1)km)$, for every $e \in E(G)$, and for every $j \in [1, k]$, there exists $i \in [1, k]$ such that $V(C^e_j) \setminus S = V(B^e_{i,j})$. Moreover, for every $e \in E(G)$, $V(J^e) \cap S = \emptyset$.*

▶ **Lemma 10.** *If Property 3 and Property 4 hold and there exists a solution $S$ of $\{H\}$-TM-Deletion on $(F_H, \ell = 2h(k-1)km)$, then, for any $e \in E(G)$, the set $T^e = \{(i, j) \mid V(B^e_{i,j}) \cap S = \emptyset\}$ is a solution of $k \times k$ Permutation Independent Set on $(G, k)$.*

Given a solution $T$ of $k \times k$ Permutation Independent Set on $(G, k)$, we define $S_T = \{v \in V(F_H) \mid v \in B^e_{i,j} : e \in E(G), (i, j) \in [1, k]^2 \setminus T\}$. Note that $|S_T| = 2h(k-1)km$.

## 4.2 The reduction for $P_5$

We are ready to present the hardness reduction when $H = P_5$.

▶ **Theorem 11.** $\{P_5\}$-M-Deletion *cannot be solved in time $2^{o(\mathsf{tw} \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ unless the ETH fails.*

**Proof.** Let $H = P_5$. Let $(G, k)$ be an instance of $k \times k$ Permutation Independent Set and let $(F, \ell)$ be the $H$-framework of $(G, k)$, as defined in Section 4.1. Note that $\ell = 10(k-1)km$. In this theorem, we define $F_{P_5} = F$ without any modification.

Let $T$ be a solution of $k \times k$ Permutation Independent Set on $(G, k)$. One can check that every connected component of $F \setminus S_T$ is of size 4. Indeed, the connected components of $F \setminus S_T$ are either the copies of the graph $K$, which is of size 4, or the subgraph induced by the edges $\{a^e_{i,j}, r^e_i\}, \{r^e_i, b^{\sigma^{-1}(e)}_{i,j}\}, \{b^{\sigma^{-1}(e)}_{i,j}, c^e_j\}$, and $\{c^e_j, a^e_{i,j}\}$, for every $(i, j) \in T$. Thus $F \setminus S_T$ does not contain any $P_5$ as a minor and $S_T$ is a solution of $\{P_5\}$-M-Deletion of size $10(k-1)km$.

Assume now that $S$ is a solution of $\{P_5\}$-M-Deletion on $G$ of size $10(k-1)km$. We first prove that Property 1 holds. Let $e \in E(G)$ and $j \in [1, k]$ that maximize the size of $|V(C^e_j) \setminus S|$. By the pigeonhole principle, $|V(C^e_j) \setminus S| \geq 10$. Let $U_M$ be a set $V(B^e_{i,j}) \setminus S$, $i \in [1, k]$, with the maximum number of elements, and let $U_A = V(C^e_j) \setminus (S \cup U_M)$. If $|U_M| = 1$, then, as $|V(C^e_j) \setminus S| \geq 10$, we obtain that $K_{10}$ is a subgraph of $C^e_j$, contradicting the definition of $S$. If $|U_M| \geq 2$ and $|U_A| \geq 3$ or if $|U_M| \geq 3$ and $|U_A| \geq 2$, then $C^e_j$ contains a $K_{2,3}$ as a subgraph, also contradicting the definition of $S$. Finally if $|U_A| = 1$, we have that $|U_M| = 9$ and so, $C^e_j[U_M]$ contains a $K_4$ that, combined with the element of $U_A$, produces a $K_5$ that is forbidden by the definition of $S$. Thus $|U_A| = 0$ and Property 1 holds.

Let $e \in E(G)$ and let $i, j \in [1, k]$ such that $b^e_{i,j} \notin S$. Let $i' \in [1, k]$ such that $i \neq i'$. If $a^{\sigma(e)}_{i',j} \notin S$, then, as by Lemma 7 $S \cap V(J^{\sigma(e)}) = \emptyset$, we have that the path $r^{\sigma(e)}_{i'}, a^{\sigma(e)}_{i',j}, c^{\sigma(e)}_j, b^e_{i,j}, r^e_i$ is a subgraph of $F \setminus S$. Since, by definition of $S$, $F \setminus S$ does not contain $P_5$ as a minor, we have that $a^{\sigma(e)}_{i',j} \in S$. Thus Property 2 holds and the theorem follows. ◀

## 4.3 The reduction for $K_{1,s}$

The next theorem should be compared to the result in the full version stating that there exist single-exponential algorithms for hitting $K_{1,s}$ as a *topological* minor for every $s \geq 1$, while in Theorem 12 we prove that it is not the case for hitting $K_{1,s}$ as a minor, for every $s \geq 4$. It should be noted that the bound on $s$ of Theorem 12 is tight, as if $s \leq 3$, then $\{K_{1,s}\}$-M-Deletion is exactly $\{K_{1,s}\}$-TM-Deletion, and therefore it can be solved in single-exponential time; see the full version for the details.

▶ **Theorem 12.** *Given $s \geq 4$, $\{K_{1,s}\}$-M-Deletion cannot be solved in time $2^{o(\mathsf{tw} \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ unless the ETH fails.*

**Proof.** Let $s \geq 4$ and let $H = K_{1,s}$. Let $(G, k)$ be an instance of $k \times k$ Permutation Independent Set and let $(F, \ell)$ be the $H$-framework of $(G, k)$, as defined in Section 4.1. Note that $\ell = 2(s+1)(k-1)km$. We construct the graph $F_H$ from $F$ by adding a pendant vertex to every vertex $a^e_{i,j}$, $e \in E(G)$, $i, j \in [1, k]$, and by adding $s-3$ pendant vertices to every vertex $b^e_{i,j}$, $e \in E(G)$, $i, j \in [1, k]$.

Let $T$ be a solution of $k \times k$ Permutation Independent Set on $(G, k)$. Then every connected component of $F_H \setminus S_T$ is either a copy of the graph $K$, which is of size $s$, or the subgraph induced by $a^e_{i,j}$, $r^e_i$, $b^{\sigma^{-1}(e)}_{i,j}$, $c^e_j$, and the vertices that are pendant to $a^e_{i,j}$ and $b^{\sigma^{-1}(e)}_{i,j}$, for every $(i, j) \in T$. This latter subgraph, depicted in Figure 4, does not contain $K_{1,s}$ as a minor. Thus $F_H \setminus S_T$ does not contain any $K_{1,s}$ as a minor and $S_T$ is a solution of $\{K_{1,s}\}$-M-Deletion of size $2(s+1)(k-1)km$.

Assume now that $S$ is a solution of $\{K_{1,s}\}$-M-Deletion on $G$ of size $2(s+1)(k-1)km$. We first prove that Property 1 holds. Let $e \in E(G)$ and $j \in [1, k]$ that maximize the size of

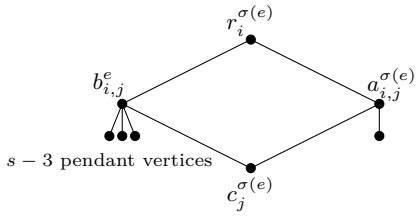



**Figure 4** A connected component of $F_H \setminus S$ that is not a copy of $K$, with $s = 6$.

$|V(C_j^e) \setminus S|$. By the pigeonhole principle, $|V(C_j^e) \setminus S| \geq 2(s+1)$. Let $U_M$ be a set $V(B_{i,j}^e) \setminus S$, $i \in [1,k]$, with the maximum number of elements, and let $U_A = V(C_j^e) \setminus (S \cup U_M)$. If $1 \leq |U_M| < s$, then $|U_A| \geq s$ and $K_{1,s}$ is a subgraph of $C_j^e$, contradicting the definition of $S$. If $|U_M| \geq s$ and $|U_A| \geq 1$, then again $K_{1,s}$ is a subgraph of $C_j^e$, contradicting again the definition of $S$. Thus $|U_A| = 0$ and Property 1 holds.

Let $e \in E(G)$ and let $i, j \in [1,k]$ such that $b_{i,j}^e \notin S$. Let $i' \in [1,k]$ such that $i \neq i'$. If $a_{i',j}^{\sigma(e)} \notin S$, then, as by Lemma 7 it holds that $S \cap V(J^{\sigma(e)}) = \emptyset$, we have that the path $r_{i'}^{\sigma(e)}, a_{i',j}^{\sigma(e)}, c_j^{\sigma(e)}, b_{i,j}^e, r_i^e$ combined with the vertex pendant to $a_{i',j}^{\sigma(e)}$ and the $s-3$ vertices pendant to $b_{i,j}^e$ is a subgraph of $F_H \setminus S$ and $K_{1,s}$ is a minor of it. As, by definition of $S$, $F_H \setminus S$ does not contains $K_{1,s}$ as a minor, we have that $a_{i',j}^{\sigma(e)} \in S$. Thus Property 2 holds and the theorem follows. ◀

## 5 Conclusions and further research

The ultimate goal in this line of research is to establish the *tight* complexity of $\{H\}$-M-Deletion and $\{H\}$-TM-Deletion for any graph $H$, but we are still very far from it. In particular, we do not know whether there exists some $H$ for which a double-exponential lower bound can be proved. Very recently, Kociumaka and Pilipczuk [13] studied the problem of deleting a minimum number of vertices to obtain a graph of Euler genus at most $g$, and presented an algorithm running in time $2^{\mathcal{O}_g(\mathsf{tw}\cdot\log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$. Generalizing their technique to $H$-minor-free graphs (which would correspond to the general $\{H\}$-M-Deletion problem) seems quite challenging, as this would involve a huge amount of technical details.

We managed to classify the complexity of $\{H\}$-M-Deletion when $H$ is a connected planar graph on at most 5 vertices (cf. Figure 1). While we consider this dichotomy a significant result, most of the algorithms and the reductions are ad hoc, and therefore our approach does not seem to be easily applicable for dealing with larger graphs $H$. The case where $H$ is planar and connected is already very interesting, since the results in [1] imply that $\{H\}$-M-Deletion cannot be solved in time $2^{o(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ under the ETH and can be solved in time $2^{\mathcal{O}(\mathsf{tw}\cdot\log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$. Thus, it makes sense to guess that, in this case, the complexity of $\{H\}$-M-Deletion is either $2^{\Theta(\mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ or $2^{\Theta(\mathsf{tw}\cdot\log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$, as it happens if $|V(H)| \leq 5$. We conjecture that for *every* connected simple planar graph $H$ with $|V(H)| \geq 6$, the latter case holds. In fact, we conjecture the following property, which is easily seen to imply the previous conjecture: if $H$ and $H'$ are graphs such that $H \preceq_{\mathsf{m}} H'$ and $\{H\}$-M-Deletion is not solvable under the ETH in time $f(\mathsf{tw}) \cdot n^{\mathcal{O}(1)}$ for some function $f$, then $\{H'\}$-M-Deletion is not solvable under the ETH in time $f(\mathsf{tw}) \cdot n^{\mathcal{O}(1)}$ either. We think that the equivalent property for the topological minor version also holds. Note that for establishing a dichotomy for $\{H\}$-TM-Deletion when $H$ is a connected planar graph on at most 5 vertices, it remains to obtain algorithms in time $2^{\mathcal{O}(\mathsf{tw}\cdot\log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ for the graphs in Figure 1 that have maximum degree 4, like the gem or the dart, as for those graphs [1, Theorem 6] cannot be applied.

Finally, note that the only connected (simple) graph on at most 5 vertices missing in Figure 1 is $K_5$. We think that, using techniques similar as those developed in [11], $\{K_5\}$-M-Deletion is solvable in time $2^{\mathcal{O}(\mathsf{tw}\cdot\log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$, which would be tight.

## References


1. Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. Optimal Algorithms for Hitting (Topological) Minors on Graphs of Bounded Treewidth. In *Proc. of the 12th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 89 of *LIPIcs*, pages 4:1–4:12, 2017.
2. Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Information and Computation*, 243:86–111, 2015.
3. Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A $c^k n$ 5-Approximation Algorithm for Treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016.
4. Édouard Bonnet, Nick Brettell, O-joung Kwon, and Dániel Marx. Generalized Feedback Vertex Set Problems on Bounded-Treewidth Graphs: Chordality Is the Key to Single-Exponential Parameterized Algorithms. In *Proc. of the 12th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 89 of *LIPIcs*, pages 7:1–7:13, 2017.
5. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
6. Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving Connectivity Problems Parameterized by Treewidth in Single Exponential Time. In *Proc. of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 150–159, 2011.
7. Reinhard Diestel. *Graph Theory*, volume 173. Springer-Verlag, 4th edition, 2010.
8. R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.
9. Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient Computation of Representative Families with Applications in Parameterized and Exact Algorithms. *Journal of the ACM*, 63(4):29:1–29:60, 2016.
10. Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.
11. Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. A Near-Optimal Planarization Algorithm. In *Proc. of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1802–1811, 2014.
12. T. Kloks. *Treewidth. Computations and Approximations*. Springer-Verlag LNCS, 1994.
13. Tomasz Kociumaka and Marcin Pilipczuk. Deleting vertices to graphs of bounded genus. *CoRR*, abs/1706.04065, 2017. `arXiv:1706.04065`.
14. Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly Superexponential Parameterized Problems. In *Proc. of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 760–776, 2011.
15. Marcin Pilipczuk. A tight lower bound for Vertex Planarization on graphs of bounded treewidth. *Discrete Applied Mathematics*, 231:211–216, 2017.

# Lower Bounds for Dynamic Programming on Planar Graphs of Bounded Cutwidth

**Bas A. M. van Geffen**
University of Oxford
bas.vangeffen@kellogg.ox.ac.uk

**Bart M. P. Jansen**[1]
Eindhoven University of Technology
b.m.p.jansen@tue.nl
https://orcid.org/0000-0001-8204-1268

**Arnoud A. W. M. de Kroon**
University of Oxford
noud.kroon@keble.ox.ac.uk

**Rolf Morel**
University of Oxford
rolf.morel@sjc.ox.ac.uk

## Abstract

Many combinatorial problems can be solved in time $\mathcal{O}^*(c^{tw})$ on graphs of treewidth $tw$, for a problem-specific constant $c$. In several cases, matching upper and lower bounds on $c$ are known based on the Strong Exponential Time Hypothesis (SETH). In this paper we investigate the complexity of solving problems on graphs of bounded cutwidth, a graph parameter that takes larger values than treewidth. We strengthen earlier treewidth-based lower bounds to show that, assuming SETH, Independent Set cannot be solved in $O^*((2-\varepsilon)^{\mathrm{ctw}})$ time, and Dominating Set cannot be solved in $O^*((3-\varepsilon)^{\mathrm{ctw}})$ time. By designing a new crossover gadget, we extend these lower bounds even to planar graphs of bounded cutwidth or treewidth. Hence planarity does not help when solving Independent Set or Dominating Set on graphs of bounded width. This sharply contrasts the fact that in many settings, planarity allows problems to be solved much more efficiently.

**2012 ACM Subject Classification** Theory of computation → Graph algorithms analysis, Theory of computation → Parameterized complexity and exact algorithms

**Keywords and phrases** planarization, dominating set, cutwidth, lower bounds, strong exponential time hypothesis

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.3

**Related Version** A full version of the paper is available on arXiv [16], `https://arxiv.org/abs/1806.10513`.

[1] Supported by NWO Gravitation grant "Networks" and NWO Veni grant "Frontiers in Parameterized Preprocessing".

© Bas A. M. van Geffen, Bart M. P. Jansen, Arnoud A. W. M. de Kroon, and Rolf Morel;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 3; pp. 3:1–3:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Dynamic programming on graphs of bounded treewidth is a powerful tool in the algorithm designer's toolbox, which has many applications (cf. [5]) and is captured by several meta-theorems [7, 25]. Through clever use of techniques such as Möbius transformation, fast subset convolution [2, 28], cut & count [9], and representative sets [4, 8, 14], algorithms were developed that can solve numerous combinatorial problems on graphs of treewidth $tw$ in $\mathcal{O}^*(c^{tw})$ time, for a problem-specific constant $c$. In recent work [23], it was shown that under the *Strong Exponential Time Hypothesis* (SETH, see [18, 19]), the base of the exponent $c$ achieved by the best-known algorithm is actually optimal for Dominating Set ($c = 3$) and Independent Set ($c = 2$), amongst others. This prompts the following questions:

1. Do faster algorithms exist for bounded-treewidth graphs that are *planar*?
2. Do faster algorithms exist for a more restrictive graph parameter, such as cutwidth?

It turns out that these questions are related, because the nature of cutwidth allows crossover gadgets to be inserted to planarize a graph without increasing its width significantly.

Before going into our results, we briefly motivate these questions. There is a rich bidimensionality theory (cf. [10]) of how the planarity of a graph can be exploited to obtain better algorithms than in the nonplanar case, leading to what has been called the *square-root phenomenon* [24]: in several settings, parameterized algorithms on planar graphs can be faster by a square-root factor in the exponent, compared to their nonplanar counterparts. Hence it may be tempting to believe that problems on bounded-width graphs can be solved more efficiently when they are planar. Lokshtanov et al. [23, §9] explicitly ask whether their SETH-based lower bounds continue to apply for planar graphs. The same problem is posed by Baste and Sau [1, p. 3] in their investigation on the influence of planarity when solving connectivity problems parameterized by treewidth. This motivates question 1.

When faced with lower bounds for the parameterization by treewidth, it is natural to investigate whether these continue to hold for more restrictive graph parameters. We work with the parameter cutwidth since it is one of the classic graph layout parameters (cf. [11]) which takes larger values than treewidth [22], and has been the subject of frequent study [17, 26, 27]. In their original work, Lokshtanov et al. [23] showed that their lower bounds also hold for pathwidth instead of treewidth. However, the parameterization by *cutwidth* has so far not been considered, which leads us to question 2. (See Section 2 for the definition of cutwidth.)

**Our results.** We answer questions 1 and 2 for the problems Independent Set and Dominating Set, which are formally defined in Section 2. Our conceptual contribution towards answering question 1 comes from the following insight: any graph $G$ can be drawn in the plane (generally with crossings) such that the graph $G'$ obtained by replacing each crossing by a vertex of degree four, does not have larger cutwidth than $G$. Hence the property of having bounded cutwidth can be preserved while planarizing the graph, which was independently[2] discovered by Eppstein [13]. When we planarize by replacing each crossing by a planar crossover gadget $H$ instead of a single vertex, then we obtain $\mathrm{ctw}(G') \leq \mathrm{ctw}(G) + \mathrm{ctw}(H) + 4$ if the endpoints of the crossing edges each obtain at most one neighbor in the crossover gadget. This gives a means to reduce a problem instance on a general graph of bounded

[2] We learned of Eppstein's result while a previous version of this work was under submission at a different venue; see Footnote 2 in [13]. Our previous manuscript, cited by Eppstein, was later split into two separate parts due to its excessive length. The present paper is one part, and [21] is the other.

cutwidth to a planar graph of bounded cutwidth, if a suitable crossover gadget is available. The parameter cutwidth is special in this regard: one cannot planarize a drawing of $K_{3,n}$ while keeping the pathwidth or treewidth constant [12, 13].

For the Independent Set problem, the crossover gadget developed by Garey, Johnson, and Stockmeyer [15] can be used in the process described above. Together with the observation that the SETH-based lower bound construction by Lokshtanov et al. [23] for the treewidth parameterization also works for the cutwidth parameterization, this yields our first result.[3]

▶ **Theorem 1.** *Assuming SETH, there is no $\varepsilon > 0$ such that* Independent Set *on a planar graph $G$ given along with a linear layout of cutwidth $k$ can be solved in time $\mathcal{O}^*((2-\varepsilon)^k)$.*

For the Dominating Set problem, more work is needed to obtain a lower bound for planar graphs of bounded cutwidth. While the lower bound construction of Lokshtanov et al. [23] also works for the parameter cutwidth after a minor tweak, no crossover gadget for the Dominating Set problem was known. Our main technical contribution therefore consists of the design of a crossover gadget for Dominating Set, which we believe to be of independent interest. Together with the framework above, this gives our second result.

▶ **Theorem 2.** *Assuming SETH, there is no $\varepsilon > 0$ such that* Dominating Set *on a planar graph $G$ given along with a linear layout of cutwidth $k$ can be solved in time $\mathcal{O}^*((3-\varepsilon)^k)$.*

Since any linear ordering of cutwidth $k$ can be transformed into a tree decomposition of width at most $k$ in polynomial time (cf. [3, Theorem 47]), the lower bounds of Theorems 1 and 2 also apply to the parameterization by treewidth. Hence our work resolves the question raised by Lokshtanov et al. [23] and by Baste and Sau [1] whether the SETH-lower bounds for Independent Set and Dominating Set parameterized by treewidth also apply for planar graphs.

**Organization.** In Section 2 we provide preliminaries. In Section 3 we present a general theorem for planarizing graphs of bounded cutwidth, using a crossover gadget. It leads to a proof of Theorem 1. In Section 4 we prove Theorem 2. Finally, we provide some conclusions in Section 5. Due to space restrictions, proofs for statements marked (★) have been deferred to the full version [16].

## 2 Preliminaries

We use $\mathbb{N}$ to denote the natural numbers, including 0. For a positive integer $n$ and a set $X$ we use $\binom{X}{n}$ to denote the collection of all subsets of $X$ of size $n$. The *power set* of $X$ is denoted $2^X$. The set $\{1, \ldots, n\}$ is abbreviated as $[n]$. The $\mathcal{O}^*$ notation suppresses polynomial factors in the input size $n$, such that $\mathcal{O}^*(f(k))$ is shorthand for $\mathcal{O}(f(k)n^{\mathcal{O}(1)})$. All our logarithms have base two.

We consider finite, simple, and undirected graphs $G$, consisting of a vertex set $V(G)$ and edge set $E(G) \subseteq \binom{V(G)}{2}$. The neighbors of a vertex $v$ in $G$ are denoted $N_G(v)$. The closed neighborhood of $v$ is $N_G[v] := N_G(v) \cup \{v\}$. For a vertex set $S \subseteq V(G)$ the open neighborhood is $N_G(S) := \bigcup_{v \in S} N_G(v) \setminus S$ and the closed neighborhood is $N_G[S] := N_G(S) \cup S$. The subgraph of $G$ induced by a vertex subset $U \subseteq V(G)$ is denoted $G[U]$. The operation of *identifying vertices $u$ and $v$* in a graph $G$ results in the graph $G'$ that is obtained from $G$ by replacing the two vertices $u$ and $v$ by a new vertex $w$ with $N_{G'}(w) = N_G(\{u, v\})$.

[3] The analogous lower bound of $\Omega((2-\varepsilon)^k)$ for solving Independent Set on planar graphs of pathwidth $k$ was already observed by Jansen and Wulms [20], based on an elaborate ad-hoc argument.

An *independent set* is a set of pairwise nonadjacent vertices. A *vertex cover* in a graph $G$ is a set $S \subseteq V(G)$ such that $S$ contains at least one endpoint from every edge. A set $S \subseteq V(G)$ *dominates* the vertices $N_G[S]$. A *dominating set* is a vertex set $S$ such that $N_G[S] = V(G)$. The associated decision problems ask, given a graph $G$ and integer $t$, whether an independent set (dominating set) of size $t$ exists in $G$. The size of a maximum independent set (resp. minimum dominating set) in $G$ is denoted $\text{OPT}_{\text{IS}}(G)$ (resp. $\text{OPT}_{\text{DS}}(G)$). The $q$-SAT problem asks whether a given Boolean formula, in conjunctive normal form with clauses of size at most $q$, has a satisfying assignment.

▶ **Strong Exponential Time Hypothesis** ([18, 19]). *For every $\varepsilon > 0$, there is a constant $q$ such that $q$-SAT on $n$ variables cannot be solved in time $\mathcal{O}^*((2-\varepsilon)^n)$.*

**Drawings.** A *drawing* of a graph $G$ is a function $\psi$ that assigns a unique point $\psi(v) \in \mathbb{R}^2$ to each vertex $v \in V(G)$, and a curve $\psi(e) \subseteq \mathbb{R}^2$ to each edge $e \in E(G)$, such that the following four conditions hold. (1) For $e = \{u, v\} \in E(G)$, the endpoints of $\psi(e)$ are exactly $\psi(u)$ and $\psi(v)$. (2) The interior of a curve $\psi(e)$ does not contain the image of any vertex. (3) No three curves representing edges intersect in a common point, except possibly at their endpoints. (4) The interiors of the curves $\psi(e), \psi(e')$ for distinct edges intersect in at most one point. If the interiors of all the curves representing edges are pairwise-disjoint, then we have a *planar drawing*. In this paper we combine (nonplanar) drawings with crossover gadgets to build planar drawings. A graph is *planar* if it admits a planar drawing.

**Cutwidth.** For an $n$-vertex graph $G$, a *linear layout* of $G$ is a linear ordering of its vertex set, as given by a bijection $\pi\colon V(G) \to [n]$. The *cutwidth* of $G$ with respect to the layout $\pi$ is:

$$\text{ctw}_\pi(G) = \max_{1 \le i < n} \left|\left\{\{u, v\} \in E(G) \,\middle|\, \pi(u) \le i \wedge \pi(v) > i\right\}\right|.$$

The cutwidth $\text{ctw}(G)$ of a graph $G$ is the minimum cutwidth attained by any linear layout. It is well-known that $\text{ctw}(G) \ge \text{pw}(G) \ge \text{tw}(G)$, where the latter denote the pathwidth and treewidth of $G$, respectively (cf. [3]).

## 3 Planarizing graphs while preserving cutwidth

In this section we show how to planarize a graph without blowing up its cutwidth. An intuitive way to think about cutwidth is to consider the vertices as being placed on a horizontal line in the order dictated by the layout $\pi$, with edges drawn as $x$-monotone curves. For any position $i$ we consider the gap between vertex $\pi^{-1}(i)$ and $\pi^{-1}(i+1)$, and count the edges that *cross* the gap by having one endpoint at position at most $i$ and the other at position after $i$. The cutwidth of a layout is the maximum number of edges crossing any single gap; see Figure 1. The simple but useful fact on which our approach hinges is the following. If we obtain $G'$ by replacing a crossing in the drawing by a new vertex of degree four, and we let $\pi'$ be the left-to-right order of the vertices in the resulting drawing, then $\text{ctw}_\pi(G) = \text{ctw}_{\pi'}(G')$. Hence by repeating this procedure we can eliminate all crossings to obtain a planarized version of $G$ without increasing the cutwidth. To utilize this idea in reductions, we formalize a version of this approach where we planarize the graph by inserting gadgets, rather than simply replacing crossings by degree-four vertices.

▶ **Definition 3.** A *crossover gadget* is a graph $H$ with terminal vertices $u, u', v, v'$ such that:

1. there is a planar drawing $\psi$ of $H$ in which all terminals lie on the outer face, and
2. there is a closed curve intersecting the drawing $\psi$ only in the terminals, which visits the terminals in the order $u, v, u', v'$.

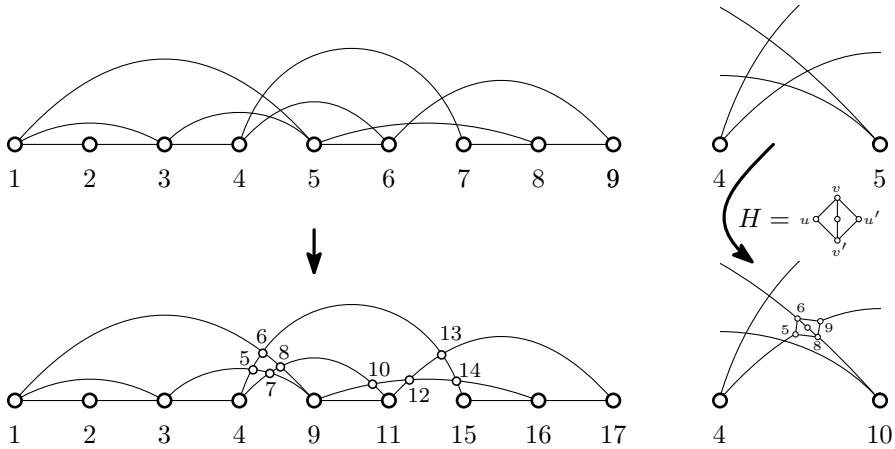



**Figure 1** Top-left: a linear layout $\pi$ of a graph $G$ with $\mathrm{ctw}_\pi(G) = 4$. The largest cutsize is attained after vertices 4 and 5. Bottom-left: after inserting vertices at the crossings to obtain $G'$ and extending $\pi$ to $\pi'$ based on the $x$-coordinates of the inserted vertices, we have $\mathrm{ctw}_\pi(G) = \mathrm{ctw}_{\pi'}(G')$. Top-right: enlarged view. Bottom-right: replacing a crossing by gadget $H$.

▶ **Definition 4.** Let $\{a, b\}$ and $\{c, d\}$ be disjoint edges of a graph $G$, and let $H$ be a crossover gadget. The operation of *replacing* $\{\{a, b\}, \{c, d\}\}$ *by* $H$ removes edges $\{a, b\}$ and $\{c, d\}$, inserts a new copy of the graph $H$, and inserts the edges $\{a, u\}, \{u', b\}, \{c, v\}, \{v', d\}$.

For a crossover gadget to be useful to planarize instances of a decision problem, a replacement should have a predictable effect on the answer. To formalize this, we say that a *decision problem* $\Pi$ *on graphs* is a decision problem whose input consists of a graph $G$ and integer $t$.

▶ **Definition 5.** A crossover gadget $H$ is *useful for a decision problem* $\Pi$ *on graphs* if there exists an integer $c_\Pi$ such that the following holds. If $(G, t)$ is an instance of $\Pi$ containing disjoint edges $\{a, b\}, \{c, d\}$, and $G'$ is the result of replacing these edges by $H$, then $(G, t)$ is a YES-instance of $\Pi$ if and only if $(G', t + c_\Pi)$ is a YES-instance of $\Pi$.

The following theorem proves that a useful crossover gadget can be used to efficiently planarize instances without blowing up their cutwidth.

▶ **Theorem 6.** *If $H$ is a crossover gadget that is useful for decision problem $\Pi$ on graphs, then there is a polynomial-time algorithm that, given an instance $(G, t)$ of $\Pi$ and a linear layout $\pi$ of $G$, outputs an instance $(G', t')$ and a linear layout $\pi'$ of $G'$ such that:*
1. *$G'$ is planar,*
2. $\mathrm{ctw}_{\pi'}(G') \leq \mathrm{ctw}_\pi(G) + \mathrm{ctw}(H) + 4$, *and*
3. *$(G, t)$ is a YES-instance of $\Pi$ if and only if $(G', t')$ is a YES-instance of $\Pi$.*

**Proof.** Consider the crossover gadget $H$ for $\Pi$ with terminals $u, u, v, v'$. Let $\pi_H$ be a linear layout of $H$ of minimum cutwidth, which is hardcoded into the algorithm along with the integer $c_\Pi$ described in Definition 5.

Let $(G, t)$ be an instance of $\Pi$ with a linear layout $\pi$. We start by constructing a (nonplanar) drawing $\psi$ of $G$ with the following properties.
1. The vertices of $G$ are placed on the $x$-axis, in the order dictated by $\pi$.
2. The image of each edge of $G$ is a strictly $x$-monotone curve.
3. If the drawings of two edges intersect in their interior, then their endpoints are all distinct and the corresponding curves properly cross; they do not only touch.

4. For each pair of edges, their drawings intersect in at most one point.
5. The $x$-coordinates of all crossings are distinct from each other, and from the $x$-coordinates of the vertices.

It is easy to see that such a drawing always exists and can be found in polynomial time; we omit the details as they are not interesting. Properties 1 and 2 together ensure that for any $i \in [|V(G)| - 1]$, the set of edges that cross the gap after vertex $\pi^{-1}(i)$ in the linear layout is exactly the set of edges intersected by a vertical line between $\pi^{-1}(i)$ and $\pi^{-1}(i+1)$, which therefore has size at most $\mathrm{ctw}_\pi(G)$. We will use this property later.

The algorithm replaces the crossings one by one. If two edges $\{a, b\}$ and $\{c, d\}$ intersect in their interior, then their endpoints are all distinct by (3) and they properly cross. Hence we can replace these two edges by a copy of $H$ as in Definition 4. Since there is a planar drawing of $H$ with the terminals alternating along the outer face, after possibly swapping the labels of $a$ and $b$, and of $c$ and $d$, the drawing can be updated so that the crossing between $\{a, b\}$ and $\{c, d\}$ is eliminated. Since each of $a, b, c, d$ is made adjacent to exactly one vertex of $H$, the replacement can be done such that the remaining crossings are in exactly the same locations as before; see the right side of Figure 1. When inserting the crossover gadget, we scale it down sufficiently far that the following holds: all vertices and crossings that were originally on the left of the crossing between $\{a, b\}$ and $\{c, d\}$ lie to the left of all vertices that are inserted to replace this crossing; and all vertices and crossings that were on the right of the $\{\{a, b\}, \{c, d\}\}$ crossing, lie to the right of all vertices inserted for its replacement.

Since each pair of edges intersects at most once by (4), the number of crossings is $\mathcal{O}(|V(G)|^2)$. Hence in polynomial time we can replace all crossings by copies of $H$ to arrive at a graph $G'$. If $\ell$ is the number of replaced crossings, then we set $t' := t + \ell \cdot c_\Pi$. By Definition 5 and transitivity it follows that $(G, t)$ is a YES-instance of $\Pi$ if and only if $(G', t')$ is a YES-instance. By construction, $G'$ is planar. It remains to define a linear layout of $G'$ and bound its cutwidth.

The layout $\pi'$ of $G'$ is defined as follows. Let the *elements* of the original drawing $\psi$ of $G$ consist of its vertices and its crossings. The elements of $\psi$ are linearly ordered by their $x$-coordinates, by (5). The linear layout $\pi'$ of $G'$ has one block per element of $\psi$, and these blocks are ordered according to the $x$-coordinates of the corresponding element. For elements that consist of a vertex $v$, the block simply consists of $v$. For elements that consist of a crossing $X$, the block consists of the vertices of the copy of $H$ that was inserted to replace $X$, in the order dictated by $\pi_H$. It is easy to see that $\pi'$ can be constructed in polynomial time.

We classify the edges of $G'$ into two types. We have *internal* edges, which are edges within an inserted copy of $H$, and we have *external edges* which connect two different copies of $H$, or which connect a vertex of $V(G) \cap V(G')$ to a copy of $H$. Using this classification we argue that for an arbitrary vertex $v^*$ of $G'$, the cut crossing the gap after vertex $v^*$ in $\pi'$ contains at most $\mathrm{ctw}_\pi(G) + \mathrm{ctw}(H) + 4$ edges. To do so, we distinguish two cases depending on whether $v^*$ is an original vertex from $G$, or was inserted as part of a copy of $H$.

▶ **Claim 7.** *If $v^* \in V(G) \cap V(G')$, then the size of the cut after $v^*$ in $\pi'$ is at most* $\mathrm{ctw}_\pi(G)$.

**Proof.** The layout $\pi'$ consists of blocks, and $v^* \in V(G) \cap V(G')$ is a block. So for each copy $C$ of a crossover gadget, the vertices of $C$ all appear on the same side of $v^*$ in the ordering. Hence no internal edge of $C$ crosses the cut after $v^*$, implying that no internal edge is in the cut. Each external edge crossing the cut is (a segment of) an edge of $G$ that is intersected by a vertical line after $v^*$ in the drawing $\psi$; see Figure 1. As such a line intersects at most $\mathrm{ctw}_\pi(G)$ edges as observed above, the cut after $v^*$ has size at most $\mathrm{ctw}_\pi(G)$. ⌟

▶ **Claim 8.** *If $v^* \in V(G')$ is a vertex of a copy $C$ of a crossover gadget that was inserted to replace a crossing $X$, then the size of the cut after $v^*$ in $\pi'$ is at most* $\mathrm{ctw}_\pi(G) + \mathrm{ctw}(H) + 4$.

**Proof.** The number of internal edges in the cut after $v^*$ is at most $\mathrm{ctw}(H)$, since the only internal edges in the cut all belong to the same copy $C$ that contains $v^*$ and we ordered them according to an optimal layout $\pi_H$. There are at most four external edges incident on a vertex of $C$, which contribute at most four to the cut. Finally, for each of the remaining external edges in the cut there is a unique edge of $G$ intersected by a vertical line through crossing $X$ in the drawing $\psi$. As at most $\mathrm{ctw}_\pi(G)$ edges are intersected by any vertical line, as observed above, it follows that the size of the cut is at most $\mathrm{ctw}_\pi(G) + \mathrm{ctw}(H) + 4$. ⌟

The two claims together show that any gap in the ordering $\pi'$ is crossed by at most $\mathrm{ctw}_\pi(G) + \mathrm{ctw}(H) + 4$ edges, which bounds the cutwidth of $G'$ as required. ◀

Using Theorem 6 we can now elegantly prove Theorem 1 by combining two known results.

▶ **Theorem 1.** *Assuming SETH, there is no $\varepsilon > 0$ such that* Independent Set *on a planar graph $G$ given along with a linear layout of cutwidth $k$ can be solved in time $\mathcal{O}^*((2-\varepsilon)^k)$.*

**Proof.** First, we observe that the crossover gadget for Vertex Cover due to Garey, Johnson, and Stockmeyer [15, Thm. 2.7] satisfies our conditions of a useful crossover gadget. Since an $n$-vertex graph has a vertex cover of size $k$ if and only if it has an independent set of size $n-k$, it also acts as a useful crossover gadget for Independent Set with $c_\pi = 9$ (cf. [20, Proposition 20]). Second, we observe that by a different analysis of a construction due to Lokshtanov et al. [23], it follows that (assuming SETH) there is no $\varepsilon > 0$ such that Independent Set on a graph $G$ with a linear layout of cutwidth $k$ can be solved in time $\mathcal{O}^*((2-\varepsilon)^k)$. We prove this in the full version [16]. By Theorem 6, if such a runtime could be achieved on *planar* graphs of a given cutwidth, it could be achieved for a general graph as well, since the insertion of crossover gadgets increases the cutwidth by only a constant. Hence Theorem 1 follows. ◀

## 4 Lower bound for dominating set on planar graphs of bounded cutwidth

In this section we prove a runtime lower bound for solving Dominating Set on planar graphs of bounded cutwidth. Our starting point is the insight that through a minor modification, the lower bound by Lokshtanov et al. [23] for the parameterization by pathwidth can be lifted to apply to the parameterization by cutwidth as well.

▶ **Theorem 9 (★).** *Assuming SETH, there is no $\varepsilon > 0$ such that* Dominating Set *on a (nonplanar) graph $G$ given along with a linear layout of cutwidth $k$ can be solved in time $\mathcal{O}^*((3-\varepsilon)^k)$.*

Our contribution is to extend the lower bound of Theorem 9 to apply to *planar* graphs. Following the strategy outlined in Section 3, to achieve this it suffices to develop a useful crossover gadget for Dominating Set as per Definition 5. Since our crossover gadget is fairly complicated (it has more than 100 vertices), we describe its design in steps. The main idea is as follows. We first show that an edge in a Dominating Set instance can be replaced by a longer double-path structure, which contains several triangles. Then we show that when two triangles cross, we can replace their crossing by a suitable adaptation of the Vertex Cover crossover gadget due to Garey, Johnson, and Stockmeyer [15, Thm. 2.7].

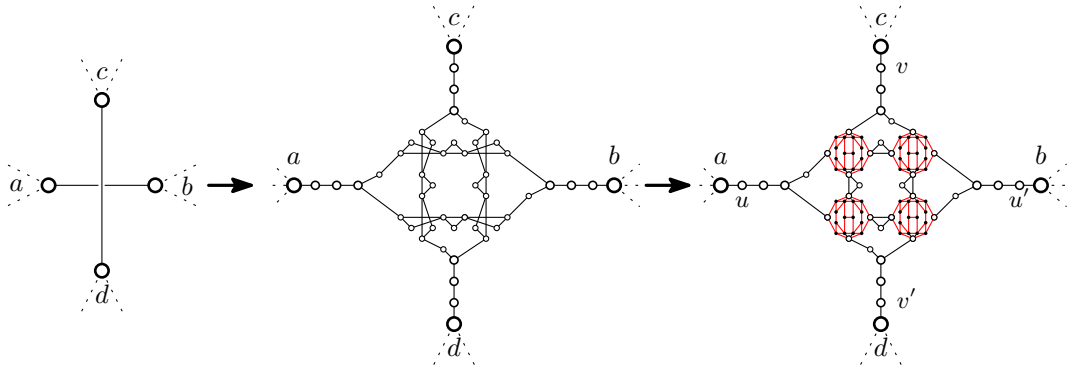



**Figure 2** Overview of the method to eliminate an edge crossing in an instance of Dominating Set. Each edge is transformed into a double-path structure, to turn a single crossing edge into four crossing triangles (middle figure). Then each crossing triangle is replaced by a planar gadget (right figure). Some vertices have been omitted for readability. For each red edge $\{u, v\}$, there is a hidden degree-two vertex in the graph that forms a triangle with $u$ and $v$.

This two-step approach is illustrated in Figure 2. We follow the same two steps in proving its correctness, starting with the insertion of the double-path structure.

▶ **Lemma 10.** *Let $\{x, y\}$ be an edge in a graph $G$. If $G'$ is obtained from $G$ by replacing $\{x, y\}$ by a double-path structure as shown in Figure 3, then $\text{OPT}_{\text{DS}}(G') = \text{OPT}_{\text{DS}}(G) + 6$.*

**Proof.** We prove equality by establishing matching upper- and lower bounds.

($\leq$ To show $\text{OPT}_{\text{DS}}(G') \leq \text{OPT}_{\text{DS}}(G) + 6$, consider a minimum dominating set $S \subseteq V(G)$ of $G$. If $S \cap \{x, y\} = \emptyset$, or $S \cap \{x, y\} = \{x, y\}$, then the edge $\{x, y\}$ is not used to dominate vertex $x$ or $y$, and therefore $S \cup \{b_x, b_y, e_x, e_y, g_x, g_y\}$ is a dominating set of size $|S| + 6 = \text{OPT}_{\text{DS}}(G) + 6$ in graph $G'$; see Figure 3. If $S \cap \{x, y\} = \{x\}$, then $S \cup \{c_x, f_x, h_x, e_y, g_y, a_y\}$ is a dominating set of size $\text{OPT}_{\text{DS}}(G) + 6$ in $G'$: the vertex $a_y$ takes over the role of dominating $y$ after the direct edge $\{x, y\}$ is removed, while $a_x$ is dominated from $x$. Symmetrically, if $S \cap \{x, y\} = \{y\}$, then $S \cup \{c_y, f_y, h_y, e_x, g_x, a_x\}$ is a dominating set in $G'$ of size $\text{OPT}_{\text{DS}}(G) + 6$.

($\geq$ To show $\text{OPT}_{\text{DS}}(G') \geq \text{OPT}_{\text{DS}}(G) + 6$, we instead prove $\text{OPT}_{\text{DS}}(G) \leq \text{OPT}_{\text{DS}}(G') - 6$. Consider a minimum dominating set $S' \subseteq V(G')$ of $G'$. Let $B$ be the vertices in the interior of the double-path structure that was inserted into $G'$ to replace edge $\{x, y\}$. If $|S' \cap B| \geq 7$, then $(S \setminus B) \cup \{x\}$ is a dominating set of size at most $|S'| - 6 \leq \text{OPT}_{\text{DS}}(G') - 6$ in $G$, since $x$ dominates itself and $y$ using the edge $\{x, y\}$. We assume $|S' \cap B| \leq 6$ in the remainder. Then we have $|S' \cap B| = 6$: the closed neighborhoods of the six vertices $\{b_x, b_y, t_x, t_y, t''_x, t''_y\}$ are contained entirely within $B$, and are pairwise disjoint. Hence these six vertices must be dominated by six distinct vertices from $B$. If $S' \cap \{a_x, a_y\} = \emptyset$ then the vertices $x$ and $y$ are not dominated from within the double-path structure, implying that $S' \setminus B$ is a dominating set in $G$ of size $|S'| - 6 \leq \text{OPT}_{\text{DS}}(G') - 6$. It remains to consider the case that $S'$ contains $a_x$, or $a_y$, or both.

▶ **Claim 11.** *Let $B' \subseteq B$ be a set of size six that dominates the vertices $B \setminus \{a_x, a_y\}$. If $B'$ contains $a_x$, then $B'$ does not dominate $a_y$. Analogously, if $B'$ contains $a_y$ then it does not dominate $a_x$.*

**Proof.** We prove that if $a_y \in B'$, then $B'$ does not dominate $a_x$. The other statement follows by symmetry. So assume for a contradiction that $B'$ contains $a_y$ and dominates $a_x$, which implies it contains $a_x$ or $b_x$. Since $B'$ dominates the interior of the double-path structure, it

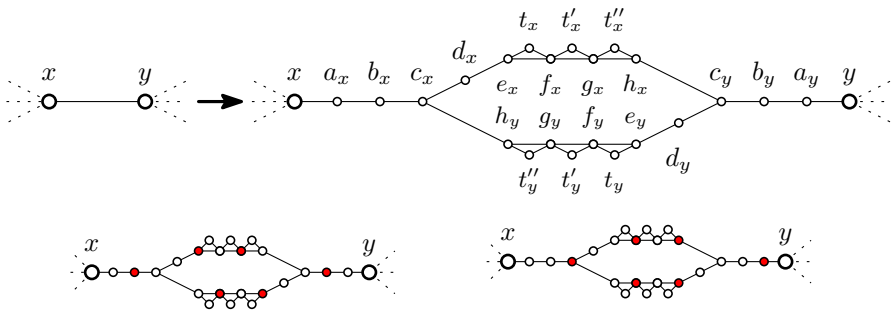



**Figure 3** The double-path structure for Dominating Set is the subgraph on the top right minus the vertices $x$ and $y$. Top: an edge $\{x, y\}$ is replaced by a double-path structure. Bottom-left: the interior of the double-path structure can be dominated by six vertices (in red). Bottom-right: there is a set of six vertices that dominates $y$ and all vertices of the double-path structure except $a_x$.

contains at least one vertex from the closed neighborhoods of $t_x, t_y, t''_x, t''_y$. Since these are pairwise disjoint, and do not contain $a_y$, $a_x$, or $b_x$, the set $B'$ contains a vertex from the closed neighborhood of each of $\{t_x, t_y, t''_x, t''_y\}$. Since $B'$ has size six, besides the four vertices from these closed neighborhoods, the vertex $a_y$, and the one vertex in $\{a_x, b_x\}$ there can be no further vertices in $B'$. Hence $B'$ does not contain $c_x$ or $d_x$, as these do not occur in the stated closed neighborhoods. This implies that to dominate $d_x$, the set $B'$ contains $e_x$. Then vertex $t'_x$ is not dominated by the vertex from $N_{G'}[t_x]$, and must therefore be dominated by the vertex in $B'$ from $N_{G'}[t''_x]$, implying that $g_x \in B'$. But the vertices mentioned so far do not dominate $c_y$, and regardless of how a vertex is chosen from the closed neighborhoods of $t_y$ and $t''_y$, the resulting choice does not dominate $c_y$ since no vertex from the closed neighborhoods of $t_y, t''_y$ is adjacent to $c_y$. So $c_y$ is not dominated by $B'$; a contradiction. ⌟

Using the claim we finish the proof. The set $B' := S \cap B$ has size six and dominates all of $B \setminus \{a_x, a_y\}$, since those vertices cannot be dominated from elsewhere. If $B'$ contains $a_x$, then $B'$ does not dominate $a_y$. Since $S'$ is a dominating set, and $y$ is the only neighbor of $a_y$ outside of $B$, it follows that $y \in S'$. But then $S' \setminus B$ is a dominating set in $G$ of size $|S'| - 6 = \text{OPT}_{\text{DS}}(G') - 6$ in $G$: the edge $\{x, y\}$ in $G$ ensures that $y$ dominates $x$. If $B'$ contains $a_y$ instead, then the symmetric argument applies. Hence $\text{OPT}_{\text{DS}}(G) \leq \text{OPT}_{\text{DS}}(G') - 6$. ◀

Using Lemma 10, we can replace a direct edge by a double-path structure while controlling the domination number. This allows two crossing edges to be reduced to four crossing triangles as in Figure 2. Even though more crossings are created in this way, these crossing triangles actually help to planarize the graph. The key point is that crossing triangles enforce a dominating set to locally act like a vertex cover, which allows us to exploit a known gadget for Vertex Cover. The following two statements are useful to formalize these ideas. Recall that a vertex $v$ is *simplicial* in a graph $G$ if $N_G(v)$ forms a clique.

▶ **Observation 12.** *If $I$ is an independent set of simplicial degree-two vertices in a graph $G$, then $G$ has a minimum dominating set that contains no vertex of $I$.*

▶ **Proposition 13.** *Let $U$ be a set of vertices in a graph $G$, such that for each edge $\{x, y\} \in E(G[U])$ there is a vertex $v \in V(G) \setminus U$ with $N_G(v) = \{x, y\}$. Then there is a minimum dominating set $S$ of $G$ such that $S$ forms a vertex cover of $G[U]$.*

**Proof.** Construct a set $I$ as follows. For each $\{x, y\} \in E(G[U])$, add a vertex $v \in V(G) \setminus U$ with $N_G(v) = \{x, y\}$ to $I$. Then $I$ is an independent set of simplicial degree-two vertices.

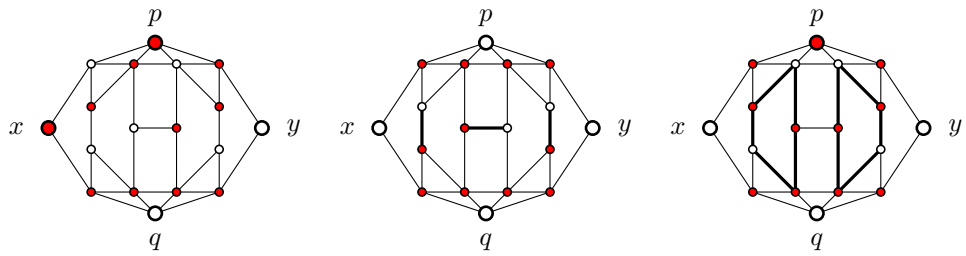



**Figure 4** Three copies of the 18-vertex gadget graph $H_{\text{VC}}$, which has four terminals $\{x, y, p, q\}$. Left: A vertex cover for $H_{\text{VC}}$ that contains $p$ and $q$ and has size eleven is shown in red. Middle: Any vertex cover for $H_{\text{VC}}$ that does not contain $p$ or $q$ contains the neighbors of $p$ and $q$ and at least one endpoint of the three thick edges, and contains at least eleven non-terminals. Right: Any vertex cover for $H_{\text{VC}}$ that does not contain $x$ or $y$ contains the four neighbors of $x$ and $y$ and at least three vertices from each of the two highlighted five-cycles, and contains at least ten non-terminals.

By Observation 12 there is a minimum dominating set $S$ of $G$ that contains no vertex of $I$. Then $S \cap U$ is a vertex cover of $G[U]$: for an arbitrary edge $\{x, y\} \in E(G[U])$ there is a vertex $v$ in $I$ whose open neighborhood is $\{x, y\}$. Since $I \cap S = \emptyset$, at least one of $x$ and $y$ belongs to $S$ to dominate $v$. Hence the edge $\{x, y\}$ is covered by $S$. ◀

Proposition 13 relates minimum dominating sets to vertex covers. We therefore use a simplified version of a Vertex Cover crossover gadget in our design. We exploit the graph $H_{\text{VC}}$ with four terminals $\{x, y, p, q\}$ that is shown in Figure 4. It was obtained by applying the "folding" reduction rule for Vertex Cover [6, Lemma 2.3] on the gadget by Garey et al. [15] and omitting two superfluous edges. We use the following property of the graph $H_{\text{VC}}$. It states that in $H_{\text{VC}}$, for every axis from which a vertex cover contains no terminal vertex, the number of non-terminal vertices used in a vertex cover increases.

▶ **Proposition 14.** *Let $S$ be a vertex cover of $H_{\text{VC}}$ and let $\ell \in \{0, 1, 2\}$ be the number of pairs among $\{p, q\}$ and $\{x, y\}$ from which $S$ contains no vertices. Then $|S \setminus \{p, q, x, y\}| \geq 9 + \ell$.*

**Proof.** We first show $|S \setminus \{p, q, x, y\}| \geq 9$ for any vertex cover $S$ of $H_{\text{VC}}$, proving the claim for $\ell = 0$. The non-terminal vertices of $H_{\text{VC}}$ can be partitioned into four vertex-disjoint triangles and an edge that is vertex-disjoint from the triangles. From any triangle, a vertex cover contains at least two vertices. From the remaining edge, it contains at least one vertex.

If $S$ contains no vertex of $\{p, q\}$, then as illustrated in the middle of Figure 4, $S$ contains at least eleven non-terminals. Hence $|S \setminus \{p, q, x, y\}| \geq 11 \geq 9 + \ell$.

If the previous case does not apply, then $\ell \leq 1$ since $S$ contains a vertex of $\{p, q\}$. If $S$ contains no vertex of $\{x, y\}$, then as illustrated on the right of Figure 4, $S$ contains at least ten non-terminals. Hence $|S \setminus \{p, q, x, y\}| \geq 10 \geq 9 + \ell$. ◀

Using Proposition 14 we prove that replacing two crossing triangles in a Dominating Set instance by the gadget, increases the optimum by exactly nine.

▶ **Lemma 15.** *Let $G$ be a graph, and let $\{x, y, z\}$ and $\{p, q, r\}$ be two vertex-disjoint triangles in $G$ such that $z$ and $r$ have degree two in $G$. Then the graph $G'$ obtained from $G$ by replacing $z$ and $r$ by $H_{\text{VC}}$ as in Figure 5 satisfies $\text{OPT}_{\text{DS}}(G') = \text{OPT}_{\text{DS}}(G) + 9$.*

**Proof.** We prove equality by establishing matching upper- and lower bounds.

($\leq$ Consider a minimum dominating set $S$ in $G$ that does not contain $r$ or $z$, which exists by Observation 12. Then $S$ contains at least one of $\{p, q\}$ to dominate $z$, and at least one

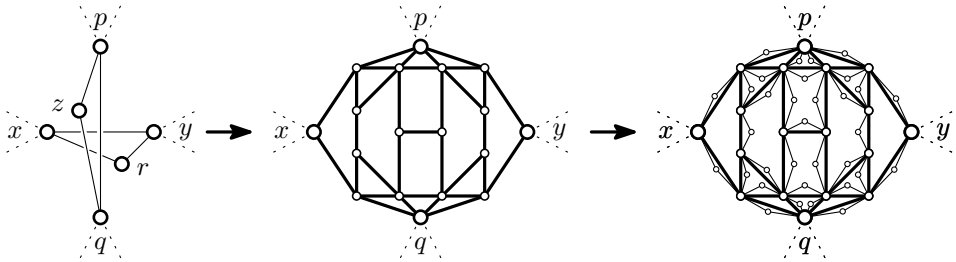



**Figure 5** Illustration of how a crossing between two triangles is eliminated in an instance of Dominating Set. A copy of $H_{\text{VC}}$ is inserted, whose four terminals are identified with the four endpoints of the crossing edge. For each edge $\{u, v\}$ of the inserted copy of $H_{\text{VC}}$, an additional degree-two vertex is inserted that forms a triangle with $u$ and $v$.

of $\{x, y\}$ to dominate $r$. We assume without loss of generality, by symmetry, that $p \in S$ and $x \in S$. As shown in Figure 4, there is a vertex cover for $H_{\text{VC}}$ of size 11 that contains $p$ and $x$, and therefore contains nine vertices from the interior of $H_{\text{VC}}$. Let $T$ be this set of nine vertices, and note that $T$ includes a neighbor of $q$ and a neighbor of $y$. We claim that $S' := S \cup T$ is a dominating set for $G'$ of size $|S| + 9 \leq \text{OPT}_{\text{DS}}(G) + 9$. Since $T \cup \{p, x\}$ is a vertex cover of $H_{\text{VC}}$ and $H_{\text{VC}}$ has no isolated vertices, each vertex of $H_{\text{VC}}$ has a neighbor in $S'$ and is dominated. The degree-two vertices that are inserted into $G'$ in the last step are dominated by the vertex that covers the edge with which they form a triangle. Vertices $q$ and $y$ are dominated from their neighbors in $T$. Finally, the remaining vertices of $G'$ are dominated in the same way as in $G$.

($\geq$ To prove $\text{OPT}_{\text{DS}}(G') \geq \text{OPT}_{\text{DS}}(G) + 9$, we instead show $\text{OPT}_{\text{DS}}(G) \leq \text{OPT}_{\text{DS}}(G') - 9$. Let $U \subseteq V(G')$ denote the vertices from the copy of $H_{\text{VC}}$ that was inserted; $U$ contains $p, q, x, y$, but $U$ does not contain the degree-two vertices that were inserted as the last step of the transformation. By Proposition 13, there is a minimum dominating set $S'$ of $G'$ such that $S' \cap U$ is a vertex cover of $G'[U]$. Let $\ell \in \{0, 1, 2\}$ be the number of pairs among $\{p, q\}$ and $\{x, y\}$ from which $S'$ contains no vertices. Since $S' \cap U$ is a vertex cover of $G'[U]$, which is isomorphic to $H_{\text{VC}}$, by Proposition 14 we know that $|(S' \cap U) \setminus \{p, q, x, y\}| \geq 9 + \ell$. Now let $S$ be obtained from $S'$ by removing all vertices of $(S' \cap U) \setminus \{p, q, x, y\}$, adding vertex $x$ if $S' \cap \{x, y\} = \emptyset$, and adding vertex $y$ if $S' \cap \{p, q\} = \emptyset$. Then $|S| \leq |S'| - 9$ since we remove $9 + \ell$ vertices and add $\ell$ new ones. Since $S$ contains at least one vertex from $\{p, q\}$ and at least one vertex from $\{x, y\}$, it dominates the two triangles in $G$. Since it contains a superset of the terminal vertices that $S'$ contains, the remaining vertices of the graph are dominated as before. Hence $S$ is a dominating set in $G$ and $\text{OPT}_{\text{DS}}(G) \leq \text{OPT}_{\text{DS}}(G') - 9$. ◀

Using the material so far, we can prove that the transformation operation in Figure 2 increases the size of an optimal dominating set by exactly 48.

▶ **Lemma 16.** *Let $\{a, b\}$ and $\{c, d\}$ be two disjoint edges of a graph $G$. Let $G'$ be the graph obtained by replacing these two edges as in Figure 2. Then $\text{OPT}_{\text{DS}}(G') = \text{OPT}_{\text{DS}}(G) + 48$.*

**Proof.** The transformation depicted in Figure 2 can be broken down into six steps: transform $\{a, b\}$ into a double-path structure, transform $\{c, d\}$ into a double-path structure, and perform four operations in which crossing triangles are replaced by gadgets. By Lemma 10, the two double-path insertions increase the size of a minimum dominating set by exactly $2 \cdot 6$. By Lemma 15, the four steps in which crossing triangles are eliminated increase the size of a minimum domination set by exactly $4 \cdot 9$. Hence $\text{OPT}_{\text{DS}}(G') = \text{OPT}_{\text{DS}}(G) + 12 + 36$. ◀

Using Lemma 16 we easily obtain the following.

▶ **Lemma 17.** *There is a useful crossover gadget for* Dominating Set.

**Proof.** The gadget that is inserted to replace two edges $\{a, b\}$ and $\{c, d\}$ in the procedure of Figure 2 is planar and has its terminals $u, u', v, v'$ on the outer face in the appropriate cyclic ordering. Since Lemma 16 shows that the replacement increases the size of a minimum dominating set by exactly 48, it follows that the structure serves as a useful crossover gadget for Dominating Set as per Definition 5. ◀

Theorem 2 now follows by combining Lemma 17 with the planarization argument of Theorem 6 and the lower bound for the nonplanar case given by Theorem 9.

▶ **Theorem 2.** *Assuming SETH, there is no $\varepsilon > 0$ such that* Dominating Set *on a planar graph $G$ given along with a linear layout of cutwidth $k$ can be solved in time $\mathcal{O}^*((3-\varepsilon)^k)$.*

**Proof.** Suppose Dominating Set on a planar graph with a given linear layout of cutwidth $k$ can be solved in $\mathcal{O}^*((3-\varepsilon)^k)$ time for some $\varepsilon > 0$, by an algorithm called $A$. Then Dominating Set on a nonplanar graph with a given layout of cutwidth $k$ can be solved in time $\mathcal{O}^*((3-\varepsilon)^k)$ by reducing it to a planar graph with a linear layout of cutwidth $k + \mathcal{O}(1)$ (using Theorem 6 and the existence of a useful crossover gadget; this blows up the graph size by at most a polynomial factor) and then running $A$. By Theorem 9, this contradicts SETH. ◀

## 5 Conclusion

In this work we have investigated whether SETH-based lower bounds for solving problems on graphs of bounded treewidth also apply for (1) planar graphs and (2) graphs of bounded cutwidth. To answer these questions, we showed that the graph parameter cutwidth can be preserved when reducing to a planar instance using suitably restricted crossover gadgets.

For both problems considered in this work, the runtime lower bound for solving the problem on graphs of bounded cutwidth continues to hold for *planar* graphs of bounded cutwidth. Hence planarity seems to offer no algorithmic advantage when working with graphs of bounded cutwidth. Moreover, for both Independent Set and Dominating Set the runtime lower bound for the treewidth parameterization also applies for cutwidth.

Future work may explore other combinatorial problems on graphs of bounded cutwidth. For example, what is the optimal running time for Feedback Vertex Set, Odd Cycle Transversal, or Hamiltonian Cycle on graphs of bounded cutwidth? What is the complexity of the cutwidth parameterization of these problems on planar graphs? For the Graph $q$-Coloring problem, these questions are answered in a recent manuscript by an overlapping set of authors [21]: planarity offers no advantage, but the parameterization by cutwidth $k$ can be solved in time $\mathcal{O}^*(2^k)$ for all $q$, sharply contrasting that the treewidth parameterization cannot be solved in time $\mathcal{O}^*((q-\varepsilon)^k)$ under SETH.

## References

1 Julien Baste and Ignasi Sau. The role of planarity in connectivity problems parameterized by treewidth. *Theor. Comput. Sci.*, 570:1–14, 2015. doi:10.1016/j.tcs.2014.12.010.

2 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: Fast subset convolution. In David S. Johnson and Uriel Feige, editors, *Proc. 39th STOC*, pages 67–74. ACM, 2007. doi:10.1145/1250790.1250801.

3 Hans L. Bodlaender. A Partial $k$-Arboretum of Graphs with Bounded Treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.

4 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.

5 Hans L. Bodlaender and Arie M. C. A. Koster. Combinatorial Optimization on Graphs of Bounded Treewidth. *Comput. J.*, 51(3):255–269, 2008. doi:10.1093/comjnl/bxm037.

6 Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex Cover: Further Observations and Further Improvements. *J. Algorithms*, 41(2):280–301, 2001. doi:10.1006/jagm.2001.1186.

7 Bruno Courcelle. The Monadic Second-Order Logic of Graphs I: Recognizable Sets of Finite Graphs. *Inf. Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

8 Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast hamiltonicity checking via bases of perfect matchings. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Proc. 45th STOC*, pages 301–310. ACM, 2013. doi:10.1145/2488608.2488646.

9 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving Connectivity Problems Parameterized by Treewidth in Single Exponential Time. In *Proc. 52nd FOCS*, pages 150–159, 2011. doi:10.1109/FOCS.2011.23.

10 Erik D. Demaine and MohammadTaghi Hajiaghayi. The Bidimensionality Theory and Its Algorithmic Applications. *Comput. J.*, 51(3):292–302, 2008. doi:10.1093/comjnl/bxm033.

11 Josep Díaz, Jordi Petit, and Maria J. Serna. A survey of graph layout problems. *ACM Comput. Surv.*, 34(3):313–356, 2002. doi:10.1145/568522.568523.

12 David Eppstein. Pathwidth of planarized drawing of $K_{3,n}$. TheoryCS StackExchange question, 2016. URL: http://cstheory.stackexchange.com/questions/35974/.

13 David Eppstein. The Effect of Planarization on Width. In *Proc. 25th GD*, volume 10692 of *LNCS*, pages 560–572, 2017. doi:10.1007/978-3-319-73915-1_43.

14 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient Computation of Representative Families with Applications in Parameterized and Exact Algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.

15 M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976. doi:10.1016/0304-3975(76)90059-1.

16 Bas A. M. van Geffen, Bart M. P. Jansen, Arnoud A. W. M. de Kroon, and Rolf Morel. Lower Bounds for Dynamic Programming on Planar Graphs of Bounded Cutwidth. *CoRR*, abs/1806.10513, 2018. arXiv:1806.10513.

17 Archontia C. Giannopoulou, Michal Pilipczuk, Jean-Florent Raymond, Dimitrios M. Thilikos, and Marcin Wrochna. Cutwidth: Obstructions and Algorithmic Aspects. In *Proc. 11th IPEC*, volume 63 of *LIPIcs*, pages 15:1–15:13, 2016. doi:10.4230/LIPIcs.IPEC.2016.15.

18 Russel Impagliazzo and Ramamohan Paturi. On the Complexity of $k$-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

19 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

20 Bart M. P. Jansen and Jules J. H. M. Wulms. Lower Bounds for Protrusion Replacement by Counting Equivalence Classes. In Jiong Guo and Danny Hermelin, editors, *Proc. 11th IPEC*, volume 63 of *LIPIcs*, pages 17:1–17:12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.IPEC.2016.17.

21 Bart M.P. Jansen and Jesper Nederlof. Computing the Chromatic Number Using Graph Decompositions via Matrix Rank. In *Proc. 26th ESA*, 2018. In press.

22 Ephraim Korach and Nir Solel. Tree-Width, Path-Width, and Cutwidth. *Discrete Applied Mathematics*, 43(1):97–101, 1993. doi:10.1016/0166-218X(93)90171-J.

23 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. In *Proc. 22nd SODA*, pages 777–789, 2011. doi:10.1137/1.9781611973082.61.

24 Dániel Marx. The Square Root Phenomenon in Planar Graphs. In Michael R. Fellows, Xuehou Tan, and Binhai Zhu, editors, *Proc. 3rd FAW-AAIM*, volume 7924 of *Lecture Notes in Computer Science*, page 1. Springer, 2013. doi:10.1007/978-3-642-38756-2_1.

25 Michal Pilipczuk. Problems Parameterized by Treewidth Tractable in Single Exponential Time: A Logical Approach. In *Proc. 36th MFCS*, pages 520–531, 2011. doi:10.1007/978-3-642-22993-0_47.

26 Dimitrios M. Thilikos, Maria J. Serna, and Hans L. Bodlaender. Cutwidth I: A linear time fixed parameter algorithm. *J. Algorithms*, 56(1):1–24, 2005. doi:10.1016/j.jalgor.2004.12.001.

27 Dimitrios M. Thilikos, Maria J. Serna, and Hans L. Bodlaender. Cutwidth II: Algorithms for partial w-trees of bounded degree. *J. Algorithms*, 56(1):25–49, 2005. doi:10.1016/j.jalgor.2004.12.003.

28 Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic Programming on Tree Decompositions Using Generalised Fast Subset Convolution. In *Proc. 17th ESA*, pages 566–577, 2009. doi:10.1007/978-3-642-04128-0_51.

# Multivariate Analysis of Orthogonal Range Searching and Graph Distances

**Karl Bringmann**
Max-Planck-Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
kbringma@mpi-inf.mpg.de

**Thore Husfeldt**[1]
BARC, IT University of Copenhagen, Denmark, and Lund University, Sweden
thore@itu.dk
https://orcid.org/0000-0001-9078-4512

**Måns Magnusson**
Department of Computer Science, Lund University, Sweden
mans.magnusson.888@student.lu.se

**Abstract**

We show that the eccentricities, diameter, radius, and Wiener index of an undirected $n$-vertex graph with nonnegative edge lengths can be computed in time $O(n \cdot \binom{k+\lceil \log n \rceil}{k} \cdot 2^k k^2 \log n)$, where $k$ is the treewidth of the graph. For every $\epsilon > 0$, this bound is $n^{1+\epsilon} \exp O(k)$, which matches a hardness result of Abboud, Vassilevska Williams, and Wang (SODA 2015) and closes an open problem in the multivariate analysis of polynomial-time computation. To this end, we show that the analysis of an algorithm of Cabello and Knauer (Comp. Geom., 2009) in the regime of non-constant treewidth can be improved by revisiting the analysis of orthogonal range searching, improving bounds of the form $\log^d n$ to $\binom{d+\lceil \log n \rceil}{d}$, as originally observed by Monier (J. Alg. 1980).

We also investigate the parameterization by vertex cover number.

**2012 ACM Subject Classification** Theory of computation → Shortest paths, Theory of computation → Parameterized complexity and exact algorithms, Theory of computation → Computational geometry, Mathematics of computing → Paths and connectivity problems

**Keywords and phrases** Diameter, radius, Wiener index, orthogonal range searching, treewidth, vertex cover number

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.4

**Acknowledgements** We thank Amir Abboud and Rasmus Pagh for useful discussions.

## 1 Introduction

Pairwise distances in an undirected, unweighted graph can be computed by performing a graph exploration, such as breadth-first search, from every vertex. This straightforward procedure determines the diameter of a given graph with $n$ vertices and $m$ edges in time $O(nm)$. It is surprisingly difficult to improve upon this idea in general. In fact, Roditty and Vassilevska Williams [14] have shown that an algorithm that can distinguish between diameter 2 and 3 in an undirected sparse graph in subquadratic time refutes the Orthogonal Vectors conjecture.

[1] Swedish Research Council grant VR-2016-03855 and Villum Foundation grant 16582.

© Karl Bringmann, and Thore Husfeldt, and Måns Magnusson;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 4; pp. 4:1–4:13
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

However, for very sparse graphs, the running time becomes linear. In particular, the diameter of a tree can be computed in linear time $O(n)$ by a folklore result that traverses the graph twice. In fact, an algorithm by Cabello and Knauer shows that for constant treewidth $k \geq 3$, the diameter (and other distance parameters) can be computed in time $O(n \log^{k-1} n)$, where the Landau symbol absorbs the dependency on $k$ as well as the time required for computing a tree decomposition. The question raised in [1] is how the complexity of this problem grows with the treewidth of the graph. We show the following result:

▶ **Theorem 1.** *The eccentricities, diameter, radius, and Wiener index of a given undirected $n$-vertex graph $G$ of treewidth* $\mathrm{tw}(G)$ *and nonnegative edge lengths can be computed in time linear in*

$$n \cdot \binom{k + \lceil \log n \rceil}{k} \cdot 2^k k^2 \log n \tag{1}$$

*where* $k = 5\,\mathrm{tw}(G) + 4$.

For every $\epsilon > 0$, the bound (1) is $n^{1+\epsilon} \exp O(\mathrm{tw}(G))$. This improves the dependency on the treewidth over the running time $n^{1+\epsilon} \exp O\big(\mathrm{tw}(G) \log \mathrm{tw}(G)\big)$ of Abboud, Vassilevska Williams, and Wang [1]. Our improvement is tight in the following sense. Abboud *et al.* [1] also showed that under the Strong Exponential Time Hypothesis of Impagliazzo, Paturi, and Zane [10], there can be no algorithm that computes the diameter with running time

$$n^{2-\delta} \exp o(\mathrm{tw}(G)) \qquad \text{for any } \delta > 0\,. \tag{2}$$

In fact, this holds under the potentially weaker Orthogonal Vectors conjecture, see [17] for an introduction to these arguments. Thus, under this assumption, the dependency on $\mathrm{tw}(G)$ in Theorem 1 cannot be significantly improved, even if the dependency on $n$ is relaxed from just above linear to just below quadratic. Our analysis encompasses the Wiener index, an important structural graph parameter left unexplored by [1].

Perhaps surprisingly, the main insight needed to establish Theorem 1 has nothing to do with graph distances or treewidth. Instead, we make – or re-discover – the following observation about the running time of $d$-dimensional range trees:

▶ **Lemma 2** ([13]). *A $d$-dimensional range tree over $n$ points supporting orthogonal range queries for the aggregate value over a commutative monoid has query time $O(2^d \cdot B(n,d))$ and can be built in time $O(nd \cdot B(n,d))$, where*

$$B(n,d) = \binom{d + \lceil \log n \rceil}{d}\,.$$

This is a more careful statement than the standard textbook analysis, which gives the query time as $O(\log^d n)$ and the construction time as $O(n \log^d n)$. For many values of $d$, the asymptotic complexities of these bounds agree – in particular, this is true for constant $d$ and for very large $d$, which are the main regimes of interest to computational geometers. But crucially, $B(n,d)$ is always $n^\epsilon \exp O(d)$ for any $\epsilon > 0$, while $\log^d n$ is not.

After Lemma 2 is realised, Theorem 1 follows via divide-and-conquer in decomposable graphs, closely following the idea of Cabello and Knauer [6] and augmented with known arguments [1, 5]. We choose to give a careful presentation of the entire construction, as some of the analysis is quite fragile.

Using known reductions, this implies that the following multivariate lower bound on orthogonal range searching is tight:

▶ **Theorem 3** (Implicit in [1]). *A data structure for the orthogonal range query problem for the monoid* $(\mathbf{Z}, \max)$ *with construction time* $n \cdot q'(n,d)$ *and query time* $q'(n,d)$*, where*

$$q'(n,d) = n^{1-\epsilon} \exp o(d)$$

*for some* $\epsilon > 0$*, refutes the Strong Exponential Time hypothesis.*

We also investigate the same problems parameterized by vertex cover number:

▶ **Theorem 4.** *The eccentricities, diameter, and radius of a given undirected, unweighted* $n$*-vertex graph* $G$ *with vertex cover number* $k$ *can be computed in time* $O(nk + 2^k k^2)$*. The Wiener index can be computed in time* $O(nk2^k)$*.*

Both of these bounds are $n \exp O(k)$. It follows from [1] that a lower bound of the form (2) holds for this parameter as well.

## 1.1 Related work

Abboud *et al.* [1] show that given a graph and an optimal tree decomposition, various graph distances can be computed in time $O(k^2 n \log^{k-1} n)$, where $k = \mathrm{tw}(G)$. This bound is $n^{1+\epsilon} \exp O(k \log k)$ for any $\epsilon > 0$. This subsumes the running time for finding an approximate tree decomposition with $k = O(\mathrm{tw}(G))$ from the input graph [5], which is $n \exp O(k)$. Their algorithm extends the construction of Cabello and Knauer [6] to superconstant treewidth. According to [6], the idea of expressing graph distances as coordinates was first mentioned by Shi [15].

If the diameter in the input graph is constant, the diameter can be computed in time $n \exp O(\mathrm{tw}(G))$ [9]. This is tight in both parameters in the sense that [1] rules out the running time (2) even for distinguishing diameter 2 from 3, and every algorithm needs to inspect $\Omega(n)$ vertices even for treewidth 1. For non-constant diameter $\Delta$, the bound from [9] deteriorates as $n \exp O(\mathrm{tw}(G) \log \Delta)$. However, the construction cannot be used to compute the Wiener index.

The literature on algorithms for graph distance parameters such as diameter or Wiener index is very rich, and we refer to the introduction of [1] for an overview of results directly relating to the present work. A recent paper by Bentert and Nichterlein [2] gives a comprehensive overview of many other parameterisations.

Orthogonal range searching using a multidimensional range tree was first described by Bentley [3], Lueker [12], Willard [16], and Lee and Wong [11], who showed that this data structure supports query time $O(\log^d n)$ and construction time $O(n \log^{d-1} n)$. Several papers have improved this in various ways by factors logarithmic in $n$; for instance, Chazelle's construction [8] achieves query time $O(\log^{d-1} n)$.

## 1.2 Discussion

In hindsight, the present result is a somewhat undramatic resolution of an open problem that has been viewed as potentially fruitful by many people [1], including the second author [9]. In particular, the resolution has led neither to an exciting new technique for showing conditional lower bounds of the form $n^{2-\epsilon} \exp \omega(k)$, nor a clever new algorithm for graph diameter. Instead, our solution follows the ideas of Cabello and Knauer [6] for constant treewidth, much like in [1]. All that was needed was a better understanding of the asymptotics of bivariate functions, rediscovering a 40-year old analysis of spatial data structures [13] (see the discussion in Sec. 3.3), and using a recent algorithm for approximate tree decompositions [5].

Of course, we can derive some satisfaction from the presentation of asymptotically tight bounds for fundamental graph parameters under a well-studied parameterization. In particular, the surprisingly elegant reductions in [1] cannot be improved. However, as we show in the appendix, when we parameterize by vertex cover number instead of treewidth, we can establish even cleaner and tight bounds without much effort.

Instead, the conceptual value of the present work may be in applying the multivariate perspective on high-dimensional computational geometry, reviving an overlooked analysis for non-constant dimension. To see the difference in perspective, Chazelle's improvement [8] of $d$-dimensional range queries from $\log^d n$ to $\log^{d-1} n$ makes a lot of sense for small $d$, but from the multivariate point of view, both bounds are $n^\epsilon \exp \Omega(d \log d)$. The range of relationships between $d$ and $n$ where the multivariate perspective on range trees gives some new insight is when $d$ is asymptotically just shy of $\log n$, see Sec. 2.1.

It remains open to find an algorithm for diameter with running time $n \exp O(\mathrm{tw}(G))$, or an argument that such an algorithm is unlikely to exist under standard hypotheses. This requires better understanding of the regime $d = o(\log n)$.

# 2 Preliminaries

## 2.1 Asymptotics

We summarise the asymptotic relationships between various functions appearing in the present paper:

▶ **Lemma 5.**

$$B(n,d) = O(\log^d n)\,. \tag{3}$$

*For any* $\epsilon > 0$,

$$B(n,d) = n^\epsilon \exp O(d)\,, \tag{4}$$

$$\log^d n = n^\epsilon \exp \Omega(d \log d)\,, \tag{5}$$

$$\log^d n = n^\epsilon \exp O(d \log d)\,. \tag{6}$$

The first expression shows that $B(n,d)$ is always at least as informative as $O(\log^d n)$. The next two expressions show that from the perspective of parameterised complexity, the two bounds differ asymptotically: $B(n,d)$ depends single-exponentially on $d$ (no matter how small $\epsilon > 0$ is chosen), while $\log^d n$ does not (no matter how *large* $\epsilon$ is chosen). Expression (6) just shows that (5) is maximally pessimistic.

**Proof.** Write $h = \lceil \log n \rceil$. To see (3), consider first the case where $d < h$. Using $\binom{a}{b} \le a^b/b!$ we see that

$$\binom{d+h}{d} \le \binom{2h}{d} \le \frac{(2h)^d}{d!} = \frac{2^d}{d!} h^d = O(\log^d n)\,. \tag{7}$$

Next, if $d \ge h$ then

$$\binom{d+h}{d} = \binom{d+h}{h} \le \binom{2d}{h} = \frac{2^h}{h!} d^h \le d^h\,,$$

provided $h \ge 4$. It remains to observe that $d^h \le h^d = O(\log^d n)$. Indeed, since the function $\alpha \mapsto \alpha / \ln \alpha$ is increasing for $\alpha \ge \mathrm{e}$, we have $h/\ln h \le d/\ln d$, which implies $\exp(h \ln d) \le \exp(d \ln h)$ as needed.

For (4), we let $\delta = d/h$ and consider two cases. First, from Stirling's formula we know $\binom{a}{b} \le \left(\frac{ea}{b}\right)^b$, so

$$\binom{d+h}{d} = \binom{(1+\delta)h}{\delta h} \le \left(\frac{e(1+\delta)h}{\delta h}\right)^{\delta h} \le \left(\frac{e(1+\delta)}{\delta}\right)^{2\delta \log n} = n^{2\delta \log(e(1+\delta)\delta^{-1})}.$$

Using that $\delta \mapsto 2\delta \log(e(1+\delta)\delta^{-1})$ is a monotone increasing function in the interval $\left(0, \frac{1}{2}\right]$ that tends to 0 for $\delta \to 0$, we obtain $\binom{d+h}{d} \le n^\epsilon$ for any sufficiently small $\delta$.

It remains to consider the case that $\delta \ge c$ for some positive constant $c$ depending only on $\epsilon$. In this case, we have

$$\binom{d+h}{d} \le \binom{(1+1/c)d}{d} < 2^{(1+1/c)d} = \exp O(d).$$

We turn to (5). Assume that there is a function $g$ such that

$$\log^d n = n^c g(d).$$

Then choose $b > 1$ and consider $d$ such that $d = b^{-1} \log n$. Then

$$g(d) \ge \frac{\log^d n}{n^c} = 2^{d \log\log n - c \log n} = 2^{d \log(bd) - cbd} = \exp \Omega(d \log d).$$

Finally for (6), we repeat the argument from [1]. If $d \le \epsilon \log n / \log\log n$ then $\log^d n = 2^{d \log\log n} \le n^\epsilon$. In particular, if $d = o(\log n / \log\log n)$ then $\log^d n = n^{o(1)}$. Moreover, for $d \ge \log^{1/2} n$ we have $\log\log n \le 2 \log d$ and thus $\log^d n = 2^{d \log\log n} \le 4^{d \log d}$. ◀

These calculations also show the regimes in which these considerations are at all interesting. For $d = o(\log n / \log\log n)$ then both functions are bounded by $n^{o(1)}$, and the multivariate perspective gives no insight. For $d \ge \log n$, both bounds exceed $n$, and we are better off running $n$ BFSs for computing diameters, or passing through the entire point set for range searching.

## 2.2 Model of computation

We operate in the word RAM, assuming constant-time arithmetic operations on coordinates and edge lengths, as well as constant-time operations in the monoid supported by our range queries. For ease of presentation, edge lengths are assumed to be nonnegative integers; we could work with abstract nonnegative weights instead [6].

# 3 Orthogonal Range Queries

## 3.1 Preliminaries

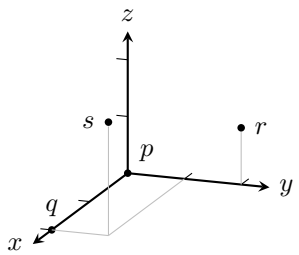
Let $P$ be a set of $d$-dimensional points. We will view $p \in P$ as a vector $p = (p_1, \ldots, p_d)$.

A commutative *monoid* is a set $M$ with an associative and commutative binary operator $\oplus$ with identity. The reader is invited to think of $M$ as the integers with $-\infty$ as identity and $a \oplus b = \max\{a, b\}$.

Let $f\colon P \to M$ be a function and define for each subset $Q \subseteq P$

$$f(Q) = \bigoplus \{ f(q) \colon q \in Q \}$$

with the understanding that $f(\emptyset)$ is the identity in $M$.




| | | |
|---|---|---|
| $p$ | $(0,0,0)$ | $f(p) = 5$ |
| $q$ | $(2,0,0)$ | $f(q) = 6$ |
| $r$ | $(0,2,1)$ | $f(r) = 7$ |
| $s$ | $(2,1,2)$ | $f(s) = 8$ |

**Figure 1** Four points in three dimensions. With the monoid $(\mathbf{Z}, \max)$ we have $f(\{p, r, s\}) = 8$.

## 3.2 Range Trees

Consider dimension $i \in \{1, \ldots, d\}$ and enumerate the points in $Q$ as $q^{(1)}, \ldots, q^{(r)}$ such that $q_i^{(j)} \leq q_i^{(j+1)}$, for instance by ordering after the $i$th coordinate and breaking ties lexicographically. Define $\mathrm{med}_i(Q)$ to be the *median* point $q^{(\lceil r/2 \rceil)}$, and similarly $\min_i(Q) = q^{(1)}$ and $\max_i(Q) = q^{(r)}$. Set

$$Q_L = \{q^{(1)}, \ldots, q^{(\lceil r/2 \rceil)}\}, \qquad Q_R = \{q^{(1+\lceil r/2 \rceil)}, \ldots, q^{(r)}\}. \tag{8}$$

For $i \in \{1, \ldots, d\}$, the *range tree* $R_i(Q)$ for $Q$ is a node $x$ with the following attributes:

- $L[x]$, a reference to range tree $T_i(Q_L)$, called the *left child* of $x$. Only exists if $|Q| > 1$.
- $R[x]$, a reference to range tree $T_i(Q_R)$, called the *right child* of $x$. Only exists if $|Q| > 1$.
- $D[x]$, a reference to range tree $T_{i+1}(Q)$, called the *secondary*, *associate*, or *higher-dimensional* structure. Only exists for $i < d$.
- $l[x] = \min_i(Q)$.
- $r[x] = \max_i(Q)$.
- $f[x] = f(Q)$. Only exists for $i = d$.

### Construction

Constructing a range tree for $Q$ is a straightforward recursive procedure:

▶ **Algorithm C** (Construction). *Given integer $i \in \{1, \ldots, d\}$ and a list $Q$ of points, this algorithm constructs the range tree $R_i(Q)$ with root $x$.*

**C1** [Base case $Q = \{q\}$.] Recursively construct $D[x] = T_{i+1}(Q)$ if $i < d$, otherwise set $f[x] = f(q)$. Set $l[x] = r[x] = q_i$. Return $x$.

**C2** [Find median.] Determine $q = \mathrm{med}_i\, Q$, $l[x] = \min_i(Q)$, $r[x] = \max_i(Q)$.

**C3** [Split $Q$.] Let $Q_L$ and $Q_R$ as given by (8), note that both are nonempty.

**C4** [Recurse.] Recursively construct $L[x] = R_i(Q_L)$ from $Q_L$. Recursively construct $R[x] = R_i(Q_R)$ from $Q_R$. If $i < d$ then recursively construct $D[x] = T_{i+1}(Q)$. If $i = d$ then set $f[x] = f[L[x]] \oplus f[R[x]]$.

The data structure can be viewed as a collection of binary trees whose nodes $x$ represent various subsets $P_x$ of the original point set $P$. In the interest of analysis, we now introduce a scheme for naming the individual nodes $x$, and thereby also the subsets $P_x$. Each node $x$ is identified by a string of letters from $\{\mathrm{L}, \mathrm{R}, \mathrm{D}\}$ as follows. Associate with $x$ a set of points, often called the *canonical subset* of $x$, as follows. For the empty string $\epsilon$ we set $P_\epsilon = P$. In general, if $Q = P_x$ then $P_{x\mathrm{L}} = Q_L$, $P_{x\mathrm{R}} = Q_R$ and $P_{x\mathrm{D}} = Q$. The strings over $\{\mathrm{L}, \mathrm{R}, \mathrm{D}\}$ can be understood as uniquely describing a path through in the data structure; for instance, L means 'go *left*, i.e., to the left subtree, the one stored at $L[x]$' and D means 'go to the next *dimension*, i.e., to the subtree stored at $D[x]$.' The name of a node now describes the unique path that reaches it.

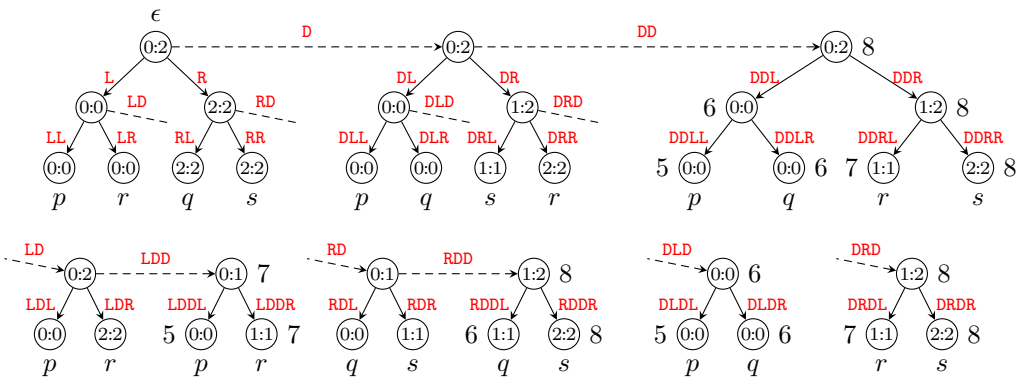



**Figure 2** Part of the range tree for the points from Fig. 1. The label of node $x$ appears in red on the arrow pointing to $x$. Nodes contain $l[x]{:}r[x]$. The references $L[x]$ and $R[x]$ appear as children in a binary tree using usual drawing conventions. The reference $D[x]$ appears as a dashed arrow (possibly interrupted); the placement on the page follows no other logic than economy of layout and readability. References $D[x]$ from leaf nodes, such as $D[\mathrm{LL}]$ leading to node LLD, are not shown; this conceals 12 single-node trees. The '3rd-dimensional nodes,' whose names contain two Ds, show the values $f[x]$ next to the node. To ease comprehension, leaf nodes are decorated with their canonical subset, which is a singleton from $\{p, q, r, s\}$. The reader can infer the canonical subset for an internal node as the union of leaves of the subtree; for instance, $P_{\mathrm{DR}} = \{r, s\}$. However, note that these point sets are *not* explicitly stored in the data structure.

▶ **Lemma 6.** *Let $n = |P|$. Algorithm C computes the $d$-dimensional range tree for $P$ in time linear in $nd \cdot B(n, d)$.*

**Proof.** We run Algorithm C on input $P$ and $i = 1$.

Disregarding the recursive calls, the running time of algorithm C on input $i$ and $Q$ is dominated by Steps C2 and C3, i.e., splitting $Q$ into two sets of equal size. It is known that this task can be performed in time linear in $|Q|$ [4]. Thus, the running time for constructing $R_i(Q)$ is linear in $|Q|$ plus the time spent in recursive calls.

This means that we can bound the running time for constructing $T_1(P)$ by bounding the sizes of the sets $P_x$ associated with every node $x$ in the data structure. If for a moment $X$ denotes the set of all these nodes then we want to bound

$$\sum_{x \in X} |P_x| = \sum_{x \in X} |\{\, p \in P \colon p \in P_x \,\}| = \sum_{p \in P} |\{\, x \in X \colon p \in P_x \,\}| \,.$$

Thus, we need to determine, for given $p \in P$, the number of subsets $P_x$ in which $p$ appears. By construction, there are fewer than $d$ occurrences of D in $x$. Moreover, if $x$ contains more than $h$ occurrences of either L or R then $P_x$ is empty. Thus, $x$ has at most $h + d$ letters. For two different strings $x$ and $x'$ that agree on the positions of D, the sets $P_x$ and $P_{x'}$ are disjoint, so $p$ appears in at most one of them. We conclude that the number of sets $P_x$ such that $p \in P_x$ is bounded by the number of ways to arrange fewer than $d$ many Ds and at most $h$ non-Ds. Using the identity $\binom{a+0}{0} + \cdots + \binom{a+b}{b} = \binom{a+b+1}{b}$ repeatedly, we compute

$$\sum_{i=0}^{d-1} \sum_{j=0}^{h} \binom{i+j}{j} = \sum_{i=0}^{d-1} \binom{i+h+1}{h} = \sum_{i=0}^{d-1} \binom{i+h+1}{i+1} =$$

$$(-1) + \sum_{i=0}^{d} \binom{i+h}{i} = \binom{h+d+1}{d} - 1 = \frac{h+d+1}{h+1} \binom{h+d}{d} - 1 \le d \binom{d+h}{d} \,.$$

The bound follows from aggregating this contribution over all $p \in P$. ◀

## Search

In this section, we fix two sequences of integers $l_1, \ldots, l_d$ and $r_1, \ldots, r_d$ describing the *query box* $B$ given by

$$B = [l_1, r_1] \times \cdots \times [l_d, r_d]\,.$$

▶ **Algorithm Q** (Query). *Given integer $i \in \{1, \ldots, d\}$, a query box $B$ as above and a range tree $R_i(Q)$ with root $x$ for a set of points $Q$ such that every point $q \in Q$ satisfies $l_j \le q_j \le r_j$ for $j \in \{1, \ldots, i-1\}$, this algorithm returns $\bigoplus\{\, f(q) \colon q \in Q \cap B \,\}$.*

**Q1** [Empty?] If the data structure is empty, or $l_i > r[x]$, or $l[x] > r_i$, then return the identity in the underlying monoid $M$.

**Q2** [Done?] If $i = d$ and $l_d \le \min_d[x]$ and $\max_d[x] \le r_d$ then return $f[x]$.

**Q3** [Next dimension?] If $i < d$ and $l_i \le l[x]$ and $r[x] \le r_i$ then query the range tree at $D[x]$ for dimension $i+1$. Return the resulting value.

**Q4** [Split.] Query the range tree $L[x]$ for dimension $i$; the result is a value $f_L$. Query the range tree $R[x]$ for dimension $i$; the result is a value $f_R$. Return $f_L \oplus f_R$.

To prove correctness, we show that this algorithm is correct for each point set $Q = P_x$.

▶ **Lemma 7.** *Let $i = D(x) + 1$, where $D(x)$ is the number of* Ds *in $x$. Assume that $P_x$ is such that $l_j \le p_i \le r_j$ for all $j \in \{1, \ldots, i-1\}$ for each $p \in P_x$. Then the query algorithm on input $x$ and $i$ returns $f(B \cap P_x)$.*

**Proof.** Backwards induction in $|x|$.

If $|x| = h + d$ then $P_x$ is the empty set, in which case the algorithm correctly returns the identity in $M$.

If the algorithm executes Step Q2 then $B$ is satisfied for all $q \in P_x$, in which case the algorithm correctly returns $f[x] = f(P_x)$.

If the algorithm executes Step Q3 then $B$ satisfies the condition in the lemma for $i+1$, and the number of Ds in $P_{x\mathrm{D}}$ is $i+1$, and $D[x]$ store the $(i+1)$th range tree for $P_{x\mathrm{D}}$. Thus, by induction the algorithm returns $f(P_{x\mathrm{D}} \cap B)$, which equals $f(P_x \cap B)$ because $P_{x\mathrm{D}} = P_x$.

Otherwise, by induction, $f_L = f(P_{x\mathrm{L}} \cap B)$ and $f_R = f(P_{x\mathrm{R}} \cap B)$. Since $P_{x\mathrm{L}} \cup P_{x\mathrm{R}} = P_x$, we have $f(P_x \cap B) = f((P_{x\mathrm{L}} \cap B) \cup (P_{x\mathrm{R}} \cap P)) = f_L \oplus f_R$. ◀

▶ **Lemma 8.** *If $x$ is the root of the range tree for $P$ then on input $i = 1$, $x$, and $B$, the query algorithm returns $f(P \cap B)$ in time linear in $2^d B(n, d)$.*

**Proof.** Correctness follows from the previous lemma.

For the running time, we first observe that the query algorithm does constant work in each visited node. Thus it suffices to bound the number of visited nodes as

$$2^d \binom{h+d}{d} \qquad (d \ge 1, h \ge 0)\,. \tag{9}$$

We will show by induction in $d$ that (9) holds for every call to a $d$-dimensional range tree for a point set $P_x$, where $h = \lceil \log |P_x| \rceil$. The two easy cases are Q1 and Q2, which incur no additional nodes to be visited, so the number of visited nodes is 1, which is bounded by (9). Step Q3 leads to a recursive call for a $(d-1)$-dimensional range tree over the same point set $P_{x\mathrm{D}} = P_x$, and we verify

$$1 + 2^{d-1} \binom{h+d-1}{d-1} \le 2^d \binom{h+d}{d}.$$

The interesting case is Step Q4. We need to follow two paths from $x$ to the leaves of the binary tree of $x$. Consider the leaves $l$ and $r$ in the subtree rooted at $x$ associated with the points $\min_i(P_x)$ and $\max_i(P_x)$ as defined in Sec. 3.2. We describe the situation of the path $Y$ from $l$ to $x$; the other case is symmetrical. At each internal node $y \in Y$, the algorithm chooses Step Q4 (because $l_i \geq l[y]$). There are two cases for what happens at $y$L and $y$R. If $l_i \leq \mathrm{med}_i(P_y)$ then $P_{y\mathrm{R}}$ satisfies $l_i \leq \min_i(P_{y\mathrm{R}}) \leq r_i$, so the call to $y$R will choose Step Q3. By induction, this incurs $2^{d-1}\binom{d-1+i}{d-1}$ visits, where $i$ is the height of $y$. In the other case, the call to $y$L will choose Step Q1, which incurs no extra visits. Thus, the number of nodes visited on the left path is at most

$$h + \sum_{i=0}^{h-1} 2^{d-1}\binom{d-1+i}{d-1},$$

and the total number of nodes visited is at most twice that:

$$2h + 2^d \sum_{i=0}^{h-1}\binom{d-1+i}{d-1} \leq 2^d \sum_{i=0}^{h}\binom{d-1+i}{d-1} = 2^d\binom{d+h}{d}. \qquad \blacktriangleleft$$

## 3.3 Discussion

The textbook analysis of range trees, and similar $d$-dimensional spatial algorithms and data structures sets up a recurrence relation like

$$r(n,d) = 2r(n/2,d) + r(n,d-1),$$

for the construction and

$$r(n,d) = \max\{\, r(n/2,d), r(n,d-1)\,\},$$

for the query time. One then observes that $n \log^d n$ and $\log^d n$ are the solutions to these recurrences. This analysis goes back to Bentley's original paper [3].

Along the lines of the previous section, one can show that the functions $n \cdot B(n,d)$ and $B(n,d)$ solve these recurrences as well. A detailed derivation can be found in [13], which also contains combinatorial arguments of how to interpret the binomial coefficients in the context of spatial data structures. A later paper of Chan [7] also takes the recurrences as a starting point, and observes asymptotically improved solution for the related question of dominance queries.

# 4 Graph Distances

We present the algorithm for computing the diameter. The construction closely follows Cabello and Knauer [6], but uses the range tree bounds from Section 3. The analysis is extended to superconstant dimension as in Abboud *et al.* [1]. Using the approximate treewidth construction of Bodlaender *et al.* [5], we can pay more attention to the parameters of the recursive decomposition into small-size separators.

## 4.1 Preliminaries

We consider an undirected graph $G$ with $n$ vertices and $m$ edges with nonnegative integer weights. The set of vertices is $V(G)$. For a vertex subset $U$ we write $G[U]$ for the induced subgraph.

A path from $u$ to $v$ is called a $u, v$-path and denoted $P$. The *length* of a path, denoted $l(P)$, is the sum of its edge lengths.

The *distance* from vertex $u$ to vertex $v$, denoted $d(u, v)$, is the minimum length of shortest $u, v$-path. The *Wiener index* of $G$, denoted $\mathrm{wien}(G)$ is $\sum_{u,v \in V(G)} d(u, v)$. The *eccentricity* of a vertex $u$, denoted $e(u)$ is given by $e(u) = \max\{\, d(u, v) \colon v \in V(G) \,\}$. The *diameter* of $G$, denoted $\mathrm{diam}(G)$ is $\max\{\, e(u) \colon u \in V(G) \,\}$. The *radius* of $G$, denoted $\mathrm{rad}(G)$ is $\min\{\, e(u) \colon u \in V(G) \,\}$.

## 4.2 Separation

A vertex subset $Z$ separates $X$ and $Y$ if every $x, y$-path with $x \in X$ and $y \in Y$ contains a vertex from $Z$. A *skew $k$-separator tree* $T$ of $G$ is a binary tree such that each node $t$ of $T$ is associated with a vertex set $Z_t \subseteq V(G)$ such that

- $|Z_t| \leq k$,
- If $L_t$ and $R_t$ denote the vertices of $G$ associated with the left and right subtrees of $t$, respectively, then $Z_t$ separates $L_t$ and $R_t$ and

$$\frac{n}{k+1} \leq |L_t \cup Z_t| \leq \frac{nk}{k+1}\,, \tag{10}$$

- $T$ remains a skew $k$-separator even if edges between vertices of $Z_t$ are added.

It is known that such a tree can be found from a tree decomposition, and an approximate tree decomposition can be found in single-exponential time. We summarise these results in the following lemma:

▶ **Lemma 9** ([6, Lemma 3] with [5, Theorem 1]). *For a given $n$-vertex input graph $G$, a skew $(5\,\mathrm{tw}(G) + 4)$-separator tree can be computed in time $n \exp O(\mathrm{tw}(G))$.*

## 4.3 Algorithm

We follow the construction of [6].

Given graph $G$, let $\mathcal{S}_{x,w}$ denote the set of shortest $x, w$-paths. We refine the notion of eccentricity to a subset $W$ of vertices. Formally,

$$e(x, W) = \max_{w \in W}\{\, l(P) \colon P \in \mathcal{S}_{x,w} \,\}\,.$$

We will consider a situation where $V(G) = X \cup Y$ with separator $Z = X \cap Y$. We can then compute $e(x)$ as $\max\{e(x, X), e(x, Y)\}$ for each $x \in X$. The first term is found recursively in $G[X]$; the interesting part is the computation of $e(x, Y)$.

Enumerate $Z = \{z_1, \ldots, z_k\}$. For $i \in \{1, \ldots, k\}$ define the $i$th eccentricity $e_i(x, Y)$ as the maximum distance from $x$ to any vertex in $Y$ 'via $z_i$.' Formally,

$$e_i(x, Y) = \max_{y \in Y}\{\, l(P) \colon P \in \mathcal{S}_{x,y}, z_i \in V(P) \,\}\,.$$

See Figure 3 for a small example.

▶ **Lemma 10.** *If $Z$ separates $X$ and $Y$ then $e(x, Y) = \max_{i=1}^{k} e_i(x, Y)$ for $x \in X$.*

**Proof.** A shortest $x, y$-path with $y \in Y$ must contain a vertex from $Z$, say $z_i$. Thus, $e(x, Y) \leq e_i(x, Y)$. Conversely, $e(x, Y) \geq e_j(x, Y)$ for all $j \in \{1, \ldots, k\}$ from the definition. ◀

Now we can write the eccentricity *via* $z_i$ as the distance *to* $z_i$ plus a range query:

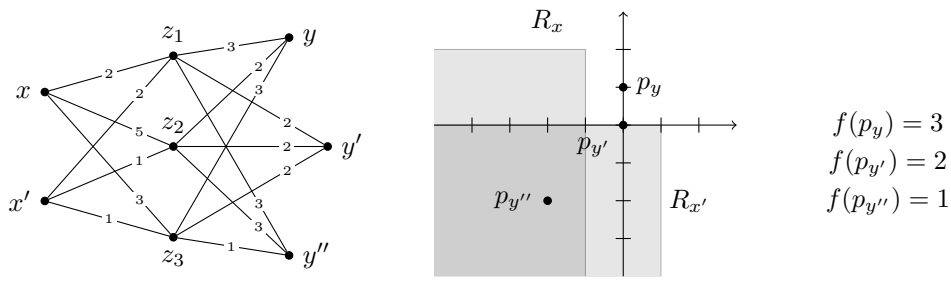



**Figure 3** Left: Example with $Z = \{z_1, z_2, z_3\}$ and $Y = Z \cup \{y, y', y''\}$. We have $e(x, Y) = 5$ (along $xz_1y$) and $e(x', Y) = 3$. For the case $i = 3$ we see $e_3(x, Y) = 4$ along $xz_3y''$, because there are no shortest paths from $x$ via $z_3$ to $y$ or $y'$, and the one-edge path $xz_3$ itself is shorter. Similarly, $e_3(x', Y) = 3$ (along $x'z_3y'$). Right: The corresponding points in $\mathbf{Z}^2$, only the first two coordinates are shown, and only for the points in $Y \setminus Z$. The points corresponding to $y'$ and $y''$ both belong to the rectangle for $x'$, certifying that there are shortest $x', y'$- and $x', y''$-paths through $z_3$. Right: Over $R_{x'}$, the point $p_{y'}$ maximises $f$. We have $e_3(x', Y) = l(x'z_3y') = d(x', z_3) + f(p_{y'}) = 1 + 2 = 3$.

▶ **Lemma 11.** *Let $i \in \{1, \dots, k\}$ and assume $\{z_1, \dots, z_k\}$ separates $X$ and $Y$. Define for each $y \in Y$ the $k$-dimensional point*

$$p_y = \begin{pmatrix} d(z_i, y) - d(z_1, y) \\ \vdots \\ d(z_i, y) - d(z_k, y) \end{pmatrix} \quad \text{with } f(p_y) = d(z_i, y)\,. \tag{11}$$

*Define for each $x \in X$ the rectangle*

$$R_x = \mathop{\times}_{j=1}^{k} [-\infty, d(x, z_j) - d(x, z_i)]\,. \tag{12}$$

*Then*

$$e_i(x, Y) = d(x, z_i) + \max_{y\colon p_y \in R_x} f(p_y)\,.$$

**Proof.** Consider a shortest $x, y$-path $P$ containing $z_i \in Z$. No other $x, y$-path is shorter than $P$, so in particular we have

$$d(x, z_i) + d(z_i, y) \le d(x, z_j) + d(z_j, y)\,, \qquad j \in \{1, \dots, k\}\,,$$

equivalently,

$$d(z_i, y) - d(z_j, y) \le d(x, z_j) - d(x, z_i)\,, \qquad j \in \{1, \dots, k\}\,. \tag{13}$$

which means $p_y \in R_x$. Moreover, if $y$ is chosen so that $P$ attains the eccentricity $e_i(x, Y)$ then $e_i(x, Y) = l(P) = d(x, z_i) + d(z_i, y)$ and $p_y$ maximises $f(p_y) = d(z_i, y)$ over the points in $R_x$. ◀

One observes that the $i$th coordinate of $p_y$ is always 0 and of $R_y$ is always $[-\infty, 0]$, so the reduction is actually to a $(k-1)$-dimensional range query instance. However, we are mainly interested in the asymptotic dependency on $k$, so we avoid this possible (but tedious) improvement.

We are ready for the algorithm.

▶ **Algorithm E** (Eccentricities). *Given an undirected, connected graph $G$ with nonnegative integer weights and a skew $k$-separator tree with root $t$, this algorithm computes the eccentricity $e(v)$ of every vertex $v \in V(G)$. We write $Z = Z_t$, $X = L_t \cup Z_t$, and $Y = R_t \cup Z_t$.*

**E1** [Base case.] If $n/\ln n < 4k(k+1)$ find all distances using Dijkstra's algorithm. Terminate.

**E2** [Distances from separator.] Compute $d(z,v)$ for each $z \in Z, v \in V(G)$ using $k$ applications of Dijkstra's algorithm. Compute $e(z,Y) = \max_{y \in Y} d(z,y)$ for each $z \in Z$.

**E3** [Add shortcuts.] For each pair $z, z' \in Z$, add the edge $zz'$ to $G$, weighted by $d(z,z')$. Remove duplicate edges, retaining the shortest.

**E4.1** [Start iterating over $\{z_1, \ldots, z_k\}$.] Let $i = 1$.

**E4.2** [Build range tree for $z_i$.] Construct a $k$-dimensional range tree for the points $\{\, p_y : y \in Y \,\}$ given by (11) using the monoid $(\mathbf{Z}, \max)$.

**E4.3** [Query range tree.] For each $x \in X$, query the rectangle $R_x$ given by (12) and add $d(x, z_i)$. The result is $e_i(x,Y)$ by Lemma 11.

**E4.4** [Next $z_i$.] If $i < k$ then increase $i$ and go to E4.1.

**E5** [Recurse on $G[X]$ and combine.] Recursively compute the distances in $G[X]$ using the left subtree of $t$ as a skew $k$-separator tree. The result are eccentricities $e(x,X)$ for each $x \in X$. For each $x \in X$, set $e(x,Y) = \max_{i=1}^{k} e_i(x,Y)$ from Step E4.3, then set $e(x) = \max\{e(x,X), e(x,Y)\}$.

**E6** [Flip.] Repeat Steps E4–5 with the roles of $X$ and $Y$ exchanged.

## 4.4 Running Time

▶ **Lemma 12.** *The running time of Algorithm E is $O(n \cdot B(n,k) \cdot 2^k k^2 \log n)$.*

We omit the proof. We can now establish Theorem 1 for diameter and radius.

**Proof of Thm. 1, distances.** To compute all eccentricities for a given graph we find a $k$-skew separator for $k = 5\,\mathrm{tw}(G) + 4$ using Lemma 9 in time $n \exp O(\mathrm{tw}(G))$. We then run Algorithm E, using Lemma 12 to bound the running time. From the eccentricities, the radius and diameter can be computed in linear time using their definition. ◀

Algorithm E can be modified to compute the Wiener index, as described in [6, Sec. 4], completing the proof of Theorem 1. The main observation is that the sum of distances between all pair $u, v \in V(G)$ can be written as pairwise distances within $X$, within $Y$, and between $X$ and $Y$, carefully subtracting contributions from these sums that were included twice. The orthogonal range queries for vertex $x \in X$ now need to report the sum of distances to every $y \in Y$, rather than just the value of the maximum distance $e(x;Y)$. To this end, we use the monoid of positive integer tuples $(d,r)$ with the operation $(d,r) \oplus (d',r') = (d+d', r+r')$ with identity element $(0,0)$. The value associated with vertex $x$ in Step E4.2 is $f(p(y)) = (1, d(z_i, y))$. To avoid overcounting, the definition of $R_x$ and $e_i(x,Y)$ have to be changed carefully.

## References


1. Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proceedings of the Twenty-Seventh Annual ACM–SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, Va, USA, January 10–12, 2016*, pages 377–391. SIAM, 2016. doi:10.1137/1.9781611974331.ch28.
2. Matthias Bentert and André Nichterlein. Parameterized Complexity of Diameter. *CoRR*, abs/1802.10048, 2018. arXiv:1802.10048.
3. Jon Louis Bentley. Multidimensional Divide-and-Conquer. *Commun. ACM*, 23(4):214–229, 1980. doi:10.1145/358841.358850.

4 Manuel Blum, Robert W. Floyd, Vaughan R. Pratt, Ronald L. Rivest, and Robert Endre Tarjan. Time Bounds for Selection. *J. Comput. Syst. Sci.*, 7(4):448–461, 1973. doi:10.1016/S0022-0000(73)80033-9.

5 Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michał Pilipczuk. An $O(c^k n)$ 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016. doi:10.1137/130947374.

6 Sergio Cabello and Christian Knauer. Algorithms for bounded treewidth with orthogonal range searching. *Comput. Geom.*, 42(9):815–824, 2009. doi:10.1016/j.comgeo.2009.02.001.

7 Timothy M. Chan. All-Pairs Shortest Paths with Real Weights in $O(n^3/\log n)$ Time. *Algorithmica*, 50(2):236–243, 2008. doi:10.1007/s00453-007-9062-1.

8 Bernard Chazelle. Lower Bounds for Orthogonal Range Searching: I. The Reporting Case. *J. Assoc. Comput. Mach.*, 37(2):200–212, 1990. doi:10.1145/77600.77614.

9 Thore Husfeldt. Computing Graph Distances Parameterized by Treewidth and Diameter. In *11th International Symposium on Parameterized and Exact Computation (IPEC 2016)*, volume 63 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:11, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.IPEC.2016.16.

10 Russel Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

11 Der-Tsai Lee and Chakkuen K. Wong. Quintary Trees: A File Structure for Multidimensional Database Systems. *ACM Trans. Database Syst.*, 5(3):339–353, 1980. doi:10.1145/320613.320618.

12 George S. Lueker. A Data Structure for Orthogonal Range Queries. In *19th Annual Symposium on Foundations of Computer Science, Ann Arbor, Michigan, USA, 16-18 October 1978*, pages 28–34. IEEE Computer Society, 1978. doi:10.1109/SFCS.1978.1.

13 Louis Monier. Combinatorial Solutions of Multidimensional Divide-and-Conquer Recurrences. *J. Algorithms*, 1(1):60–74, 1980. doi:10.1016/0196-6774(80)90005-X.

14 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 515–524, 2013. doi:10.1145/2488608.2488673.

15 Qiaosheng Shi. *Efficient algorithms for network center/covering location optimization problems.* PhD thesis, School of Computing Science, Simon Fraser University, 2008.

16 Dan E. Willard. New data structures for orthogonal range queries. *SIAM J. Comput.*, 14(1):232–253, 1985. doi:10.1137/0214019.

17 Virginia Vassilevska Williams. Hardness of Easy Problems: Basing Hardness on Popular Conjectures such as the Strong Exponential Time Hypothesis (Invited Talk). In *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, pages 17–29, 2015. doi:10.4230/LIPIcs.IPEC.2015.17.

# A Faster Tree-Decomposition Based Algorithm for Counting Linear Extensions

**Kustaa Kangas**
Department of Computer Science, Aalto University, Espoo, Finland
juho-kustaa.kangas@aalto.fi

**Mikko Koivisto**
Department of Computer Science, University of Helsinki, Helsinki, Finland
mikko.koivisto@helsinki.fi

**Sami Salonen**
Department of Computer Science, University of Helsinki, Helsinki, Finland
sami.m.salonen@helsinki.fi

## Abstract

We consider the problem of counting the linear extensions of an $n$-element poset whose cover graph has treewidth at most $t$. We show that the problem can be solved in time $\tilde{O}(n^{t+3})$, where $\tilde{O}$ suppresses logarithmic factors. Our algorithm is based on fast multiplication of multivariate polynomials, and so differs radically from a previous $\tilde{O}(n^{t+4})$-time inclusion–exclusion algorithm. We also investigate the algorithm from a practical point of view. We observe that the running time is not well characterized by the parameters $n$ and $t$ alone, fixing of which leaves large variance in running times due to uncontrolled features of the selected optimal-width tree decomposition. For selecting an efficient tree decomposition we adopt the method of empirical hardness models, and show that it typically enables picking a tree decomposition that is significantly more efficient than a random optimal-width tree decomposition.

**2012 ACM Subject Classification** Theory of computation → Algorithm design techniques, Theory of computation → Parameterized complexity and exact algorithms

**Keywords and phrases** Algorithm selection, empirical hardness, linear extension, multiplication of polynomials, tree decomposition

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.5

**Funding** This work was partially supported by the Academy of Finland, Grant 276864, "Supple Exponential Algorithms."

**Acknowledgements** We thank the anonymous reviewers for constructive suggestions, most of which are implemented in the present version of the paper.

## 1 Introduction

Consider a partially ordered set $(V, \prec)$, or *poset* for short, formed by an $n$-element set $V$ and an irreflexive and transitive binary relation $\prec$ on $V$, called a *partial order*. Another partial order $<$ on $V$ is a *linear extension* of $\prec$ if it contains $\prec$ and for any distinct elements $x, y \in V$ either $x < y$ or $y < x$. The problem of counting linear extensions (#LE) asks the number of linear extensions of a given poset; equivalent to #LE is the problem of counting the topological sorts of a given directed (acyclic) graph. The problem has applications in numerous areas, for example, in sequence analysis [22], sorting [27], preference reasoning [21], convex rank tests [24], partial order plans [25], and learning graphical models [32, 26].

© Kustaa Kangas, Mikko Koivisto, and Sami Salonen;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 5; pp. 5:1–5:13
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The #LE problem is #**P**-complete [11] but admits a fully polynomial approximation scheme [13]. The best known asymptotic bounds for the expected running time are $O(\epsilon^{-2}n^3 \log^2 \ell \log n)$ [6] and $O(\epsilon^{-2}n^5 \log^2 n)$ [30], where $\ell$ is the number of linear extensions and $\epsilon$ the allowed relative error. These bounds, while polynomial, become prohibitively large in practice if, say, one requires an accuracy of $\epsilon = 0.01$ and $n$ is around one hundred.

Exact and parameterized algorithms offer an alternative approach to design practical algorithms for the problem. For the required number of arithmetic operations, the best known worst-case bound is $O(2^n n)$ in general, which is achieved by a simple dynamic programming algorithm. For several special instance classes better bounds are known: $O(n^w w)$ for width-$w$ posets, $O(n^2)$ for series–parallel posets [23], and $O(n^2)$ also for posets whose cover graph is a forest [5]; the *cover graph* of a poset $(V, \prec)$ is the directed graph $(V, E)$ where the edge set $E$ is the transitive reduction of $\prec$ (a.k.a. Hasse diagram). If parameterized by the treewidth of the cover graph, $t$, the problem can be solved with $O(n^{t+3})$ arithmetic operations by an inclusion–exclusion algorithm [17]. On the other hand, the problem parameterized by $t$ is $W[1]$-hard [14], and so it may be difficult, or even impossible, to find an algorithm that runs in time $O(f(t)n^d)$ for some computable function $f$ and constant $d$.

## 1.1 Theoretical contributions

In this paper, we give a new algorithm that exploits small treewidth. Write $\tilde{O}(f)$ as a shorthand for $f(\log f)^{O(1)}$. We prove the following in Section 3.

▶ **Theorem 1** (Main). *The linear extensions of a given $n$-element poset can be counted with $\tilde{O}(n^{t+3})$ bit-operations, where $t$ is the treewidth of the cover graph of the poset.*

We state the bound in terms of bit-complexity to emphasize the fact that the dominating computations deal with large integers. Indeed, the bounds stated in the previous paragraph refer to the number of arithmetic operations. Thus, in particular, our bound improves the previous bound of Kangas et al. [17] by a factor of $n$. For large $n$ and small $t$ the improvement is relatively significant; for instance, for $t = 2$ the bound is reduced from $\tilde{O}(n^6)$ to $\tilde{O}(n^5)$.

Perhaps more importantly, the design of our algorithm is very different from that of the inclusion–exclusion algorithm. In the latter, the idea is to view a linear extension as a bijective mapping and then remove the global bijectivity constraint by inclusion–exclusion, similarly to previous applications to matrix permanent [29], Hamiltonian path [18], and set partitioning [19], but incurring only a polynomial overhead. Once the bijectivity constraint is removed, what remains is a collection of simpler subproblems with local constraints. The subproblems can be handled by standard routines that exploit small treewidth [7, 12]; see Section 2.4 for some additional details.

The present algorithm, in contrast, takes care of the bijectivity constraint within dynamic programming along a tree decomposition and is, with this respect, similar to a folklore $t^{O(t)}n$-time algorithm for the Hamiltonian path problem. However, #LE being W[1]-hard one may expect it to require a significantly larger dynamic programming table. We give a formulation, where each node of a tree decomposition is associated with $\Theta(n^{t+1})$ counts. This formulation leads to a challenge: a step in the dynamic program that combines two (or more) arrays of such counts appears to require a quadratic number of arithmetic operations, $\Theta(n^{2t+2})$, if implemented in a straightforward manner. Fortunately, we discover that the key ingredient of the step takes a form of multidimensional convolution, which we can compute efficiently using known (deep) results for fast multiplication of multivariate polynomials.

Concerning space complexity we only make a couple of observations here: Both our algorithm and the inclusion–exclusion algorithm by Kangas et al. [17] require $\tilde{O}(n^{t+2})$ bits of

space; this can be reduced by a factor about linear in $n$ by carrying the computations modulo several small relative primes and constructing the final output using the Chinese remainder theorem. The simple dynamic programming algorithm requires $\tilde{O}(2^n)$ bits of space.

## 1.2 Empirical contributions

In the second part of the paper we address the practical value of the algorithm. Given that the present algorithm is technically more convoluted than the inclusion–exclusion algorithm, it is natural to ask, whether the improvement in the asymptotic worst case time requirement can be realized in practice. Our interest is particularly in instances where $t$ is small (at most four) and $n$ ranging up to a few hundred.

A well known challenge in practical implementation of tree-decomposition based algorithms is that finding an optimal-width tree decomposition may be insufficient for minimizing the computational cost: the running time of the dynamic programming algorithm can be sensitive to the shape of the tree decomposition. Bodlaender and Fomin [9] addressed this issue from a theoretical viewpoint by studying the complexity of finding a tree decomposition that minimizes a sum of costs associated with each node of a tree decomposition. In their *$f$-cost* framework the cost of a node is allowed to depend only on the width of the node (i.e., the size of the associated bag; see Section 2). Recently, Abseher et al. [1, 2] presented a more general and more practical heuristic approach. Their `htd` library [1] allows a user to generate a variety of optimal-width tree decompositions and also (locally) optimize a given cost function. Moreover, they proposed and evaluated [2] a method to learn an appropriate cost function, or regression model, from collected empirical data on running times on varying instances. The method can be viewed as an instantation of the method of empirical hardness models [20] for the algorithm selection problem [28].

Following these ideas we have implemented and tested our algorithm for #LE using a collection of synthetically generated instances (posets) together with a variety of tree decompositions generated by `htd` for each instance. We will report on and discuss our preliminary observations, which suggest that selecting the tree decomposition using a learned regression model can make a difference, at least for the smallest treewidth ($t = 2$): compared to the median running time over generated tree decompositions, the selected one typically yields almost an order-of-magnitude speedup.

## 1.3 Organization

Some preliminary material is given in Section 2. Section 3 is devoted to proving Theorem 1. In Section 4 we descibe some implementation details and report on empirical results. We conclude in Section 5 by highlighting the main observations and some open questions.

# 2 Preliminaries

In this section we introduce some basic terminology, notation, and facts.

## 2.1 Basic notation

We denote by $\mathbb{N}$ the set of natural numbers $\{0, 1, 2, \ldots\}$. For two sets $S$ and $U$ we write $S^U$ for the set of functions from $U$ to $S$. If $m \in \mathbb{N}$ we write $[m]$ for the set $\{1, \ldots, m\}$. By $S^m$ we denote the set of $m$-tuples $a = (a_i)_{i=1}^m$ with $a_i \in S$.

The *restriction* of a function $\alpha : U \to S$ to a subset $A \subset U$ is defined in the standard manner and denoted by $\alpha_{|A}$; conversely, we say that $\alpha$ is an *extension* of $\alpha_{|A}$. We also denote by $\alpha^{v \mapsto i}$ the extension of $\alpha$ that we obtain by mapping an added element $v \notin U$ to $i$.

We use the Iverson's bracket notation: for a proposition $P$, the expression $[P]$ evaluates to 1 if $P$ is true, and to 0 otherwise.

## 2.2 Factorials and multiplication of polynomials

Let $a = (a_1, \ldots, a_k) \in \mathbb{N}^k$. We denote $a! := a_1! \cdots a_k!$ and $|a| := a_1 + \cdots + a_k$. Since $|a|!/a!$ is a multinomial coefficient, we have the following.

▶ **Fact 2.** *If $a$ is a tuple of nonegative integers, then $a!$ divides $|a|!$.*

Another fact we need concerns the complexity of multiplying two multivariate polynomials that are sparse in the sense that their total degree is given an upper bound, while the degrees of the individual variables may be large (not larger than the total degree, of course). We assume that a polynomial is represented by a list of its coefficients. The following result can be obtained by a straightforward specialization of a more general and detailed bound due to van der Hoeven and Lecerf [31, Cor. 4].

▶ **Fact 3.** *Two $k$-variate polynomials with $\tilde{O}(n)$-bit integer coefficients and total degree at most $n$ can be multiplied with $\tilde{O}\big(\binom{n+k}{k}(n+k)\big)$ bit-operations.*[1]

For univariate polynomials the bound is quadratic in $n$ because the coefficients can be large.

## 2.3 Tree decomposition

▶ **Definition 4** (Tree decomposition)**.** A *tree decomposition* of a graph $G = (V, E)$ is a pair $(T, B)$ where $T = (I, F)$ is a tree and $B$ maps each $x \in I$ to a *bag* $B_x \subseteq V$ such that

1. for each $v \in V$, the set $\{x \in I : v \in B_x\}$ induces a nonempty connected subtree of $T$,
2. for each $uv \in E$, there exists $x \in I$ with $u, v \in B_x$.

The *width* of the tree decomposition is the size of its largest bag minus one, $\max_{x \in I} |B_x| - 1$, and the *treewidth* of a graph is the minimum width over all its tree decompositions.

For a graph of treewidth $t$, we can find a tree decomposition of width $t$ in time $t^{O(t^3)}n$ using Bodlaender's algorithm [8] or in time $\tilde{O}(n^{t+2})$ using the approach of Arnborg et al. [4].

A tree decomposition is *rooted* if the edges of the tree are directed so that there is a unique node, the *root*, that has no parent. Clearly, one obtains a rooted tree decomposition by simply choosing one node as the root and directing the edges accordingly.

▶ **Definition 5** (Nice tree decomposition)**.** A rooted tree decomposition $(T, B)$ of a graph $G = (V, E)$ is *nice* if each node $x$ of $T$ is of one of the following types:

**(leaf)** $x$ has no children and $|B_x| = 1$;
**(introduce)** $x$ has a unique child $y$ and $B_x = B_y \cup \{v\}$ for some $v \in V \setminus B_y$;
**(forget)** $x$ has a unique child $y$ and $B_x = B_y \setminus \{v\}$ for some $v \in B_y$;
**(join)** $x$ has exactly two children $y, z$ and $B_x = B_y = B_z$.

We can convert a given tree decomposition of width $t$ and $O(n)$ nodes into a nice tree decomposition of width $t$ and $O(tn)$ nodes in time $t^{O(1)}n$ [10].

[1] In fact, this result assumes we have access to a $\tilde{\Omega}(n)$-bit prime $p$ and to a primitive element (i.e., generator) of $\mathbb{F}_p^*$, a multiplicative field of order $p$. Finding such numbers seem hard, theoretically: no deterministic polynomial time algorithm is known for the former, and no polynomial time algorithm is known for the latter. The result thus concerns the non-uniform model of computational complexity.

## 2.4 Counting linear extensions via inclusion–exclusion

Kangas et al. [17] showed that the number of linear extensions of a poset $(V, \prec)$ whose cover graph is $G = (V, E)$ is given by the formula

$$\sum_{k=1}^{n} \binom{n}{k} (-1)^{n-k} \sum_{\tau} \prod_{uv \in E} [\tau(u) < \tau(v)] ,$$

where $\tau$ runs over all functions from $V$ to $[k]$.

Supposing $G$ has treewidth $t$, it is well known that there is an elimination ordering $v_1, \ldots, v_n$ of the vertices, such that when removing the vertices from the graph in this order and always connecting the neighbors of the removed vertex, the size of the largest clique in the obtained graphs is $t + 1$. The $n$-dimensional inner summation over the variables $\tau(v_i)$, for $i \in [n]$, can be processed iteratively along such an ordering, the $i$th one-dimensional summation over $\tau(v_i)$ requiring $O(n_i k^{t+1})$ additions and multiplications of $O(n \log n)$-bit numbers, for some $n_1, \ldots, n_n$ that sum up to $O(n)$. In total, the evaluation of the inclusion–exclusion formula thus requires $\tilde{O}(n^{t+4})$ bit-operations. We omit a more detailed treatment of the algorithm, as the method applied for computing the inner summation is standard. (The original analysis of Kangas et al. [17] use a looser bound of $n_i = O(n)$ for each $i$, arriving at a bound that is larger by a factor of $n$.)

# 3 The algorithm – proof of Theorem 1

We implement a standard recipe of designing a tree-decomposition based algorithm. The outline of the algorithm is as follows.

**A1** Compute the cover graph of the input poset.
**A2** Find a minimum-width nice tree decomposition of the cover graph.
**A3** Run dynamic programming over the nice tree decomposition.

We will next consider each step in detail. We will see that the last step dominates our asymptotic running time bounds.

## 3.1 Computing the cover graph

The cover graph $G = (V, E)$ is obtained by computing the transitive reduction of the input poset $(V, \prec)$. The transitive reduction can be computed in time $O(|V| \cdot |\prec|)$ [3], which is $O(n^{t+2})$ for all $t \geq 1$.

## 3.2 Finding a minimum-width nice tree decomposition

As mentioned in Section 2, if the cover graph has treewidth $t$, then a width-$t$ nice tree decomposition of the cover graph can be found in $\tilde{O}(n^{t+2})$ time.

## 3.3 Dynamic programming

Suppose now that a width-$t$ nice tree decomposition $(T, B)$ of the cover graph $G$ is available. Our idea will be to associate each node of $T$ with an array of numbers such that (i) the numbers at the root node are sufficient for computing the number of linear extensions and (ii) the array of a node can be computed from the arrays of its child nodes.

The following notation will be useful. Denote by $V_x$ the set of vertices covered by the subtree of $T$ rooted at $x$, that is, $V_x$ is the union of the bags $B_y$ of nodes $y$ to which there is a directed path from $x$. Write $n_x$ for the size $|V_x|$ and $E_x$ for set of edges in the induced graph $G[V_x]$.

Now, for each node $x \in T$ and injection $\alpha \in [n_x]^{B_x}$, define $\ell_x(\alpha)$ as the number of bijections $\pi \in [n_x]^{V_x}$ such that $\pi(v) = \alpha(v)$ for all $v \in B_x$, and $\pi(u) < \pi(v)$ whenever $uv \in E_x$. In other words, $\ell_x(\alpha)$ is the number of ways to extend $\alpha$ to a linear extension of the induced poset $(V_x, \prec \cap (V_x \times V_x))$, where we view a linear extension as a bijection from $V_x$ to $[n_x]$ that satisfies the ordering constraints.

We begin by showing that the values $\ell_x(\alpha)$ are sufficient for computing the number of linear extensions of the poset, that is, they satisfy the listed conditions (i) and (ii). After that we consider the time requirement of computing the values $\ell_x(\alpha)$ for each node of the nice tree decomposition.

Consider first the root node.

▶ **Lemma 6** (Root). *We have $\ell(V) = \sum_\alpha \ell_r(\alpha)$, where $\alpha$ runs over all injections in $[n_x]^{B_r}$.*

**Proof.** Since $V_r = V$, $E_r = E$, and $n_r = n$, we have that

$$\sum_\alpha \ell_r(\alpha) = \sum_\pi \prod_{uv \in E} [\pi(u) < \pi(v)] = \ell(V)\,,$$

where $\alpha$ and $\pi$ run over all injections in $[n]^{B_r}$ and $[n]^V$, respectively. ◀

Next we will show separately for each node type of the nice tree decomposition, how the values $\ell_x(\alpha)$ are determined by the corresponding values for the child node or child nodes of $x$. For all but join nodes the results are immediate, and we omit the proofs.

▶ **Lemma 7** (Leaf). *If $x$ is a leaf node, then $\ell_x(\alpha) = 1$ for the unique injection $\alpha$ in $[n_x]^{B_x}$.*

For an introduce node, we simply restrict the injection $\alpha$ to the bag in question and check that the ordering constraint holds.

▶ **Lemma 8** (Introduce). *If $x$ is an introduce node with child $y$, then*

$$\ell_x(\alpha) = \ell_y(\alpha_{|B_y}) \prod_{u,v \in B_x : uv \in E} [\alpha(u) < \alpha(v)]\,.$$

For a forget node, we extend the injection $\alpha$ to the larger bag by mapping the new vertex to some value.

▶ **Lemma 9** (Forget). *If $x$ is a forget node with child $y$ and $B_x = B_y \setminus \{v\}$, then*

$$\ell_x(\alpha) = \sum_{a=1}^{n_y} \ell_y(\alpha^{v \mapsto a})\,.$$

To handle a join node, we introduce some convenient notation. Let $\alpha$ be an injection from a $k$-element set $S$ to a range of integers $[m]$. Label the elements of $S$ such that $\alpha(v_1) < \cdots < \alpha(v_k)$. For $i = 1, \ldots, k-1$, denote by $\alpha_i$ the number of integers between $\alpha(v_i)$ and $\alpha(v_{i+1})$, that is, $\alpha_i := \alpha(v_{i+1}) - \alpha(v_i) - 1$; in addition, denote $\alpha_0 := \alpha(v_1) - 1$ and $\alpha_k := m - \alpha(v_k)$. Observe that $\alpha_0 + \cdots + \alpha_k = m - k$. Furthemore, if $\beta$ is another injection from $S'$ to $[m']$, write $\beta \sim \alpha$ if $\beta$ and $\alpha$ specify the same linear order on $S \cap S'$, that is, $\beta(u) < \beta(v)$ if and only if $\alpha(u) < \alpha(v)$ for all $u, v \in S \cap S'$.

▶ **Lemma 10** (Join). *If $x$ is a join node with children $y$ and $z$, then*

$$\ell_x(\alpha) = \sum_{\beta}\sum_{\gamma}[\alpha \sim \beta \sim \gamma]\prod_{i=0}^{|B_x|}[\alpha_i = \beta_i + \gamma_i]\binom{\alpha_i}{\beta_i}\ell_y(\beta)\,\ell_z(\gamma)\,,$$

*where $\beta$ and $\gamma$ run over all injections in $[n_y]^{B_y}$ and $[n_z]^{B_z}$, respectively.*

**Proof.** By definition,

$$\ell_x(\alpha) = \sum_{\pi}\prod_{uv\in E_x}[\pi(u) < \pi(v)]\,,$$

where $\pi$ runs over all bijections from $V_x$ to $[n_x]$ that extend $\alpha$. Observe that by the tree decomposition properties, the sets $V_y \setminus B_x$ and $V_z \setminus B_x$ are disjoint and their union is $V_x \setminus B_x$. Thus we may represent any bijection $\pi : V_x \to [n_x]$ that extends $\alpha$ uniquely by a pair of injections $\beta' : V_y \to [n_x]$ and $\gamma' : V_z \to [n_x]$ whose restrictions to $B_x$ are equal to $\alpha$ and whose images $\beta'(V_y)$ and $\gamma'(V_z)$ cover $[n_x]$.

$$\ell_x(\alpha) = \sum_{\beta'}\sum_{\gamma'}\left[\beta'(V_y)\cup\gamma'(V_z) = [n_x]\right]\prod_{uv\in E_y}\left[\beta'(u) < \beta'(v)\right]\prod_{uv\in E_z}\left[\gamma'(u) < \gamma'(v)\right],$$

where $\beta'$ and $\gamma'$ run over all injections that extend $\alpha$ in $[n_x]^{V_y}$ and $[n_x]^{V_z}$, respectively.

Consider then a mapping that "compresses" any such injection $\beta'$ into a bijection $\beta'' : V_y \to [n_y]$ by letting $\beta''(v) := |\{u \in V_y : \beta'(u) \le \beta'(v)\}|$; let $\gamma''$ denote the bijection obtained similarly from an injection $\gamma'$. Let $\beta$ denote the restriction of $\beta''$ to $B_x$ and $\gamma$ the restriction of $\gamma''$ to $B_x$. We have that $\beta'$ and $\gamma'$ extend $\alpha$ if and only if $\beta \sim \alpha$ and $\gamma \sim \alpha$. Thus we get that

$$\ell_x(\alpha) = \sum_{\beta'':\beta\sim\alpha}\sum_{\gamma'':\gamma\sim\alpha}\prod_{i=0}^{|B_x|}[\alpha_i = \beta_i+\gamma_i]\binom{\alpha_i}{\beta_i}\prod_{uv\in E_y}\left[\beta''(u) < \beta''(v)\right]\prod_{uv\in E_z}\left[\gamma''(u) < \gamma''(v)\right],$$

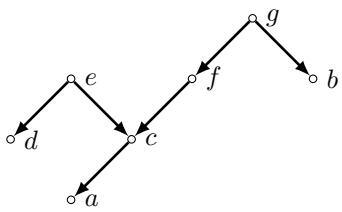
where $\beta''$ and $\gamma''$ run over all bijections in $[n_y]^{B_y}$ and $[n_z]^{B_z}$, respectively. The product of the binomial coefficients $\binom{\alpha_i}{\beta_i}$ is the number of pairs $(\beta', \gamma')$ that map to the same pair $(\beta'', \gamma'')$, that is, the number of interleavings of the $\beta_i + \gamma_i$ elements to the range of $\alpha_i$ elements.

To complete the proof, it suffices to write the summation over $\beta''$ as a double-summation: the outer summation being over all injections $\beta : B_y \to [n_y]$ and the inner summation being over all bijections $\beta'' : V_y \to [n_y]$ that extend $\beta$; similarly for the summation over $\gamma''$. ◀

▶ **Example 11** (Join in a tree). Consider the example illustrated in Figure 1. The cover graph is a tree with vertex set $V = \{a, b, c, d, e, f, g\}$. In a nice tree decomposition (not shown) the root node $x$ is a join of nodes $y$ and $z$, with $B_y = B_z = B_x = \{c, f\}$. The vertex sets associated with the nodes are $V_x = V$, $V_y = \{a, c, d, e, f\}$, and $V_z = \{b, c, f, g\}$. For the shown injection $\alpha$, the value of $\ell_x(\alpha)$ is obtained by a sum of $\ell_y(\beta) \cdot \ell_z(\gamma)$ over valid pairs $(\beta, \gamma)$, multiplied by the number of possible interleavings, which is given by a product of binomial coefficient. Shown is one pair $(\beta, \gamma)$, for which $\ell_y(\beta) = 1$ and $\ell_z(\gamma) = 1$ and the number of interleavings is equal to $\binom{2}{1}\binom{2}{2}\binom{1}{0} = 4$.

It remains to bound the running time of the algorithm.

▶ **Lemma 12** (Time complexity). *Given a width-$t$ nice tree decomposition of the cover graph of an $n$-element poset, the linear extensions can be counted in $\tilde{O}(n^{t+3})$ bit-operations.*




| Injection | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $\alpha$ | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ | $g$ |
| $\beta$ | $a$ | $c$ | $d$ | $e$ | $f$ | | |
| $\gamma$ | $b$ | $c$ | $f$ | $g$ | | | |

**Figure 1** An illustration of the recurrence for a join node; see Example 11 for a description.

**Proof.** Let $x$ be a node in the nice tree decomposition. For brevity, denote $k := |B_x|$. If $x$ is a leaf node, introduce node, or forget node, then the values $\ell_x(\alpha)$ for all injections $\alpha \in [n_x]^{B_x}$ can clearly be computed using $O(kn^k) = O(tn^{t+1})$ basic operations, some of which are additions of two $O(n \log n)$-bit numbers, thus using $\tilde{O}(n^{t+2})$ bit-operations (observe that the factor $t$ is $O(\log n^{t+2})$ and can thus be omitted).

Consider then the remaining case: $x$ is a join node. Let $y$ and $z$ be the two children of $x$. Recall that $B_x = B_y = B_z$.

Represent an injection $\alpha$ in $[n_x]^{B_x}$ as a pair $(\sigma, a)$, where $a = (a_i)_{i=1}^{k+1}$ with $a_i = \alpha_{i-1}$ and $\sigma$ is a bijection from $B_x$ to $[k]$ that captures the specified linear order, that is, $\sigma(u) < \sigma(v)$ if $\alpha(u) < \alpha(v)$. Clearly, the mapping $\alpha \mapsto (\sigma, a)$ is a bijection when we require that $a_i \in \mathbb{N}$ and $|a| = n_x - k$. Using this representation and Lemma 10, write

$$\ell_x(\sigma, a) = \sum_b \sum_c \prod_{i=0}^{k} [a_i = b_i + c_i] \binom{a_i}{b_i} \ell_y(\sigma, b)\, \ell_z(\sigma, c)\,,$$

where $b$ and $c$ run over $\mathbb{N}^{k+1}$. By writing $\ell'_x(\sigma, a) := \ell_x(\sigma, a)/a!$, we get the convolution form

$$\ell'_x(\sigma, a) = \sum_{a=b+c} \ell'_y(\sigma, b)\, \ell'_z(\sigma, c)\,.$$

To treat this as a multiplication of multivariate polynomials, consider a fixed bijection $\sigma$ and let $P_x(r_1, \ldots, r_k)$ be the $k$-variate polynomial where the coefficient of $r_1^{a_1} \cdots r_k^{a_k}$ equals $n! \cdot \ell'_x(\sigma, a)$; we define $P_y$ and $P_z$ similarly. Note that $k$ variables suffice, since $a_{k+1}$ is determined by the fixed $|a|$. Here we multiplied by the factorial $n!$ to get integer coefficients (by Fact 2, since $n \geq |a|, |b|, |c|$). We have that $n! \cdot P_x = P_y P_z$, that is, we obtain $P_x$ by multiplying $P_y$ and $P_z$ and dividing each coefficient of the resulting polynomial by $n!$.

We bound the bit-complexity of the polynomial multiplication using Fact 3. The total degrees of the polynomials are at most $n - k$. Each coefficient of the polynomials is a $\tilde{O}(n)$-bit integer. Thus the multiplication takes $\tilde{O}(\binom{n}{k} n)$ bit-operations.

Multiplying the obtained bound by the number of bijections $\sigma$, we get that all $\ell_x(\sigma, a)$ can be computed using $\tilde{O}(n^{k+1}) = \tilde{O}(n^{t+2})$ bit-operations.

Because there are $O(tn)$ nodes in the nice tree decomposition, $\tilde{O}(n^{t+3})$ bit-operations suffice in total. ◀

## 4 Experiments

This section is devoted to empirical results. We first describe our implementation of the algorithm and the test instances used in the experiments. Then we show how the performance on the algorithm depends on the way we choose the tree decomposition.

## 4.1 Implementation

We have implemented the algorithm described in Section 3 in a `C++` program `Countle`.[2] For multiplication of polynomials we used a `C` library `FLINT` [16]. For finding an optimal tree decomposition we used the `C++` library `htd` [1]. We ran all experiments on machines with Intel Xeon E5540 CPUs; the same machines were used by Kangas et al. [17], which makes their running time measurements directly comparable to the present results.

Two implementation details are worth mentioning. First, for multiplication of multivariate polynomials, we transformed the polynomials into univariate polynomials using an appropriate Kronecker substitution. Specifically, separately for each node $x$ of the tree decomposition and the considered vertex ordering (bijection) $\sigma$, we encode a $k$-variate polynomial in variables $r_1, \ldots, r_k$ as a univariate polynomial in variable $s$ by substituting $r_j := s^{(d_1+1)\cdots(d_{j-1}+1)}$, where each $d_i$ is an upper bound for the degree of $r_i$ in the polynomial. Using knowledge associated with the node $x$ and ordering $\sigma$, we aim at finding a value $d_i$ that is smaller than the trivial upper bound $n_x - k$. To this end, we set $d_i$ to the sum of the largest realized exponents of $r_i$ in the already computed polynomials for the two child nodes of $x$.

Second, we wish to ignore any impossible ordering $\sigma$ at a node $x$ of the tree decomposition, and so save both time and space. The key observation is that, even if the value $\ell_x(\sigma, a)$ is nonzero, we can ignore it if $\sigma$ assigns some two vertices in the bag $B_x$ an order that violates the partial order $\prec$, that is, for some $u, v \in B_x$ we have $u \prec v$ and $\sigma(u) > \sigma(v)$.

## 4.2 Instances

We generated random instances (posets) of different sizes for small values of treewidth $t$. We varied the number of elements $n$ from 10 to 199 ($t = 2$), 109 ($t = 3$), and 59 ($t = 4$). For each pair $(t, n)$ we generated 5 posets; following Kangas et al. [17], we let each poset be a "grid tree," constructed by randomly joining $t$-by-$t$ grids along the (boundary) edges, orienting the edges so that no directed cycles are introduced, and finally taking the transitive closure.

## 4.3 Results

Because the running time of `Countle` may be sensitive to the particular (nice) tree decomposition selected, we ran the program on 50 optimal-width tree decompositions, which we generated using `htd`; we checked that for every encountered instance, `htd` indeed generated tree decompositions of optimal width. We allowed each individual run to take up to 20 minutes of CPU time and 30 GB of memory.

First we considered the scaling of `Countle` in terms of the number of elements $n$ and treewidth $t$. We observed that, while the growth of the running time follows the rate suggested by the worst case bound, there is significant variance in the running times for any fixed $(n, t)$, due to differences in the five posets and the 50 tree decompositions per poset (compare *median* to *best* in Figure 2). Compared to an implementation of the inclusion–exclusion algorithm by Kangas et al. [17], `VEIE`, we find that `Countle` is an order of magnitude faster. For example, `Countle` can solve a typical (median) poset with a typical (median) tree decomposition in about 20 seconds if $n = 100$ and $t = 2$, or if $n = 50$ and $t = 3$, while `VEIE` requires about 200 seconds on such instances [17, Fig. 8] (not reproduced here). We have observed that the same pattern also holds for $t = 4$; we omit the detailed results here, as such posets are handled faster by another, exponential time algorithm of Kangas et al. [17, Fig. 8].

[2] `Countle` is free and publicly available at `https://bitbucket.org/samsalo/countledist/`.

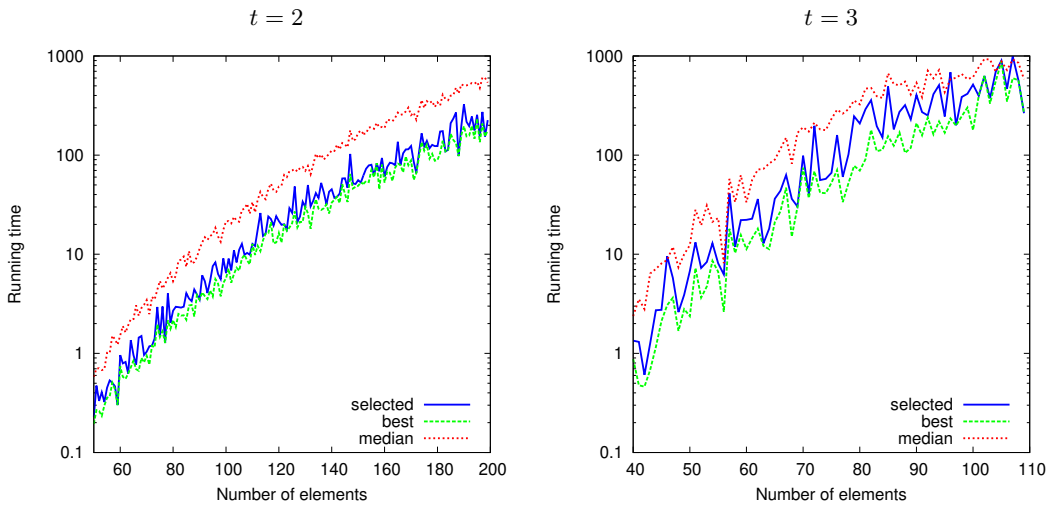



**Figure 2** The running time of `Countle` (in seconds) on random "grid tree" posets of treewidth 2 and 3, with a varying number of elements $n$. For each poset, we generated 50 optimal-width tree decompositions and collected the median running time and the running times for the selected and the best tree decomposition. Shown are the medians of these three statistics over five independent posets for each value of $n$.

Then we investigated whether one can efficiently select a near-optimal tree decomposition from a collection of generated candidates. The observed variance in running times suggests that, if successful, this could lead to a significant expedition of `Countle`, by up to one order of magnitude. To constuct a "selector" we applied the method of Abseher et al. [2] in a straightforward manner:

**C1** We collected a data set of measured running times for multiple pairs of posets and tree decompositions. We used the procedure described in the previous section, except that we used a single poset (instead of five) for each combination of $n$ and $t$. If a run was not completed within the 20-minute time limit, we simply discarded the instance (and thus introduced some bias).

**C2** We computed for each tree decomposition the values of several features, such as statistics of bag sizes (by node type), node depths (by node type), and distances between join nodes; for a full feature list, see Abseher et al. [2].

**C3** We fitted a multivariate linear regression model, separately for each $t = 2, 3, 4$, with the features as the predictor variables and the logarithm of the running time as the response variable. We used the machine learning software `WEKA 3.6.13` [15] with default options.

To select a tree decomposition for a given new poset, we first generated 50 candidate tree decompositions for the poset, and then selected the one for which the model predicted the shortest running time.

We observed (Figure 2) that, for $t = 2$, the model is almost always able to select a top-3 tree decomposition, which yields a nearly as good (i.e., short) running time as the best among the 50 tree decompositions. For $t = 3$ the performance degrades: the model is usually able to select a top-10 tree decomposition, which yields a running time that is systematically better than for a typical (median) tree decomposition, yet not quite achieving the performance of the best among the generated candidates. For $t = 4$ the performance degrades further, yet being better than by selecting a random tree decomposition (results not shown). In more quantitative terms, the proportions of tree decompositions (among 50) better than the selected one were 2.5 %, 12 %, 28 % for $t = 2, 3, 4$, respectively; these numbers are medians of averages over 5 test posets (one per fixed $n$ and $t$).

## 5 Concluding remarks

We have presented a new tree-decomposition based algorithm for counting linear extensions. The algorithm relies on fast multiplication of multivariate polynomials, thus differing radically from the inclusion–exclusion approach of Kangas et al. [17]. For any constant treewidth $t$ the obtained asymptotic speedup is about linear in the number of elements $n$.

A question not settled here is whether one could save another factor of $n$, that is, solve the problem in time $\tilde{O}(n^{t+2})$. The present authors find this question particularly intriguing for two reasons: One is that for finding an optimal-width tree decomposition, the best known time complexity bound is $\tilde{O}(n^{t+2})$, assuming we let $t$ grow at least logarithmically in $n$. The other reason is that for posets whose cover graph is a tree ($t = 1$), Atkinson's [5] algorithm takes – at least seemingly – a different approach and runs in time $\tilde{O}(n^3)$. Furthermore, Atkinson's algorithm is monotonic in the sense that all arithmetic operations are carried out with nonnegative numbers. This is in sharp contrast to both the present algorithm and the inclusion–exclusion algorithm, which crucially rely on a richer algebraic structure.

Our preliminary empirical study confirmed that the improvement in the asymptotic bound consistently transfers to the running times measured in practice. That said, the observed speedup, for $n$ around one hundred, was by one order of magnitude rather than two. This "leak" of efficiency can, at least in part, be explained by the present algorithm's higher sensitivity to the shape of the selected tree decomposition. Indeed, we observed that the best of 50 generated tree decompositions typically yields a 5- to 10-fold speedup in relation to a median tree decomposition. We also showed that there is an efficient way to select the best or close-to-best tree decomposition using a linear regression model that was fitted to a collected data set of instances along with the measured running times, following the method of Abseher et al. [2].

However, we observed that the performance of the regression method rapidly degraded as the treewidth $t$ increases. This suggests that the general-purpose method may not suit well for the problem of counting linear extensions. A potential reason for suboptimal performance is that the default set of features [2] does not include perhaps the most informative quantity associated with a node $x$ in a tree decomposition, namely the term $n_x^k$ (or some variant of it), which combines the size $k$ of the bag of $x$ with the number of vertices in the subtree rooted at $x$. This issue could be addressed by extending the feature set accordingly, or, potentially, by using some nonlinear regression model. On the other hand, we did inspect how well a single feature can predict a well-performing tree decomposition. We observed that for $t = 2$ and $t = 3$ the average depth of join nodes alone yielded predictions that were almost as good as the predictions by the full regression model that used all the features. Nevertheless, these observations also encourage looking for a simpler solution: manually constructing a heuristic function that approximates the actual running time by adding up estimated contributions of each tree decomposition node. Note that here the *f-cost* framework [9] is insufficient, as in that framework the contribution of each node can only depend on the size of the bag.

## References

1 Michael Abseher, Nysret Musliu, and Stefan Woltran. htd – A Free, Open-Source Framework for (Customized) Tree Decompositions and Beyond. In *Proceedings of the 14th International Conference on Integration of AI and OR Techniques in Constraint Programming*, volume 10335 of *Lecture Notes in Computer Science*, pages 376–386. Springer, 2017.

2 Michael Abseher, Nysret Musliu, and Stefan Woltran. Improving the Efficiency of Dynamic Programming on Tree Decompositions via Machine Learning. *J. Artif. Intell. Res.*, 58:829–858, 2017.

3 Alfred V. Aho, M. R. Garey, and Jeffrey D. Ullman. The Transitive Reduction of a Directed Graph. *SIAM J. Comput.*, 1(2):131–137, 1972.

4 Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of Finding Embeddings in a K-tree. *SIAM J. Algebra. Discr.*, 8(2):277–284, April 1987.

5 Mike D. Atkinson. On computing the number of linear extensions of a tree. *Order*, 7(1):23–25, 1990.

6 Jacqueline Banks, Scott Garrabrant, Mark L. Huber, and Anne Perizzolo. Using TPA to count linear extensions. *arXiv preprint arXiv:1010.4981*, 2017.

7 Umberto Bertelè and Francesco Brioschi. *Nonserial Dynamic Programming.* Academic Press, 1972.

8 Hans L. Bodlaender. A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

9 Hans L. Bodlaender and Fedor V. Fomin. Tree decompositions with small cost. *Discrete Appl. Math.*, 145(2):143–154, 2005.

10 Hans L. Bodlaender and Ton Kloks. Efficient and Constructive Algorithms for the Pathwidth and Treewidth of Graphs. *J. Algorithms*, 21(2):358–402, 1996.

11 Graham Brightwell and Peter Winkler. Counting linear extensions. *Order*, 8(3):225–242, 1991.

12 Rina Dechter. Bucket elimination: A unifying framework for reasoning. *Artif. Intell.*, 113(1–2):41–85, 1999.

13 Martin Dyer, Alan Frieze, and Ravi Kannan. A Random Polynomial-time Algorithm for Approximating the Volume of Convex Bodies. *J. ACM*, 38(1):1–17, 1991.

14 Eduard Eiben, Robert Ganian, Kustaa Kangas, and Sebastian Ordyniak. Counting Linear Extensions: Parameterizations by Treewidth. In *Proceedings of the 24th Annual European Symposium on Algorithms*, volume 57 of *LIPIcs*, pages 39:1–39:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.

15 Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

16 William B. Hart. Fast Library for Number Theory: An Introduction. In *Proceedings of the Third International Congress on Mathematical Software*, pages 88–91, Berlin, Heidelberg, 2010. Springer-Verlag. URL: `http://flintlib.org`.

17 Kustaa Kangas, Teemu Hankala, Teppo Niinimäki, and Mikko Koivisto. Counting linear extensions of sparse posets. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 603–609. IJCAI/AAAI Press, 2016.

18 Richard M. Karp. Dynamic Programming Meets the Principle of Inclusion and Exclusion. *Oper. Res. Lett.*, 1(2):49–51, April 1982.

19 Mikko Koivisto. An $O^*(2^n)$ Algorithm for Graph Coloring and Other Partitioning Problems via Inclusion–Exclusion. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 583–590. IEEE Computer Society, 2006.

20 Kevin Leyton-Brown, Eugene Nudelman, and Yoav Shoham. Empirical hardness models: Methodology and a case study on combinatorial auctions. *J. ACM*, 56(4), 2009.

21 Thomas Lukasiewicz, Maria V. Martinez, and Gerardo I. Simari. Probabilistic Preference Logic Networks. In *Proceedings of the 21st European Conference on Artificial Intelligence*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 561–566. IOS Press, 2014.

**22** Heikki Mannila and Christopher Meek. Global Partial Orders from Sequential Data. In *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining*, pages 161–168. ACM, 2000.

**23** Rolf H. Möhring. Computationally tractable classes of ordered sets. In I. Rival, editor, *Algorithms and Order*, pages 105–193. Kluwer Academic Publishers, 1989.

**24** Jason Morton, Lior Pachter, Anne Shiu, Bernd Sturmfels, and Oliver Wienand. Convex Rank Tests and Semigraphoids. *SIAM J. Discrete Math.*, 23(3):1117–1134, 2009.

**25** Christian J. Muise, J. Christopher Beck, and Sheila A. McIlraith. Optimal Partial-Order Plan Relaxation via MaxSAT. *J. Artif. Intell. Res.*, 57:113–149, 2016.

**26** Teppo Niinimäki, Pekka Parviainen, and Mikko Koivisto. Structure Discovery in Bayesian Networks by Sampling Partial Orders. *J. Mach. Learn. Res.*, 17:57:1–57:47, 2016.

**27** Marcin Peczarski. New Results in Minimum-Comparison Sorting. *Algorithmica*, 40(2):133–145, 2004.

**28** John R. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

**29** Herbert J. Ryser. *Combinatorial Mathematics*, volume 14 of *Carus Mathematical Monographs*. The Mathematical Association of America, 1963.

**30** Topi Talvitie, Kustaa Kangas, Teppo Niinimäki, and Mikko Koivisto. Counting Linear Extensions in Practice: MCMC versus Exponential Monte Carlo. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.

**31** Joris van der Hoeven and Grégoire Lecerf. On the bit-complexity of sparse polynomial and series multiplication. *J. Symb. Comput.*, 50:227–254, 2013.

**32** Chris S. Wallace, Kevin B. Korb, and Honghua Dai. Causal Discovery via MML. In *Proceedings of the 13th International Conference on Machine Learning*, pages 516–524. Morgan Kaufmann, 1996.

# Generalized Distance Domination Problems and Their Complexity on Graphs of Bounded mim-width

**Lars Jaffke**[1]
Department of Informatics, University of Bergen, Norway
lars.jaffke@uib.no
https://orcid.org/0000-0003-4856-5863

**O-joung Kwon**
Department of Mathematics, Incheon National University, Incheon, South Korea
ojoungkwon@gmail.com
https://orcid.org/0000-0003-1820-1962

**Torstein J. F. Strømme**
Department of Informatics, University of Bergen, Norway
torstein.stromme@uib.no
https://orcid.org/0000-0002-3896-3166

**Jan Arne Telle**
Department of Informatics, University of Bergen, Norway
jan.arne.telle@uib.no

## Abstract

We generalize the family of $(\sigma, \rho)$-problems and locally checkable vertex partition problems to their distance versions, which naturally captures well-known problems such as distance-$r$ dominating set and distance-$r$ independent set. We show that these distance problems are XP parameterized by the structural parameter *mim-width*, and hence polynomial on graph classes where mim-width is bounded and quickly computable, such as $k$-trapezoid graphs, Dilworth $k$-graphs, (circular) permutation graphs, interval graphs and their complements, convex graphs and their complements, $k$-polygon graphs, circular arc graphs, complements of $d$-degenerate graphs, and $H$-graphs if given an $H$-representation. To supplement these findings, we show that many classes of (distance) $(\sigma, \rho)$-problems are W[1]-hard parameterized by mim-width + solution size.

**2012 ACM Subject Classification** Mathematics of computing → Graph algorithms, Theory of computation → Parameterized complexity and exact algorithms

**Keywords and phrases** Graph Width Parameters, Graph Classes, Distance Domination Problems, Parameterized Complexity

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.6

**Related Version** A full version is available at [12], `https://arxiv.org/abs/1803.03514`.

## 1 Introduction

Telle and Proskurowski [19] defined the $(\sigma, \rho)$-domination problems, and the more general *locally checkable vertex partitioning* problems (LCVP). In $(\sigma, \rho)$-domination problems, feasible

[1] Supported by the Bergen Research Foundation (BFS).

© Lars Jaffke, O-joung Kwon, Torstein J. F. Strømme, and Jan Arne Telle;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 6; pp. 6:1–6:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

solutions are vertex sets with constraints on how many neighbours each vertex of the graph has in the set. The framework generalizes important and well-studied problems such as Maximum Independent Set and Minimum Dominating Set, as well as Perfect Code, Minimum subgraph with minimum degree $d$ and a multitude of other problems. See Table 1. Bui-Xuan, Telle and Vatshelle [6] showed that $(\sigma, \rho)$-domination and locally checkable vertex partitioning problems can be solved in time XP parameterized by *mim-width*, if we are given a corresponding decomposition tree. Roughly speaking, the structural parameter mim-width measures how easy it is to decompose a graph along vertex cuts inducing a bipartite graph with small maximum induced matching size [20].

In this paper, we consider distance versions of problems related to independence and domination, like Distance-$r$ Independent Set and Distance-$r$ Dominating Set. The Distance-$r$ Independent Set problem, also studied under the names $r$-Scattered Set and $r$-Dispersion (see e.g. [2] and the references therein), asks to find a set of at least $k$ vertices whose vertices have pairwise distance strictly longer than $r$. Agnarsson et al. [1] pointed out that it is identical to the original Independent Set problem on the $r$-th power graph $G^r$ of the input graph $G$, and also showed that for fixed $r$, it can be solved in linear time for interval graphs, and circular arc graphs. The Distance-$r$ Dominating Set problem was introduced by Slater [18] and Henning et al. [11]. They also discussed that it is identical to solve the original Dominating Set problem on the $r$-th power graph. Slater presented a linear-time algorithm to solve Distance-$r$ Dominating Set problem on forests.

We generalize all of the $(\sigma, \rho)$-domination and LCVP problems to their distance versions, which naturally captures Distance-$r$ Independent Set and Distance-$r$ Dominating Set. Where the original problems put constraints on the size of the immediate neighborhood of a vertex, we consider the constraints to be applied to the ball of radius $r$ around it. Consider for instance the Minimum Subgraph with Minimum Degree $d$ problem; where the original problem is asking for the smallest (number of vertices) subgraph of minimum degree $d$, we are instead looking for the smallest subgraph such that for each vertex there are at least $d$ vertices at distance at least 1 and at most $r$. In the Perfect Code problem, the target is to choose a subset of vertices such that each vertex has exactly one chosen vertex in its closed neighbourhood. In the distance-$r$ version of the problem, we replace the closed neighbourhood by the closed $r$-neighbourhood. This problem is known as Perfect $r$-Code, and was introduced by Biggs [4] in 1973. Similarily, for every problem in Table 1 its distance-$r$ generalization either introduces a new problem or is already well-known.

We show that all these distance problems are XP parameterized by mim-width if a decomposition tree is given. The main result of the paper is of structural nature, namely that for any positive integer $r$ the mim-width of a graph power $G^r$ is at most twice the mim-width of $G$. It follows that we can reduce the distance-$r$ version of a $(\sigma, \rho)$-domination problem to its non-distance variant by taking the graph power $G^r$, whilst preserving small mim-width.

The downside to showing results using the parameter mim-width, is that we do not know an XP algorithm computing mim-width. Computing a decomposition tree with optimal mim-width is NP-complete in general and W[1]-hard parameterized by itself. Determining the optimal mim-width is not in APX unless NP = ZPP, making it unlikely to have a polynomial-time constant-factor approximation algorithm [17], but saying nothing about an XP algorithm. However, for several graph classes we are able to find a decomposition tree of constant mim-width in polynomial time, using the results of Belmonte and Vatshelle [3]. These include; permutation graphs, convex graphs and their complements, interval graphs and their complements (all of which have *linear* mim-width 1); (circular $k$-) trapezoid graphs, circular permutation graphs, Dilworth-$k$ graphs, $k$-polygon graphs, circular arc graphs and

complements of $d$-degenerate graphs. Fomin, Golovach and Raymond [10] show that we can find linear decomposition trees of constant mim-width for the very general class of $H$-graphs, see Definition 10, in polynomial time, *given*[2] an $H$-representation of the input graph. For all of the above graph classes, our results imply that the distance-$r$ $(\sigma, \rho)$-domination and LCVP problems become polynomial time solvable.

Graphs represented by intersections of objects in some model are often closed under taking powers. For instance, interval graphs, and generally $d$-trapezoid graphs [9, 1], circular arc graphs [16, 1], and leaf power graphs (by definition) are such graphs. We refer to [5, Chapter 10.6] for a survey of such results. For these classes, we already know that the distance-$r$ version of a $(\sigma, \rho)$-domination problem can be solved in polynomial time. However, this closure property does not always hold; for instance, permutation graphs are not closed under taking powers. Our result provides that to obtain such algorithmic results, we do not need to know that these classes are closed under taking powers; it is sufficient to know that classes have bounded mim-width. To the best of our knowledge, for the most well-studied distance-$r$ $(\sigma, \rho)$-domination problem, Distance-$r$ Dominating Set, we obtain the first polynomial time algorithms on Dilworth $k$-graphs, convex graphs and their complements, complements of interval graphs, $k$-polygon graphs, $H$-graphs (given an $H$-representation of the input graph), and complements of $d$-degenerate graphs.

The natural question to ask after obtaining an XP algorithm, is whether we can do better, e. g. can we show that for all fixed $r$, the distance-$r$ $(\sigma, \rho)$-domination problems are in FPT? Fomin et al. [10] answered this in the negative by showing that (the standard, i.e. distance-1 variants of) Maximum Independent Set, Minimum Dominating Set and Minimum Independent Dominating Set problems are W[1]-hard parameterized by (linear) mim-width + solution size. We modify their reductions to extend these results to several families of $(\sigma, \rho)$-domination problems, including the maximization variants of Induced Matching, Induced $d$-Regular Subgraph and Induced Subgraph of Max Degree $\leq d$, the minimization variants of Total Dominating Set and $d$-Dominating Set and both the maximization and the minimization variant of Dominating Induced Matching.

The remainder of the paper is organized as follows. In Section 2 we introduce the $(\sigma, \rho)$ problems and define their distance-$r$ generalization. In Section 3 we introduce mim-width, and state previously known results. In Section 4 we show that the mim-width of a graph grows by at most a factor 2 when taking (arbitrary large) powers and give algorithmic consequences. We discuss LCVP problems, their distance-$r$ versions and algorithmic consequences regarding them in Section 5 and in Section 6 we present the above mentioned lower bounds. Finally, we give some concluding remarks in Section 7. Proofs of statements marked with '★' are deferred to the full version [12].

## 2 Distance-$r$ $(\sigma, \rho)$-Domination Problems

Let $\sigma$ and $\rho$ be finite or co-finite subsets of the natural numbers $\sigma, \rho \subseteq \mathbb{N}$. Furthermore, for a graph $G$, one of its vertices $v \in V(G)$, and a positive integer $r$, let $N^r(v)$ denote the *ball of radius $r$ around $v$*, i. e. $N^r(v) := \{w \in V(G) \setminus \{v\} \mid \text{DIST}_G(v, w) \leq r\}$. A vertex set $S \subseteq V(G)$ is called a *distance-$r$ $(\sigma, \rho)$-dominating set*, if

- for each vertex $v \in S$ it holds that $|N^r(v) \cap S| \in \sigma$, and

[2] We would like to remark that it is NP-complete to decide whether a graph is an $H$-graph whenever $H$ is not a cactus [7].

**Table 1** Some vertex subset properties expressible as $(\sigma, \rho)$ sets, with $\mathbb{N} = \{0, 1, ...\}$ and $\mathbb{N}^+ = \{1, 2, ...\}$. Column $d$ shows $d = \max(d(\sigma), d(\rho))$. For each problem, at least one of the minimization, the maximization and the existence problem is NP-complete. For problems marked with $\star$ (resp., $\star\star$), W[1]-hardness of the maximization (resp., minimization) problem parameterized by mim-width + solution size is shown in the present paper. For problems marked with $*$ (resp., $**$) the W[1]-hardness of maximization (resp., minimization) in the same parameterization was shown by Fomin et al. [10].

| $\sigma$ | $\rho$ | $d$ | Standard name |
|---|---|---|---|
| $\{0\}$ | $\mathbb{N}$ | 1 | Independent set $*$ |
| $\mathbb{N}$ | $\mathbb{N}^+$ | 1 | Dominating set $**$ |
| $\{0\}$ | $\mathbb{N}^+$ | 1 | Maximal Independent set $**$ |
| $\mathbb{N}^+$ | $\mathbb{N}^+$ | 1 | Total Dominating set $\star\star$ |
| $\{0\}$ | $\{0, 1\}$ | 2 | Strong Stable set or 2-Packing |
| $\{0\}$ | $\{1\}$ | 2 | Perfect Code or Efficient Dom. set |
| $\{0, 1\}$ | $\{0, 1\}$ | 2 | Total Nearly Perfect set |
| $\{0, 1\}$ | $\{1\}$ | 2 | Weakly Perfect Dominating set |
| $\{1\}$ | $\{1\}$ | 2 | Total Perfect Dominating set |
| $\{1\}$ | $\mathbb{N}$ | 2 | Induced Matching $\star$ |
| $\{1\}$ | $\mathbb{N}^+$ | 2 | Dominating Induced Matching $\star$, $\star\star$ |
| $\mathbb{N}$ | $\{1\}$ | 2 | Perfect Dominating set |
| $\mathbb{N}$ | $\{d, d+1, ...\}$ | $d$ | $d$-Dominating set $\star\star$ |
| $\{d\}$ | $\mathbb{N}$ | $d+1$ | Induced $d$-Regular Subgraph $\star$ |
| $\{d, d+1, ...\}$ | $\mathbb{N}$ | $d$ | Subgraph of Min Degree $\geq d$ |
| $\{0, 1, ..., d\}$ | $\mathbb{N}$ | $d+1$ | Induced Subg. of Max Degree $\leq d$ $\star$ |

- for each vertex $v \in V(G) \setminus S$ it holds that $|N^r(v) \cap S| \in \rho$.

For the special case of $r = 1$, we call a distance-1 $(\sigma, \rho)$-dominating set simply a $(\sigma, \rho)$-dominating set or $(\sigma, \rho)$ set. This recovers many well-studied naturally defined vertex sets of graphs. For instance, a $(\{0\}, \mathbb{N})$ set is an independent set as there are no edges inside of the set, and we do not care about adjacencies between $S$ and $V(G) \setminus S$; and a $(\mathbb{N}, \mathbb{N}^+)$ set is a dominating set since each vertex in $V(G) \setminus S$ has to have at least one neighbor in $S$.

There are three types of distance-$r$ $(\sigma, \rho)$-domination problems: minimization, maximization, and existence. For $r = 1$, we denote the problem of finding a minimum (maximum) $(\sigma, \rho)$ set as the Min-$(\sigma, \rho)$ (Max-$(\sigma, \rho)$) problem, see Table 1 for examples.

The *$d$-value* of a distance-$r$ $(\sigma, \rho)$ problem is a constant which will ultimately affect the runtime of the algorithm. For a set $\mu \subseteq \mathbb{N}$, the value $d(\mu)$ should be understood as the highest value in $\mathbb{N}$ we need to enumerate in order to describe $\mu$. Hence, if $\mu$ is finite, it is simply the maximum value in $\mu$, and if $\mu$ is co-finite, it is the maximum natural number *not* in $\mu$ (1 is added for technical reasons).

▶ **Definition 1** ($d$-value). Let $d(\mathbb{N}) = 0$. For every non-empty finite or co-finite set $\mu \subseteq \mathbb{N}$, let $d(\mu) = 1 + \min(\max\{x \mid x \in \mu\}, \max\{x \mid x \in \mathbb{N} \setminus \mu\})$.

For a given distance-$r$ $(\sigma, \rho)$ problem $\Pi_{\sigma,\rho}$, its $d$-value is defined as $d(\Pi_{\sigma,\rho}) := \max\{d(\sigma), d(\rho)\}$, see column $d$ in Table 1.

## 3 Mim-width and Applications

*Maximum induced matching width*, or *mim-width* for short, was introduced in the Ph. D. thesis of Vatshelle [20], used implicitly by Belmonte and Vatshelle [3], and is a structural

graph parameter described over *decomposition trees* (sometimes called *branch decompositions*), similar to graph parameters such as *rank-width* and *module-width*. Decomposition trees naturally appear in divide and conquer style algorithms where one recursively partitions the pieces of a problem into two parts. When the algorithm is at the point where it combines solutions of its subproblems to form a full solution, the structure of the cuts are (unsurprisingly) important to the runtime; this is especially true of dynamic programming when one needs to store multiple sub-solutions at each intermediate node. We will briefly introduce the necessary machinery here, but for a more comprehensive introduction we refer the reader to [20].

A graph of maximum degree at most 3 is called *subcubic*. A *decomposition tree* for a graph $G$ is a pair $(T, \delta)$ where $T$ is a subcubic tree and $\delta : V(G) \to L(T)$ is a bijection between the vertices of $G$ and the leaves of $T$. Each edge $e \in E(T)$ naturally splits the leaves of the tree in two groups depending on their connected component when $e$ is removed. In this way, each edge $e \in E(T)$ also represent a partition of $V(G)$ into two partition classes $A_e$ and $\overline{A}_e$. One way to measure the cut structure is by the *maximum induced matching* across a cut of $(T, \delta)$. A set of edges $M$ is called an *induced matching* if no pair of edges in $M$ shares an endpoint and if the subgraph induced by the endpoints of $M$ does not contain any additional edges.

▶ **Definition 2** (mim-width)**.** Let $G$ be a graph, and let $(T, \delta)$ be a decomposition tree for $G$. For each edge $e \in E(T)$ and corresponding partition of the vertices $A_e, \overline{A}_e$, we let $cutmim_G(A_e, \overline{A}_e)$ denote the size of a maximum induced matching of the bipartite graph on the edges crossing the cut. Let the *mim-width of the decomposition tree* be

$$mimw_G(T, \delta) = \max_{e \in E(T)} \{cutmim(A_e, \overline{A}_e)\}$$

The *mim-width of the graph* $G$, denoted $mimw(G)$, is the minimum value of $mimw_G(T, \delta)$ over all possible decompositions trees $(T, \delta)$. The *linear mim-width of the graph* $G$ is the minimum value of $mimw_G(T, \delta)$ over all possible decompositions trees $(T, \delta)$ where $T$ is a caterpillar.

In previous work, Bui-Xuan et al. [6] and Belmonte and Vatshelle [3] showed that all $(\sigma, \rho)$ problems can be solved in time $n^{\mathcal{O}(w)}$ where $w$ denotes the mim-width of a decomposition tree that is provided as part of the input. More precisely, they show the following.[3]

▶ **Proposition 3** ([3, 6])**.** *There is an algorithm that given a graph $G$ and a decomposition tree $(T, \delta)$ of $G$ with $w := mimw_G(T, \delta)$ solves each $(\sigma, \rho)$ problem $\Pi$ with $d := d(\Pi)$*

**(i)** *in time $\mathcal{O}(n^{4+2d\cdot w})$, if $T$ is a caterpillar, and*

**(ii)** *in time $\mathcal{O}(n^{4+3d\cdot w})$, otherwise.*

## 4 Mim-width on Graph Powers

▶ **Definition 4** (Graph power)**.** Let $G = (V, E)$ be a graph. Then the $k$-th power of $G$, denoted $G^k$, is a graph on the same vertex set where there is an edge between two vertices if and only if the distance between them is at most $k$ in $G$. Formally, $V(G^k) = V(G)$ and $E(G^k) = \{uv \mid \text{DIST}_G(u, v) \leq k\}$.

[3] We would like to remark that the original results in [6] are stated in terms of the number of $d$-neighborhood equivalence classes across the cuts in the decomposition tree ($nec_d(T, \delta)$) giving a runtime of $n^4 \cdot nec_d(T, \delta)^c$ (where $c = 2$ if the given decomposition is a caterpillar and $c = 3$ otherwise). In [3, Lemma 2], Belmonte and Vatshelle show that $nec_d(T, \delta) \leq n^{d \cdot mimw_G(T,\delta)}$.

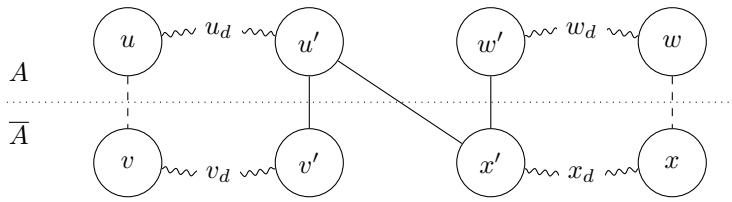



**Figure 1** Structure of two paths $P_{uv}$ and $P_{wx}$ when the edge $u'x'$ exists in $G$. Dashed edges appear in $G^k$, solid edges appear in $G$, squiggly lines are (shortest) paths existing in $G$ (possibly of length 0, and possibly crossing back and forth across the cut).

▶ **Theorem 5.** *For any graph $G$ and positive integer $k$, $mimw(G^k) \leq 2 \cdot mimw(G)$.*

**Proof.** Assume that there is a decomposition tree of mim-width $w$ for the graph $G$. We show that the same decomposition tree has mim-width at most $2w$ for $G^k$.

We consider a cut $A, \overline{A}$ of the decomposition tree. Let $M$ be a maximum induced matching across the cut for $G^k$. To prove our claim, it suffices to construct an induced matching across the cut $M'$ in $G$ such that $|M'| \geq \frac{|M|}{2}$.

We begin by noticing that for an edge $uv \in M$, the distance between $u$ and $v$ is at most $k$ in $G$. For each such edge $uv \in M$, we let $P_{uv}$ denote some shortest path between $u$ and $v$ in $G$ (including the endpoints $u$ and $v$).

▶ **Claim 5.1.** *Let $uv, wx \in M$ be two distinct edges of the matching. Then $P_{uv}$ and $P_{wx}$ are vertex disjoint.*

**Proof.** We may assume that $u, w \in A$ and $v, x \in \overline{A}$. Now assume for the sake of contradiction there exists a vertex $y \in P_{uv} \cap P_{wx}$. Because both paths have length at most $k$, we have that $\textsc{dist}_G(u, y) + \textsc{dist}_G(y, v) \leq k$, and $\textsc{dist}_G(w, y) + \textsc{dist}_G(y, x) \leq k$. Adding these together, we get

$$\textsc{dist}_G(u, y) + \textsc{dist}_G(y, v) + \textsc{dist}_G(w, y) + \textsc{dist}_G(y, x) \leq 2k.$$

Since $uv$ and $wx$ are both in $M$, there can not exist edges $ux$ and $wv$ in $G^k$. Hence, their distance in $G$ is strictly greater than $k$, i.e. $\textsc{dist}_G(u, y) + \textsc{dist}_G(y, x) \geq \textsc{dist}_G(u, x) > k$, and $\textsc{dist}_G(w, y) + \textsc{dist}_G(y, v) > k$. Putting these together, we obtian our contradiction:

$$\textsc{dist}_G(u, y) + \textsc{dist}_G(y, x) + \textsc{dist}_G(w, y) + \textsc{dist}_G(y, v) > 2k$$

This concludes the proof of the claim. ⌟

Our next observation is that for each $uv \in M$, the path $P_{uv}$ starts (without loss of generality) in $A$, and ends in $\overline{A}$. There must hence exist at least one point at which the path cross from $A$ to $\overline{A}$. For each $uv \in M$, we can thus safely let $u'v' \in E(P_{uv})$ denote an edge in $G$ such that $u' \in A$ and $v' \in \overline{A}$.

We plan to construct our matching $M'$ by picking a subset of such edges. However, we can not simply take all of them, since some pairs may be incompatible in the sense that they will not form an induced matching across the cut $A, \overline{A}$. We examine the structures that arise when two such edges $u'v'$ and $w'x'$ are incompatible, and can not both be included in the same induced matching across the cut. For easier readability, we let $\alpha_d$ be a shorthand notation for $\textsc{dist}_G(\alpha, \alpha')$ for $\alpha \in \{u, v, w, x\}$.

▶ **Claim 5.2.** *Let $uv, wx \in M$ be two distinct edges of $M$ and let $u'v'$ and $w'x'$ be edges on the shortest paths as defined above. If there is an edge $u'x' \in E(G)$, then all of the following hold. See Figure 1.*

**(a)** $u_d + x_d = k$

**(b)** $u_d + v_d = w_d + x_d = k - 1$

**(c)** $w_d = u_d - 1$

**Proof.** (a) Since $ux$ is not an edge in $G^k$, the distance between $u$ and $x$ must be at least $k+1$ in $G$, and so $u_d + x_d$ must be at least $k$. It remains to show that $u_d + x_d \leq k$ for equality to hold. Similarily to the proof of Claim 5.1, we know that $P_{uv}$ and $P_{wx}$ both are of length at most $k$. We get

$$u_d + v_d + w_d + x_d \leq 2k - 2 \tag{1}$$

The $-2$ at the end is because we do not include the length contributed by edges $u'v'$ and $w'x'$ in our sum. Now assume for the sake of contradiction that $u_d + x_d \geq k + 1$. Then we get that

$$v_d + w_d \leq 2k - 2 - k - 1 = k - 3$$

Because $\text{DIST}_G(v', w') \leq 3$ (follow the edges $u'v' \to u'x' \to w'x'$), this implies that $\text{DIST}_G(v, w) \leq k$, and the edge $vw$ would hence exist in $G^k$. This contradicts that $uv$ and $wx$ were both in the same induced matching $M$.

(b) Assume for the sake of contradiction that $u_d + v_d \leq k - 2$. Then, rather than Equation 1, we get the following bound

$$u_d + v_d + w_d + x_d \leq 2k - 3$$

By (a) we know that $u_d + x_d = k$, so by a similar argument as above we get that $v_d + x_d \leq k - 3$, obtaining a contradiction. An anolgous argument holds for $w_d + x_d$.

(c) This follows immidiately by substituting (a) into (b). ⌟

We will now construct our induced matching $M'$. We construct two candidates for $M'$, and we will pick the biggest one. First, we construct $M'_0$ by including $u'v'$ for each edge $uv \in M$ where $\text{DIST}_G(u, u')$ is even. Symetrically, $M'_1$ is constructed by including $u'v'$ if $\text{DIST}_G(u, u')$ is odd. Clearly, at least one of $M'_0, M'_1$ contains $\geq \frac{|M|}{2}$ egdes. It remains to show that $M'$ indeed forms an induced matching across the cut $A, \overline{A}$ in $G$.

Consider two distinct edges $u'v'$ and $w'x'$ from $M'$. By Claim 5.1, the two edges are vertex disjoint. If there is an edge violating that $u'v'$ and $w'x'$ are both in the same induced matching, it must be either $u'x'$ or $v'w'$. Without loss of generality we may assume it is an edge of the type $u'x'$. By Claim 5.2 (c), we then have that the parities of $\text{DIST}_G(u, u')$ and $\text{DIST}_G(w, w')$ are different. But by how $M'$ was constructed, this is not possible. This concludes the proof. ◀

▶ **Observation 6.** *For a positive integer $r$, a graph $G$ and a vertex $u \in V(G)$, the $r$-neighbourhood of $u$ is equal to the neighbourhood of $u$ in $G^r$, i.e. $N^r_G(u) = N_{G^r}(u)$.*

The observation above shows that solving a distance-$r$ $(\sigma, \rho)$ problem on $G$ is the same as solving the same standard distance-1 variation of the problem on $G^r$. Hence, we may reduce our problem to the standard version by simply computing the graph power. Combining Theorem 5 with the algorithms provided in Proposition 3, we have the following consequence.

▶ **Corollary 7** (★). *There is an algorithm that for all $r \in \mathbb{N}$, given a graph $G$ and a decomposition tree $(T, \delta)$ of $G$ with $w := mimw_G(T, \delta)$ solves each distance-$r$ $(\sigma, \rho)$ problem $\Pi$ with $d := d(\Pi)$*

**(i)** *in time $\mathcal{O}(n^{4+4d\cdot w})$, if $T$ is a caterpillar, and*

**(ii)** *in time $\mathcal{O}(n^{4+6d\cdot w})$, otherwise.*

## 5 LCVP Problems

A generalization of $(\sigma, \rho)$ problems are the *locally checkable vertex partitioning* (LCVP) problems. A *degree constraint matrix* $D$ is a $q \times q$ matrix where each entry is a finite or co-finite subset of $\mathbb{N}$. For a graph $G$ and a partition of its vertices $\mathcal{V} = \{V_1, V_2, \dots V_q\}$, we say that it is a $D$-partition if and only if, for each $i, j \in [q]$ and each vertex $v \in V_i$, it holds that $|N(v) \cap V_j| \in D[i, j]$. Empty partition classes are allowed.

For instance, if a graph can be partitioned according to the $3 \times 3$ matrix whose diagonal entries are $\{0\}$ and the non-diagonal ones are $\mathbb{N}$, then the graph is 3-colorable. Typically, the natural algorithmic questions associated with LCVP properties are existential.[4] Interesting problems which can be phrased in such terms include the $H$-Covering and Graph $H$-Homomorphism problems where $H$ is fixed, as well as $q$-coloring, Perfect Matching Cut and more. We refer to [19] for an overview.

We generalize LCVP properties to their distance-$r$ version, by considering the ball of radius $r$ around each vertex rather than just the immediate neighbourhood.

▶ **Definition 8** (Distance-$r$ neighbourhood constraint matrix). A distance-$r$ neighbourhood constraint matrix $D$ is a $q \times q$ matrix where each entry is a finite or co-finite subset of $\mathbb{N}$. For a graph $G$ and a partition of its vertices $\mathcal{V} = \{V_1, V_2, \dots V_q\}$, we say that it is a $D$-distance-$r$-partition if and only if, for each $i, j \in [q]$ and each vertex $v \in V_i$, it holds that $|N^r(v) \cap V_j| \in D[i, j]$. Empty partition classes are allowed.

We say that an algorithmic problem is a *distance-$r$* LCVP *problem* if the property in question can be described by a distance-$r$ neighbourhood constraint matrix. For example, the distance-$r$ version of a problem such as $q$-coloring can be interpreted as an assignment of at most $q$ colours to vertices of a graph such that no two vertices are assigned the same colour if they are at distance $r$ or closer.

For a given distance-$r$ LCVP problem $\Pi$, its $d$-value $d(\Pi)$ is the maximum $d$-value over all the sets in the corresponding neighbourhood constraint matrix.

As in the case of $(\sigma, \rho)$ problems, combining Theorem 5 with Observation 6 and the works [3, 6] we have the following result.

▶ **Corollary 9.** *There is an algorithm that for all $r \in \mathbb{N}$, given a graph $G$ and a decomposition tree $(T, \delta)$ of $G$ with $w := mimw_G(T, \delta)$ solves each distance-$r$ LCVP problem $\Pi$ with $d := d(\Pi)$*

**(i)** *in time $\mathcal{O}(n^{4+4qd\cdot w})$, if $T$ is a caterpillar, and*

**(ii)** *in time $\mathcal{O}(n^{4+6qd\cdot w})$, otherwise.*

---

[4] Note however that each $(\sigma, \rho)$ problem can be stated as an LCVP problem via the matrix $D_{(\sigma,\rho)} = \begin{bmatrix} \sigma & \mathbb{N} \\ \rho & \mathbb{N} \end{bmatrix}$, so maximization or minimization of some block of the partition can be natural as well.

## 6 Lower Bounds

We show that several $(\sigma, \rho)$-problems are W[1]-hard parameterized by linear mim-width plus solution size. Our reductions are based on two recent reductions due to Fomin, Golovach and Raymond [10] who showed that Independent Set and Dominating Set are W[1]-hard parameterized by linear mim-width plus solution size. In fact they show hardness for the above mentioned problems on *$H$-graphs* (the parameter being the number of edges in $H$ plus solution size) which we now define formally.

▶ **Definition 10** ($H$-Graph)**.** Let $X$ be a set and $\mathcal{S}$ a family of subsets of $X$. The *intersection graph* of $\mathcal{S}$ is a graph with vertex set $\mathcal{S}$ such that $S, T \in \mathcal{S}$ are adjacent if and only if $S \cap T \neq \emptyset$. Let $H$ be a (multi-) graph. We say that $G$ is an *$H$-graph* if there is a subdivision $H'$ of $H$ and a family of subsets $\mathcal{M} := \{M_v\}_{v \in V(G)}$ (called an *$H$-representation*) of $V(H')$ where $H'[M_v]$ is connected for all $v \in V(G)$, such that $G$ is isomorphic to the intersection graph of $\mathcal{M}$.

All of the hardness results presented in this section are obtained via reductions to the respective problems on $H$-graphs, and the hardness for linear mim-width follows from the following proposition.

▶ **Proposition 11** (Theorem 2 in [10])**.** *Let $G$ be an $H$-graph. Then, $G$ has linear mim-width at most $2 \cdot ||H||$ and a corresponding decomposition tree can be computed in polynomial time given an $H$-representation of $G$.*

The first lower bound concerns several maximization problems that can be expressed in the $(\sigma, \rho)$ framework. Recall that the Independent Set problem can be formulated as Max-$(\{0\}, \mathbb{N})$. The following result states that a class of problems that generalize the Independent Set problem where each vertex in the solution is allowed to have at most some fixed number of $d$ neighbors of the solution, and several variants thereof, is W[1]-hard on $H$-graphs parameterized by $||H||$ plus solution size.

▶ **Theorem 12.** *For any fixed $d \in \mathbb{N}$ and $x \leq d+1$, the following holds. Let $\sigma^* \subseteq \mathbb{N}_{\leq d}$ with $d \in \sigma^*$. Then,* Max-$(\sigma^*, \mathbb{N}_{\geq x})$ Domination *is* W[1]*-hard on $H$-graphs parameterized by the number of edges in $H$ plus solution size, and the hardness holds even if an $H$-representation of the input graph is given.*

**Proof.** To prove the theorem, we provide a reduction from Multicolored Clique where given a graph $G$ and a partition $V_1, \ldots, V_k$ of $V(G)$, the question is whether $G$ contains a clique of size $k$ using precisely one vertex from each $V_i$ ($i \in [k]$). This problem is known to be W[1]-complete [8, 14].

Let $(G, V_1, \ldots, V_k)$ be an instance of Multicolored Clique. We can assume that $k \geq 2$ and that $|V_i| = p$ for $i \in [k]$. If the second assumption does not hold, let $p := \max_{i \in [k]} |V_i|$ and add $p - |V_i|$ isolated vertices to $V_i$, for each $i \in [k]$. (Note that adding isolated vertices does not change the answer to the problem.) For $i \in [k]$, we denote by $v_1^i, \ldots, v_p^i$ the vertices of $V_i$. We first describe the reduction of Fomin et al. [10] and then explain how to modify it to prove the theorem.

**The Construction of Fomin, Golovach and Raymond [10].** The graph $H$ is obtained as follows.

1. Construct $k$ nodes $u_1, \ldots, u_k$.

**2.** For every $1 \leq i < j \leq k$, construct a node $w_{i,j}$ and two pairs of parallel edges $u_i w_{i,j}$ and $u_j w_{i,j}$.

We then construct the subdivision $H'$ of $H$ by first subdividing each edge $p$ times. We denote the subdivision nodes for 4 edges of $H$ constructed for each pair $1 \leq i < j \leq k$ in Step 2 by $x_1^{(i,j)}, \ldots, x_p^{(i,j)}$, $y_1^{(i,j)}, \ldots, y_p^{(i,j)}$, $x_1^{(j,i)}, \ldots, x_p^{(j,i)}$, and $y_1^{(j,i)}, \ldots, y_p^{(j,i)}$. To simplify notation, we assume that $u_i = x_0^{(i,j)} = y_0^{(i,j)}$, $u_j = x_0^{(j,i)} = y_0^{(j,i)}$ and $w_{i,j} = x_{p+1}^{(i,j)} = y_{p+1}^{(i,j)} = x_{p+1}^{(j,i)} = y_{p+1}^{(j,i)}$.

We now construct the $H$-graph $G''$ by defining its $H$-representation $\mathcal{M} = \{M_v\}_{v \in V(G'')}$ where each $M_v$ is a connected subset of $V(H')$. (Recall that $G$ denotes the graph of the Multicolored Clique instance.)

**1.** For each $i \in [k]$ and $s \in [p]$, construct a vertex $z_s^i$ with model

$$M_{z_s^i} := \bigcup_{j \in [k], j \neq i} \left( \left\{ x_0^{(i,j)}, \ldots, x_{s-1}^{(i,j)} \right\} \cup \left\{ y_0^{(i,j)}, \ldots, y_{p-s}^{(i,j)} \right\} \right).$$

**2.** For each edge $v_s^i v_t^j \in E(G)$ for $s, t \in [p]$ and $1 \leq i < j \leq k$, construct a vertex $r_{s,t}^{(i,j)}$ with:

$$\begin{aligned} M_{r_{s,t}^{(i,j)}} := & \left\{ x_s^{(i,j)}, \ldots, x_{p+1}^{(i,j)} \right\} \cup \left\{ y_{p-s+1}^{(i,j)}, \ldots, y_{p+1}^{(i,j)} \right\} \\ & \cup \left\{ x_t^{(j,i)}, \ldots, x_{p+1}^{(j,i)} \right\} \cup \left\{ y_{p-t+1}^{(j,i)}, \ldots, y_{p+1}^{(j,i)} \right\}. \end{aligned}$$

Throughout the following, for $i \in [k]$ and $1 \leq i < j \leq k$, respectively, we use the notation

$$Z(i) := \bigcup_{s \in [p]} \left\{ z_s^i \right\} \text{ and } R(i,j) := \bigcup_{\substack{v_s^i v_t^j \in E(G), \\ s,t \in [p]}} \left\{ r_{s,t}^{(i,j)} \right\}.$$

We now observe the crucial property of $G''$.

▶ **Observation 12.1** (Claim 18 in [10]). *For every $1 \leq i < j \leq k$, a vertex $z_h^i \in V(G')$ (a vertex $z_h^j \in V(G')$) is* not *adjacent to a vertex $r_{s,t}^{(i,j)} \in V(G')$ corresponding to the edge $v_s^i v_t^j \in E(G)$ if and only if $h = s$ ($h = t$, respectively).*

**The New Gadget.** We now describe how to obtain from $G''$ a graph $G'$ that will be the graph of the instance of Max-$(\sigma^*, \mathbb{N}_{\geq x})$ Domination. We do so by adding a gadget to each set $Z(i)$ and $R(i,j)$ (for all $1 \leq i < j \leq k$). We first describe the gadget and then explain how to modify $H'$ to a new graph $K'$ such that $G'$ is a $K$-graph (where $K$ denotes the graph obtained from $K'$ by undoing the above described subdivisions that were made in $H$ to obtain $H'$). Let $X$ be any set of vertices of $G''$. The gadget $\mathfrak{B}(X)$ is a complete bipartite graph on $2d-1$ vertices and bipartition $(\{\beta_{1,1}, \ldots, \beta_{1,d}\}, \{\beta_{2,1}, \ldots, \beta_{2,d-1}\})$. such that for $h \in [d]$, each vertex $\beta_{1,h}$ is additionally adjacent to each vertex in $X$. For $1 \leq i < j \leq k$, we use the notation $\mathfrak{B}(i) := \mathfrak{B}(Z(i))$ and $\mathfrak{B}(i,j) := \mathfrak{B}(R(i,j))$ and we denote their vertices by $\beta_{\cdot,\cdot}^i$ and $\beta_{\cdot,\cdot}^{(i,j)}$, respectively.

We obtain $K'$ by 'hardcoding' each gadget $\mathfrak{B}(\cdot)$ into $H'$. That is, for $i \in [k]$, we add the graph $\mathfrak{B}(i)$ and connect it to the remaining vertices via the edges $u_i \beta_{1,h}^i$ for $h \in [d]$. For $1 \leq i < j \leq k$, we proceed analogously in encoding $\mathfrak{B}(i,j)$ into $H'$. For an illustration of the graph $K$, see Figure 2. We observe that $|K| = 2d\left(k + \binom{k}{2}\right) = kd(k+1)$ and

$$||K|| = 4\binom{k}{2} + \left(k + \binom{k}{2}\right) \cdot (d + d(d-1)) = \frac{1}{2}k\left(d^2(k+1) + 4(k-1)\right). \tag{2}$$

We subdivide all newly introduced edges, i.e. all edges in $E(K') \setminus E(H')$ and for an edge $xy \in E(K') \setminus E(H')$, we denote the resulting vertex by $s(x,y)$. We are now ready to describe (the $K$-representation of) $G'$.

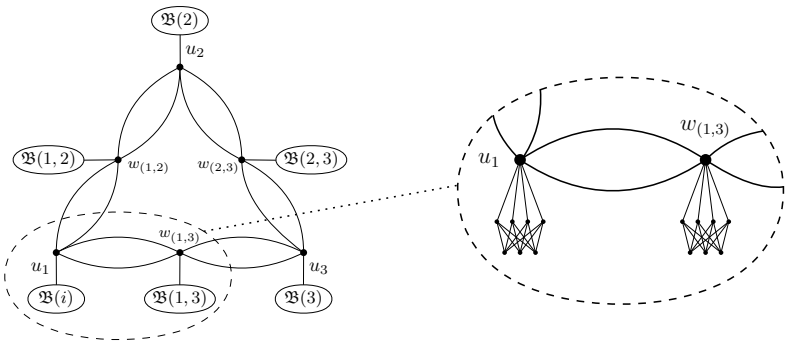



**Figure 2** The graph $K$ with respect to which the graph $G'$ constructed in the proof of Theorem 12 is a $K$-graph. In this example, we have $k = 3$ and $d = 4$.

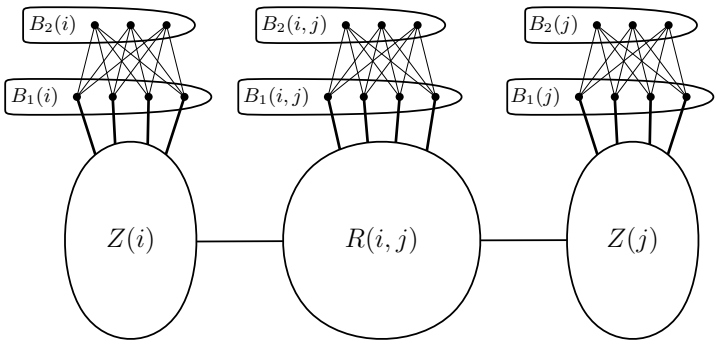



**Figure 3** A part of the graph $G'$, where $1 \leq i < j \leq k$ and $d = 4$.

1. For all $i \in [k]$ and $s \in [p]$, we add the vertices $s(u_i, \beta^i_{1,h})$ (where $h \in [d]$) to the model of $z^i_s$. For all $1 \leq i < j \leq k$ and $s, t \in [p]$ with $v^i_s v^j_t \in E(G)$, we add the vertices $s(w_{(i,j)}, \beta^{(i,j)}_{1,h})$ for $h \in [d]$ to the model of $r^{(i,j)}_{s,t}$.
2. For all $i \in [k]$ and $h \in [d]$, we add a vertex $b^i_{1,h}$ with model $\{\beta^i_{1,h}, s(u_i, \beta^i_{1,h})\} \cup \bigcup_{h' \in [d-1]} \{s(\beta^i_{1,h}, \beta^i_{2,h'})\}$.
3. For all $i \in [k]$ and $h \in [d-1]$, we add a vertex $b^i_{2,h}$ with model $\{\beta^i_{2,h}\} \cup \bigcup_{h' \in [d]} \{s(\beta^i_{2,h}, \beta^i_{1,h'})\}$.
4. For all $v^i_s v^j_t \in E(G)$ (where $1 \leq i < j \leq k$ and $s, t \in [p]$) and $h \in [d]$, we add a vertex $b^{(i,j)}_{1,h}$ with model $\{\beta^{(i,j)}_{1,h}, s(w_{(i,j)}, \beta^{(i,j)}_{1,h})\} \cup \bigcup_{h' \in [d-1]} \{s(\beta^{(i,j)}_{1,h}, \beta^{(i,j)}_{2,h'})\}$.
5. For all $v^i_s v^j_t \in E(G)$ (where $1 \leq i < j \leq k$ and $s, t \in [p]$) and $h \in [d-1]$, we add a vertex $b^{(i,j)}_{1,h}$ with model $\{\beta^{(i,j)}_{2,h}\} \cup \bigcup_{h' \in [d]} \{s(\beta^{(i,j)}_{2,h}, \beta^{(i,j)}_{1,h'})\}$.

One can verify that these five steps introduce the above described vertices to $G'$. For an illustration of $G'$, see Figure 3. The correctness proof of the reduction is given in the appendix; it is essentially a proof of the following claim.

▶ **Claim 12.2 (★).** *$G$ has a multicolored clique if and only if $G'$ has a $(\sigma^*, \mathbb{N}_{\geq x})$ set of size $k' = 2d \cdot (k + \binom{k}{2})$.*

We observe that $|V(G')| = \mathcal{O}(|V(G)| + d^2 \cdot k^2)$ and clearly, $G'$ can be constructed from $G$ in time polynomial in $|V(G)|$, $d$ and $k$ as well. Furthermore, by (2), $||K|| = \mathcal{O}(d^2 \cdot k^2)$ and the theorem follows. ◀

By Proposition 11, the previous theorem implies

▶ **Corollary 13.** *For any fixed $d \in \mathbb{N}$ and $x \leq d+1$, the following holds. Let $\sigma^* \subseteq \mathbb{N}_{\leq d}$ with $d \in \sigma^*$. Then,* Max-$(\sigma^*, \mathbb{N}_{\geq x})$ Domination *is* W[1]*-hard parameterized by linear mim-width plus solution size, and the hardness holds even if a corresponding decomposition tree is given.*

We now turn to hardness of minimization problems that can be expressed in $(\sigma, \rho)$ notation. First, with a slight modification of the reduction due to Fomin et al. [10], we obtain hardness for problems such as Total Dominating Set and Dominating Induced Matching.

▶ **Theorem 14 (★).** *For $\sigma^* \subseteq \mathbb{N}^+$ with $1 \in \sigma^*$ and $\rho^* \subseteq \mathbb{N}^+$ with $\{1,2\} \subseteq \rho^*$,* Min-$(\sigma^*, \rho^*)$ Domination *is* W[1]*-hard on $H$-graphs parameterized by the number of edges in $H$ plus solution size, and the hardness holds even when an $H$-representation of the input graph is given.*

As a somewhat orthogonal result to Theorem 12, we now show hardness of several problems related to the $d$-Dominating Set problem, where each vertex that is not in the solution set has to be dominated by at least some fixed number of $d$ neighbors in the solution.

▶ **Theorem 15 (★).** *For any fixed $d \in \mathbb{N}_{\geq 2}$,*[5] *the following holds. Let $\sigma^* \subseteq \mathbb{N}$ with $\{0, 1, d-1\} \subseteq \sigma^*$ and $\rho^* \subseteq \mathbb{N}_{\geq d}$ with $\{d, d+1\} \subseteq \rho^*$. Then,* Min-$(\sigma^*, \rho^*)$ Domination *is* W[1]*-hard on $H$-graphs parameterized by the number of edges in $H$ plus solution size, and the hardness even holds when an $H$-representation of the input graph is given.*

Similarly to above, a combination of the previous two theorems with Proposition 11 yields the following hardness results for $(\sigma, \rho)$ mimization problems on graphs of bounded linear mim-width.

▶ **Corollary 16.** *Let $\sigma^* \subseteq \mathbb{N}$ and $\rho^* \subseteq \mathbb{N}$. Then,* Min-$(\sigma^*, \rho^*)$ Domination *is* W[1]*-hard parameterized by linear mim-width plus solution size, if one of the following holds.*

**(i)** *$\sigma^* \subseteq \mathbb{N}^+$ with $1 \in \sigma^*$ and $\rho^* \subseteq \mathbb{N}^+$ with $\{1,2\} \subseteq \rho^*$.*

**(ii)** *For some fixed $d \in \mathbb{N}_{\geq 2}$, $\{0, 1, d-1\} \subseteq \sigma^*$ and $\rho^* \subseteq \mathbb{N}_{\geq d}$ with $\{d, d+1\} \subseteq \rho^*$.*

*Furthermore, the hardness holds even if a corresponding decomposition tree is given.*

## 7 Concluding Remarks

We have introduced the class of distance-$r$ $(\sigma, \rho)$ and LCVP problems. This generalizes well-known graph distance problems like distance-$r$ domination, distance-$r$ independence, distance-$r$ coloring and perfect $r$-codes. It also introduces many new distance problems for which the standard distance-1 version naturally captures a well-known graph property.

Using the graph parameter mim-width, we showed that all these problems are solvable in polynomial time for many interesting graph classes. These meta-algorithms will have runtimes which can likely be improved significantly for a particular problem on a particular graph class. For instance, blindly applying our results to solve Distance-$r$ Dominating Set on permutation graphs yields an algorithm that runs in time $\mathcal{O}(n^8)$: Permutation graphs have linear mim-width 1 (with a corresponding decomposition tree that can be computed in linear time) [3, Lemmas 2 and 5], so we can apply Corollary 7(i). However, there is an algorithm that solves Distance-$r$ Dominating Set on permutation graphs in time $\mathcal{O}(n^2)$ [15]; a much faster runtime.

We would like to draw attention to the most important and previously stated [13, 17, 20] open question regarding the mim-width parameter: Is there an XP approximation algorithm

[5] Note that the analogous statement for $d = 1$ follows from the reduction given in [10].

for computing mim-width? An important first step could be to devise a polynomial-time algorithm deciding if a graph has mim-width 1, or even linear mim-width 1.

Regarding lower bounds, we expanded on the previous results by Fomin et al. [10] and showed that many $(\sigma, \rho)$ problems are W[1]-hard parameterized by mim-width. However, it remains open whether there exists a problem which is NP-hard in general, yet FPT by mim-width. In particular, there are currently no hardness results when $\sigma$ and $\rho$ are both finite. Even so, we conjecture that every NP-hard (distance) $(\sigma, \rho)$ problem is W[1]-hard parameterized by mim-width.

## References


1. Geir Agnarsson, Peter Damaschke, and Magnús M. Halldórsson. Powers of geometric intersection graphs and dispersion algorithms. *Discrete Appl. Math.*, 132(1-3):3–16, 2003. doi:10.1016/S0166-218X(03)00386-X.
2. Gábor Bacsó, Dániel Marx, and Zsolt Tuza. H-free graphs, independent sets, and subexponential-time algorithms. In *Proc. IPEC 2016*, pages 3:1–3:12, 2017.
3. Rémy Belmonte and Martin Vatshelle. Graph classes with structured neighborhoods and algorithmic applications. *Theor. Comput. Sci.*, 511:54–65, 2013. doi:10.1016/j.tcs.2013.01.011.
4. Norman Biggs. Perfect codes in graphs. *J. Combin. Theory, Ser. B*, 15(3):289–296, 1973. doi:10.1016/0095-8956(73)90042-7.
5. A. Brandstädt, V. Le, and J. Spinrad. *Graph Classes: A Survey*. SIAM, 1999. doi:10.1137/1.9780898719796.
6. Binh-Minh Bui-Xuan, Jan Arne Telle, and Martin Vatshelle. Fast dynamic programming for locally checkable vertex subset and vertex partitioning problems. *Theor. Comput. Sci.*, 511:66–76, 2013. doi:10.1016/j.tcs.2013.01.009.
7. Steven Chaplick, Martin Töpfer, Jan Voborník, and Peter Zeman. On H-topological intersection graphs. In *Proc. WG 2017*, pages 167–179. Springer, 2017.
8. Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410(1):53–61, 2009.
9. Carsten Flotow. On powers of m-trapezoid graphs. *Discrete Appl. Math.*, 63(2):187–192, 1995. doi:10.1016/0166-218X(95)00062-V.
10. Fedor V. Fomin, Petr A. Golovach, and Jean-Florent Raymond. On the tractability of optimization problems on H-graphs. In *Proc. ESA 2018*, pages 30:1–30:14, 2018.
11. M. A. Henning, Ortrud R. Oellermann, and Henda C. Swart. Bounds on distance domination parameters. *J. Combin. Inform. Syst. Sci.*, 16(1):11–18, 1991.
12. Lars Jaffke, O-joung Kwon, Torstein J. F. Strømme, and Jan Arne Telle. Generalized Distance-Domination Problems and Their Complexity on Graphs of Bounded Mim-Width. *arXiv preprint*, 2018. arXiv:1803.03514.
13. Lars Jaffke, O-joung Kwon, and Jan Arne Telle. A Unified Polynomial-Time Algorithm for Feedback Vertex Set on Graphs of Bounded Mim-Width. In *Proc. STACS 2018*, pages 42:1–42:14, 2018.
14. Krzysztof Pietrzak. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *J. Comput. Syst. Sci.*, 67(4):757–771, 2003.
15. Akul Rana, Anita Pal, and Madhumangal Pal. An efficient algorithm to solve the distance $k$-domination problem on permutation graphs. *J. Discrete Math. Sci. Cryptography*, 19(2):241–255, 2016.

16 Arundhati Raychaudhuri. On powers of strongly chordal and circular arc graphs. *Ars Combin.*, 34:147–160, 1992.

17 Sigve Hortemo Sæther and Martin Vatshelle. Hardness of computing width parameters based on branch decompositions over the vertex set. *Theor. Comput. Sci.*, 615:120–125, 2016. doi:10.1016/j.tcs.2015.11.039.

18 Peter J. Slater. $R$-domination in graphs. *J. ACM*, 23(3):446–450, 1976.

19 Jan Arne Telle and Andrzej Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM J. Discrete Math.*, 10(4):529–550, 1997.

20 Martin Vatshelle. *New width parameters of graphs.* PhD thesis, University of Bergen, 2012.

# A Parameterized Complexity View on Collapsing $k$-Cores

**Junjie Luo**[1]
Algorithmics and Computational Complexity, Faculty IV, TU Berlin, Berlin, Germany
Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China
School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing, China
junjie.luo@campus.tu-berlin.de, luojunjie@amss.ac.cn

**Hendrik Molter**[2]
Algorithmics and Computational Complexity, Faculty IV, TU Berlin, Berlin, Germany
h.molter@tu-berlin.de

**Ondřej Suchý**[3]
Department of Theoretical Computer Science, Faculty of Information Technology,
Czech Technical University in Prague, Prague, Czech Republic
ondrej.suchy@fit.cvut.cz

## Abstract

We study the NP-hard graph problem Collapsed $k$-Core where, given an undirected graph $G$ and integers $b$, $x$, and $k$, we are asked to remove $b$ vertices such that the $k$-core of remaining graph, that is, the (uniquely determined) largest induced subgraph with minimum degree $k$, has size at most $x$. Collapsed $k$-Core was introduced by Zhang et al. [AAAI 2017] and it is motivated by the study of engagement behavior of users in a social network and measuring the resilience of a network against user drop outs. Collapsed $k$-Core is a generalization of $r$-Degenerate Vertex Deletion (which is known to be NP-hard for all $r \geq 0$) where, given an undirected graph $G$ and integers $b$ and $r$, we are asked to remove $b$ vertices such that the remaining graph is $r$-degenerate, that is, every its subgraph has minimum degree at most $r$.

We investigate the parameterized complexity of Collapsed $k$-Core with respect to the parameters $b$, $x$, and $k$, and several structural parameters of the input graph. We reveal a dichotomy in the computational complexity of Collapsed $k$-Core for $k \leq 2$ and $k \geq 3$. For the latter case it is known that for all $x \geq 0$ Collapsed $k$-Core is W[P]-hard when parameterized by $b$. We show that Collapsed $k$-Core is W[1]-hard when parameterized by $b$ and in FPT when parameterized by $(b + x)$ if $k \leq 2$. Furthermore, we show that Collapsed $k$-Core is in FPT when parameterized by the treewidth of the input graph and presumably does not admit a polynomial kernel when parameterized by the vertex cover number of the input graph.

**2012 ACM Subject Classification** Theory of computation → Graph algorithms analysis, Theory of computation → Parameterized complexity and exact algorithms

**Keywords and phrases** $r$-Degenerate Vertex Deletion, Feedback Vertex Set, Fixed-Parameter Tractability, Kernelization Lower Bounds, Graph Algorithms, Social Network Analysis

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.7

**Related Version** A full version is available at [18], `https://arxiv.org/abs/1805.12453`.

[1] Supported by CAS-DAAD Joint Fellowship Program for Doctoral Students of UCAS.
[2] Supported by the DFG, project MATE (NI 369/17).
[3] Supported by grant 17-20065S of the Czech Science Foundation.

© Junjie Luo, Hendrik Molter, and Ondřej Suchý;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 7; pp. 7:1–7:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

**Acknowledgements** This research was initiated at the annual research retreat of the Algorithmics and Computational Complexity (AKT) group of TU Berlin, held in Darlingerode, Germany, from March 19 till March 23, 2018. The authors would like to thank Anne-Sophie Himmel for initial discussions leading to the results in this paper.

## 1 Introduction

In recent years, modelling user engagement in social networks has received substantial interest [23, 24]. A popular assumption is that a user engages in a social network platform if she has at least a certain number of contacts, say $k$, on the platform. Further, she is inclined to abandon the social network if she has less than $k$ contacts [2, 7, 8, 11, 19, 25]. In compliance with this assumption, a suitable graph-theoretic model for the "stable" part of a social network is the so-called *$k$-core* of the social network graph, that is, the largest induced subgraph with minimum degree $k$ [21].[4]

Now, given a stable social network, that is, a graph with minimum degree $k$, the departure of a user decreases the degree of her neighbors in the graph by one which then might be smaller than $k$ for some of them. Following our assumption these users now will abandon the network, too. This causes a cascading effect of users dropping out (collapse) of the network until a new stable state is reached. From an adversarial perspective a natural question is how to maximally destabilize a competing social network platform by compelling $b$ users to abandon the network. This problem was introduced as Collapsed $k$-Core by Zhang et al. [25] and the decision version is formally defined as follows.

> Collapsed $k$-Core
> **Input:** An undirected graph $G = (V, E)$, and integers $b$, $x$, and $k$.
> **Question:** Is there a set $S \subseteq V$ with $|S| \leq b$ such that the $k$-core of $G - S$ has size at most $x$?

In the mentioned motivation, one would aim to minimize $x$ for a given $b$ and $k$. Alternatively, we can also interpret this problem as a measure for resilience agains user drop outs of a social network by determining the smallest $b$ for a given $k$ and $x$.

**Related Work.** In 2017 Zhang et al. [25] showed that Collapsed $k$-Core is NP-hard for any $k \geq 1$ and gave a greedy algorithm to compute suboptimal solutions for the problem. However, for $x = 0$ and any fixed $k$, solving Collapsed $k$-Core is equivalent to finding $b$ vertices such that after removing said vertices, the remaining graph is $(k-1)$-degenerate[5]. This problem is known as $r$-Degenerate Vertex Deletion and it is defined as follows.

> $r$-Degenerate Vertex Deletion
> **Input:** An undirected graph $G = (V, E)$, and integers $b$ and $r$.
> **Question:** Is there a set $S \subseteq V$ with $|S| \leq b$ such that $G - S$ is $r$-degenerate?

It is easy to see that Collapsed $k$-Core is a generalization of $r$-Degenerate Vertex Deletion. In 2010 Mathieson [20] showed that $r$-Degenerate Vertex Deletion is NP-complete and W[P]-complete when parameterized by the budget $b$ for all $r \geq 2$ even if the input graph is already $(r+1)$-degenerate and has maximum degree $2r+1$. In the mid-90s Abrahamson et al. [1] already claimed W[P]-completeness for $r$-Degenerate Vertex Deletion with $r = 2$ when parameterized by $b$ under the name Degree 3 Subgraph

[4] Note that the $k$-core of a graph is uniquely determined.
[5] A graph $G$ is $r$-degenerate if every subgraph of $G$ has a vertex with degree at most $r$ [10].

Annihilator. For $r = 1$, the problem is equivalent to Feedback Vertex Set and for $r = 0$ it is equivalent to Vertex Cover, both of which are known to be NP-complete and fixed-parameter tractable when parameterized by the solution size [9, 12]. The aforementioned results concerning $r$-Degenerate Vertex Deletion in fact imply the hardness results shown by Zhang et al. [25] for Collapsed $k$-Core.

**Our Contribution.** We complete the parameterized complexity landscape of Collapsed $k$-Core with respect to the parameters $b$, $k$, and $x$. Specifically, we correct errors in the literature [1, 20] concerning the W[P]-completeness of $r$-Degenerate Vertex Deletion when parameterized by $b$ for $r = 2$. We clarify the parameterized complexity of Collapsed $k$-Core for $k \leq 2$ by showing W[1]-hardness for parameter $b$ and fixed-parameter tractability for the combination of $b$ and $x$. Together with previously known results, this reveals a dichotomy in the computational complexity of Collapsed $k$-Core for $k \leq 2$ and $k \geq 3$.

We present two single exponential linear time FPT algorithms, one for Collapsed $k$-Core with $k = 1$ and one for $k = 2$. In both cases the parameter is $(b + x)$. In particular, the algorithm for $k = 2$ runs in $O(1.755^{x+4b} \cdot n)$ time which means that it solves Feedback Vertex Set in $O(9.487^b \cdot n)$ time (here, $b$ is the solution size of Feedback Vertex Set). Cao [5] independently developed a very similar algorithm for Feedback Vertex Set. In comparison, we additionally generalize the algorithm for Collapsed $k$-Core and give a more thorough running time analysis. To the best of our knowledge, despite of its simplicity this algorithm improves many previous linear time parameterized algorithm for Feedback Vertex Set [13, 17]. However, recently a linear time computable polynomial kernel for Feedback Vertex Set was shown [14], which together with e.g. [16] yields an even faster linear time parameterized algorithm.

Furthermore, we conduct a thorough parameterized complexity analysis with respect to structural parameters of the input graph. On the positive side, we show that Collapsed $k$-Core is fixed-parameter tractable when parameterized by the treewidth of the input graph and show that it presumably does not admit a polynomial kernel when parameterized by either the vertex cover number or the bandwidth of the input graph. We also show that the problem is fixed-parameter tractable when parameterized by the combination of the cliquewidth of the input graph and $b$ or $x$. Further results include W[1]-hardness when parameterized by the clique cover number of the input graph and para-NP-hardness for the domination number of the input graph.

Due to space contraints, results marked with ($\star$) are (in parts) deferred to a full version [18].

## 2 Hardness Results from the Literature

In this section, we gather and discuss known hardness results for Collapsed $k$-Core. Recall that that Collapsed $k$-Core with $x = 0$ is the same problem as $r$-Degenerate Vertex Deletion with $r = k - 1$. Hence, the hardness of Collapsed $k$-Core was first established by Mathieson [20] who showed that $r$-Degenerate Vertex Deletion is NP-complete and W[P]-complete when parameterized by the budget $b$ for all $r \geq 2$ even if the input graph is already $(r + 1)$-degenerate and has maximum degree $2r + 1$. However, in the proof of Mathieson [20] the reduction is incorrect for the case $r = 2$. Abrahamson et al. [1] claim W[P]-completeness for $r = 2$ but their reduction is also flawed. We provide counterexamples for both cases and show how to adjust the reduction of Mathieson [20] in a full version [18].

▶ **Theorem 1** (Corrected from [20]) ($\star$). *For any $r \geq 2$ $r$-Degenerate Vertex Deletion is NP-hard and W[P]-complete when parameterized by $b$, even if the degeneracy of the input graph is $r + 1$ and the maximum degree of the input graph is $2r + 1$.*

The following observation shows that the hardness result by Mathieson [20] (Theorem 1) easily transfers to Collapsed $k$-Core (also in the cases where $x \neq 0$).

▶ **Observation 2** ($\star$). *Let $x' > 0$ be a positive integer. There is a reduction which transforms instances $(G, b, x, k)$ of* Collapsed $k$-Core *with $x = 0$ into equivalent instances $(G', b, x', k)$ of* Collapsed $k$-Core.

With that, we arrive at the following corollary.

▶ **Corollary 3.** Collapsed $k$-Core *is NP-hard and W[P]-hard when parameterized by $b$ for all $x \geq 0$ and $k \geq 3$, even if the degeneracy of the input graph is $\max\{k, x-1\}$ and the maximum degree of the input graph is $\max\{2k-1, x-1\}$.*

Note that $r$-Degenerate Vertex Deletion is known to be NP-hard for all $r \geq 0$.[6] Hence, we also know that Collapsed $k$-Core is NP-hard for $k \leq 2$ and all $x \geq 0$. However, the parameterized complexity with respect to $b$ is open in this case. We settle this in the next section.

## 3 Algorithms and Complexity for $k = 1$ and $k = 2$

In this section we investigate the parameterized complexity of Collapsed $k$-Core for the case that $k \leq 2$. Since Theorem 3 only applies for $k \geq 3$ we first show in the following that the problem is W[1]-hard with respect to the combination of $b$ and $(n - x)$ for all $k \geq 1$. Furthermore, we present two algorithms; one that solves Collapsed $k$-Core with $k = 1$ and one for the $k = 2$ case. Both algorithms run in single exponential linear FPT-time with respect to the parameter combination $(b + x)$.

We first give a parameterized reduction from Clique to Collapsed $k$-Core. Note that since this hardness result holds for the combination of $b$ and the dual parameter of $x$, it is incomparable to Theorem 3 even for $k \geq 3$.

▶ **Proposition 4** ($\star$). Collapsed $k$-Core *is W[1]-hard when parameterized by the combination of $b$ and $(n-x)$ for all $k \geq 1$, even if the input graph is bipartite and $\max\{2, k\}$-degenerate.*

**Proof.** We reduce from W[1]-hard problem Clique [9], where given a graph $G = (V, E)$ and an integer $p$, the task is to decide whether $G$ contains a clique of size at least $p$. Let $(G, p)$ be an instance of Clique and $k$ be a given constant. We build an instance $(G', b, x, k)$ of Collapsed $k$-Core as follows. We can assume that $p \leq |V(G)|$, as otherwise we can output a trivial no-instance. We further assume that each vertex of $G$ has degree at least $p + 1$. Vertex of degree less than $p - 1$ is not part of a clique of size at least $p$, while for all vertices of degree $p - 1$ or $p$ we can check in $O(n \cdot p^3)$ time whether there is a clique of size at least $p$ containing any of them.

We let $V(G') = V \cup U \cup W$, where $V = V(G)$ are the vertices of $G$, $U = \{u_e^i \mid e \in E, i \in \{1, \ldots, k\}\}$, and $W = \{w_e^i \mid e \in E, i \in \{1, \ldots, k-1\}\}$. We also let $E(G') = E_U \cup E_W$, where $E_U = \{\{v, u_e^i\} \mid e \in E, i \in \{1, \ldots, k\}, v \in e\}$, and $E_W = \{\{u_e^i, w_e^{i'}\} \mid e \in E, i \in \{1, \ldots, k\}, i' \in \{1, \ldots, k-1\}\}$. We actually only introduce the sets $W$ and $E_W$ if $k \geq 2$.

Finally we set $b = p$ and $x = n - (p + (2k-1)\binom{p}{2})$, where $n = |V(G')|$ is the number of vertices of graph $G'$.

[6] Theorem 1 states NP-hardness of $r$-Degenerate Vertex Deletion for $r \geq 2$. Recall that for $r = 1$ $r$-Degenerate Vertex Deletion is equivalent to Feedback Vertex Set and for $r = 0$ it is equivalent to Vertex Cover, both of which are known to be NP-hard [12].

**Algorithm 1:** Algorithm for Collapsed $k$-Core with $k = 1$.

```
SolveRec(G, S, Q, b, x)
if |S| > b or |Q| > b + x then return No solution;
Let G' be the 1-core of G \ S.
if |V(G')| ≤ x then return S;
if V(G') ⊆ Q then return No solution;
Let v be the vertex with the highest degree in G' which is not in Q
T ← SolveRec(G, S ∪ {v}, Q, b, x)
if T ≠ No solution then return T;
else return SolveRec(G, S, Q ∪ {v}, b, x);
```

We claim that $(G', b, x, k)$ is a yes-instance of Collapsed $k$-Core if and only if $(G, p)$ is a yes-instance of Clique. We defer the proof of this claim to a full version [18]. ◀

Now we proceed with the algorithm for Collapsed $k$-Core with $k = 1$. While there is a simple algorithm with $O(3^{x+b}(m + n))$ running time[7] for this case, we consider an algorithm with the slightly worse running time as stated, since we then generalize this algorithm to the case $k = 2$ with some modifications.

▶ **Proposition 5** ($\star$). Collapsed $k$-Core *with* $k = 1$ *can be solved in* $O(2^{x+2b}(m + n))$ *time and in* $O(2^{O(b+\sqrt{bx})} \cdot n)$ *time. Assuming the Exponential Time Hypothesis, there is no* $2^{o(b)\cdot f(x)} n^{O(1)}$ *time algorithm for* Collapsed $k$-Core *with* $k = 1$*, for any function* $f$*.*

*Algorithm:* We present a recursive algorithm (see Algorithm 1 for pseudocode) that maintains two sets $S$ and $Q$. The recursive function is supposed to return a solution to the instance, whenever there is a solution $B$ containing all of $S$ and there is no solution containing $S$ and anything of $Q$. If some of the conditions is not met, then the function should return "No solution". In other words, $S$ is the set of deleted vertices and $Q$ is the set of vertices the algorithm has decided not to delete in the previous steps but may be collapsed in the future. Hence, the solution to the instance, or the information that there is none, is obtained by calling the recursive function with both sets $S$ and $Q$ empty.

The correctness proof and running time analysis of Algorithm 1 is deferred to a full version [18]. In the remainder of the section, we show how to adapt this algorithm for Collapsed $k$-Core with $k = 2$.

▶ **Theorem 6.** Collapsed $k$-Core *with* $k = 2$ *can be solved in* $O(1.755^{x+4b} \cdot n)$ *time and in* $O(2^{O(b+\sqrt{bx})} \cdot n)$ *time. Assuming the Exponential Time Hypothesis, there is no* $2^{o(b)\cdot f(x)} n^{O(1)}$ *time algorithm for* Collapsed $k$-Core *with* $k = 2$*, for any function* $f$*.*

The above theorem in particular yields an $O(9.487^b \cdot n)$ algorithm for Feedback Vertex Set.

*Algorithm:* Our algorithm for $k = 2$ is similar to Algorithm 1 with two main differences (see Algorithm 2 for pseudocode). First $|Q| > b + x$ is replaced by $|Q| > 3b + x$. Second when selecting the maximum degree vertex from $V(G') \setminus Q$, we need to make sure that this vertex

[7] An informal description of the algorithm: We use an initially empty set $X$ that should contain vertices of the remaining 1-core. We branch over edges where both endpoints are not in $X$ and either remove one of the endpoints or put both endpoint into $X$. Then we branch over all edges that have exactly one endpoint in $X$ and either remove the other endpoint or put the other endpoint into $X$ as well.

**Algorithm 2:** Algorithm for Collapsed $k$-Core with $k = 2$.

1 SolveRec2$(G, S, Q, b, x)$
2 **if** $|S| > b$ *or* $|Q| > 3b + x$ **then return** No solution;
3 Let $G'$ be the 2-core of $G \setminus S$.
4 **if** $|V(G')| \leq x$ **then return** $S$;
5 **if** $V(G') \subseteq Q$ **then return** No solution;
6 Let $v$ be the vertex with the highest degree in $G'$ which is not in $Q$
7 **if** $\deg_{G'}(v) \leq 2$ **then**
8 Let $C_1, \ldots, C_r$ be the connected components of $G'$ not containing vertices of $Q$, ordered such that $|V(C_1)| \geq |V(C_2)| \geq \ldots \geq |V(C_r)|$;
9 Let $r' \leftarrow \min\{r, b - |S|\}$;
10 **if** $|V(G')| - \sum_{i=1}^{r'} |V(C_i)| \leq x$ **then**
11 **for** $i = 1, \ldots, r'$ **do** Select an arbitrary vertex from $C_i$ and add it to $S$;
12 **return** $S$
13 **end**
14 **else return** No solution;
15 **end**
16 $T \leftarrow$ SolveRec2$(G, S \cup \{v\}, Q, b, x)$ ;
17 **if** $T \neq$ No solution **then return** $T$;
18 **else return** SolveRec2$(G, S, Q \cup \{v\}, b, x)$ ;

has degree greater than 2. Otherwise, either we can directly select vertices from $V(G') \setminus Q$ to break cycles in $G'$ and get a 2-core of size at most $x$, or the algorithm rejects this branch.

We start by claiming that Algorithm 2 has the following running time.

▶ **Lemma 7** ($\star$). *Algorithm 2 runs in $O(1.755^{x+4b} \cdot n)$ time and in $O(2^{O(b+\sqrt{bx})} \cdot n)$ time.*

Next we claim the following conditional lower bound on the running time for any algorithm for Collapsed $k$-Core with $k = 2$.

▶ **Lemma 8** ($\star$). *Assuming the Exponential Time Hypothesis, there is no $2^{o(b) \cdot f(x)} n^{O(1)}$ time algorithm for* Collapsed $k$-Core *with $k = 2$, for any function $f$.*

Before showing the correctness of Algorithm 2, we give the following lemmata which will be helpful in the correctness proof. The following lemma claims that, except for some specific connected components, we can limit the solution to contain vertices of degree at least three.

▶ **Lemma 9** ($\star$). *For any instance $(G, b, x, 2)$ of* Collapsed $k$-Core *with $k = 2$, where $G$ is a 2-core and does not contain a cycle as a connected component, if there is a solution $B$ for $(G, b, x, 2)$, then there is also a solution $B'$ for $(G, b, x, 2)$ which contains only vertices with degree larger than 2.*

The next lemma helps to show that line 14 is correct.

▶ **Lemma 10** ($\star$). *If line 14 of Algorithm 2 applies, and for every $q \in Q$ there no solution containing whole $S \cup \{Q\}$, then there is no solution containing the whole $S$.*

Next, we show that the second part of line 2 of Algorithm 2 is correct.

▶ **Lemma 11.** *If $Q$ is of size more than $3b + x$, then there is no solution containing whole $S$ and no vertex from $Q$.*

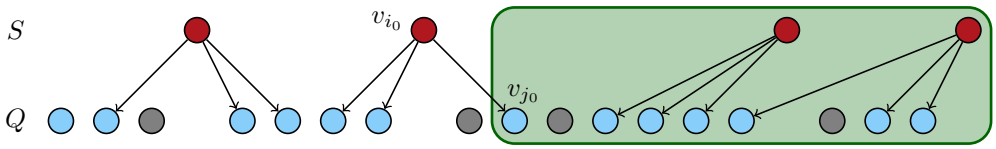



**Figure 1** Illustration of function $f$. Vertices in $Q$ are separated into two parts: gray vertices from $X$ and blue vertices from $Q' = Q \setminus X$. Every vertex in $S$ is mapped to a set of three consecutive blue vertices in $Q'$ from the right to the left. Graph $G_{j_0}$ with the property that $i > \max\{j \mid j \in f(i)\}$ for every $i$ with $v_i \in B \cap V(G_{j_0})$ is contained in the green box.

**Proof.** Suppose for contradiction that there is a set $Q$ of size at least $3b + x + 1$ and a solution $B$ such that $|B| \leq b$, $S \subseteq B$ and $B \cap Q = \emptyset$. Let $v_1, v_2, \ldots, v_{r'}$ be the vertices of the set $S \cup Q$ in the order as they were added to the set by successive recursive calls. Moreover, if $B \setminus S$ is empty, then let $r = r'$. Otherwise, let $B \setminus S = \{v_{r'+1}, \ldots, v_r\}$, i.e., in both cases $\{v_1, \ldots, v_r\} = B \cup Q$. Without loss of generality, we can assume that for every vertex $v_i \in B$, $v_i$ is contained in the 2-core of $G \setminus (B \cap \{v_1, v_2, \ldots, v_{i-1}\})$, since otherwise $v_i$ must come from $B \setminus S$, then $B' = B \setminus \{v_i\}$ is also a solution such that $|B'| \leq b$, $S \subseteq B'$ and $B' \cap Q = \emptyset$. Let $G'$ be the 2-core of $G \setminus B$ and let $X = V(G') \cap Q$. Since $B$ is a solution, we know that $|X| \leq x$.

We construct a function $f$ that maps every vertex of $B$ to a set of three consecutive vertices of $Q \setminus X$ (see also Figure 1). First let $v_i$ be the vertex in $B$ with the largest $i$. We let $f(i)$ be the set $\{j_1, j_2, j_3\}$, where $j_1 = \max\{j \mid v_j \in Q \setminus X\}$, $j_2 = \max\{j < j_1 \mid v_j \in Q \setminus X\}$, and $j_3 = \max\{j < j_2 \mid v_j \in Q \setminus X\}$. Now let $v_i$ be the vertex from $B$ with the largest $i$ such that $f(i)$ was not set yet and $v_{i'}$ be the vertex from $B$ with the least $i'$ such that $i' > i$. We set $f(i) = \{j_1, j_2, j_3\}$, where $j_1 = \max\{j < \min\{k \in f(i')\} \mid v_j \in Q \setminus X\}$, $j_2 = \max\{j < j_1 \mid v_j \in Q \setminus X\}$, and $j_3 = \max\{j < j_2 \mid v_j \in Q \setminus X\}$. Since the set $Q \setminus X$ contains at least $3b + 1$ vertices, while $B$ contains at most $b$, this way we find a mapping for every vertex in $B$, keeping the images of different vertices disjoint. Moreover, denote $p = |Q \setminus X| - 3|B| \geq 1$. There remains $p$ vertices in $Q \setminus X$ not being in the union of images of $f$, let us denote them $v_{q_1}, \ldots, v_{q_p}$.

For $t \in \{1, \ldots, r\}$ let $G_t$ be the 2-core of the graph $G \setminus (B \cap \{v_1, \ldots, v_{t-1}\})$. For $v \in V(G_t)$ let $\deg_t(v)$ be the degree of the vertex $v$ in $G_t$. By the way we selected $v_t$ we again know that $\deg_t(v_t) \geq \deg_t(v_{t'}) \geq \deg_{t'}(v_{t'})$ for every $t < t' \leq r'$ and $\deg_t(v_t) \geq \deg_t(v_{t'}) \geq \deg_{t'}(v_{t'})$ for every $t \leq r' < t'$. Note also that $\deg_t(v_t) \geq 3$ for all $v_t \in Q$.

If for every vertex $v_i \in B$ we have $i > \max\{j \mid j \in f(i)\}$, then $\deg_i(v_i) \leq \deg_j(v_j)$ for vertex $v_i$ in $B$ and every $j \in f(i)$. Together with $\deg_t(v_t) \geq 3$ for all $v_t \in Q$, we have $\deg_i(v_i) - \sum_{j \in f(i)} \deg_j(v_j) + 6 \leq 0$ for every vertex $v_i$ in $B$. Let $V = B \uplus Q' \uplus X \uplus Y \uplus Z$, where $Q' = Q \setminus X$, $Y$ is the set of collapsed vertices not contained in $Q$ and $Z = V(G') \setminus X$ the part of the core of $G \setminus B$ not in $Q$. Now we transform graph $G$ into a partial directed graph by considering the collapsing process (see also Figure 2). More precisely, for every edge in $G$, except for edges which have two endpoints in $X \cup Z$, we will assign it a direction. To this end, we just need to give an order of vertices in $V(G) \setminus (X \cup Z)$. We define this order based on the time vertices being deleted or collapsed. It may happen that several vertices in $Q'$ or $Y$ collapse at the same time. For this situation, we just order these vertices according to an arbitrary but fixed order. Since $k = 2$, every collapsed vertex in $Q' \cup Y$ has at most one outgoing edge. Let us consider the number, denoted by $N$, of edges of the form $\overrightarrow{v_i v_j}$ such that the head $v_j$ is in $Q'$. Since every vertex in $Q'$ has at most one outgoing edge, we have $N \geq \sum_{v_j \in Q'} \deg_j(v_j) - |Q'|$.

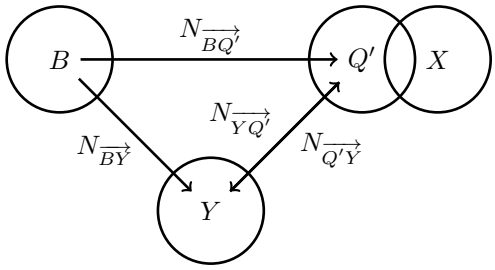



**Figure 2** Illustration of the partial directed graph when considering the collapsing process. The set $B$ contains the deleted vertices and $X$ is the set of vertices remaining in the 2-core of $G \setminus B$. The set $Q'$ contains vertices the algorithm has decided not to delete but eventually collapse and $Y$ is the set of other collapsed vertices.

On the other hand, let

- $N_{\overrightarrow{BQ'}}$ be the number of edges going from $B$ to $Q'$;
- $N_{\overrightarrow{BY}}$ be the number of edges going from $B$ to $Y$;
- $N_{\overrightarrow{YQ'}}$ be the number of edges going from $Y$ to $Q'$;
- $N_{\overrightarrow{Q'Y}}$ be the number of edges going from $Q'$ to $Y$.

We claim that $N_{\overrightarrow{YQ'}} \leq N_{\overrightarrow{BY}} + N_{\overrightarrow{Q'Y}}$. To show that, denote the number of edges in $G[Y]$ by $N_{\overrightarrow{YY}}$. Since every vertex in $Y$ has at least one incoming edge but at most one outgoing edge, we have $\sum_{v \in Y} \deg^-(v) \leq \sum_{v \in Y} \deg^+(v)$, where $\deg^+(v)$ ($\deg^-(v)$) is the number of incoming (outgoing) edges of vertex $v$ in $G$, respectively. This means $N_{\overrightarrow{YQ'}} + N_{\overrightarrow{YY}} \leq N_{\overrightarrow{BY}} + N_{\overrightarrow{Q'Y}} + N_{\overrightarrow{YY}}$, therefore, $N_{\overrightarrow{YQ'}} \leq N_{\overrightarrow{BY}} + N_{\overrightarrow{Q'Y}}$, finishing the proof of the claim.

Since the edges which have their heads in $Q'$ have their tails from $B \cup Y \cup Q'$,

$$\begin{aligned} N &\leq N_{\overrightarrow{BQ'}} + N_{\overrightarrow{YQ'}} + (|Q'| - N_{\overrightarrow{Q'Y}}) \leq N_{\overrightarrow{BQ'}} + |Q'| + N_{\overrightarrow{BY}} \\ &\leq \sum_{v_i \in B} \deg^-(v_i) + |Q'| \leq \sum_{v_i \in B} \deg_i(v_i) + |Q'|. \end{aligned}$$

The last inequality holds since for every vertex $v_i \in B$, $v_i$ is contained in the 2-core of $G \setminus (B \cap \{v_1, v_2, \ldots, v_{i-1}\})$, which means all outgoing edges of $v_i$ are counted in $\deg_i(v_i)$. Then we have that $\sum_{v_i \in B} \deg^-(v_i) \leq \sum_{v_i \in B} \deg_i(v_i)$.

$$\begin{aligned} 0 &\leq \sum_{v_i \in B} \deg_i(v_i) + |Q'| - \Big( \sum_{v_j \in Q'} \deg_j(v_j) - |Q'| \Big) \\ &= \sum_{v_i \in B} \deg_i(v_i) - \sum_{v_j \in Q'} (\deg_j(v_j) - 2) \\ &\leq \sum_{v_i \in B} \deg_i(v_i) - \Big( \sum_{v_i \in B} \sum_{j \in f(i)} (\deg_j(v_j) - 2) + \sum_{i=1}^{p} (\deg_{q_i}(v_{q_i}) - 2) \Big) \\ &< \sum_{v_i \in B} \deg_i(v_i) - \sum_{v_i \in B} \sum_{j \in f(i)} (\deg_j(v_j) - 2) \\ &= \sum_{v_i \in B} \Big( \deg_i(v_i) - \sum_{j \in f(i)} \deg_j(v_j) + 6 \Big) \leq 0 \end{aligned}$$

which is a contradiction.

The remaining case where there exist a $i$ such that $i < \max\{j \mid j \in f(i)\}$ is analogous and deferred to a full version [18]. ◀

The last part of the proof of Theorem 6 is deferred to a full version [18].

## 4 Structural Graph Parameters

In this section, we investigate the parameterized complexity of Collapsed $k$-Core with respect to several structural parameters of the input graph. Theorem 3 already implies hardness for constant values of several structural graph parameters. We expand this picture by observing that the problem remains NP-hard on graphs with a dominating set of size one and by showing that the problem is W[1]-hard when parameterized by the combination of $b$ and the clique cover number of the input graph. On the positive side, we show that the problem is in FPT when parameterized by the treewidth of the input graph or the clique-width of the input graph and $k$ combined with either $b$, $x$, $n - x$, or $n - b$. Lastly, we show that the problem presumably does not admit a polynomial kernel for any $k \geq 2$ when parameterized by the combination of $b$ and the vertex cover number of the input graph, or by the combination of $b$, $k$, and the bandwidth of the input graph.

We start with an easy observation that we will make use of in most of the hardness results in this section.

▶ **Observation 12** ($\star$). *If $(G, b, x, k)$ is an instance of* Collapsed $k$-Core *and vertex $v$ is a part of the $(k + b)$-core of $G$, and $S \subseteq V$ is of size at most $b$, then either $v \in S$ or $v$ is part of the $k$-core of $G \setminus S$.*

The following observation yields that we can reduce the size of a dominating set of any instance of Collapsed $k$-Core to one by introducing a universal vertex. Note that, for example, this only increases the degeneracy by one.

▶ **Observation 13** ($\star$). *Let $(G, b, x, k)$ be an instance of* Collapsed $k$-Core *and $G'$ be the graph obtained from $G$ by adding a universal vertex, then $(G', b + 1, x, k)$ is an equivalent instance of* Collapsed $k$-Core.

Considering a larger parameter than, e.g., the size of the dominating set, namely the clique cover number[8], we can show W[1]-hardness, even in combination with $b$. This can be done with a parameterized reduction from Multicolored Clique parameterized by the solution size.

▶ **Proposition 14** ($\star$). Collapsed $k$-Core *is W[1]-hard when parameterized by the combination of $b$ and the clique cover number of the input graph.*

On the positive side, we sketch a dynamic program on the tree decomposition of the input graph $G$ which implies that Collapsed $k$-Core is in FPT when parameterized by the treewidth of the input graph.

▶ **Proposition 15** ($\star$). Collapsed $k$-Core *is in FPT when parameterized by the treewidth of the input graph.*

**Proof Sketch.** Let $(G, b, x, k)$ be an instance of Collapsed $k$-Core. Observe that either $k \leq \mathrm{tw}(G)$ or the $k$-core of $G$ is (already) empty and we can answer Yes. Hence, for the rest of the proof we assume that $k \leq \mathrm{tw}(G)$. We assume that we are given a nice tree decomposition

[8] The clique cover number of a graph $G$ is the minimum number of induced cliques such that their union contains all vertices of $G$.

of $G$ [15, 3] and use dynamic programming on the nice tree decomposition of $G$. The indices of the table are formed for each bag of the decomposition by the number of vertices of the solution already forgotten, the number of vertices in the core already forgotten, a partition of the bag into three set $B$, $X$, and $Q$, an (elimination) order for the vertices in $Q$, and for each vertex in $Q$ the number of its neighbors in $X$ or higher in the order. This number is always in $0, \ldots, k-1$, as otherwise it would not be possible to eliminate the vertex.

The set $B$ represents the partial solution (or rather its intersection with the bag), i.e., the vertices to be deleted. The set $X$ represents the vertices which (are free to) remain in the core. The vertices in $Q$ should collapse after removing the vertices of the solution and the collapse of the vertices preceding them in the order.

There are $3^{\mathrm{tw}(G)} \cdot (\mathrm{tw}(G))^{O(\mathrm{tw}(G))} \cdot k^{\mathrm{tw}(G)} = (\mathrm{tw}(G))^{O(\mathrm{tw}(G))}$ possible indices for each bag. Hence the slightly superexponential running time of $(\mathrm{tw}(G))^{O(\mathrm{tw}(G))} \cdot n^{O(1)}$ follows. ◀

Using monadic second order (MSO) logic formulas, it can be shown that for a smaller structural parameter, namely the cliquewidth of the input graph, there are also positive results. Here however, we can only show fixed-parameter tractability for the combination of the cliquewidth of the input graph with $k$ and either $b$, $x$, $n-x$, or $n-b$.

▶ **Proposition 16** ($\star$). Collapsed $k$-Core *is in FPT when parameterized by the cliquewidth of the input graph combined with $k$ and either $b$, $x$, $n-x$, or $n-b$.*

In the remainder of this section, we show that Collapsed $k$-Core does not admit a polynomial kernel when parameterized by rather large parameter combinations. We first show an OR-cross composition [4, 9] from Cubic Vertex Cover.

▶ **Theorem 17.** *For all $k \geq 2$* Collapsed $k$-Core *does not admit a polynomial kernel when parameterized by the combination of $b$ and the vertex cover number of the input graph unless NP $\subseteq$ coNP/poly.*

**Proof.** We apply an OR-cross composition [4, 9] from the NP-hard problem Cubic Vertex Cover [12]. In Cubic Vertex Cover, we are given a 3-regular graph $G$ and an integer $s$, and the task is to find a vertex subset of size at most $s$ which contains at least one endpoint of each edge of $G$.

We say an instance of Cubic Vertex Cover is *malformed* if the string does not represent a pair $(G, s)$, where $G$ is a 3-regular graph and $s$ is a positive integer. It is *trivial*, if $s \geq |V(G)|$. We define the equivalence relation $\mathcal{R}$ as follows: all malformed instances are equivalent, all trivial instances are equivalent and two well-formed non-trivial instances $(G, s)$ and $(G', s')$ are $\mathcal{R}$-equivalent if $|V(G)| = |V(G')|$ and $s = s'$. Observe that $\mathcal{R}$ is a polynomial equivalence relation.

Let the input consist of $T$ $\mathcal{R}$-equivalent instances of Cubic Vertex Cover. If the instances are malformed or trivial, we return a constant size no- or yes- instance of Collapsed $k$-Core, respectively. Let $(G_i, s)_{0 \leq i \leq T-1}$ be well-formed non-trivial $\mathcal{R}$-equivalent instances of Cubic Vertex Cover. Since all instances have the same size of the vertex set, we can assume they share the same vertex set $V = \{v_1, v_2, \ldots, v_n\}$ . We assume $T$ to be a power of 2, as otherwise we can duplicate some instances. Now we create an instance $(G, b, x, k)$ of Collapsed $k$-Core for some arbitrary but fixed $k \geq 2$ as follows.

- Set $b = s + 2ks \log T$ and $x = N - N'$, where $N = n + \frac{3}{2}nT + 4ks \log T + k + s + \frac{3}{2}n(k-2)$ is the number of all vertices in graph $G$ we will construct and $N' = s + 2ks \log T + \frac{3}{2}n$.
- First for every vertex $v_i$ in $V$, create a vertex $v_i$ in $G$.

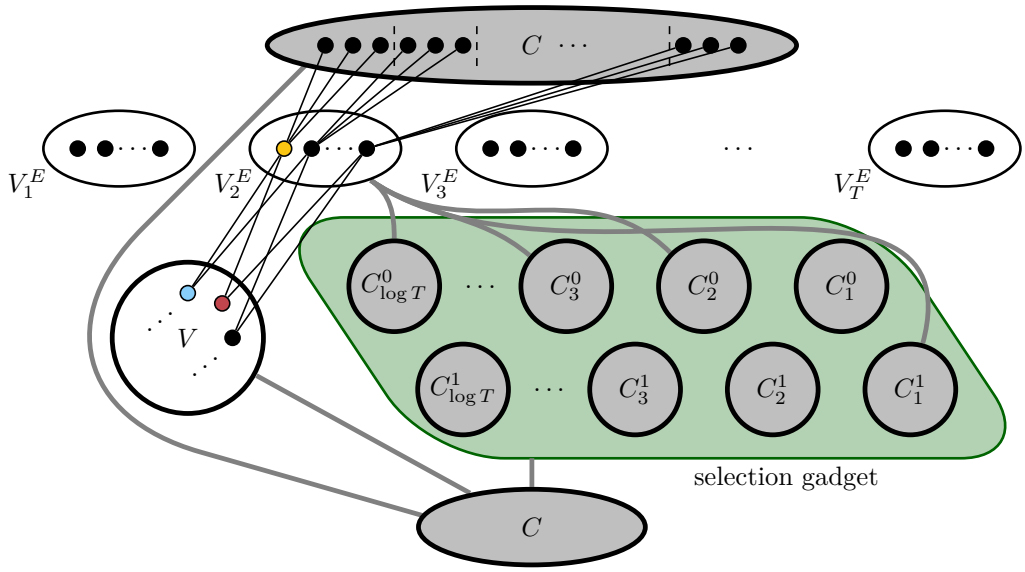



**Figure 3** Illustration of the OR-cross composition from Cubic Vertex Cover to Collapsed $k$-Core with $k = 5$. The selection gadget is all the circles contained in the green box. Every gray edge in this figure means that all vertices in one endpoint of this edge are connected to all vertices in the other endpoint. Every vertex in $V_i^E$ connects to two endpoints of its corresponding edge in $G_i$. For example, the yellow vertex in $V_2^E$ is connected to the blue and the red vertex in $V$, which represents the two endpoints of the corresponding edge in $G_2$. The big clique $C$ is separated into two parts. Every vertex in $V_i^E$ is connected to $k-2$ vertices in the upper part of $C$. Since $k = 5$, the yellow vertex in $V_2^E$ is connected to three vertices in $C$. Vertices contained in thick outlined vertex sets form a vertex cover. To keep the picture simple, edges that contain vertices from $V_i^E$ with $i \neq 2$ are not depicted.

- For every edge set $E(G_i)$, create a vertex set $V_i^E$ in $G$, in which a vertex $v_{p,q}$ represents an edge $v_pv_q$ in $E(G_i)$. Then we have $T$ of these vertex sets and each set has $\frac{3}{2}n$ vertices.
- For every edge $v_pv_q$ in $E(G_i)$, add 2 edges $v_{p,q}v_p$ and $v_{p,q}v_q$ in $G$.
- Now create the selection gadget in $G$. It contains $\log T$ pairs of cliques $C_i^d$ ($1 \leq i \leq \log T, d \in \{0,1\}$), and all of them have the same size of $2ks$. For every vertex set $V_i^E$, let $i = (d_{\log T-1}d_{\log T-2}\dots d_0)_2$ be the binary representation of the index $i$, where $d_j \in \{0,1\}$ for $0 \leq j \leq \log T - 1$ and we add leading zeros so that the length of the representation is exactly $\log T$. We add edges between all vertices in $V_i^E$ and all vertices in $\bigcup_{j=0}^{\log T-1} C_j^{d_j}$.
- Finally we create a clique $C$ with $|C| = k + b + \frac{3}{2}n(k-2)$, which contains two parts of vertices. The first part contains $k + b$ vertices and each of them connects to all vertices in $V \cup \bigcup_{j=0}^{\log T-1}\bigcup_{d=0}^{1} C_j^d$. The second part of $\frac{3}{2}n(k-2)$ vertices is connected to vertices in $V_i^E$ in the following way. For every vertex $v_{p,q}$ in $V_i^E$, add edges between $v_{p,q}$ and $k-2$ vertices in $C$. We make sure that all vertices in the same $V_i^E$ connect to different vertices in $C$. In other words, every vertex in the second part of $C$ connects to exactly one vertex in every $V_i^E$.

Notice that the vertex cover number of $G$ is $n+4ks\log T+k+b+\frac{3}{2}n(k-2)$. The construction is illustrated in Figure 3. We now show that at least one instance $(G_i, s)$ is a yes-instance if and only if the instance $(G, b, x, k)$ of Collapsed $k$-Core constructed above is a yes-instance.

$\Rightarrow$: If $(G_i, s)$ is a yes-instance, which means that there is a vertex subset $V^*$ of size $s$ that covers all edges in $G_i$, then we delete the corresponding $s$ vertices in $G$ and all vertices in $\bigcup_{j=0}^{\log T-1} C_j^{d_j}$, where $i = (d_{\log T-1}\dots d_0)_2$ is the binary representation of $i$. So far, we deleted

$s + 2ks \log T$ vertices, and all vertices in $V_i^E$ will collapse, since they just have at most $k-1$ edges remaining, $k-2$ of which connect to vertices in $C$ and at most one to vertices in $V$. Therefore, the number of remaining vertices is $x$ and instance $(G, b, x, k)$ is a yes-instance.

$\Leftarrow$: If $(G, b, x, k)$ is a yes-instance, we need to show that there is at least one instance which has a vertex cover of size at most $s$. Let $S$ be the set of deleted vertices of size at most $b$ and let $S'$ be the set of all collapsed vertices. Since $N' = s + 2ks \log T + \frac{3}{2}n$, we have $|S'| \geq \frac{3}{2}n$. In the subgraph of $G$ induced by $V$, $C$ and $\bigcup_{j=0}^{\log T-1} \bigcup_{d=0}^{1} C_j^d$ all vertices in $V \cup \bigcup_{j=0}^{\log T-1} \bigcup_{d=0}^{1} C_j^d \cup C$ have degree larger than $k+b$. Hence, by Theorem 12 they will not collapse and all collapsed vertices come from $\bigcup_{i=0}^{T-1} V_i^E$.

We show that all collapsed vertices can only come from one single $V_i^E$ for some $i$. Suppose two vertices $v$ and $v'$ from different sets of $V_i^E$ $(0 \leq i \leq T-1)$ collapse after deleting $S$, then there is at least one pair of cliques $C_{j_0}^0$ and $C_{j_0}^1$ such that $v$ is connected to all vertices in $C_{j_0}^{d_0}$ for some $d_0 \in \{0, 1\}$ and $v'$ is connected to all vertices in $C_{j_0}^{1-d_0}$. To make $v$ collapse, at least $2ks \log T - (k-1)$ vertices from the corresponding cliques in the selection gadget need to be deleted. Then to make $v'$ collapse, at least $2ks - (k-1)$ vertices from $C_{j_0}^{1-d_0}$ need to be deleted. Therefore, at least $2ks \log T + 2ks - 2(k-1)$ vertices need to be deleted, which is strictly more than $b$. This means that the collapsed vertices come from one single $V_i^E$. Since $|S'| \geq \frac{3}{2}n$ and $|V_i^E| = \frac{3}{2}n$, we have $S' = V_i^E$ and $S \cap V_i^E = \emptyset$ for some $i$.

We consider the vertex set $S$. We know that after deleting $S$, all vertices in $V_i^E$ collapse. Denote $V_I$ the vertex set of all vertices in $\bigcup_{j=0}^{\log T-1} C_j^{d_j}$, where $i = (d_{\log T-1} \dots d_0)_2$ is the binary representation of $i$. Since every vertex in $V_I$ is connected to all vertices in $V_i^E$, to make $V_i^E$ collapse, it is always better to choose vertices from $V_I$ than any other vertex. If $S$ does not contain all vertices from $V_I$, we can update $S$ by replacing any $|V_I \setminus S|$ vertices in $S$ with vertices in $V_I \setminus S$. Then $V_I \subseteq S$.

Suppose there is a vertex $v_{p,q}$ in $V_i^E$ such that both $v_p$ and $v_q$ are not in $S \cap V$, then $S$ contains at least one vertex $v_c$ in $C$ connected to $v_{p,q}$, as otherwise $v_{p,q}$ has degree at least $k$ and will not collapse. We update $S$ by replacing $v_c$ with $v_p$. This will not influence the size of $S$ and more importantly, this will not influence the collapsed set $S' = V_i^E$, since $v_c$ in $C$ is connected to only one vertex $v_{p,q}$ in $V_i^E$, and $v_{p,q}$ will still collapse under the new $S$. By updating $S$ in the same way for other vertices in $V_i^E$ not covered by vertices in $S \cap V$, we get a vertex set $S \cap V$ which covers all vertices in $V_i^E$ at least once. And $|S \cap V| \leq s$, since $V_I \subseteq S$ and $|V_I| = 2ks \log T$. This corresponds to a vertex cover of size $s$ in $G_i$. ◀

Lastly, we claim that there is a simple OR-cross composition [4, 9] from Collapsed $k$-Core onto itself. Note that the parameter combination of the following result is incomparable to that of Theorem 17.

▶ **Proposition 18** ($\star$). Collapsed $k$-Core *does not admit a polynomial kernel when parameterized by the combination of $b$, $k$, and the bandwidth of the input graph unless NP $\subseteq$ coNP/poly.*

## 5 Conclusion

Our results highlight a dichotomy in the computational complexity of Collapsed $k$-Core for $k \leq 2$ and $k \geq 3$. Along the way, we correct some inaccuracies in the literature concerning the parameterized complexity of Collapsed $k$-Core with $k = 3$ and $x = 0$ and give a simple single exponential linear time parameterized algorithm for Collapsed $k$-Core with $k = 2$, which almost matches the simplest known, independently found, single exponential linear time algorithm for Feedback Vertex Set. We leave as an open question whether

Collapsed $k$-Core with $k = 2$ on graphs of maximum degree 3 is polynomial time solvable similarly as Feedback Vertex Set [6, 22]. This would improve also the running time of our algorithm for Collapsed $k$-Core with $k = 2$ on general graphs. We further investigate the parameterized complexity with respect to several structural parameters of the input graph. As a highlight we show that Collapsed $k$-Core does not admit polynomial kernels for rather large parameter combinations. We leave the complexity of Collapsed $k$-Core when parameterized solely by the cliquewidth of the input graph open.

## References


1. Karl A. Abrahamson, Rodney G. Downey, and Michael R. Fellows. Fixed-parameter tractability and completeness IV: On completeness for W[P] and PSPACE analogues. *Annals of Pure and Applied Logic*, 73(3):235–276, 1995.
2. Kshipra Bhawalkar, Jon Kleinberg, Kevin Lewi, Tim Roughgarden, and Aneesh Sharma. Preventing unraveling in social networks: the anchored $k$-core problem. *SIAM Journal on Discrete Mathematics*, 29(3):1452–1475, 2015.
3. Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michał Pilipczuk. A $c^k n$ 5-Approximation Algorithm for Treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016.
4. Hans L Bodlaender, Bart MP Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014.
5. Yixin Cao. A Naive Algorithm for Feedback Vertex Set. In *Proceedings of the 1st Symposium on Simplicity in Algorithms, (SOSA '18)*, volume 61 of *OASICS*, pages 1:1–1:9. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.
6. Yixin Cao, Jianer Chen, and Yang Liu. On feedback vertex set: New measure and new structures. *Algorithmica*, 73(1):63–86, 2015.
7. Rajesh Chitnis, Fedor V. Fomin, and Petr A. Golovach. Parameterized complexity of the anchored $k$-core problem for directed graphs. *Information and Computation*, 247:11–22, 2016.
8. Rajesh Hemant Chitnis, Fedor V. Fomin, and Petr A. Golovach. Preventing Unraveling in Social Networks Gets Harder. In *Proceedins of the 27th AAAI Conference on Artificial Intelligence (AAAI '13)*, pages 1085–1091. AAAI Press, 2013.
9. Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
10. Reinhard Diestel. *Graph Theory, 5th Edition*, volume 173 of *Graduate Texts in Mathematics*. Springer, 2016.
11. David Garcia, Pavlin Mavrodiev, and Frank Schweitzer. Social resilience in online communities: The autopsy of friendster. In *Proceedings of the 1st ACM Conference on Online Social Networks (COSN '13)*, pages 39–50. ACM, 2013.
12. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.
13. Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *Journal of Computer and System Sciences*, 72(8):1386–1396, 2006.
14. Yoichi Iwata. Linear-Time Kernelization for Feedback Vertex Set. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 68:1–68:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2017.68.

15 Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

16 Tomasz Kociumaka and Marcin Pilipczuk. Faster deterministic feedback vertex set. *Information Processing Letters*, 114(10):556–560, 2014.

17 Daniel Lokshtanov, MS Ramanujan, and Saket Saurabh. Linear Time Parameterized Algorithms for Subset Feedback Vertex Set. *ACM Transactions on Algorithms (TALG)*, 14(1):7, 2018.

18 Junjie Luo, Hendrik Molter, and Ondřej Suchý. A Parameterized Complexity View on Collapsing $k$-Cores. *CoRR*, 2018. `arXiv:1805.12453`.

19 Fragkiskos D. Malliaros and Michalis Vazirgiannis. To stay or not to stay: modeling engagement dynamics in social graphs. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM '13)*, pages 469–478. ACM, 2013.

20 Luke Mathieson. The parameterized complexity of editing graphs for bounded degeneracy. *Theoretical Computer Science*, 411(34-36):3181–3187, 2010.

21 Stephen B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269–287, 1983.

22 Shuichi Ueno, Yoji Kajitani, and Shin'ya Gotoh. On the nonseparating independent set problem and feedback set problem for graphs with no vertex degree exceeding three. *Discrete Mathematics*, 72(1-3):355–360, 1988.

23 Xin Wang, Roger Donaldson, Christopher Nell, Peter Gorniak, Martin Ester, and Jiajun Bu. Recommending Groups to Users Using User-Group Engagement and Time-Dependent Matrix Factorization. In *Proceedins of the 30th AAAI Conference on Artificial Intelligence (AAAI '16)*, pages 1331–1337. AAAI Press, 2016.

24 Shaomei Wu, Atish Das Sarma, Alex Fabrikant, Silvio Lattanzi, and Andrew Tomkins. Arrival and departure dynamics in social networks. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining (WSDM '13)*, pages 233–242. ACM, 2013.

25 Fan Zhang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. Finding Critical Users for Social Network Engagement: The Collapsed $k$-Core Problem. In *Proceedins of the 31st AAAI Conference on Artificial Intelligence (AAAI '17)*, pages 245–251. AAAI Press, 2017.

# Parameterized Complexity of Multi-Node Hubs

**Saket Saurabh**
University of Bergen, Norway, and Institute of Mathematical Science, HBNI, India
saket@imsc.res.in

**Meirav Zehavi**
Ben-Gurion University, Israel
meiravze@bgu.ac.il

## Abstract

Hubs are high-degree nodes within a network. The examination of the emergence and centrality of hubs lies at the heart of many studies of complex networks such as telecommunication networks, biological networks, social networks and semantic networks. Furthermore, identifying and allocating hubs are routine tasks in applications. In this paper, we do not seek a hub that is a single node, but a hub that consists of $k$ nodes. Formally, given a graph $G = (V, E)$, we a seek a set $A \subseteq V$ of size $k$ that induces a connected subgraph from which at least $p$ edges emanate. Thus, we identify $k$ nodes which can act as a unit (due to the connectivity constraint) that is a hub (due to the cut constraint). This problem, which we call Multi-Node Hub (MNH), can also be viewed as a variant of the classic Max Cut problem. While it is easy to see that MNH is W[1]-hard with respect to the parameter $k$, our main contribution is the first parameterized algorithm that shows that MNH is FPT with respect to the parameter $p$.

Despite recent breakthrough advances for cut-problems like Multicut and Minimum Bisection, MNH is still very challenging. Not only does a *connectivity constraint* has to be handled on top of the involved machinery developed for these problems, but also the fact that MNH is a *maximization* problem seems to prevent the applicability of this machinery in the first place. To deal with the latter issue, we give non-trivial reduction rules that show how MNH can be preprocessed into a problem where it is necessary to delete a bounded-in-parameter number of vertices. Then, to handle the connectivity constraint, we use a novel application of the form of tree decomposition introduced by Cygan et al. [STOC 2014] to solve Minimum Bisection, where we demonstrate how *connectivity constraints* can be replaced by simpler *size constraints*. Our approach may be relevant to the design of algorithms for other cut-problems of this nature.

**2012 ACM Subject Classification** Theory of computation → Fixed parameter tractability

**Keywords and phrases** hub, bisection, tree decomposition

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.8

## 1 Introduction

Hubs are high-degree nodes within a network. The examination of the emergence and centrality of hubs lies at the heart of many studies of complex networks such as telecommunication networks, biological networks, social networks and semantic networks [25, 5, 1, 19, 35]. For example, hubs have been examined in the context of complex phylogenetic diseases such as asthma [33]; indeed, deletion of a hub protein is more likely to be lethal than deletion of a non-hub protein [21, 24]. Furthermore, identifying and allocating hubs are routine tasks in a wide-variety of applications [6]. In fact, the concept of hubs is also of significant interest in studies of hyperlinked environments [27], viral marketing [30], outbreaks of epidemics [4] and the Blogosphere [41].

© Saket Saurabh and Meirav Zehavi;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 8; pp. 8:1–8:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

It is natural to ask whether one can not only efficiently find a hub that is merely a single node, but a hub that consists of at most $k$ nodes or (more generally) exactly $k$ nodes.[1] More precisely, given a threshold parameter $p$, a hub can be defined as a node of degree at least $p$. Here, we introduce the concept of a *multi-node hub*, which is a connected set of nodes of degree at least $p$. In other words, we aim to identify $k$ nodes which can act as a unit (due to the connectivity constraint) that is a hub (due to the cut constraint). For example, in the context of biological networks, we might want to seek a module or a protein complex that acts as a hub, and in the context of social networks, we might want to identify a group of people that together act as a hub. Formally, we solve the following problem.

| Multi-Node Hub (MNH) **Parameters:** $k, p$ |
|---|
| **Input:** An undirected graph $G$ such that $\lvert V(G)\rvert = n$, and positive integers $k$ and $p$. |
| **Question:** Does there exist a subset $A \subseteq V(G)$ of size exactly $k$ such that $G[A]$ is connected, and $E(G)$ contains at least $p$ edges with exactly one endpoint in $A$? |

Multi-node hubs are of interest also for large values of $k$: when $k$ is close to $n$, a multi-node hub allows us to "shave" a few nodes from the network while ensuring that the core stays connected and has a large number of links to the nodes that were shaved. Moreover, we believe that multi-node hubs deserve to be studied also because they are simple, elegant combinatorial objects. The MNH problem involves three different types of constraints, namely a cut constraint, a size constraint and a connectivity constraint. On the one hand, it is easily seen to be MNH is W[1]-hard with respect to $k$; that is, it is unlikely that MNH is FPT with respect to $k$. On the other hand, it can be shown to be FPT with respect to $k+p$ (by a simple application of color-coding [3]). This brings us to the parameter $p$. By relying on a preprocessing phase inspired by algorithms for the Max Leaf Spanning Tree problem along with a novel application of special tree decompositions, namely "unbreakable tree decompositions", we are able to establish that MNH is FPT with respect to the parameter $p$.

**Related Work.** MNH is closely related to the classical NP-hard Max-Cut problem. Here, the input consists of a graph $G$ and a positive integer $p$, and the objective is to check whether there is a cut of size at least $p$. A cut of a graph is a partition of the vertices of the graph into two disjoint subsets. The size of the cut is the number of edges whose endpoints belong to different subsets of the partition. Max-Cut has been the focus of extensive study from the algorithmic perspective of computer science as well as the extremal perspective of combinatorics. Papadimitriou and Yannakakis showed that Max-Cut is APX-hard [37]. It is also well known that given an arbitrary partition of the vertex-set of $G$, by repeatedly moving vertices from one subset of the partition to the other as long as the size of the cut increases, one obtains a cut of size at least $|E(G)|/2$. A breakthrough result by Goemans and Williamson [20] gave a 0.878-approximation algorithm, which is optimal under the Unique Games Conjecture [26]. Max-Cut has also been well studied from the viewpoint of parameterized complexity [12, 13, 34, 38]. In this context, a notable work is the parameterized algorithm for an above-guarantee version of Max-Cut [13].

The $(k, n-k)$-Max Cut problem, which is a variant of Max Cut that has been extensively studied in the literature, is linked even more tightly to MNH. This variant asks whether there exists a subset $A$ of the vertex-set of a given graph $G = (V, E)$ such that

[1] If we solve the problem with the condition "exactly", then we solve the version with the condition "at most" by trying every option to choose $k$. The condition "exactly" is of use when we have $k$ tokens that should be placed on distinct nodes, which together should act as a hub.

$|A| = k$ and $E$ contains at least $p$ edges with exactly one endpoint in $A$. For this specific variant, Ageev and Sviridenko gave a 0.5-approximation algorithm [2], which has been slightly improved by Feige and Langberg [17]. It is also known that this variant is W[1]-hard with respect to the parameter $k$ [9]. With respect to the parameter $p$, $(k, n-k)$-Max Cut can be solved in time $\mathcal{O}^*(p^p)$ [8]. This bound was improved to $\mathcal{O}^*(4^{p+o(p)})$ in [40], and to $\mathcal{O}^*(2^p)$ in [39]. The latter work also gives a polynomial kernel for $(k, n-k)$-Max Cut with respect to the parameter $p$. Bonnet et al. [8] also gave a parameterized approximation scheme for $(k, n-k)$-Max Cut with respect to the parameter $k$.

Furthermore, Connected Max-Cut (CMC), the variant of Max Cut where $G[A]$ should be a connected graph, has also been investigated in the literature. This variant is of interest, for example, in image segmentation [42]. Essentially, MNH can be defined as the variant of $(k, n-k)$-Max Cut that adopts the connectivity constraint of CMC. Hajiaghayi et al. [23] studied CMC from the view point of approximation, and obtained an $\Omega(\frac{1}{\log n})$-approximation algorithm for this problem. They explicitly note that since CMC has the flavors of both cut and connectivity problems simultaneously, well-known approaches used to develop algorithms for either connectivity problems or cut problems such as Minimum Bisection are *not* applicable to CMC. Recently, Lee et al. [29] developed a 0.5-approximation algorithm for the special case of CMC where the input graph has bounded treewidth. We remark that the study of the parameterized complexity of natural connectivity variants of well-known problems, such as Connected Vertex Cover [7, 14, 16, 18, 22, 36], is an active research area that has received considerable attention.

**Our Contribution.** In this paper, we initiate the study of multi-node hubs in general, and of algorithmic aspects of MNH in particular. Since the concept of multi-node hubs is a natural generalization of the concept of hubs, it can be of significant interest in various studies of complex networks where hubs play a central role. We believe that connectivity is the most fundamental condition to demand from multi-node hubs; in particular, it is the most basic requirement that can allow the nodes to act as a single unit. In specialized scenarios, one may want the hub to be "highly-connected" or to be structured in a manner compatible with some application-specific demands. We believe that our paper can lead to several follow-up works, not only from the viewpoint of Parameterized Complexity, on (possibly enriched) multi-node hubs.

While it is easy to observe that MNH is W[1]-hard with respect to the parameter $k$, our main contribution is the first parameterized algorithm that shows that MNH is FPT with respect to $p$ (Section 4). Our algorithm is primarily a classification result and should be viewed as a proof of concept that the problem is FPT with respect to $p$. As a corollary to our algorithm we also get that CMC is FPT with respect to $p$.

Despite the recent breakthrough advances for cut-problems like Multicut and Minimum Bisection, MNH is still very challenging. Not only does a *connectivity constraint* has to be handled on top of the involved machinery developed for these problems, but also the fact that MNH is a *maximization* problem seems to prevent the applicability of this machinery in the first place. To deal with the latter issue, we give non-trivial reduction rules that show how MNH can be preprocessed into a problem where it is necessary to delete a bounded-in-parameter number of vertices. Then, to handle the connectivity constraint, we use a novel application of the form of tree decomposition introduced by Cygan et al. [15] to solve Minimum Bisection, where we demonstrate how *connectivity constraints* can be replaced by simpler *size constraints*. A more detailed description of this approach can be found in Section 3. We believe that our approach can be relevant to the design of algorithms for other cut-problems of this nature.

## 2 Preliminaries

**Tree Decompositions.** A *tree decomposition* of a graph $G$ is a pair $(D, \beta)$, where $D$ is a rooted tree and $\beta : V(D) \to 2^{V(G)}$ is a mapping that satisfies the following conditions.

- For each vertex $v \in V(G)$, the set $\{d \in V(D) : v \in \beta(d)\}$ induces a nonempty and connected subtree of $D$.
- For each edge $\{v, u\} \in E(G)$, there exists $d \in V(D)$ such that $\{v, u\} \subseteq \beta(d)$.

The set $\beta(d)$ is called the *bag at* $d$, while sets $\beta(d) \cap \beta(t)$ for $\{d, t\} \in E(D)$ are called *adhesions*. Furthermore, given a node $d \in V(D)$, we denote $\sigma(d) = \beta(d) \cap \beta(\text{parent}(d))$ in case $d$ is not the root, and $\sigma(d) = \emptyset$ otherwise. We also denote $\gamma(d) = (\bigcup_{t \in \mathsf{des}(d)} \beta(t)) \cup \beta(d)$.

**Unbreakability.** Given a graph $G$, we say that a pair $(A, B)$ such that $A \cup B = V(G)$ is a *separation* if $E(A \setminus B, B \setminus A) = \emptyset$. The *order* of the separation is $|A \cap B|$.

Roughly speaking, a vertex-set $U \subseteq V(G)$ is unbreakable in the graph $G$ if every separation of $V(G)$ of a small order must contain a large chunk of $U$ in one of its two sets. Intuitively, this means that the set $U$ is "highly-connected": any attempt to "break" it severely by using a separation of a small order is futile. Formally, an unbreakable set is defined as follows.

▶ **Definition 1.** Given a graph $G$, a set $U \subseteq V(G)$ is $(q, p)$*-unbreakable in* $G$ if for any separation $(A, B)$ of order at most $p$, we have that $|(A \setminus B) \cap U| \leq q$ or $|(B \setminus A) \cap U| \leq q$.

A $(q, p)$*-unbreakable tree decomposition*, a concept introduced in [15], is a tree decomposition which satisfies several properties relating to the unbreakability and connectendness of the "parts" to which it partitions the graph. These properties will be extremely valuable in Section 4, where we solve the MNH problem. In particular, such a decomposition consists of $(q, p)$-unbreakable bags; thus, for example, if we remove at most $p$ vertices from the graph, and we want to ensure that there remains a path between two vertices that belong to the same bag, it is sufficient to ensure that each of these vertices is contained in a large enough connected component (consisting of vertices from the bag). Formally, an unbreakable tree decomposition is defined as follows.

▶ **Definition 2.** We say that a tree decomposition $(D, \beta)$ of a graph $G$ is $(q, p)$*-unbreakable* if $|V(D)| \leq |V(G)|$ and the following conditions hold.

1. For each $d \in V(D)$, the graph $G[\gamma(d) \setminus \sigma(d)]$ is connected and $N(\gamma(d) \setminus \sigma(d)) = \sigma(d)$.
2. For each $d \in V(D)$, the set $\beta(d)$ is $(q, p)$-unbreakable in $G[\gamma(d)]$.
3. For each non-root $d \in V(D)$, we have that $|\sigma(d)| \leq q$ and $\sigma(d)$ is $(2p, p)$-unbreakable in $G[\gamma(\mathsf{parent}(d))]$.

Cygan *et al.* [15] gave an efficient computation of unbreakable tree decompositions, summarized in the result below.

▶ **Theorem 3.** *For some constant $c > 0$, there is an $\mathcal{O}^*(2^{\mathcal{O}(p^2)})$-time algorithm that, given a connected graph $G$, computes a $(2^{cp}, p)$-unbreakable tree decomposition of $G$.*

## 3 Our Technique

To handle the cut constraint ($|E(A, V(G) \setminus A)| \geq p$), we employ the randomized contractions technique [10], for which an excellent high-level description is given in Section 1.1 of [11]. Since our problem also involves a size constraint ($|A| = k$), we also rely on Theorem 3 (Section 2). In short, Cygan et al. [15] showed how to replace the recursion scheme of

the randomized contractions technique with a dynamic programming computation over a static tree decomposition; this enabled a possibility for a DP state to depend on a few counters (like sizes of parts in the case of Minimum Bisection of [15]), unachievable in the pure randomized contractions framework. The randomized contractions framework, regardless of whether one uses the original recursion scheme or the tree decomposition of [15], always requires significant technical work, but in most so-far known cases, this work is not particularly novel – it follows the bumpy but well-paved way of [10]. This is not the case in the study of the MNH problem.

First, note that the MNH problem asks for a cut of size *at least* $p$, whereas all known uses of randomized contractions focused on cuts of size bounded-in-parameter. Thus, directly, it seems that randomized contractions or the decomposition of [15] would not give much. Our first main insight is that if $n \geq k + p$, then a subgraph being a small enough tree with at least $p$ leaves witnesses a YES-instance, in which case the problem becomes similar to the Max Leaf Spanning Tree problem [28], and can be solved using the method of bounded search trees. Thus we are left with the case where $|V(G) \setminus A| \leq p$, that is, we are to delete bounded-in-parameter number of vertices from the graph (delete $V(G) \setminus A$), such that the graph stays connected, but at least $p$ edges "stick out" of the remaining graph. Now we think of the set $V(G) \setminus A$ as the "cut-set" for the randomized contractions technique, and finally we can start up the entire machinery. To this end, we give one more important problem-specific insight that greatly simplifies the use of randomized contractions: once we know $n < k + p$, then it is easy to check if a vertex $v$ of degree $> 2p$ can be put outside $A$ – any set $A$ of size $k$ not containing $v$ needs to have $\geq p$ edges that touch $v$. Thus, after easy filtering, we can put all vertices of degree $> 2p$ into $A$. By replacing each of these vertices by a binary tree, we obtain a graph of degree bounded by $2p$. This degree bound makes the use of "flower cuts" (see [10]) unnecessary; intuitively, it makes the picture much closer to edge-cuts than vertex-cuts.

At this point, we finally employ the special form of tree decomposition of Cygan et al. [15]. We use this decomposition in a non-standard manner, which relies on a twist of the understanding of the meaning of a cut-set in the randomized contractions framework, that may be of independent interest. In particular, we demonstrate how this decomposition can be used to tackle difficult *connectivity constraints* by replacing them by simpler *size constraints*: since each bag in the tree decomposition is a "highly-connected" subgraph of $G$ (see Section 2), removing a few vertices from $G$ cannot create two (separate) connected components that contain many vertices from the bag. Thus, instead of using direct computations to ensure the existence of paths between certain vertices, which may be very inefficient in case the paths are long, we can sometimes merely verify that the vertices are contained in large connected components.

Each node of the tree decomposition will be associated with a problem reflecting the "work" that should be performed when handling its bag. The definition is quite technically involved, since it is has to capture *both connectivity and cut constraints*, and since it has to incorporate information provided by solutions for problems associated with the bags of the children (which also capture both connectivity and cut constraints), while providing enough information that will next allow us to solve the problem associated with the bag of the parent. By proving several observations that aim to bound the sizes of "problematic" sets of vertices (e.g., vertices in the "surroundings" of vertices that should not be put in $A$), we can develop a procedure that solves the problem associated with a single bag, based on the classic color coding method [3]. More precisely, by relying on the properties of an unbreakable tree decomposition as well as the fact that $n < k + p$ and $\Delta_G < 2p$, we implicitly

show that only for a bounded-in-parameter number of the problems associated with the bags of the children we may not simply take all of the vertices of the graph, and that it is sufficient to classify correctly (via color coding) only a bounded-in-parameter number of vertices in the "surroundings" of vertices that should not be put in $A$. We also bound the number of vertices that surround these "surroundings", and thus, essentially, we obtain connected components that can be handled in a partially independent manner that relies on dynamic programming. The solutions to the problems associated with the individual bags are combined by using a somewhat technically involved procedure that is also based on dynamic programming.

## 4 Multi-Node Hub is FPT with Respect to $p$

In this section we prove our main result:

▶ **Theorem 4.** *The* MNH *problem is solvable in time* $\mathcal{O}^*(2^{2^{\mathcal{O}(p)}})$.

We first solve simple cases of MNH (Section 4.1), where either the graph $G$ contains many vertices (more precisely, $n \geq k + p$) or there exists a solution $A$ such that $V(G) \setminus A$ contains a vertex of high-degree. Then, we turn to focus on the difficult instances whose handling relies on the properties of an unbreakable tree decomposition (Section 4.2).

### 4.1 The Preprocessing Phase

We first handle instances where $G$ contains many vertices (Lemma 6). Then, we handle the remaining instances to which there is a solution $A$ that excludes a high-degree vertex (Lemma 7). Thus, we overall define special instances (called useful instances) of a new problem (called Special MNH (SMNH)), which encapsulates the difficult instances of MNH (Lemma 8).

Assume WLOG that $G$ is a connected graph. To solve instances where $n \geq k + p$, we need the following notation. Given a rooted tree $T$, $\mathsf{leaves}(T)$ and $\mathsf{internal}(T)$ denote the leaf-set and the set of internal vertices of $T$, respectively. Given two trees, $T$ and $T_e$, we say that $T_e$ *extends* $T$ if $T$ is a subtree of $T_e$. Observe that if $T_e$ extends $T$, then the number of leaves of $T_e$ is larger or equal to the number of leaves of $T$. This leads us to the next simple observation.

▶ **Observation 5.** *If a graph $G$ on at least $k + p$ vertices contains a subtree $T$ with at most $k$ internal vertices and at least $p$ leaves, then $(G, k, p)$ is a yes-instance of* MNH.

**Proof.** Recall that $G$ is a connected graph. Thus, since $n \geq k+p$, as long as $|\mathsf{internal}(T)| < k$ and $|V(T)| < k+p$, we can extend $T$ to a tree $T_e$ such that $|V(T_e)| = |V(T)| + 1$. At the end of this process, we obtain a tree $T'$ with at least $p$ leaves (since the original tree $T$ had at least $p$ leaves) such that $|\mathsf{internal}(T')| \leq k$ and $|V(T')| \geq k + p$. We let $A = \mathsf{internal}(T') \cup U$ where $U$ is an arbitrarily chosen subset of $\mathsf{leaves}(T')$ of size exactly $k - |\mathsf{internal}(T')|$. Thus, we obtain a solution $A$ to the instance $(G, k, p)$ of MNH. ◀

We are now ready to handle instances where $n \geq k + p$.

▶ **Lemma 6.** MNH *restricted to instances where $n \geq k + p$ is solvable in time* $\mathcal{O}^*(2^{\mathcal{O}(p)})$.

**Proof.** Let $(G, k, p)$ be an instance of MNH where $n \geq k + p$. The main reason why such an instance is simple lies in Observation 5, which implies that once we find a tree with a small number of internal vertices and a large number of leaves, we are done. This observation is exploited by our bounded search tree-based procedure, called MNH. During its execution, we maintain a tree $T$, and make "difficult" decisions (that is, decisions made by using branching

rather than reduction rules) relating to the insertion of vertices into a solution $A$ when handling vertices that should next be internal vertices with at least two children or fixed as leaves in the tree $T$. Since a tree with at most $p$ leaves has at most $p$ internal vertices with at least two children, Observation 5 bounds the number of difficult decisions we need to make (along each branch of the search tree) before we can conclude whether there is a solution.

Each call to $\mathsf{MNH}$ has the form $\mathsf{MNH}(G, k, p, T, O, I)$, where $T$ is a rooted subtree of $G$, $O \subseteq \mathsf{leaves}(T)$, and $I \subseteq \mathsf{leaves}(T) \setminus O$ such that $N_G(I) \subseteq V(T)$. Informally, $\mathsf{internal}(T) \cup I$ contains the vertices we decided to insert into the solution $A$ that we attempt to construct, while $O$ contains those we decided to leave outside $A$. We further ensure that for every vertex $v \in \mathsf{internal}(T)$, $T$ contains the entire set $N_G(v)$ (i.e., $N_G(v) \subseteq V(T)$). In other words, given a vertex which we decided to insert into $A$ (by inserting it into either $I$ or $\mathsf{internal}(T)$), we can assume that all of its neighbors belong to $T$. That is, we rely on the following assumption:

$$\boxed{\text{For every vertex } v \in I \cup \mathsf{internal}(T), \text{ we have that } N_G(v) \subseteq V(T).}$$

Since $T$, $O$ and $I$ are the only arguments changed during the execution, we use the abbreviation $\mathsf{MNH}(T, O, I)$. The goal of such a call is to accept if there is a solution $A$ to the instance $(G, k, p)$ of MNH such that $\mathsf{internal}(T) \cup I \subseteq A$ and $O \cap A = \emptyset$, and to reject if there is no solution $A$ to this instance (in the other cases, we can either accept or reject). Clearly, we can use $\mathsf{MNH}$ to solve MNH in the following manner. We call it with every subtree $T$ of $G$ that consists of exactly one internal vertex along with all of its neighbors as children (which will be leaves), and $O = I = \emptyset$. Then, we accept if and only if at least one of the calls to $\mathsf{MNH}$ accepts.

To analyze the rules of $\mathsf{MNH}$, we use the measure $3p - |E(\mathsf{internal}(T) \cup I, O)| - |\mathsf{leaves}(T)| - |I \cup O|$. The necessity of incorporating $|E(\mathsf{internal}(T) \cup I, O)|$ in the measure will be clear at the analysis of the branching vector of our last rule. Initially, the measure is at most $3p$. At the latest, when the measure drops to a non-positive value, we will ensure that $\mathsf{MNH}$ returns an answer in polynomial time. We will also ensure that the branching rules performed by $\mathsf{MNH}$ are associated with branching vectors whose roots are bounded by a *fixed* $c$. Thus, we get that the overall running time is bounded by $\mathcal{O}^*(2^{\mathcal{O}(p)})$. Now, it remains to give the list of rules of $\mathsf{MNH}(T, O, I)$, and prove that they satisfy the above mentioned properties. We note that although our problem resembles the Max Leaf Spanning Tree problem [28], there are important differences between the two problems, which forces us to devise a different, somewhat technically involved, set of rules (which is, nevertheless, inspired by the approach of [28]). In particular, recall that our goal is *not* to accept if and only if $G$ contains a subtree with at least $p$ leaves – however, if we do find a *small enough* such subtree, we can terminate the execution of our procedure.

Due to lack of space, the list of rules is omitted. ◀

We are only left with the case $n < k + p$. Next, we show a lemma that will help us in assuming that the maximum degree of the input graph is bounded by $2p$.

▶ **Lemma 7.** *Given an instance $(G, k, p)$ of* MNH *where $n < k+p$, in polynomial time we can decide whether there exists a solution $A$ that does not contain $\widetilde{R} = \{v \in V(G) : |N(v)| \geq 2p\}$.*

**Proof.** Suppose there exists a solution $A$ such that $\widetilde{R} \setminus A \neq \emptyset$. Furthermore, let $v^* \in (\widetilde{R} \setminus A)$. Then, there is a connected component $C^*$ in $G[V \setminus \{v^*\}]$ on at least $k$ vertices. As long as $|V(C^*)| > k$, compute a tree that spans $C^*$ and remove from $C^*$ a vertex that is a leaf in this tree. At the end of this process, we obtain a connected subgraph $A$ of $G[V \setminus \{v^*\}]$ on exactly $k$ vertices. Since $n < k + p$ and $|N(v)| \geq 2p$, this subgraph contains at least $p$ vertices from $N(v)$. Therefore, $|E(V(A), V(G) \setminus V(A))| \geq p$. Thus, we get that $V(A)$ is a

solution to $(G, k, p)$ of the desired kind. Our argument shows that to decide whether there is a solution $A$ such that $\widetilde{R} \setminus A \neq \emptyset$, we can simply check whether there is a vertex $v \in \widetilde{R}$ such that $G \setminus \{v\}$ has a connected component of size at least $k$. This can be done in polynomial time. ◀

When we will solve difficult instances, it will be useful to assume that we handle graphs $G$ such that $\Delta_G < 2p$. To this end, we define the Special MNH (SMNH) problem as follows. The input $(G, k, p, R)$ consists of an instance $(G, k, p)$ of MNH and a subset $R \subseteq V(G)$. The goal is to decide if there is a solution $A$ to the instance $(G, k, p)$ of MNH such that $R \subseteq A$. We say that an instance $(G, k, p, R)$ for SMNH is *useful* if $n < k + p$ and $\Delta_G < 2p$. Next, we show that we can focus on solving useful instances of SMNH rather than instances of MNH.

▶ **Lemma 8.** *Given an instance $(G, k, p)$ of* MNH*, there is an $\mathcal{O}^*(2^{\mathcal{O}(p)})$-time algorithm that either solves $(G, k, p)$ or returns a useful instance $(G', k', p, R)$ of* SMNH *such that (1) $|V(G')| = \mathcal{O}(|E(G)|)$, and (2) $(G, k, p)$ is a yes-instance if and only if $(G', k', p, R)$ is a yes-instance.*

**Proof.** Let $\widetilde{R} = \{v \in V(G) : |N_G(v)| \geq 2p\}$. By Lemmas 6 and 7, we can assume that (a) $n < k + p$, and (b) there is no solution $A$ for $(G, k, p)$ such that $\widetilde{R} \setminus A \neq \emptyset$. For each vertex $v \in \widetilde{R}$, let $T_v$ be a binary tree on $|N_G(v)|$ leaves and $|N_G(v)| - 1$ internal vertices. While $G$ contains a vertex $v \in \widetilde{R}$, insert $T_v$ into $G$ such that each leaf of $T_v$ is connected by an edge to a distinct vertex in $N_G(v)$, and then remove $v$ (and the edges containing $v$) from $G$. Let the resulting graph be $G'$. Clearly, $\Delta_{G'} < 2p$ and property (1) holds. Let $k' = |V(G')| - (n - k)$, and $R = \bigcup_{v \in \widetilde{R}} V(T_v)$. By assumption (a), $|V(G')| = k' + (n - k) < k' + p$, and thus we conclude that $(G', k', p, R)$ is useful. By assumption (b) and our definition of $G'$, $k'$ and $R$, $A$ is a solution for $(G, k, p)$ if and only if $(A \setminus \widetilde{R}) \cup R$ is a solution for $(G', k', p, R)$. Thus, property (2) holds. ◀

## 4.2 An Algorithm for the Difficult Instances

First, we briefly discuss some of the details of the dynamic programming computation that combines solutions to problems associated with individual bags (Section 4.2.1). Then, we formally define the problem associated with a single bag (Section 4.2.2). Due to lack of space, the algorithm that solves it is omitted.

### 4.2.1 Combining Individual Bags (Overview)

Let $(G, k, p, R)$ be a useful instance of SMNH (i.e., $n < k + p$ and $\Delta_G < 2p$). Recall that we assume WLOG that $G$ is a connected graph. Then, by Theorem 3, we can construct a $(2^{\mathcal{O}(p)}, p)$-unbreakable tree decomposition $(D, \beta)$ in time $\mathcal{O}^*(2^{\mathcal{O}(p^2)})$. We let $c$ be the constant such that $(D, \beta)$ is $(2^{cp}, p)$-unbreakable. Let $r \in V(D)$ be the root of $D$, and recall that $\sigma(r) = \emptyset$. As usual for tree decompositions, we will use dynamic programming. Unlike dynamic programming over normal tree decompositions where the size of the bag is bounded, we have that for all $d \in V(D)$, $\beta(d)$ is $(2^{cp}, p)$-unbreakable. Thus, the problem we need to solve even for one bag becomes a problem in itself.

**The Definition of $\mathcal{M}$.** At every node $d$ of the tree decomposition we keep a matrix (table) that stores how an optimal solution $A^*$ when restricted to $G[\gamma(d)]$ could potentially look like. Towards this we use a matrix $\mathcal{M}$ that has an entry $[d, T, q, P^*]$ for all $d \in V(D)$, $R \cap \beta(d) \subseteq T \subseteq (R \cap \beta(d)) \cup \sigma(d)$, $q \in \{\max\{0, k - |(V(G) \setminus \gamma(d)) \cup (T \cap \sigma(d))|\}, \ldots, \min\{|\gamma(d) \setminus$

$\sigma(d)|, k - |T \cap \sigma(d)|\}\}$, and $P^* \subseteq \{\{v, u\} : v, u \in \sigma(d) \cap T\}$. The set $T$ is used to guess the intersection of $A^*$ with $\sigma(d)$, and the edge-set $P^*$ is used to determine the partition of $T$ into connected components. That is, if we look at the graph, say $T^*$, induced by $T \cap \sigma(d)$ together with edges from $P^*$, we get a set of connected components, and the idea is that when we restrict $A^*$ to $G[\gamma(d)]$, say $A^*_d$, then every connected component of $A^*_d$ must contain exactly one connected component of $T^*$. In other words, we use $T^*$ to remember the connected components of $A^*_d$ in $G[\gamma(d)]$. Next, consider some entry $\mathcal{M}[d, T, q, P^*]$.

We say that a subset $A \subseteq \gamma(d)$ is $(d, T, q, P^*)$-*valid* if the following conditions hold.

1. $T \subseteq A$, $(\sigma(d) \setminus T) \cap A = \emptyset$ and $R \cap \gamma(d) \subseteq A$.
2. If $\sigma(d) \cap T \neq \emptyset$, then every connected component of $G[A]$ contains a vertex from $\sigma(d)$, and otherwise $G[A]$ is a connected graph.
3. For all $\{v, u\} \in P^*$: $G[A]$ has a connected component containing both $v$ and $u$.
4. $|A \setminus \sigma(d)| = q$.
5. If $A \neq \emptyset$, then $A \cap \beta(d) \neq \emptyset$.

For a $(d, T, q, P^*)$-*valid* set $A$, we define $\widetilde{p}(A) = |E(A, \gamma(d) \setminus A) \setminus E(A \cap \sigma(d), \sigma(d) \setminus A)|$. The entry $\mathcal{M}[d, T, q, P^*]$ is meant to store $\widetilde{p}_{d,T,q,P^*}$, the maximum value such that there exists a $(d, T, q, P^*)$-*valid* set $A$ satisfying $\widetilde{p}(A) = \widetilde{p}_{d,T,q,P^*}$.

Having computed $\mathcal{M}$, we can solve the useful instance $(G, k, p, R)$ of SMNH. Indeed, $(G, k, p, R)$ is a yes-instance if and only if a value that is at least $p$ is stored in at least one entry of the form $\mathcal{M}[d, T, k, \emptyset]$ where $\sigma(d) \cap T = \emptyset$. Since $\mathcal{M}$ contains $\mathcal{O}^*(2^{2^{\mathcal{O}(p)}})$ entries, it is sufficient to show that each entry of $\mathcal{M}$ can be computed in time $\mathcal{O}^*(2^{2^{\mathcal{O}(p)}})$.

**Constructing an Instance of Single Bag.** We compute the entries of $\mathcal{M}$ by using a postorder traversal; the order of computation between entries having the same argument $d$ is arbitrary. Now, consider some entry $\mathcal{M}[d, T, q, P^*]$, where it is possible that $d$ is a leaf, and assume that all of the preceding entries of $\mathcal{M}$ have been already computed correctly. To compute the entry $\mathcal{M}[d, T, q, P^*]$, we construct an instance $(H, k, p, T, U^*, P^*, \mathcal{U}, \mathsf{edges}, \mathsf{size}, \mathsf{cut})$ of a problem that we call Single Bag, and let $\mathcal{M}[d, T, q, P^*]$ store the result $W_q$ obtained by calling the algorithm we shall develop for Single Bag, which runs in the desired time $\mathcal{O}^*(2^{2^{\mathcal{O}(p)}})$. Here, $\mathsf{edges}, \mathsf{size}$ and $\mathsf{cut}$ are appropriate functions that are specified using the previously computed matrices at the children nodes. The parameters $k$ and $p$ are the same as those in the original instance of SMNH, and $T$ and $P^*$ are specified by the entry of $\mathcal{M}$. We let $H = G[\beta(d)]$ and $U^* = \sigma(d)$. Since $\Delta_G < 2p$, we have that $\Delta_H < 2p$. Moreover, we let $\mathcal{U}$ denote the *multiset* $\{\sigma(d') : d' \in \mathsf{children}_D(d)\}$. For each occurrence $U \in \mathcal{U}$, we let $d_U$ denote a different child such that $\sigma(d_U) = U$. The functions are defined as follows.

- For each $U \in \mathcal{U}$, $\mathsf{edges}(U)$ contains every edge-set in $2^{\{\{v,u\}:v,u \in U\}}$. This is used to represent how a partial solution is connected in the graph $G[\sigma(d_U)]$.
- For each $U \in \mathcal{U}$, let $\mathsf{size}(U) = |\gamma(d_U) \setminus \sigma(d_U)|$. Moreover, let $\mathsf{size}(U^*) = |V(G) \setminus \gamma(d)|$.
- $\mathsf{cut} : \{(U, A_U, E_U, s_U) : U \in \mathcal{U}, A_U \subseteq U, E_U \in \mathsf{edges}(U), s_U \in \{0, 1, \ldots, \mathsf{size}(U)\}\} \to \{-\infty, 0, 1, 2, \ldots\}$. The function $\mathsf{cut}$ assigns a value to every tuple $(U, A_U, E_U, s_U)$. Such a tuple represents a problem already solved that is associated with the bag $\beta(d_U)$. The tuple implies that in this problem, it is required that $A_U$ is contained in the partial solution, connectivity of the partial solution is similar to the one given by edges in $E_U$, and exactly $s_U$ vertices from $\gamma(d_U) \setminus \beta(d)$ are part of this partial solution. The value assigned by $\mathsf{cut}$ is the maximum number of edges that could emanate from a partial solution satisfying the connectivity and the size requirements given by $A_U, E_U$ and $s_U$.

### 4.2.2 The Problem Associated with a Single Bag

In this section we define the problem, called Single Bag, which reflects the "work" that should be performed when handling each bag individually.

**Input.** The input is of the form $(H, k, p, T, U^*, P^*, \mathcal{U}, \mathsf{edges}, \mathsf{size}, \mathsf{cut})$, where

1. $(H, k, p, T)$ is an instance of SMNH, where $\Delta_H < 2p$.
2. $U^* \subseteq V(H)$, $P^* \subseteq \{\{v, u\} : v, u \in U^* \cap T\}$.
3. $\mathcal{U}$ is a multiset of subsets of $V(H)$.
4. $\mathsf{edges}$ is a function that assigns to each set $U \in \mathcal{U}$ a subfamily of the family of edge-sets $2^{\{\{v,u\}:v,u\in U\}}$. The family $2^{\{\{v,u\}:v,u\in U\}}$ includes the edge-set $E^U_{\mathsf{com}} = \{\{v, u\} : v, u \in U\}$ such that the graph on the vertex-set $U$ and the edge-set $E^U_{\mathsf{com}}$ is a clique.
5. $\mathsf{size} : \{U^*\} \cup \mathcal{U} \to \{0, 1, \ldots, n\}$. Recall that $n = |V(G)|$.
6. $\mathsf{cut} : \{(U, A_U, E_U, s_U) : U \in \mathcal{U}, A_U \subseteq U, E_U \in \mathsf{edges}(U), s_U \in \{0, 1, \ldots, \mathsf{size}(U)\}\} \to \{-\infty, 0, 1, 2, \ldots\}$.

We now attempt to give informal explanations of the intuition underlying the choice of the arguments of the input. Roughly speaking, they relate to a bag as follows. Let $\beta(d)$ be the bag associated with the node $d$ of the tree decomposition that is currently being examined, $\beta(d^p)$ be the bag associated with the parent $d_p$, $\beta(d^c_1), \beta(d^c_2), \ldots, \beta(d^c_t)$ be the bags associated with the children, and $\gamma(d^c_i)$ be the union of $\beta(d^c_i)$ and the bags of the descendants of the child $d^c_i$ for all $i \in \{1, 2, \ldots, t\}$.

Recall that $G$ is the graph in the original instance of SMNH. In that context, we will let the graph $H$ be the subgraph of $G$ induced by the set of vertices in the current bag $\beta(d)$, i.e., $H = G[\beta(d)]$. The parameters $k$ and $p$ are the same as those in the original instance of SMNH, while $T$ may contain, in addition to $R$, vertices from $U^*$. The arguments $U^*, P^*$ and $\mathsf{size}(U^*)$ allow using solutions to the current problem (defined by $d$) when we next solve the problem defined by $d_p$, while the remaining arguments allow using solutions to the problems defined by $d^c_1, d^c_2, \ldots, d^c_t$ when solving the current problem. More precisely, $U^*$ is the adhesion between $d$ and $d^p$ (i.e., $U^* = \beta(d) \cap \beta(d^p)$), while each set $U \in \mathcal{U}$ is the adhesion between $\beta(d)$ and the child $\beta(d^c_i)$. The function $\mathsf{size}$ assigns $U^*$ the size of $V(G) \setminus \gamma(d)$ (which contains vertices yet to be handled), while it assigns each set $U \in \mathcal{U}$ the size of $\beta(d^c_i) \setminus U$ for the appropriate $i$ (which contains vertices already handled). The set $P^*$ contains pairs of vertices that should be connected in the solution for the current problem, to ensure the connectivity of the the graph induced by the solution $A$ that we ultimately attempt to construct (upon handling all of the bags). When we solve the problem associated with $d^p$, it might not be possible/worthy to connect these vertices, and thus we need to have a set that specifies connectivity requirements that must already be addressed.

For each set $U \in \mathcal{U}$, $\mathsf{edges}$ assigns a family of edge-sets. Exactly one edge-set can be added to the graph $H$ (see "Valid Triples"), and it is aimed to allow finding solutions that cannot be found without it (since a connectivity constraint will be broken). Each edge in such an edge-set represents a path that exists in a solution already computed (for a problem associated with a bag of a descendant). For example, when we next solve the problem associated with $d^p$, we will be able to use $P^*$ as such an edge-set. Note that each edge-set is associated with a certain value, specified by $\mathsf{cut}$, and thus it is not true that it is always best to choose a large, or even a non-empty, edge-set from $\mathsf{edges}(U)$. Finally, the function $\mathsf{cut}$ assigns a value to each tuple $(U, A_U, E_U, s_U)$. Such a tuple represents a problem already solved, associated with $d^c_i$ for the appropriate $i$ (in particular, $\beta(d) \cap \beta(d^c_i) = U$). The tuple implies that in this problem, it was required that $A_U$ is contained in the solution, paths

between every two vertices in $E_U$ are located, and exactly $s_U$ vertices from $\gamma(d_i^c) \setminus \beta(d)$ are inserted to the solution. The value assigned by $\mathsf{cut}$ is the number of edges between vertices in the solution (to the problem already solved) and the other vertices in $\gamma(d_i^c)$ (excluding edges between vertices in $U$).

**Restrictions.** Next, we denote $tSize = |V(H)| + \sum_{U \in \{U^*\} \cup \mathcal{U}} \mathsf{size}(U)$. For the sake of efficiency of our algorithm for Single Bag, we impose several restrictions on an input instance.

1. For all $U \in \mathcal{U} \cup \{U^*\}$: $|U| \leq 2^{cp}$, and each vertex in $V(H)$ is contained in at most $2p$ (not necessarily distinct) sets in the multiset $\mathcal{U}$.
2. $tSize = n < k + p$.
3. For all $(U, A_U, E_U, s_U) \in \mathsf{domain}(\mathsf{cut})$ such that $E_U \neq \{\{v, u\} : v, u \in U\}$, $U = A_U$ and $s_U = \mathsf{size}(U)$: $\mathsf{cut}(U, A_U, E_U, s_U) = -\infty$.

The first restriction bounds the size of sets that are part of an input instance, and thus it is clear that this restriction can potentially make the instance easier to solve efficiently. The second restriction will allow us to assume that there are not too many sets $U \in \mathcal{U}$ for which we may not use the default options $A_U = U$ and $s_U = \mathsf{size}(U)$, while the third restriction will overall allow us to conclude that there are not too many sets $U \in \mathcal{U}$ for which we may not use the default option $E_U = \{\{v, u\} : v, u \in U\}$. The claims are made precise in "Goal".

**Output.** For each $q \in \{\max\{0, k - \mathsf{size}(U^*) - |U^* \cap T|\}, \ldots, \min\{tSize - \mathsf{size}(U^*) - |U^*|, k - |U^* \cap T|\}\}$, we will need to find the "best" triple $(A, f, g)$ that satisfies certain conditions. Thus, we first work towards defining which triples are valid with respect to a given $q$.

**Valid Triples.** A triple $(A, f, g)$ is *potentially valid* if $T \subseteq A \subseteq V(H) \setminus (U^* \setminus T)$, $f$ is a function such that $f(U) \in \mathsf{edges}(U)$ for all $U \in \mathcal{U}$, and $g$ is a function such that $g(U) \in \{0, 1, \ldots, \mathsf{size}(U)\}$ for all $U \in \mathcal{U}$. We say that a set $U \in \mathcal{U}$ is *$A$-damaged*, *$f$-damaged* and *$g$-damaged* if $U \setminus A \neq \emptyset$, $f(U) \neq \{\{v, u\} : v, u \in U\}$ and $g(U) \neq \mathsf{size}(U)$, respectively. Roughly speaking, a set $U \in \mathcal{U}$ is $A$-damaged if we do not "take" all of its vertices, *$f$-damaged* if we do not "take" all edges between its vertices, and $g$-damaged if we do not "take" all of the vertices "below" $U$. Overall, a set $U \in \mathcal{U}$ is *damaged* if it is *$A$-damaged*, *$f$-damaged* or *$g$-damaged*. We will show below (in "Goal") that only few sets are damaged; the efficiency of our algorithm crucially relies on this observation, which essentially says that only few sets need "special attention". Moreover, we say a vertex $v \in V(H)$ is *damaged* if $v \in A$ and at least one of the two following conditions hold: (1) $v \in U$ for a damaged set $U$; (2) $v \in N_H(u)$ for a vertex $u \in V(H) \setminus A$. Finally, we define the *damage control* graph, $H(A, f)$, as the graph $H[A]$ to which we add the edges in $\bigcup_{U \in \mathcal{U}} f(U)$. Formally, $V(H(A, f)) = A$ and $E(H(A, f)) = E(H[A]) \cup (\bigcup_{U \in \mathcal{U}} f(U) \cap \{\{v, u\} : v, u \in A\})$. We say that a connected component is *huge* if it contains at least $2^{cp}$ vertices.

We are now ready to define which triples are valid with respect to a given $q \in \{\max\{0, k - \mathsf{size}(U^*) - |U^* \cap T|\}, \ldots, \min\{tSize - \mathsf{size}(U^*) - |U^*|, k - |U^* \cap T|\}\}$.

▶ **Definition 9.** A potentially valid $(A, f, g)$ is *$q$-valid* if the following conditions hold.

1. For each damaged vertex $v$ such that the connected component $C$ of $H(A, f)$ containing $v$ is not huge, at least one of the following conditions holds.
   a. $V(C) \cap (U^* \cap A) \neq \emptyset$.
   b. $|V(H)| \leq 2^{cp} + p$ and $H(A, f)$ is a connected graph.

2. If there exists a damaged vertex $v$ such that the connected component $C$ of $H(A, f)$ containing $v$ is huge, then at least one of the following conditions holds.
   a. $H(A, f)$ has a huge connected component $C'$ such that $V(C') \cap (U^* \cap A) \neq \emptyset$.
   b. $H(A, f)$ has no connected component that contains a damaged vertex and is not huge.
3. For all $\{v, u\} \in P^*$, either $H(A, f)$ has a connected component containing both $v$ and $u$ or each of its connected components containing $v$ or $u$ is huge.
4. $q = |A \setminus U^*| + \sum_{U \in \mathcal{U}} g(U)$.

Each $q$-valid triple is a potential solution for Single Bag. The first three conditions are related to the connectivity demand of a solution to MNH, and the fourth to the size demand. The *value* of a $q$-valid triple $(A, f, g)$ is

$$W(A, f, g) = |E(A, V(H) \setminus A) \setminus E(U^* \cap A, U^* \setminus A)| + \sum_{U \in \mathcal{U}} \mathsf{cut}(U, A \cap U, f(U), g(U)).$$

Informally, $W(A, f, g)$ is the sum of the number of edges between vertices in $A$ and vertices outside $A$ (excluding edges with both endpoints in $U^*$), to which we add the sum of the values of previously computed solutions.

**Goal.** For each $q \in \{\max\{0, k - \mathsf{size}(U^*) - |U^* \cap T|\}, \ldots, \min\{tSize - \mathsf{size}(U^*) - |U^*|, k - |U^* \cap T|\}\}$, we need to compute the maximum value $W_q$ for which there exists a $q$-valid triple $(A, f, g)$ such that $W_q = W(A, f, g)$. In case $W_q$ is not defined, since there does not exist a $q$-valid tuple, we set $W_q = -\infty$.

The following observation bounds the number of damaged sets in $\mathcal{U}$ and the number of damaged vertices corresponding to a $q$-valid triple. The main intuition is that if the current bag has several children then the size of the graph (which is less than $k + p$) forces a solution to select all vertices of all the subgraphs induced by the children, except $p$ of them. This observation is used to allow us to apply the color coding method efficiently. Due to lack of space, the proof is omitted.

▶ **Observation 10.** *If $(A, f, g)$ is a $q$-valid triple such that $W_q = W(A, f, g)$, then*
- *$|V(H) \setminus A| \leq p$ and $(A, f, g)$ damages at most $2p^2$ sets in $\mathcal{U}$,*
- *$(A, f, g)$ damages at most $2p^2 \cdot (2^{cp} + 1)$ vertices, and*
- *$g(U) \geq \mathsf{size}(U) - p$ for all $U \in \mathcal{U}$.*

## 5 Conclusion

In this paper, we initiated the study of multi-node hubs in general, and of algorithmic aspects of MNH in particular. The concept of multi-node hubs is a natural generalization of the concept of hubs, therefore it might play a central role in the analysis of complex networks. We showed that although MNH is W[1]-hard parameterized by $k$, it is FPT parameterized by $p$. Our algorithm is primarily a classification result and should be viewed as a proof of concept that the problem is FPT. While our approach may lead to an algorithm with running time $\mathcal{O}^*(2^{p^{\mathcal{O}(1)}})$ (or even $\mathcal{O}^*(2^{\mathcal{O}(p \log p)})$), developing an algorithm with running time $\mathcal{O}^*(2^{\mathcal{O}(p)})$ seems challenging. It may also be interesting to consider other natural restrictions on the set $A$ such as properties expressible in monadic second order logic, or to study MNH with respect to parameters other than $k$ and $p$.

To the best of our knowledge, besides our algorithm, only algorithms for Bisection and a problem called Judicious Partition [32] are based on the decomposition of [15]. Exploring the power of this decomposition more deeply might be fruitful; in this context, it is noteworthy to mention [31]. In our application, we showed how an instance of a maximization problem

can be preprocessed to a form that enables using known tools to solve cut-problems and in particular the decomposition of [15], as well as a manner in which connectivity constraints can be handled on top of cut and size constraints. We believe that our approach can be relevant to the design of algorithms for other cut-problems of this nature.

## References


1. L A Adamic, R M Lukose, A R Puniyani, and B A Huberman. Search in power-law networks. *CoRR cs.NI/0103016*, 2001.
2. A A Ageev and M Sviridenko. Approximation algorithms formaximum coverage andmax cutwith given sizes of parts. In *IPCO*, pages 17–30, 1999.
3. N Alon, R Yuster, and U Zwick. Color coding. *J. ACM*, 42(4):844–856, 1995.
4. D Balcan, H Hu, B Goncalves, P Bajardi, C Poletto, J J Ramasco, D Paolotti, N Perra, M Tizzoni, W Van den Broeck, V Colizza, and A Vespignani. Seasonal transmission potential and activity peaks of the new influenza A(H1N1): a Monte Carlo likelihood analysis based on human mobility. *BMC Medicine*, 7(45):29, 2009.
5. A Barabasi and R Albert. Emergence of scaling in random networks. *Science*, 286:509, 1999.
6. M Berlingerio, M Coscia, F Giannotti, A Monreale, and D Pedreschi. The pursuit of hubbiness: Analysis of hubs in large multidimensional networks. *J. Comput. Science*, 2:223–237, 2011.
7. D Binkele-Raible. Amortized analysis of exponential time and parameterized algorithms: Measure and conquer and reference search trees. *PhD Thesis, University of Trier*, 2009.
8. E Bonnet, B Escoffier, V T Paschos, and E Tourniaire. Multi-parameter analysis for local graph partitioning problems: using greediness for parameterization. *Algorithmica*, 71(3):566–580, 2015.
9. L Cai. Parameter complexity of cardinality constrained optimization problems. *Comput. J.*, 51(1):102–121, 2008.
10. R H Chitnis, M Cygan, M T Hajiaghayi, M Pilipczuk, and M Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. In *FOCS*, pages 460–469, 2012.
11. R H Chitnis, M Cygan, M T Hajiaghayi, M Pilipczuk, and M Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. In *http://arxiv.org/pdf/1207.4079v1.pdf*, 2012.
12. R Crowston, G Gutin, M Jones, and G Muciaccia. Maximum balanced subgraph problem parameterized above lower bound. *Theor. Comput. Sci.*, 513:53–64, 2013.
13. R Crowston, M Jones, and M Mnich. Max-Cut Parameterized Above the Edwards-Erdős Bound. *Algorithmica*, 72(3):734–757, 2015.
14. M Cygan. Deterministic parameterized connected vertex cover. In *SWAT*, pages 95–106, 2012.
15. M Cygan, D Lokshtanov, M Pilipczuk, M Pilipczuk, and S Saurabh. Minimum bisection is fixed parameter tractable. In *STOC*, pages 323–332, 2014.
16. M Cygan, J Nederlof, M Pilipczuk, M Pilipczuk, J M M van Rooij, and J O Wojtaszczyk. Deterministic parameterized connected vertex cover. In *FOCS*, pages 150–159, 2012.
17. U Feige and M Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *J. Algorithms*, 41(2):174–211, 2001.
18. H Fernau and D Manlove. Vertex and edge covers with clustering properties: Complexity and algorithms. *J. Discrete Algorithms*, 7(2):149–167, 2009.
19. D W Franks, J Noble, P Kaufmann, and S Stagl. Extremism propagation in social networks with hubs. *Adaptive Behavior – Animals, Animats, Software Agents, Robots, Adaptive Systems*, 16:264–274, 2008.

20 M X Goemans and D P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

21 P Goymer. Network biology: Why do we need hubs? *Nature Reviews Genetics*, 9:650–651, 2008.

22 J Guo, R Niedermeier, and S Wernicke. Parameterized complexity of generalized vertex cover problems. In *WADS*, pages 36–48, 2005.

23 M T Hajiaghayi, G Kortsarz, R MacDavid, M Purohit, and K K Sarpatwar. Approximation algorithms for connected maximum cut and telated problems. In *ESA*, pages 693–704, 2015.

24 X He and J Zhang. Why Do Hubs Tend to Be Essential in Protein Networks? *PLOS Genetics*, 2(6):e88, 2006.

25 H Jeong, B Tombor, R Albert, Z N Oltvai, and A L Barabasi. The large-scale organization of metabolic networks. *Nature*, 407:651–654, 2000.

26 S Khot, G Kindler, E Mossel, and R O'Donnell. Optimization, approximation, and complexity classes. *SICOMP*, 37(1):319–357, 2007.

27 J M Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46:604–632, 1999.

28 J Kneis, A Langer, and P Rossmanith. A new algorithm for finding trees with many leaves. *Algorithmica*, 61:882–897, 2011.

29 J Lee, V Nagarajan, and X Shen. Max-Cut Under Graph Constraints. In *IPCO*, pages 50–62, 2016.

30 J Leskovec, L A Adamic, and Huberman B A. The dynamics of viral marketing. In *EC*, pages 228–237, 2006.

31 D Lokshtanov, M S Ramanujan, S Saurabh, and M Zehavi. Reducing CMSO Model Checking to Highly Connected Graphs. In *ICALP*, pages 135:1–135:14, 2018.

32 D Lokshtanov, S Saurabh, R Sharma, and M Zehavi. Balanced Judicious Bipartition is Fixed-Parameter Tractable. In *FSTTCS*, pages 40:40–40:15, 2017.

33 X Lu, V J Vipul, P W Finn, and D L Perkins. Hubs in biological interaction networks exhibit low changes in expression in experimental asthma. *Molecular Systems Biology*, 3(98), 2008.

34 M Mahajan and V Raman. Parameterizing above Guaranteed Values: MaxSat and MaxCut. *J. Algorithms*, 31(2):335–354, 1999.

35 A S Maiya and T Y Berger-Wolf. Online sampling of high centrality individuals in social networks. In *PAKDD*, pages 91–98, 2010.

36 D Molle, S Richter, and P Rossmanith. Enumerate and expand: Improved algorithms for connected vertex cover and tree cover. *Theory Comput. Syst.*, 43(2):234–253, 2008.

37 C H Papadimitriou and M Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991.

38 V Raman and S Saurabh. Improved fixed parameter tractable algorithms for two "edge" problems: MAXCUT and MAXDAG. *Inf. Process. Lett.*, 104(2):65–72, 2007.

39 S Saurabh and M Zehavi. $(k, n-k)$-Max-Cut: an $O^*(2^p)$-time algorithm and a polynomial kernel. In *LATIN*, pages 686–699, 2016.

40 H Shachnai and M Zehavi. Parameterized algorithms for graph partitioning problems. In *WG*, pages 384–395, 2014.

41 X Shi, B Tseng, and L Adamic. Looking at the Blogosphere Topology through Different Lenses. In *ICWSM*, 2007.

42 S Vicente, V Kolmogorov, and C Rother. Graph cut based image segmentation with connectivity priors. In *CVPR*, pages 1–8, 2008.

# The Parameterised Complexity of Computing the Maximum Modularity of a Graph

**Kitty Meeks**[1]
School of Computing Science, University of Glasgow, Glasgow, UK
kitty.meeks@glasgow.ac.uk

**Fiona Skerman**[2]
Department of Mathematics, Uppsala University, Uppsala, Sweden
fiona.skerman@math.uu.se

## Abstract

The *maximum modularity* of a graph is a parameter widely used to describe the level of clustering or community structure in a network. Determining the maximum modularity of a graph is known to be NP-complete in general, and in practice a range of heuristics are used to construct partitions of the vertex-set which give lower bounds on the maximum modularity but without any guarantee on how close these bounds are to the true maximum. In this paper we investigate the parameterised complexity of determining the maximum modularity with respect to various standard structural parameterisations of the input graph $G$. We show that the problem belongs to FPT when parameterised by the size of a minimum vertex cover for $G$, and is solvable in polynomial time whenever the treewidth or max leaf number of $G$ is bounded by some fixed constant; we also obtain an FPT algorithm, parameterised by treewidth, to compute any constant-factor approximation to the maximum modularity. On the other hand we show that the problem is W[1]-hard (and hence unlikely to admit an FPT algorithm) when parameterised simultaneously by pathwidth and the size of a minimum feedback vertex set.

**2012 ACM Subject Classification** Mathematics of computing → Graph algorithms, Theory of computation → Parameterized complexity and exact algorithms

**Keywords and phrases** modularity, community detection, integer quadratic programming, vertex cover, pathwidth

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.9

**Acknowledgements** The authors are grateful to Jessica Enright for some helpful initial discussions about the topic.

## 1 Introduction

The increasing availability of large network datasets has led to great interest in techniques to discover network structure. An important and frequently observed structure in networks is the existence of groups of vertices with many connections between them, often referred to as 'communities'.

Newman and Girvan introduced the modularity function in 2004 [22]. Modularity gives a measure of how well a graph can be divided into communities and is used in the most

[1] KM is supported by a Royal Society of Edinburgh Personal Research Fellowship, funded by the Scottish Government.
[2] FS is supported by grants from Swedish Research Council and the Ragnar Söderberg Foundation. Research partially conducted while at the Heilbronn Institute for Mathematical Research, Bristol.

© Kitty Meeks and Fiona Skerman;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 9; pp. 9:1–9:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

popular algorithms to cluster large networks. For example, the Louvain method, an iterative clustering technique, uses the modularity function to choose which parts from the previous step to fuse into larger parts at each step [2, 15]. The widespread use of modularity and empirical success in finding communities makes modularity an important function to study from an algorithmic point of view.

In this paper we are concerned with the computational complexity of computing the maximum modularity of a given input graph, and specifically in the following decision problem.

> Modularity
> *Input:* A graph $G$ and a constant $q \in [0, 1]$.
> *Question:* Is the maximum modularity of $G$ at least $q$?

This problem was shown to be NP-complete in general by Brandes et. al. [3], using a construction that relies on the fact that all vertices of a sufficiently large clique must be assigned to the same part of an optimal partition. They also showed that a variation of the problem in which we wish to find the optimal partition into exactly two sets is hard; their proof for this relied again on the use of large cliques, but DasGupta and Desai [5] later showed that this 2-clustering problem remains NP-complete on $d$-regular graphs for any fixed $d \geq 9$. It has also been shown that it is NP-hard to approximate the maximum modularity within any constant factor [7], although there is a polynomial-time constant-factor approximation algorithm for certain families of scale-free networks [9]. The hardness of computing constant-factor multiplicative approximations in general has motivated research into approximation algorithms with an additive error [7, 16]: the best known result is an approximation algorithm with additive error roughly 0.42084 [16].

In this paper we initiate the study of the parameterised complexity of Modularity, considering its complexity with respect to several standard structural parameterisations. On the positive side, we show that the problem is in FPT when parameterised by the cardinality of a minimum vertex cover for the input graph $G$, and that it belongs to XP when parameterised by either the treewidth or max leaf number of $G$. The XP algorithm parameterised by treewidth can easily be adapted to give an FPT algorithm, parameterised by treewidth, to compute any constant-factor approximation maximum modularity. On the other hand, we demonstrate that Modularity, parameterised by treewidth, is unlikely to belong to FPT: we prove that the problem is W[1]-hard even when parameterised simultaneously by the pathwidth of $G$ and the size of a minimum feedback vertex set for $G$. For background on parameterised complexity, and the complexity classes discussed here, we refer the reader to [4, 10].

These results follow the same pattern as those obtained for the problem Equitable Connected Partition [11], and indeed our hardness result involves a reduction from a specialisation of this problem. There are clear similarities between the two problems: in a partition that maximises the modularity, every part will induce a connected subgraph and, in certain circumstances, we achieve the maximum modularity with a partition into parts that are as equal as possible. However, the crucial difference between the two problems is that the input to Equitable Connected Partition includes the required number of parts, whereas Modularity requires us to maximise over all possible partition sizes; in fact, if we restrict to partitions with a specified parts, it is no longer necessarily true that a partition maximising the modularity must induce connected subgraphs. This difference makes reductions between the two problems non-trivial.

## 1.1 The modularity function

The definition of modularity was first introduced by Newman and Girvan in [22]. Many or indeed most popular algorithms used to search for clusterings on large datasets are based on finding partitions with high modularity [17, 14], and the heuristics within them sometimes also use local modularity optimisation, for example in the Louvain method [2]. See [23, 13] for surveys on community detection including modularity based methods.

Knowledge on the maximum modularity for classes of graphs helps to understand the behaviour of the modularity function. There is a growing literature on this which began with cycles and complete graphs in [3]. Bagrow [1] and Montgolfier et al. [6] showed some classes of trees have high maximum modularity which was extended in [19] to all trees with maximum degree $o(n)$, and furthermore to all graphs where the product of treewidth and maximum degree grows more slowly than the number of edges. Many random graph models also have high modularity, see [20, 21] for a treatment of Erdős-Renyi random graphs, [19] for random regular graphs and also [24] which includes the preferential attachment model.

Given a set $A$ of vertices, let $e(A)$ denote the number of edges within $A$, and let $\mathrm{vol}(A)$ (sometimes called the volume of $A$) denote the sum of the degree $d_v$ (in the whole graph $G$) over the vertices $v$ in $A$. For a graph $G$ with $m \geq 1$ edges and a vertex partition $\mathcal{A}$ of $G$, set the modularity of $\mathcal{A}$ on $G$ to be

$$q_{\mathcal{A}}(G) = \frac{1}{2m} \sum_{\mathsf{A} \in \mathcal{A}} \sum_{u,v \in A} \left( \mathbf{1}_{uv \in E} - \frac{d_u d_v}{2m} \right) = \frac{1}{m} \sum_{\mathsf{A} \in \mathcal{A}} e(\mathsf{A}) - \frac{1}{4m^2} \sum_{\mathsf{A} \in \mathcal{A}} \mathrm{vol}(\mathsf{A})^2;$$

the maximum modularity of $G$ is $q^*(G) = \max_{\mathcal{A}}(G)$, where the maximum is over all partitions $\mathcal{A}$ of the vertices of $G$. Graphs with no edges are defined conventially to have modularity 0.

The modularity function is designed to score partitions highly when most edges fall within the parts and penalise partitions with very few or very big parts. These two objectives are encoded as the *edge contribution* or *coverage* $q^E_{\mathcal{A}}(G) = \frac{1}{m} \sum_{\mathsf{A} \in \mathcal{A}} e(\mathsf{A})$, and *degree tax* $q^D_{\mathcal{A}}(G) = \frac{1}{4m^2} \sum_{\mathsf{A} \in \mathcal{A}} \mathrm{vol}(\mathsf{A})^2$, in the modularity of a vertex partition $\mathcal{A}$ of $G$.

Note that for any graph with $m \geq 1$ edges $0 \leq q^*(G) < 1$. To see the lower bound, notice that the trivial partition which places all vertices in the same part has modularity zero. For example, complete graphs and stars have modularity 0 as noted in [3]. A graph consisting of $c$ disjoint cliques of the same size has modularity $1 - 1/c$ with the optimal partition taking each clique to be a part.

As modularity is at most 1 it is sometimes useful to consider the *modularity defect* $\tilde{q}_{\mathcal{A}}(G) = 1 - q_{\mathcal{A}}(G)$. Denote by $\partial(A)$ the number of edges between $A$ and the rest of the graph. Then

$$\tilde{q}_{\mathcal{A}}(G) = \frac{1}{2m} \sum_{A \in \mathcal{A}} \left( \partial(A) + \frac{\mathrm{vol}(A)^2}{2m} \right)$$

and we may equivalently minimise the modularity defect as maximise the modularity. In particular

$$\tilde{q}(G) = \min_{A \in \mathcal{A}} \tilde{q}_{\mathcal{A}}(G) = 1 - q^*(G).$$

We will make use of several facts about the maximum modularity of a graph.

▶ **Fact 1.1** (Lemma 3.3 of [3]). *If $\mathcal{A}$ is a partition of $V(G)$ such that $q_{\mathcal{A}}(G) = q^*(G)$ then no part $A$ consists of a single vertex of degree 1.*

▶ **Fact 1.2** (Lemma 3.4 of [3]). *Suppose that $G$ is a graph that contains no isolated vertices. If $\mathcal{A}$ is a partition of $V(G)$ such that $q_{\mathcal{A}}(G) = q^*(G)$ then, for every $A \in \mathcal{A}$, $G[A]$ is a connected subgraph of $G$.*

▶ **Fact 1.3** (Corollary 1 of [3]). *Let $G = (V, E)$ and suppose that $V_0 \subseteq V$ is a set of isolated vertices. Then $q(G) = q(G \setminus V_0)$. Moreover, if partitions $\mathcal{A}$ and $\mathcal{A}'$ agree on all vertices of $V \setminus V_0$, then $q_{\mathcal{A}}(G) = q_{\mathcal{A}'}(G)$.*

▶ **Fact 1.4** (Lemma 1 of [8],Lemma 2.1 of [5]). *For any integer $c > 0$ and any graph $G$,*

$$\max_{|\mathcal{A}| \leq c} q_{\mathcal{A}}(G) > q^*(G)\Big(1 - \frac{1}{c}\Big).$$

## 1.2 Notation and definitions

Given a graph $G = (V, E)$, and a set $U \subseteq V$ of vertices, we write $G[U]$ for the subgraph of $G$ induced by $U$ and $G \setminus U$ for $G[V \setminus U]$. Given two disjoint subsets of vertices $A, B \subseteq V$, we write $e(A, B)$ for the number of edges with one endpoint in $A$ and the other in $B$. We shall often want to denote the number of edges between a set of vertices and the remainder of the graph so set $\partial(A) = e(A, \bar{A})$. If $\mathcal{P}$ is a partition of a set $X$, and $Y \subset X$, we write $\mathcal{P}[Y]$ for the restriction of $\mathcal{P}$ to $Y$.

A *vertex cover* of a graph $G = (V, E)$ is a set $U \subseteq V$ such that every edge has at least one endpoint in $U$; equivalently, $G \setminus U$ is an independent set (i.e. contains no edges). The *vertex cover number* of $G$ is the smallest cardinality of any vertex cover of $G$. A *feedback vertex set* for $G$ is a set $U \subseteq V$ such that $G \setminus U$ contains no cycles. Notice that the vertex cover number of $G$ gives an upper bound on the size of the smallest feedback vertex set for $G$, written $\mathrm{fvs}(G)$. The *max leaf number* of $G$ is the maximum number of leaves (degree one vertices) in any spanning tree of $G$.

A *tree decomposition* of a graph $G$ is a pair $(T, \mathcal{D})$ where $T$ is a tree and $\mathcal{D} = \{\mathcal{D}(t) : t \in V(T)\}$ is a collection of non-empty subsets of $V(G)$ (or *bags*), indexed by the nodes of $T$, satisfying:

1. $V(G) = \bigcup_{t \in V(T)} \mathcal{D}(t)$,
2. for every $e = uv \in E(G)$, there exists $t \in V(T)$ such that $u, v \in \mathcal{D}(t)$,
3. for every $v \in V(G)$, if $T(v)$ is defined to be the subgraph of $T$ induced by nodes $t$ with $v \in \mathcal{D}(t)$, then $T(v)$ is connected.

If $T$ is in fact a path, we say that $(T, \mathcal{D})$ is a *path decomposition* of $G$. The *width* of the tree decomposition $(T, \mathcal{D})$ is defined to be $\max_{t \in V(T)} |\mathcal{D}(t)| - 1$, and the *treewidth* of $G$, written $\mathrm{tw}(G)$, is the minimum width over all tree decompositions of $G$. The *pathwidth* of $G$, $\mathrm{pw}(G)$, is the minimum width over all path decompositions of $G$.

# 2 Positive results

In this section we identify a number of structural restrictions on the input graph that allow us to compute the maximum modularity of a graph, or a good approximation to this quantity, efficiently.

## 2.1 Parameterisation by vertex cover number

In this section we demonstrate that Modularity is in FPT when parameterised by the vertex cover number of the input graph.

▶ **Theorem 2.1.** Modularity, *parameterised by cardinality of a minimum vertex cover for the input graph $G$, is in* FPT.

To prove this result, we make use of recent work of Lokshtanov [18] which gives an FPT algorithm for the following problem.

> Integer Quadratic Programming
> *Input:* An $n \times n$ integer matrix $Q$, an $m \times n$ integer matrix $A$, and an $m$-dimensional vector $\mathbf{b}$.
> *Parameter:* $n + \alpha$, where $\alpha$ is the maximum absolute value of any entry in $A$ or $Q$.
> *Problem:* Find a vector $\mathbf{x} \in \mathbb{Z}^n$ which minimises $\mathbf{x}^T Q\mathbf{x}$, subject to $A\mathbf{x} \leq \mathbf{b}$.

Before proving Theorem 2.1, we introduce some notation. Suppose that the graph $G = (V, E)$ has $|E| = m$, and that $U = \{u_1, \ldots, u_k\}$ is a vertex cover for $G$. Let $\mathcal{P} = \{P_1, \ldots, P_\ell\}$ be a partition of $U$, and set $W = V \setminus U$ (so $W$ is an independent set).

We can partition the vertices of $W$ into $2^k$ sets based on their *type*: the type $\tau_U(w) \in \{0,1\}^k$ of a vertex $w \in W$ describes which of the vertices in $U$ are neighbours of $w$. Formally $\tau_U(w)_j = 1$ if $u_j w \in E(G)$ and $\tau_U(w)_j = 0$ otherwise. For each $\sigma \in \{0,1\}^k$, we set $S_\sigma$ to be the set of all vertices in $W$ with type exactly $\sigma$, that is, $S_\sigma = \{w \in W : \tau(w) = \sigma\}$.

Now let $\mathcal{A} = \{A_1, \ldots, A_r\}$ be a partition of $V$. We write $x^{\mathcal{A}}_{\sigma,i}$ for the number of vertices of type $\sigma$ which are assigned to $A_i$, that is, $x^{\mathcal{A}}_{\sigma,i} = |S_\sigma \cap A_i|$. Finally, we introduce 0-1 vectors to encode the sets $P_i \in \mathcal{P}$: for $1 \leq i \leq \ell$, we let $\pi^i \in \{0,1\}^k$ be given by $\pi^i_j = 1$ if $u_j \in P_i$, and $\pi^i_j = 0$ otherwise.

We now argue that, if the partition $\mathcal{A}$ extends $\mathcal{P}$, we can compute the modularity of $\mathcal{A}$ using only the values $x^{\mathcal{A}}_{\sigma,i}$, together with information about $\mathcal{P}$. This shows that the modularity depends only on the number of vertices of each type assigned to a given partition, and not the assignment of individual vertices. The proof is omitted due to space constraints.

▶ **Lemma 2.2.** *Let $U = \{u_1, \ldots, u_k\}$ be a vertex cover for $G = (V, E)$, where $|E| = m$, and let $\mathcal{P}$ be a partition of $U$. If $\mathcal{A}$ is any partition of $V$ which extends $\mathcal{P}$ and has the property that every $A \in \mathcal{A}$ has non-empty intersection with $U$, then*

$$q^E_{\mathcal{A}}(G) = \frac{1}{m}\sum_{i=1}^{\ell} e(P_i) + \frac{1}{m}\sum_{(\sigma,i)} x^{\mathcal{A}}_{\sigma,i}(\sigma \cdot \pi^i),$$

*and*

$$4m^2 q^D_{\mathcal{A}'} = 4\sum_i e(P_i)^2 + 4\sum_{(\sigma,i)} x^{\mathcal{A}}_{\sigma,i} e(P_i)(\sigma \cdot (\mathbf{1} + \pi^i)) + \sum_{(\sigma,i)(\sigma',j)} x^{\mathcal{A}}_{\sigma,i} x^{\mathcal{A}}_{\sigma',j}(\sigma \cdot (\mathbf{1} + \pi^i))(\sigma' \cdot (\mathbf{1} + \pi^j)).$$

**Proof of Theorem 2.1.** We will assume that the input to our instance of Modularity is a graph $G = (V, E)$, where $|E| = m$. We may assume without loss of generality that we are also given as input a vertex cover $U = \{u_1, \ldots, u_k\}$ for $G$ (as if not we can easily compute one in the allowed time). We may further assume that $G$ does not contain any isolated vertices, as we can delete any such vertices (in polynomial time) without changing the value of the maximum modularity (by Fact 1.3).

Note that the total number of possible partitions of $U$ into non-empty parts is equal to the $k^{th}$ *Bell number*, $B_k$ (and hence is certainly less than $k^k$). It therefore suffices to describe an fpt-algorithm which determines, given some partition $\mathcal{P}$ of $U$,

$$q^{\mathcal{P}}(G) = \max\{q_{\mathcal{A}}(G) : \mathcal{A}[U] = \mathcal{P}\}.$$

The maximum modularity of $G$ can then be calculated by taking

$$\max\{q^{\mathcal{P}}(G) : \mathcal{P} \text{ is a partition of } U\}.$$

From now on, we consider a fixed partition $\mathcal{P} = \{P_1, \ldots, P_\ell\}$ of $U$, and describe how to compute $q^{\mathcal{P}}(G)$.

It follows from Facts 1.1 and 1.2, together with the fact that $W$ is an independent set that, if $\mathcal{A} = \{A_1, \ldots, A_j\}$ is a partition of $V$ which achieves the maximum modularity, then every part $A_i$ has non-empty intersection with $U$. We will call a partition with this properties a $U$-partition of $G$. It then suffices to maximise the modularity over all $U$-partitions in order to determine the value of $q^{\mathcal{P}}(G)$.

Now, by Lemma 2.2, we know that we can express the modularity of a $U$-partition $\mathcal{A}$ as

$$\begin{aligned} q_{\mathcal{A}}(G) = \frac{1}{m}\sum_{i=1}^{\ell} e(P_i) + \frac{1}{m}\sum_{(\sigma,i)} x^{\mathcal{A}}_{\sigma,i}(\sigma\cdot\pi^i) - \frac{1}{m^2}\sum_i e(P_i)^2 \\ - \frac{1}{m^2}\sum_{(\sigma,i)} x^{\mathcal{A}}_{\sigma,i} e(P_i)(\sigma\cdot(\mathbf{1}+\pi^i)) \\ - \frac{1}{4m^2}\sum_{(\sigma,i)(\sigma',j)} x^{\mathcal{A}}_{\sigma,i} x^{\mathcal{A}}_{\sigma',j}(\sigma\cdot(\mathbf{1}+\pi^i))(\sigma'\cdot(\mathbf{1}+\pi^j)). \end{aligned} \tag{1}$$

As we have fixed the partition $\mathcal{P}$, all values $e(P_i)$ can be regarded as fixed constants. In order to determine the maximum modularity we can obtain with a $U$-partition, we therefore need to find the values of $x^{\mathcal{A}}_{\sigma,i}$ which maximise this expression.

We can rewrite (1) as the sum of a constant term, two linear functions $\theta$ and $\phi$ of the $x^{\mathcal{A}}_{\sigma,i}$ and a quadratic function $\psi$ of the $x^{\mathcal{A}}_{\sigma,i}$ (up to scaling by constants):

$$\begin{aligned} q_{\mathcal{A}}(G) = & \underbrace{\frac{1}{m}\sum_{i=1}^{\ell} e(P_i) - \frac{1}{m^2}\sum_i e(P_i)^2}_{\text{constant}} \\ & + \frac{1}{m}\underbrace{\sum_{(\sigma,i)} x^{\mathcal{A}}_{\sigma,i}(\sigma\cdot\pi^i)}_{\theta(\mathcal{A})} - \frac{1}{m^2}\underbrace{\sum_{(\sigma,i)} x^{\mathcal{A}}_{\sigma,i}\, e(P_i)(\sigma\cdot(\mathbf{1}+\pi^i))}_{\phi(\mathcal{A})} \\ & - \frac{1}{4m^2}\underbrace{\sum_{(i,\sigma)(j,\sigma'} x^{\mathcal{A}}_{\sigma,i} x^{\mathcal{A}}_{\sigma',j}(\sigma\cdot(\mathbf{1}+\pi^i))(\sigma'\cdot(\mathbf{1}+\pi^j))}_{\psi(\mathcal{A})}. \end{aligned}$$

To find the maximum value of $q_{\mathcal{A}}(G)$ over all $U$-partitions it therefore suffices to determine, for all possible values of $\theta(\mathcal{A})$ and $\phi(\mathcal{A})$, the minimum possible value of $\psi(\mathcal{A})$. Before describing how to do this, we observe that the number of combinations of possible values for $\theta(\mathcal{A})$ and $\phi(\mathcal{A})$ and is not too large. Note that $0 \leq \sum_{\sigma,i} x^{\mathcal{A}}_{\sigma,i}(\sigma\cdot\pi^i) < nk$, and $0 \leq \sum_{\sigma,i} x^{\mathcal{A}}_{\sigma,i} e(P_i)(\sigma\cdot(\mathbf{1}+\pi^i)) < n\binom{k}{2}2k < nk^3$, so the number of possible pairs $(\theta(\mathcal{A}), \phi(\mathcal{A}))$

is at most $n^2k^4$. Thus, if we know the minimum possible value of $\psi(\mathcal{A})$ corresponding to each possible pair $(\theta(\mathcal{A}), \phi(\mathcal{A}))$, we can compute the maximum modularity achieved by any $U$-partition $\mathcal{A}$ such that $(\theta(\mathcal{A}), \phi(\mathcal{A})) = (y, z)$, and maximising over the polynomial number of possible pairs $(y, z)$ will give $q^{\mathcal{P}}(G)$.

Now, given a possible pair of values $(y, z)$ for $(\theta(\mathcal{A}), \phi(\mathcal{A}))$, we describe how to compute

$$\min\{\psi(\mathcal{A}) : \mathcal{A} \text{ is a } U\text{-partition with } \theta(\mathcal{A}) = y \text{ and } \phi(\mathcal{A}) = z\}.$$

Our strategy is to express this minimisation problem as an instance of Integer Quadratic Programming and then apply the FPT algorithm of [18].

In this instance, we have $n = \ell 2^k \leq k2^k$, and our vector of variables $\mathbf{x} = (x_1, \ldots, x_n)^T$ is given by

$$x_i = x^{\mathcal{A}}_{(\sigma_{i \mod 2^k}), \lceil i/2^k \rceil},$$

where $\sigma_1, \ldots, \sigma_{2^k}$ is a fixed enumeration of all vectors in $\{0, 1\}^k$. The matrix $Q$ expresses the value of $\psi(\mathcal{A})$ in terms of $\mathbf{x}$: if we set $Q = \{q_{i,j}\}$ where

$$q_{i,j} = \left(\sigma_{(i \mod 2^k)} \cdot \left(\mathbf{1} + \pi^{\lceil i/2^k \rceil}\right)\right)\left(\sigma_{(j \mod 2^k)} \cdot \left(\mathbf{1} + \pi^{\lceil j/2^k \rceil}\right)\right),$$

then it is easy to see that $\psi(\mathcal{A}) = \mathbf{x}^T Q \mathbf{x}$. Note also that the maximum absolute value of any entry in $Q$ is at most $4k^2$.

We now use the linear constraints to express the conditions that

1. $\theta(\mathcal{A}) = y$,
2. $\phi(\mathcal{A}) = z$, and
3. the values $x_{i,\sigma}$ correspond to a valid $U$-partition $\mathcal{A}$.

The first of these conditions can be expressed as a single linear constraint:

$$\sum_{(\sigma,i)} x^{\mathcal{A}}_{\sigma,i}(\sigma \cdot \pi^i) = y,$$

or equivalently $\mathbf{a}_1\mathbf{x} = y$ where $\mathbf{a}_1$ is the $1 \times n$ row vector with $i^{th}$ entry equal to

$$\sigma_{(i \mod 2^k)} \cdot \pi^{\lceil i/2^k \rceil}.$$

We can similarly express the second condition as a single linear constraint:

$$\sum_{(\sigma,i)} x^{\mathcal{A}}_{\sigma,i}\, e(P_i)(\sigma \cdot (\mathbf{1} + \pi^i)) = z,$$

or equivalently $\mathbf{a}_2\mathbf{x} = z$, where $\mathbf{a}_2$ is the $1 \times n$ row vector with $i^{th}$ entry equal to

$$e\left(P_{\lceil i/2^k \rceil}\right)\left(\sigma_{(i \mod 2^k)} \cdot \left(\mathbf{1} + \pi^{\lceil i/2^k \rceil}\right)\right).$$

Note that every entry in the vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ has absolute value no more than $2k^3$. For the third condition, note that the values $x_{i,\sigma}$ correspond to a valid $U$-partition if and only if every $x_{i,\sigma}$ is non-negative, and for each $\sigma$ we have $\sum_{i=1}^{\ell} x^{\mathcal{A}}_{i,\sigma} = |S_\sigma|$.

We can therefore express all three conditions in the form $A\mathbf{x} = \mathbf{b}$, where $A$ is a $\left(4 + (\ell+1)2^k\right) \times n$ and $\mathbf{b}$ is a $\left(4 + (\ell+1)2^k\right)$-dimensional vector (notice that we use two inequalities to express each of the linear equality constraints).

Altogether, this means that a solution to this instance of Integer Quadratic Programming will determine the values of $x^{\mathcal{A}}_{i,\sigma}$ which minimize (out of all values corresponding to

some $U$-partition $\mathcal{A}$) the value of $\psi(\mathcal{A})$, subject to the additional requirement that $\theta(\mathcal{A}) = y$ and $\phi(\mathcal{A}) = z$. Note that the number of variables $n$ is at most $k2^k$ and the largest absolute value of any entry in $A$ or $Q$ is at most $2k^3$, so the parameter in the instance of Integer Quadratic Programming is bounded by a function of $k$. This completes the proof. ◀

We note the algorithm described can easily be modified to output an optimal partition.

## 2.2 Parameterisation by treewidth

In this section we demonstrate that Modularity, parameterised by the treewidth of the input graph $G$, belongs to XP and so is solvable in polynomial time on graph classes whose treewidth is bounded by some fixed constant. We further show that for any fixed $\varepsilon > 0$ there is an FPT-algorithm, parameterised by treewidth, which computes a factor $(1-\varepsilon)$-approximation; i.e. returning a value between $(1-\varepsilon)q^*$ and $q^*$ where $q^*$ is the maximum modularity of the graph.

▶ **Theorem 2.3.** Modularity *parameterised by the treewidth of the input graph $G$ is in* XP.

This result makes use of standard dynamic programming techniques, and details are omitted due to space constraints. The key property of modularity we use is that each part in a partition that maximises modularity must induce a connected subgraph, so it suffices to keep track of feasible statistics for each part that intersects the bag of the tree decomposition under consideration, together with the contribution to modularity from parts that are entirely "below" the current bag.

To obtain our FPT approximation result, we use a very similar approach; the key is to restrict our attention to partitions with only a constant number of parts. For any constant $c \in \mathbb{N}$, we write $q^*_{\leq c}(G)$ for the maximum modularity for $G$ achievable with a partition into at most $c$ parts, that is

$$q^*_{\leq c}(G) = \max_{|\mathcal{A}| \leq c} q_{\mathcal{A}}(G).$$

We refer to the problem of deciding whether $q^*_{\leq c}(G) \geq q$ for a given input graph $G$ and constant $q \in [0,1]$ as $c$-Modularity. We now argue that $c$-Modularity is in FPT parameterised by the treewidth of the input graph.

▶ **Lemma 2.4.** $c$-Modularity *is in* FPT *when parameterised by the treewidth of the input graph.*

The crucial difference from our XP algorithm above is the fact that, when we fix the number of parts in the partition, we can no longer assume that every part is connected. However, if the maximum number of parts $c$ is a constant, we can keep track of the necessary statistics for every possible part, not just those that intersect the bag under consideration.

Recall (Fact 1.4) that $q^*(G) \geq q^*_{\leq c}(G)) > q^*(G)\left(1 - \frac{1}{c}\right)$; thus, for any constant $\epsilon > 0$, we obtain a multiplicative approximation with relative error at most $\epsilon$ by solving $\lceil \frac{1}{\epsilon} \rceil$-Modularity. This immediately gives the following result.

▶ **Corollary 2.5.** *Given any constant $\epsilon > 0$, there is an FPT-algorithm, parameterised by the treewidth of the input graph $G$, that that returns a partition $\mathcal{A}$ with $q_{\mathcal{A}}(G) > (1-\varepsilon)q^*(G)$.*

We conclude this section by noting that sparse graphs, in particular graphs $G$ with low tree width, $\mathrm{tw}(G)$, and maximum degree, $\triangle(G)$, can have high maximum modularity. In particular Theorem 1.11 of [19] shows $q^*(G) \geq 1 - 2((\mathrm{tw}(G)+1)\triangle(G)/|E(G)|)^{1/2}$.

## 2.3 Parameterisation by max leaf number

In this section we demonstrate that Modularity can be solved in time linear in the number of connected subgraphs of the input graph $G$; as a consequence of this result, we deduce that the problem belongs to XP when parameterised by the max leaf number of $G$.

▶ **Theorem 2.6.** *Let $G$ be a graph on $n$ vertices with $m$ edges and at most $h$ connected subgraphs. Then* Modularity *can be solved in time $\mathcal{O}(h^2 n)$.*

This is achieved by means of a dynamic programming strategy, in which we compute the maximum value of a modularity-like function for each connected subgraph in turn, considering the subgraphs in nondecreasing order of their number of vertices. Full details of the proof are omitted due to space constraints.

It is known that, if the max leaf number of $G$ is $c$, then $G$ is a subdivision of some graph $H$ on at most $4c$ vertices [12]; a graph on $n$ vertices that is a subdivision of such a graph $H$ has at most $2^{4c} n^{(4c)^2}$ connected subgraphs (once we have decided which branch vertices belong to a subgraph, it remains only to decide where to cut each path from one of the chosen branch vertices to one we have not chosen). Thus, if the max leaf number of $G$ is bounded by a constant it follows that $G$ has at most a polynomial number of connected subgraphs, and the following result is an immediate consequence of Theorem 2.6.

▶ **Corollary 2.7.** Modularity *is in* XP *when parameterised by the max leaf number of the input graph $G$.*

We conjecture that this result is not optimal, and that Modularity is in fact in FPT with respect to this parameterisation.

# 3 Hardness results

In this section we complement our positive result about the FPT approximability of the problem parameterised by treewidth by demonstrating that computing the exact value of the maximum modularity is hard even in a more restricted setting.

▶ **Theorem 3.1.** Modularity*, parameterised simultaneously by the pathwidth and the size of a minimum feedback vertex set for the input graph, is* W[1]*-hard.*

Our proof of this result relies on the hardness of the following problem.

> Equitable Connected Partition (ECP)
> *Input:* A graph $G = (V, E)$ and $r \in \mathbb{N}$.
> *Question:* Is there a partition of $V$ into $r$ classes $V_1, \ldots, V_r$ such that $|V_i| - |V_j| \leq 1$ for all $1 \leq i < j \leq r$, and the induced subgraph $G[V_i]$ is connected for each $i \in 1, \ldots, r$?

The parameterised complexity of ECP was investigated thoroughly in [11]. Among other results, the problem is shown to be W[1]-hard even when parameterised simultaneously by $r$, $\mathrm{pw}(G)$ and $\mathrm{fvs}(G)$. In proving this hardness result, the authors implicitly consider the following variation of ECP.

Anchored Equitable Connected Partition (AECP)
*Input:* A graph $H = (V_H, E_H)$, and a set of distinguished *anchor vertices* $a_1, \ldots, a_r \in V$.
*Question:* Is there a partition of $V_H$ into $r$ classes $V_1, \ldots, V_r$ such that $a_i \in V_i$ for all $i$, $||V_i| - |V_j|| \leq 1$ for all $1 \leq i < j \leq r$, and the induced subgraph $G[V_i]$ is connected for each $i \in 1, \ldots, r$?

From the proof of [11, Theorem 1] we can extract the following statement about the hardness of AECP.

▶ **Lemma 3.2** ([11], implicit in proof of Theorem 1). *AECP is* W[1]*-hard, parameterised simultaneously by* $\mathrm{pw}(H)$ *and* $\mathrm{fvs}(H)$*, even if the following conditions hold simultaneously:*
1. *$H$ is connected;*
2. *the graph $H'$ obtained from $H$ by deleting all vertices of degree one is a subdivision of a 3-regular graph $\tilde{H}$;*
3. *the branch vertices of $H'$ (i.e. vertices of $\tilde{H}$) are precisely the anchor vertices $a_1, \ldots, a_r$;*
4. *$r \geq 4$ is even and divides $|V_H|$;*
5. *$H \setminus \{a_1, \ldots, a_r\}$ is a disjoint union of isolated vertices and paths with pendant edges.*

In the proof of Theorem 3.1, it is useful to analyse the 'per unit modularity defect' $f_m(B)$ of vertex subsets $B$. Intuitively, the following lemma (proof omitted due to space constraints) says that, if you are restricted to parts $B$ with $\delta(B) = 4$, the modularity maximising volume is $\mathrm{vol}(B) = 2\sqrt{2m}$. Moreover, while it would usually be better to take parts with $\delta(B) < 4$ these parts are actually worse (i.e. higher $f_m(B)$ value) if their volumes are too big or too small. The function $f_m(B)$ plays a similar role to the *n-cost* in Proposition 1 of [19].

▶ **Lemma 3.3.** *Assume that $m \geq 1$, $\mathrm{vol}(B) \geq 1$ and define*

$$f_m(B) = \frac{\partial(B)}{\mathrm{vol}(B)} + \frac{\mathrm{vol}(B)}{2m}.$$

*Then the following properties hold:*
0: *if $\partial(B) = 0$ and $\mathrm{vol}(B) > 4\sqrt{2m}$ then $f_m(B) > 2\sqrt{2/m}$.*
1: *if $\partial(B) = 1$ and $\mathrm{vol}(B) > 3.7321\sqrt{2m}$ or $\mathrm{vol}(B) < 0.2679\sqrt{2m}$ then $f_m(B) > 2\sqrt{2/m}$.*
2: *if $\partial(B) = 2$ and $\mathrm{vol}(B) > 3.4143\sqrt{2m}$ or $\mathrm{vol}(B) < 0.5857\sqrt{2m}$ then $f_m(B) > 2\sqrt{2/m}$.*
3: *if $\partial(B) = 3$ and $\mathrm{vol}(B) > 3\sqrt{2m}$ or $\mathrm{vol}(B) < \sqrt{2m}$ then $f_m(B) > 2\sqrt{2/m}$.*
4: *if $\partial(B) = 4$ and $\mathrm{vol}(B) = 2\sqrt{2m}$ then $f_m(B) = 2\sqrt{2/m}$.*
5: *if $\partial(B) \geq 5$ then $f_m(B) > 2\sqrt{2/m}$.*

**Proof of Theorem 3.1.** We give a reduction from AECP. Let $(H, \{a_1, \ldots, a_r\})$ be the input to an instance of AECP; we will describe how to construct a graph $G$, where $\mathrm{pw}(G)$ and $\mathrm{fvs}(G)$ are both bounded by a function of $r$, together with an explicit $q_0 \in (0, 1)$ such that $(G, q_0)$ is a yes-instance for Modularity if and only if $(H, \{a_1, \ldots, a_r\})$ is a yes-instance for AECP.

We may assume without loss of generality that our instance of AECP satisfies all of the conditions of Lemma 3.2.

We define a new graph $G$, obtained from $H$ by adding the following (see Figure 1):
- $\alpha$ new leaves adjacent to each anchor vertex $a_1, \ldots, a_r$,
- $\beta$ isolated edges disjoint from $G$, and
- an arbitrary perfect matching on the anchor vertices $a_1, \ldots, a_r$,

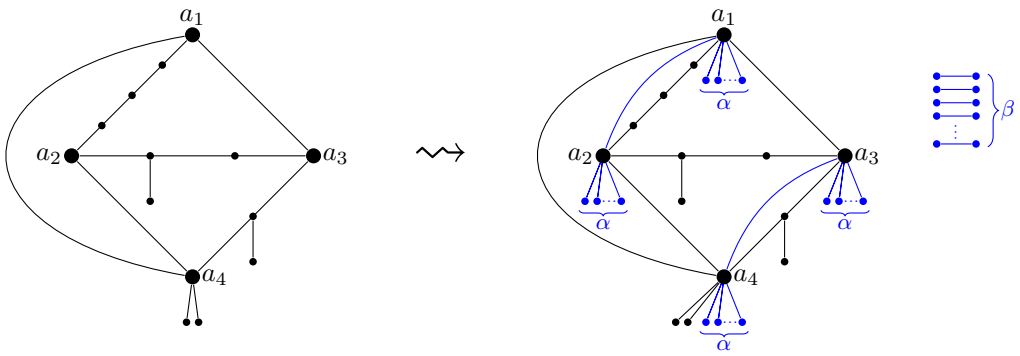



**Figure 1** Possible input graph $H$ with anchors $a_1, a_2, a_3, a_4$ and the graph $G$ constructed from it by adding $\alpha$ new leaves adjacent to each anchor, $\beta$ isolated edges and a perfect matching between the anchors.

where the values of $\alpha$ and $\beta$ will be determined later. Notice that, even with these modifications, $G \setminus \{a_1, \ldots, a_r\}$ is still a disjoint union of isolated vertices and paths with pendant edges; hence $\mathrm{pw}(G) \leq r+1$ and $\mathrm{fvs}(G) \leq r$. We set $m = |E(G)|$ so $m = |E(H)|+\alpha r+\beta+r/2$.

We now define our instance of Modularity to be $(G, q_0)$, where

$$q_0 = 1 - \frac{\beta}{m^2} - \frac{2\sqrt{2}(m-\beta)}{m^{3/2}}$$

We now argue that $(G, q_0)$ is a yes-instance if and only if $(H, \{a_1, \ldots, a_r\})$ is a yes-instance for AECP. Recall that

$$q^*(G) = 1 - \min_{\mathcal{A}} \sum_{A \in \mathcal{A}} \left( \frac{\partial(A)}{2m} + \frac{\mathrm{vol}(A)^2}{4m^2} \right).$$

and that the partition $\mathcal{A}$ which achieves the minimum in the expression above is exactly the modularity maximal $\mathcal{A}$. In any modularity optimal partition, $\mathcal{A}$, each isolated edge will form its own part: this follows from Facts 1.1 and 1.2. Write $V'$ for vertices of $G$ without the vertices supporting the $\beta$ isolated edges, and let the minimisation be over $\mathcal{A}'$ which are vertex partitions of $V'$. We then have

$$q^*(G) = 1 - \frac{\beta}{m^2} - \min_{\mathcal{A}'} \sum_{A \in \mathcal{A}'} \frac{\partial(A)}{2m} + \frac{\mathrm{vol}(A)^2}{4m^2}.$$

Rearranging, we see that

$$\begin{aligned} 1 - \frac{\beta}{m^2} - q^*(G') &= \frac{m-\beta}{m} \min_{\mathcal{A}'} \sum_{A \in \mathcal{A}'} \frac{\mathrm{vol}(A)}{2(m-\beta)} \left( \frac{\partial(A)}{\mathrm{vol}(A)} + \frac{\mathrm{vol}(A)}{2m} \right) \\ &= \frac{m-\beta}{m} \min_{\mathcal{A}'} \sum_{A \in \mathcal{A}'} \frac{\mathrm{vol}(A)}{2(m-\beta)} f_m(A) \qquad (2) \\ &\geq \frac{m-\beta}{m} \min_{A \subseteq V'} f_m(A). \qquad (3) \end{aligned}$$

The last inequality holds because $\sum_A \mathrm{vol}(A) = 2(m-\beta)$ and so (2) is a weighted sum of the $f_m(A)$ with total weight one. This, together with the fact that no $A$ has zero volume, also implies that (2)$\geq$(3) with equality if and only if $f_m(A) = \min_{B \subset V'} f_m(B)$ for every $A \in \mathcal{A}'$.

Note that, since $\mathcal{A}'$ is the restriction of some modularity optimal partition $\mathcal{A}$ to a connected component of $G$, we may assume that, for all $A \in \mathcal{A}'$, $G[A]$ is connected. Moreover, if $v$ is a pendant vertex adjacent to $u$ then $u$ and $v$ are in the same part in $\mathcal{A}'$; we call a partition with this last property (or, abusing notation, a set that would not violate this condition in a partition) 'pendant-consistent'.

We make the following claim, whose proof is omitted due to space constraints; we write $s = |H|/r$ for the desired part size in our instance of AECP.

▶ **Claim 3.4.** *Suppose that $\alpha > 32|E(H)|^2$ and that we have $\sqrt{2m} = s + \alpha + 1$. Then:*

**(a)** *for any connected, pendant-consistent set $B \subseteq V'$ we have $f_m(B) \geq 2\sqrt{2/m}$, and if $f_m(B) = 2\sqrt{2/m}$ then $B$ contains exactly one anchor and $\mathrm{vol}(B) = 2\sqrt{2m}$;*

**(b)** *if $(H, \{a_1, \ldots, a_r\})$ is a yes-instance, then there is a vertex partition $\mathcal{A}'$ of $V'$ so that $f_m(A) = 2\sqrt{2/m}$ for all $A \in \mathcal{A}'$;*

**(c)** *if there is a vertex partition $\mathcal{A}' = \{A_1, \ldots, A_r\}$ of $V'$ so that for all $A_i \in \mathcal{A}$, $f_m(A_i) = 2\sqrt{2/m}$, $A$ is pendant-consistent and $G[A]$ is connected for all $A \in \mathcal{A}'$, then $(H, \{a_1, \ldots, a_r\})$ is a yes-instance.*

Note that, by Claim 3.4(a) and line (3), we always have

$$q^*(G) \leq q_0 = 1 - \frac{\beta}{m^2} - \frac{2\sqrt{2}(m-\beta)}{m^{3/2}}.$$

Hence in particular $(G, q_0)$ is a yes-instance if and only if there is a partition $\mathcal{A}'$ of $V'$ such that $\forall A \in \mathcal{A}'$ $f_m(A) = 2\sqrt{2/m}$.

Claim 3.4(b) , together with line (2), implies that, if $(H, \{a_1, \ldots, a_r\})$ is a yes-instance, then so is $(G, q_0)$. Converesly, if $(G, q_0)$ is a yes-instance, it follows from Claim 3.4(c), that $(H, \{a_1, \ldots, a_r\})$ is a yes-instance.

It remains only to show that we can choose suitable values of $\alpha$ and $\beta$. Set $\alpha$ to be the least integer such that

$$\alpha \geq 32|E(H)|^2, \quad (\alpha + s + 1)^2 > 2|E(H)| + 2\alpha r + r \quad \text{and} \quad \alpha = s + 1 \pmod 2. \tag{4}$$

Recall that $r$ is even. This, along with our parity constraint between $\alpha$ and $s$, implies that $(\alpha + s + 1)^2 - r$ is even. Thus we can choose $\beta$ to be

$$\beta = \frac{1}{2}\left((\alpha + s + 1)^2 - r\right) - |E(H)| - \alpha r; \tag{5}$$

note $\beta$ is positive because we set $\alpha$ so that $(\alpha + s + 1)^2 > 2|E(H)| + 2\alpha r + r$. Finally, observe that we do have $\sqrt{2m} = s + \alpha + 1$ because, by the chosen value of $\beta$,

$$m = |E(H)| + \alpha r + \beta + r/2 = (s + \alpha + 1)^2/2.$$ ◀

## 4 Conclusions and Open Problems

We have shown that Modularity belongs to FPT when parameterised by the vertex cover number of the input graph, and that the problem is solvable in polynomial time on input graphs whose treewidth or max leaf number is bounded by some fixed constant; we also showed that there is an FPT algorithm, parameterised by treewidth, which computes any constant-factor approximation to the maximum modularity. In contrast with the positive approximation result, we demonstrated that the problem is unlikely to admit an exact FPT when the treewidth is taken to be the parameter, as it is W[1]-hard even when parameterised simultaneously by the treewidth and size of a minimum feedback vertex set for the input graph.

We conjecture that our XP algorithm parameterised by max leaf number is not optimal, and that Modularity in fact belongs to FPT with respect to this parameterisation. Another natural open question arising from our work is whether our approximation result can be extended to larger classes of graphs, for example those of bounded cliquewidth or bounded expansion.

## References


1. J. P. Bagrow. Communities and bottlenecks: Trees and treelike networks have high modularity. *Physical Review E*, 85(6):066118, 2012.
2. V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.
3. U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On Modularity Clustering. *IEEE Trans. on Knowl. and Data Eng.*, 20(2):172–188, February 2008. doi:10.1109/TKDE.2007.190689.
4. M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, Cham, 2015.
5. B. DasGupta and D. Desai. On the complexity of Newman's community finding approach for biological and social networks. *Journal of Computer and System Sciences*, 79(1):50–67, 2013. doi:10.1016/j.jcss.2012.04.003.
6. F. De Montgolfier, M. Soto, and L. Viennot. Asymptotic modularity of some graph classes. In *Algorithms and Computation*, pages 435–444. Springer, 2011.
7. T. N. Dinh, X. Li, and M. T. Thai. Network Clustering via Maximizing Modularity: Approximation Algorithms and Theoretical Limits. In *2015 IEEE International Conference on Data Mining*, pages 101–110, November 2015. doi:10.1109/ICDM.2015.139.
8. T. N. Dinh and M. T. Thai. Finding community structure with performance guarantees in scale-free networks. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 888–891. IEEE, 2011.
9. T. N. Dinh and M. T. Thai. Community Detection in Scale-Free Networks: Approximation Algorithms for Maximizing Modularity. *IEEE Journal on Selected Areas in Communications*, 31(6):997–1006, June 2013. doi:10.1109/JSAC.2013.130602.
10. R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer London, 2013.
11. R. Enciso, M. R. Fellows, J. Guo, I. Kanj, F. Rosamond, and O. Suchý. *What Makes Equitable Connected Partition Easy*, pages 122–133. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. doi:10.1007/978-3-642-11269-0_10.
12. V. Estivill-Castro, M. Fellows, M. Langston, and F. Rosamond. FPT is P-time extremal structure I. In *Algorithms and Complexity in Durham 2005, Proceedings of the first ACiD Workshop, volume 4 of Texts in Algorithmics*, pages 1–41. King's College Publications, 2005.
13. S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.
14. S. Fortunato and D. Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016.
15. I. S. Jutla, L. G. S. Jeub, and P. J. Mucha. A generalized Louvain method for community detection implemented in MATLAB. *URL http://netwiki.amath.unc.edu/GenLouvain*, 2011.
16. Y. Kawase, T. Matsui, and A. Miyauchi. Additive Approximation Algorithms for Modularity Maximization. In Seok-Hee Hong, editor, *27th International Symposium on Algorithms*

*and Computation (ISAAC 2016)*, volume 64 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:13, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ISAAC.2016.43.

**17** A. Lancichinetti and S. Fortunato. Limits of modularity maximization in community detection. *Physical Review E*, 84(6):066122, 2011.

**18** D. Lokshtanov. Parameterized Integer Quadratic Programming: Variables and Coefficients. arXiv:1511.00310 [cs.DS], 2015.

**19** C. McDiarmid and F. Skerman. Modularity of regular and treelike graphs. *Journal of Complex Networks*, 5, 2017.

**20** C. McDiarmid and F. Skerman. Modularity of Erdős-Rényi Random Graphs. In *29th International Conference on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms*, volume 1, 2018.

**21** C. McDiarmid and F. Skerman. Modularity of Erdős-Rényi random graphs. arXiv:1808.02243 [math.CO], 2018.

**22** M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004.

**23** M. Porter, J.-P. Onnela, and P. Mucha. Communities in networks. *Notices of the AMS*, 56(9):1082–1097, 2009.

**24** L. O. Prokhorenkova, P. Prałat, and A. Raigorodskii. Modularity of models of complex networks. *Electronic Notes in Discrete Mathematics*, 61:947–953, 2017.

# On the Distance Identifying Set Meta-Problem and Applications to the Complexity of Identifying Problems on Graphs

**Florian Barbero**
LIRMM, Université de Montpellier, 161 rue Ada, 34095, Montpellier, France
florian.barbero@lirmm.fr

**Lucas Isenmann**
LIRMM, Université de Montpellier, 161 rue Ada, 34095, Montpellier, France
lucas.isenmann@lirmm.fr

**Jocelyn Thiebaut**
LIRMM, Université de Montpellier, 161 rue Ada, 34095, Montpellier, France
jocelyn.thiebaut@lirmm.fr

## Abstract

Numerous problems consisting in identifying vertices in graphs using distances are useful in domains such as network verification and graph isomorphism. Unifying them into a meta-problem may be of main interest. We introduce here a promising solution named Distance Identifying Set. The model contains Identifying Code (IC), Locating Dominating Set (LD) and their generalizations $r$-IC and $r$-LD where the closed neighborhood is considered up to distance $r$. It also contains Metric Dimension (MD) and its refinement $r$-MD in which the distance between two vertices is considered as infinite if the real distance exceeds $r$. Note that while IC = 1-IC and LD = 1-LD, we have MD = $\infty$-MD; we say that MD is not *local*.

In this article, we prove computational lower bounds for several problems included in Distance Identifying Set by providing generic reductions from (Planar) Hitting Set to the meta-problem. We focus on two families of problem from the meta-problem: the first one, called *bipartite gifted local*, contains $r$-IC, $r$-LD and $r$-MD for each positive integer $r$ while the second one, called 1-*layered*, contains LD, MD and $r$-MD for each positive integer $r$. We have:

- the 1-layered problems are NP-hard even in bipartite apex graphs,
- the bipartite gifted local problems are NP-hard even in bipartite planar graphs,
- assuming ETH, all these problems cannot be solved in $2^{o(\sqrt{n})}$ when restricted to bipartite planar or apex graph, respectively, and they cannot be solved in $2^{o(n)}$ on bipartite graphs,
- even restricted to bipartite graphs, they do not admit parameterized algorithms in $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ except if $\mathsf{W}[0] = \mathsf{W}[2]$. Here $k$ is the solution size of a relevant identifying set.

In particular, Metric Dimension cannot be solved in $2^{o(n)}$ under ETH, answering a question of Hartung in [20].

**2012 ACM Subject Classification** Mathematics of computing → Graph algorithms, Theory of computation → Problems, reductions and completeness

**Keywords and phrases** identifying code, resolving set, metric dimension, distance identifying set, parameterized complexity, W-hierarchy, meta-problem, hitting set

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.10

**Related Version** The full version of the article can be found in [4], `https://arxiv.org/abs/1810.03868`.

**Acknowledgements** We want to thank Stéphane Bessy and Anaël Grandjean for their precious advice concerning the correctness of the claims, and the structure of the article.

© Florian Barbero, Lucas Isenmann, and Jocelyn Thiebaut;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 10; pp. 10:1–10:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction and Corresponding Works

Problems consisting in identifying each element of a combinatorial structure with a hopefully small number of elements have been widely investigated. Here, we study a meta identification problem which generalizes three of the most well-known identification problems in graphs, namely Identifying Code (IC), Locating Dominating Set (LD) and Metric Dimension (MD). These problems are used in network verification [3, 5], fault-detection in networks [22, 28], graph isomorphism [2] or logical definability of graphs [23]. The versions of these problems in hypergraphs have been studied under different names in [6], [7] and [8].

Given a graph $G$ with vertex set $V$, the classical identifying sets are defined as follows:

- IC: Introduced by Karposky et al. [22], a set $C$ of vertices of $G$ is said to be an *identifying code* if none of the sets $N[v] \cap C$ are empty, for $v \in V$ and they are all distinct.
- LD: Introduced by Slater [25, 26], a set $C$ of vertices of $G$ is said to be a *locating-dominating set* if none of the sets $N[v] \cap C$ are empty, for $v \in V \setminus C$ and they are all distinct. When not considering the dominating property ($N[v] \cap C$ may be empty), these sets have been studied in [2] as distinguing sets and in [23] as sieves.
- MD: Introduced independently by Harary et al. [18] and Slater [24], a set $C$ of vertices of $G$ is said to be a *resolving set* if $C$ contains one vertex from each connected component of $G$ and, for every distinct vertices $u$ and $v$ of $G$, there exists a vertex $w$ of $C$ such that $d(w, u) \neq d(w, v)$. The *metric dimension* of $G$ is the minimum size of its resolving sets.

The corresponding minimization problems of the previous identifying sets are defined as follows: given a graph $G$, compute a suitable set $C$ of minimal size, if one exists. In this paper, we mainly focus on the computational complexity of these minimization problems.

**Known results.** A wide collection of NP-hardness results has been proven for the problems.

For IC and LD, the minimization problems are indeed NP-hard [10, 11]. Charon et al. showed the NP-hardness when restricted to bipartite graphs [9], while Auger showed it for planar graphs with arbitrarily large girth [1]. For trees, there exists a linear algorithm [25].

The MD problem is also NP-hard, even when restricted to Gabriel unit disk graphs [17, 21]. Epstein et al. [14] showed that MD is polynomial on several classes as trees, cycles, cographs, partial wheels, and graphs of bounded cyclomatic number, but it remains NP-hard on split graphs, bipartite graphs, co-bipartite and line graphs of bipartite graphs. Additionally, Diaz et al. [12] proved a quite tight separation: the problem is polynomial on outerplanar graphs whereas it remains NP-hard on bounded degree planar graphs.

In a recent publication, Foucaud et al. [16] also proved the NP-hardness of the three problems restricted to interval graphs and permutation graphs.

These notions may be considered under the parameterized point of view; see [13] for a comprehensive study of Fixed Parameter Tractability (FPT). In the following, the parameter $k$ is chosen as the solution size of a distance identifying set.

For IC and LD, the parameterized problems are clearly FPT since the number of vertices of a positive instance is bounded by $2^k + k$ ($k$ vertices may characterize $2^k$ neighbors).

Such complexity is not likely to be achievable in the case of MD, since it would imply $\mathsf{W}[2] = \mathsf{FPT}$ ($= \mathsf{W}[0]$). Indeed, Hartung et al. [19, 20] showed MD is W[2]-hard for bipartite subcubic graphs. The problem is however FPT on families of graphs with degree $\Delta$ growing with the number of vertices because the size $k$ of a resolving set must satisfy $log_3(\Delta) < k$. Finally, Foucaud et al. [16] provided a FPT algorithm on interval graphs.

| | 1-layered problems | $r$-local 0-layered problems | $r$-local problems |
|---|---|---|---|
| Planar Hitting Set | NP-hard on bipartite apex graphs | NP-hard on bipartite planar graphs | (bipartite) planar gadget ⇒ NP-hard on (bipartite) planar graphs |
| with ETH | no algorithm running in $2^{\mathcal{O}(\sqrt{n})}$ time for relevant classes of graphs. | | |
| Hitting Set | NP-hard on bipartite graph | | (bipartite) gadget ⇒ NP-hard on (bipartite) graphs |
| with ETH | no algorithm running in $2^{\mathcal{O}(n)}$ for (bipartite) graphs. | | |
| with W[2] ≠ W[0] | no parameterized algorithm in $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ for (bipartite) graphs. | | |

**Figure 1** The computational lower bounds implied by our generic reductions.

**Our contributions.** In order to unify the previous minimization problems, we introduce the concept of *distance identifying functions*. Given a distance identifying function $f$ and a value $r$ as a positive integer or infinity, the Distance Identifying Set meta-problem consists in finding a minimal sized $r$-dominating set which distinguishes every couple of vertices of an input graph thanks to the function $f$. Here, we mainly focus on two natural subfamilies of problems of Distance Identifying Set named *local*, in which a vertex cannot discern the vertices outside of its $i$-neighborhood, for $i$ a fixed positive integer, and 1-*layered*, where a vertex is able to separate its open neighborhood from the distant vertices.

With this approach, we obtain several computational lower bounds for problems included in Distance Identifying Set by providing generic reductions from (Planar) Hitting Set to the meta-problem. The reductions rely on the set/element-gadget technique, the noteworthy adaptation of the clause/variable-gadget technique from SAT to Hitting Set.

As we provide a 1-layered generic gadget, the 1-layered reductions operate without condition. For local problems, the existence of a local gadget is not always guaranteed. Thus, a local reduction operates only if a local gadget is provided. However, the local planar reduction is slightly more efficient than its 1-layered counterpart: it indeed implies computational lower bounds for planar graphs whereas the 1-layered reduction requires an auxiliary apex, limiting the consequences to apex graphs.

The reductions in general graphs are designed to exploit the W[2]-hardness of Hitting Set parameterized by the solution size $k_{HS}$ of an hitting set, hereby using:

▶ **Theorem 1** (folklore). *Let $n_{HS}$ and $m_{HS}$ be the number of elements and sets of an* Hitting Set *instance, and $k_{HS}$ be its solution size. A parameterized problem with parameter $k$ admitting a reduction from* Hitting Set *verifying $k = \mathcal{O}(k_{HS} + log(n_{HS} + m_{HS}))$ does not have a parameterized algorithm running in $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time except if* W[2] = FPT.

**Proof.** Given a reduction from Hitting Set to a parameterized problem $\Pi$ such that the reduced parameter satisfies $k = \mathcal{O}(k_{HS} + log(n_{HS} + m_{HS}))$ and the size of the reduced instance verifies $n = (n_{HS} + m_{HS})^{\mathcal{O}(1)}$, an algorithm for $\Pi$ of running time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ is actually an algorithm for Hitting Set of running time $2^{\mathcal{O}(k_{HS})} \cdot (n_{HS} + m_{HS})^{\mathcal{O}(1)}$, meaning that Hitting Set is FPT, a contradiction to its W[2]-hardness (otherwise W[2] = FPT). ◀

Hence, as each gadget contributes to the resulting solution size of a distance identifying set, we set up a binary compression of the gadgets to limit their number to the logarithm order. From the best of our knowledge, this merging gadgets technique has never been employed.

The organization of the paper is as follows. After a short reminder of the computational properties of Hitting Set, Section 2 contains the definitions of distance identifying functions and sets, allowing us to precise the computation lower bounds we obtain. Section 3 designs

the supports of the reductions as *distance identifying graphs* and *compressed graph*. Finally, the gadgets needed for the reductions to apply are given in Section 4.

# 2 Definition of the Meta-Problem and Related Concepts

## 2.1 Preliminaries

**Notations.** Throughout the paper, we consider simple non oriented graphs.

Given a positive integer $n$, the set of positive integers smaller than $n$ is denoted by $[\![n]\!]$. By extension, we define $[\![\infty]\!] = \mathbb{N}_{>0} \cup \{\infty\}$. Given two vertices $u, v$ of a graph $G$, the distance between $u$ and $v$ corresponds to the number of vertices in the shortest path between u and v and is denoted $d(u, v)$. The *open neighborhood of* $u$ is denoted by $N(u)$, its *closed neighborhood* is $N[u] = N(u) \cup \{u\}$, and for a value $r \in [\![\infty]\!]$, the $r$*-neighborhood* of $u$ is $N_r[u]$, that is the set of vertices at distance less than $r + 1$ of $u$. For $r = \infty$, the $\infty$-neighborhood of $u$ is the set of vertices in the same connected component than $u$. We recall that a subset $D$ of $V$ is called a $r$*-dominating set* of $G$ if for all vertices $u$ of $V$, the set $N_r[u] \cap D$ is non-empty. Thus an $\infty$-dominating set of $G$ contains at least a vertex for each connected component of $G$.

Given two subsets $X$ and $Y$ of $V$, the distance $d(X, Y)$ corresponds to the value $d(X, Y) = \min\{d(x, y) \mid x \in X,\ y \in Y\}$. For a vertex $u$, we will also use $d(u, X)$ and $d(X, u)$, defined similarly. The *symmetric difference* between $X$ and $Y$ is denoted by $X \Delta Y$, and the 2-*combination* of a set $X$ is denoted $\mathcal{P}_2(X)$

Given two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$, $H$ is an *induced subgraph of* $G$ if $V_H \subseteq V_G$ and for all vertices $u$ and $v$ of $V_H$, $(u, v) \in E_G$ if and only if $(u, v) \in E_H$. We denote $H = G[V_H]$ and $V_G \setminus V_H$ by $V_{G \setminus H}$. Symmetrically, $G$ is an *induced supergraph of* $H$.

### The (Planar) Hitting Set problem

Consider a universe of $n$ elements denoted $\Omega = \{u_i \mid i \in [\![n]\!]\}$ and a set of $m$ non-empty subsets of $\Omega$ denoted $\mathcal{S} = \{S_i \mid i \in [\![m]\!]\}$ such that every element belongs to at least a subset. Then, a subset of $\Omega$ intersecting every set of $\mathcal{S}$ is called an *hitting set* of $\mathcal{S}$:

Hitting Set
**Input:** A universe $\Omega$ and a set $\mathcal{S}$ of non-empty subsets of $\Omega$ whose union covers $\Omega$.
**Output:** A minimal-sized hitting set $C$ of $\mathcal{S}$, i.e. a subset of $\Omega$ satisfying $\forall S_i \in \mathcal{S},\ S_i \cap C \neq \emptyset$.
The parameterized version Hitting Set(k) decides if there exists a hitting set of size $k$.

▶ **Theorem 2** (R.G. Downey and M.R Fellows [13]). *Hitting Set cannot be solved in $2^{o(n)}$ time under ETH even if $m = \mathcal{O}(n)$. Moreover, Hitting Set(k) is* W[2]*-hard.*

Hitting Set may be translated into a dominating problem on bipartite graphs. Given an instance $(\Omega, \mathcal{S})$ of Hitting Set, let us define $\phi(\Omega, \mathcal{S}) = (V_\Omega \cup V_\mathcal{S}, E)$ as the bipartite graph of size $n + m$ such that for each $i \in [\![n]\!]$, there exists a vertex $v_i^\Omega$ in $V_\Omega$, for each $j \in [\![m]\!]$, there exists a vertex $v_j^\mathcal{S}$ in $V_\mathcal{S}$, and the edge $(v_i^\Omega, v_j^\mathcal{S})$ is present in $E$ if and only if the element $u_i$ belongs to the subset $S_j$. Henceforth, a hitting set of $\mathcal{S}$ is equivalent to a subset $C$ of $V_\Omega$ that dominates $V_\mathcal{S}$. We call $\phi(\Omega, \mathcal{S})$ *the associated graph of* $(\Omega, \mathcal{S})$:

Planar Hitting Set
**Input:** An instance $(\Omega, \mathcal{S})$ of Hitting Set such that $\phi(\Omega, \mathcal{S})$ is planar.
**Output:** A hitting set $C$ of $\mathcal{S}$ of minimal size.
We also consider the parameterized version Planar Hitting Set(k) of the latter problem.

▶ **Theorem 3** (folklore). *There exists a reduction from* SAT *to* Planar Hitting Set*(n) producing associated graphs of quadratic size in the number $n$ of variables of the instances of* SAT*. Thus* Planar Hitting Set *cannot be solved in $2^{o(\sqrt{n})}$ under ETH even if $m = \mathcal{O}(n)$.*

**Sketch of proof.** A linear reduction from SAT to Hitting Set is known. To guarantee the planarity of $\phi(\Omega, \mathcal{S})$, we apply the reduction on a restriction of SAT named Separate Simple Planar SAT which is not solvable in $2^{o(\sqrt{n})}$ under ETH. See [27]. ◀

## 2.2 The Distance Identifying Set meta-problem

Given a graph $G = (V, E)$ and $r \in [\![\infty]\!]$, the classical identifying sets may be rewritten:

- $r$-IC: a subset $C$ of $V$ is a $r$-identifying code of $G$ if it is an $r$-dominating set and for every distinct vertices $u, v$ of $V$, a vertex $w$ in $C$ verifies $w \in N_r[u] \,\Delta\, N_r[v]$.
- $r$-LD: a subset $C$ of $V$ is a $r$-locating dominating set of $G$ if it is an $r$-dominating set and for every distinct vertices $u, v$ of $V$, a vertex $w$ in $C$ verifies $w \in (N_r[u] \,\Delta\, N_r[v]) \cup \{u, v\}$.
- $r$-MD: a subset $C$ of $V$ is a $r$-resolving set of $G$ if it is an $r$-dominating set and for every distinct vertices $u, v$ of $V$, a vertex $w$ in $C$ verifies $w \in N_r[u] \cup N_r[v]$ and $d(u, w) \neq d(v, w)$.

A pattern clearly appears: the previous identifying sets only deviate on the criterion that the vertex $w$ must verify. The pivotal idea is to consider an abstract version of the criterion which does not depend on the input graph. Hence:

▶ **Definition 4** (identifying function). A function $f$ of type: $G \to (V \times \mathcal{P}_2(V) \to \{\text{true}, \text{false}\})$, is called *an identifying function*. Given three vertices $u$, $v$ and $w$ of a graph $G$ such that $u \neq v$, we write $f_G[w](u, v)$ to get the resulting boolean. The notation $\mathcal{P}_2(V)$ implies that $f_G$ is symmetric, that is $f_G[w](u, v) = f_G[w](v, u)$.

We need to require some useful properties on identifying functions to produce generic results. By mimicking the classical identifying sets, the main property we consider is that a vertex cannot distinguish two vertices at the same distance from it. Then:

▶ **Definition 5** (distance function). A *distance identifying function* $f$ is an identifying function such that for every graph $G$ and all vertices $u$,$v$ and $w$ of $G$ with $u \neq v$:
**($\alpha$)** $f_G[w](u, v)$ is false when $d(u, w) = d(v, w)$.
Besides this mandatory criterion, we suggest two paradigms related to the neighborhood of a vertex. Let $i \in [\![\infty]\!]$. First, we may restrain the range of a vertex to its $i$-neighborhood: a vertex should not distinguish two vertices if they do not lie in its $i$-neighborhood but it should always distinguish them whenever exactly one of them lies to that $i$-neighborhood. Reciprocally, we may ensure that a vertex could distinguish the vertices of its $i$-neighborhood: a vertex should distinguish a vertex belonging to its $i$-neighborhood from all the other vertices, assuming the distances are different. Formally, we have:

▶ **Definition 6** ($i$-local function). For $i \in [\![\infty]\!]$, an *$i$-local identifying function* $f$ is an identifying function such that for every graph $G$ and all vertices $u$, $v$, $w$ of $G$ with $u \neq v$:
**($\beta_1$)** $f_G[w](u, v)$ is true when $d(u, w) \leq i < d(v, w)$ or, symmetrically, $d(v, w) \leq i < d(u, w)$.
**($\beta_2$)** $f_G[w](u, v)$ is false when $i < \min\{d(u, w), d(v, w)\}$.

▶ **Definition 7** ($i$-layered function). For $i \in [\![\infty]\!]$, an *$i$-layered identifying function* $f$ is an identifying function such that for every graph $G$ and all vertices $u$,$v$,$w$ of $G$ with $u \neq v$:
**($\gamma$)** $f_G[w](u, v)$ is true when $\min\{d(u, w), d(v, w)\} \leq i$ and $d(u, w) \neq d(v, w)$.

In the following, given an identifying function $f$ and three vertices $u$, $v$, $w$ of a graph $G$, we say that $w$ $f$-distinguishes $u$ and $v$ if and only if $f_G[w](u,v)$ is true. By extension, given three vertex sets $C$, $X$ and $Y$ of $G$, we say that $C$ $f$-distinguishes $X$ and $Y$ if for every $u$ in $X$ and $v$ in $Y$, either $u = v$ or there exists $w$ in $C$ verifying $f_G[w](u,v)$. Finally, a graph $G$ of vertex set $V$ is $f$-distinguished by $C$ when $C$ $f$-distinguishes $V$ and $V$.

We are now ready to define the Distance Identifying Set meta-problem.

▶ **Definition 8** (($f,r$)-distance identifying set)**.** For a distance identifying function $f$ and $r \in [\![\infty]\!]$, a $(f,r)$-*distance identifying set* of a graph $G$ is an $r$-dominating set of $G$ that $f$-distinguishes $G$.

Distance Identifying Set
**Input:** A distance identifying function $f$ and $r \in [\![\infty]\!]$. A graph $G$.
**Output:** A $(f,r)$-distance identifying set of $G$ of minimal size, if one exists.

Given a distance identifying function $f$ and $r \in [\![\infty]\!]$ as inputs of the meta-problem, the resulting problem is called $(f,r)$-Distance Identifying Set and denoted $(f,r)$-DIS. The problem $(f,r)$-DIS is said to be *$i$-layered* when the function $f$ is $i$-layered, and it is said to be *$i$-local* when $f$ is $i$-local and $r = i$. A problem is *local* if it is $i$-local for an integer $i$. Recall that our local reductions need a local gadget to operate: the subfamilies of local problems admitting a (bipartite) local gadget is called *(bipartite) gifted local*. We do not need to define *gifted* 1*-layered* as every 1-layered problem admits a 1-layered gadget. We also consider the parameterized version Distance Identifying Set($k$).

## 2.3 Detailed Computational Lower Bounds

Using the Distance Identifying Set meta-problem, we get the following lower bounds:

▶ **Theorem 9.** *For each* 1*-layered distance identifying function $f$ and every $r \in [\![\infty]\!]$, the* $(f,r)$-Distance Identifying Set *problem restricted to bipartite apex graphs is* NP*-hard, and does not admit an algorithm running in $2^{\mathcal{O}(\sqrt{n})}$ time under ETH.*

▶ **Theorem 10.** *The (bipartite) gifted local problems restricted to (bipartite) planar graphs are* NP*-hard, and do not admit an algorithm running in $2^{\mathcal{O}(\sqrt{n})}$ time under ETH.*

▶ **Theorem 11.** *For each $r$-local* 0*-layered distance identifying function $f$,* $(f,r)$-DIS *restricted to bipartite planar graphs is* NP*-hard, and cannot be solved in $2^{\mathcal{O}(\sqrt{n})}$ under ETH.*

▶ **Theorem 12.** *Let $f, g$ and $h$ be distance identifying functions such that $f$ is* 1*-layered, $g$ is $q$-local* 0*-layered and $h$ is $p$-local and admits a local (bipartite) gadget. Let $r \in [\![\infty]\!]$. The* $(f,r)$*-,* $(g,q)$*- and* $(h,p)$*-* DIS *problems are* NP*-hard, and do not admit:*

- *algorithms running in $2^{o(n)}$ time, except if ETH fails,*
- *parameterized algorithms running in $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time, except if $W[2] = \mathsf{FPT}$.*

*The parameter $k$ denotes here the solution size of a relevant distance identifying set.*
*All bounds still hold in the bipartite case (whenever the gadget associated with $h$ is bipartite).*

As a side result, the 1-layered general reduction answers a question of Hartung in [20]:

▶ **Corollary 13.** *Under ETH,* Metric Dimension *cannot be solved in $2^{o(n)}$.*

Finally, notice that the parameterized lower bound from Theorem 12 may be complemented by an elementary upper bound inspired from the kernel of IC and LD of size $2^k + k$:

▶ **Proposition 14.** *For every $r$-local distance identifying function $f$, the* $(f,r)$-Distance Identifying Set *problem has a kernel of size $(r+1)^k + k$ where $k$ is the solution size. Therefore, it admits a naive parameterized algorithm running in $\mathcal{O}(n^{k+3}) \in \mathcal{O}^*(r^{(k^2)})$ time.*

# 3 The Supports of the Reductions for Distance Identifying Set

## 3.1 The Distance Identifying Graphs

Consider the associated graph $\phi(\Omega, \mathcal{S})$ as defined in Section 2.1. The differences between the Distance Identifying Set meta-problem and the dominating problem related to associated graphs actually raise two issues for a reduction based on these latter notions to be effective on Distance Identifying Set. First, contrarily to the dominating problem where a vertex may only discern its close neighborhood, the meta-problem may allow a vertex to discern further than its direct neighborhood. In that case, we cannot certify that a vertex $v_i^\Omega$ does not distinguish a vertex $v_j^\mathcal{S}$ when $u_i$ is not in $S_j$, the adjacency not remaining a sufficient argument. Secondly, one may object that a vertex $v_i^\Omega$ formally has to distinguish a vertex $v_j^\mathcal{S}$ from another vertex, but that distinguishing a single vertex is not defined.

To circumvent these problems, we suggest the following fix: rather than producing a single vertex for each $S_j \in \mathcal{S}$, the set $V_\mathcal{S}$ may contain two vertices $v_j^\mathcal{S}$ and $\bar{v}_j^\mathcal{S}$. Then, the role of $v_i^\Omega$ would be to distinguish them if and only if $u_i \in S_j$. To ensure that the vertex $v_i^\Omega$ distinguishes $v_j^\mathcal{S}$ and $\bar{v}_j^\mathcal{S}$ when $u_i \in S_j$, we may use the properties $(\beta_1)$ and $(\gamma)$ of Definition 6 and 7 for the $r$-local and 1-layered problems, respectively. Precisely, when $u_i \in S_j$, $v_i^\Omega$ should be at distance $r$ to $v_j^\mathcal{S}$ (with $r = 1$ in the 1-layered cases) while $\bar{v}_j^\mathcal{S}$ should not be in the $r$-neighborhood of $v_i^\Omega$. Similarly, to ensure that $v_i^\Omega$ cannot distinguish $v_j^\mathcal{S}$ and $\bar{v}_j^\mathcal{S}$ when $u_i \notin S_j$, we may use properties $(\alpha)$ or $(\beta_2)$ of Definitions 5 and 6. Hence, when $u_i \notin S_j$, $v_j^\mathcal{S}$ should not be in the $r$-neighborhood of $v_i^\Omega$, or $d(v_i^\Omega, v_j^\mathcal{S})$ and $d(v_i^\Omega, \bar{v}_j^\mathcal{S})$ should be equal.

That fix fairly indicates how to initiate the transformation of the associated graphs in order to deliver an equivalence between a hitting set formed by elements of $\Omega$ and the vertices of a distance identifying set included in $V_\Omega$. However, it is clearly not sufficient since we also have to distinguish the couples of vertices of $V_\Omega$ for which nothing is required. To solve that problem, we suggest to append to each vertex of the associated graph a copy of some gadget with the intuitive requirement that the gadget is able to distinguish the close neighborhood of its vertices from the whole graph. We introduce the notion of *$B$-extension*:

▶ **Definition 15** ($B$-extension). Let $H = (V_H, E_H)$ be a connected graph, and $B \subseteq V_H$. An induced supergraph $G = (V_G, E_G)$ is said to be a *$B$-extension* of $H$ if it is connected and for every vertex $v$ of $V_{G \setminus H}$, the set $N(v) \cap V_H$ is either equal to $\emptyset$ or $B$.

A vertex $v$ of $V_{G \setminus H}$ such that $N(v) \cap V_H = B$ is said to be *$B$-adjacent*. The $B$-extensions of $H$ such that $V_{G \setminus H}$ contains exactly a $B$-adjacent vertex or two $B$-adjacent (but not neighbors) vertices are called *the $B$-single-extension* and *the $B$-twin-extension* of $H$, respectively.

Here, the border $B$ makes explicit the connections between a copy of a gadget $H$ and a vertex outside the copy. In particular, a $B$-single-extension is formed by a gadget with its related vertex $v_i^\Omega$, while a $B$-twin-extension contains a gadget with its two related vertices $v_j^\mathcal{S}$ and $\bar{v}_j^\mathcal{S}$. Piecing all together, we may adapt the associated graphs to the meta-problem:

▶ **Definition 16** (distance identifying graph). Let $(\Omega = \{u_i \mid i \in [\![n]\!]\}, \mathcal{S} = \{S_i \mid i \in [\![m]\!]\})$ be an instance of Hitting Set. Let $H$ be a connected graph, $B$ a subset of its vertices, and $r$ a positive integer. The *$(H, B, r)$-distance identifying graph* $\Phi[H, B, r](\Omega, \mathcal{S})$ is as follows.

- for each $i \in [\![n]\!]$, the graph $\Phi[H, B, r](\Omega, \mathcal{S})$ contains as induced subgraph a copy $H_i^\Omega$ of $H$ together with a $B_i^\Omega$-adjacent vertex $v_i^\Omega$, where $B_i^\Omega$ denotes the copy of $B$.
- similarly, for each $j \in [\![m]\!]$, the graph $\Phi[H, B, r](\Omega, \mathcal{S})$ contains a copy $H_j^\mathcal{S}$ of $H$ together with two $B_j^\mathcal{S}$-adjacent vertices $v_j^\mathcal{S}$ and $\bar{v}_j^\mathcal{S}$; the latter vertices are not adjacent.
- finally, for each $S_j \in \mathcal{S}$ and each $u_i \in S_j$, $v_i^\Omega$ is connected to $v_j^\mathcal{S}$ by a path of $r-1$ vertices denoted $l_{i,j}^k$ with $d(v_i^\Omega, l_{i,j}^k) = k$ for each $k \in [\![r-1]\!]$.

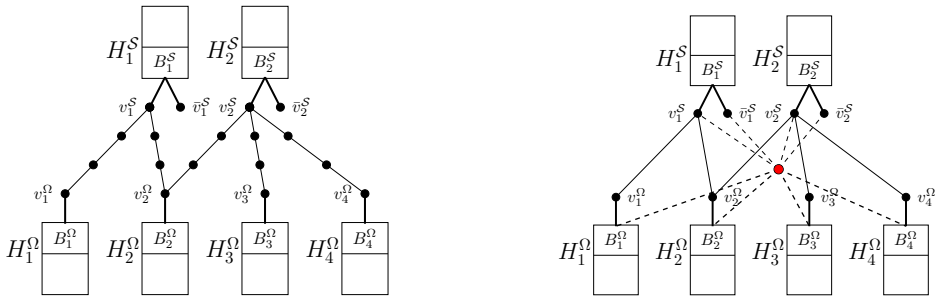



**Figure 2** A $(H, B, 3)$-distance identifying graph and a $(H, B)$-apex distance identifying graph built on the planar instance formed by $\Omega = \{1, 2, 3, 4\}$ and $\mathcal{S} = \{\{1, 2\}, \{2, 3, 4\}\}$.

When the problem is not local, we prefer the following identifying graph:

▶ **Definition 17** ($(H, B)$-apex distance identifying graph)**.** An $(H, B)$-*apex distance identifying graph* $\Phi^*[H, B](\Omega, \mathcal{S})$ is the union of a $(H, B, 1)$-distance identifying graph with an additional vertex $a$ called *apex* such that:

- for each $u_i \in \Omega$, the apex $a$ is $B_i^{\Omega}$-adjacent to $H_i^{\Omega}$.
- for each $S_j \in \mathcal{S}$, the apex $a$ is adjacent to $v_j^{\mathcal{S}}$ and $\bar{v}_j^{\mathcal{S}}$.

▶ **Proposition 18.** *Given an instance $(\Omega, \mathcal{S})$ of* Planar Hitting Set *where $|\Omega| = n$, $|\mathcal{S}| = m$, the graphs $G = \Phi[H, B, r](\Omega, \mathcal{S})$ and $G' = \Phi^*[H, B](\Omega, \mathcal{S})$*

- *are connected and have size bounded by $(|H| + 2r)(n + m)$, (with $r = 1$ for $G'$),*
- *may be built in polynomial time in their size,*
- *are bipartite if the $B$-single extension of $H$ is bipartite,*
- *are respectively planar and an apex graph if the $B$-twin-extension of $H$ is planar.*

Having defined the (apex) distance identifying graphs, the main effort to obtain generic reduction from Planar Hitting Set is done. We now define relevant gadgets:

▶ **Definition 19** ($(f, r)$-gadgets)**.** Let $f$ be a distance identifying function and $r \in [\![\infty]\!]$. Let $H = (V_H, E_H)$ be a connected graph, and $B, C$ be two subsets of $V_H$. We said that the triple $(H, B, C)$ is a $(f, r)$-*gadget* if for every $B$-extension $G$ of $H$:

$(p_h)$ $C$ $f$-distinguishes $V_H$ and $V_G$.
$(p_b)$ $C$ $f$-distinguishes $N_B$ and $V_{G \setminus H} \setminus N_B$, where $N_B$ is the set of $B$-adjacent vertices of $G$.
$(p_d)$ $C$ is an $r$-dominating set of $G[V_H \cup N_B]$.
$(p_s)$ For all $(f, r)$-distance identifying set $S$ of $G$, $|C| \leq |S \cap V_H|$.

▶ **Definition 20** (local gadgets)**.** A $(f, r)$-gadget is a *local gadget*, if $f$ is a $r$-local identifying function with $r \neq \infty$, and $(p_l)$: for every $k \in [\![r]\!]$, there exists $c \in C$ such that $d(c, B) = k - 1$.

Consistently, we say that a $(f, r)$-gadget $(H, B, C)$ is *bipartite* if the $B$-single-extension of $H$ is bipartite, and that it is *planar* if the $B$-twin-extension of $H$ is planar.

▶ **Theorem 21.** *Let $(\Omega, \mathcal{S})$ be an instance of* Hitting Set *such that $|\Omega| = n > 1$, $|\mathcal{S}| = m$. Let $(H, B, C)$ be a $(f, r)$-gadget for a 1-layered identifying function $f$ and let $(H', B', C')$ be a local $(g, q)$-gadget. The following propositions are equivalent:*

- *there exists a hitting set of $\mathcal{S}$ of size $k$.*
- *there exists a $(f, r)$-distance identifying set of $\Phi^*[H, B](\Omega, \mathcal{S})$ of size $k + |C|(n + m)$.*
- *there exists a $(g, q)$-distance identifying set of $\Phi[H', B', q](\Omega, \mathcal{S})$ of size $k + |C'|(n + m)$.*

**Proof of Theorem 21.** We focus on the equivalence between the first and second items.

Suppose first that $P$ is a hitting set of $(\Omega, \mathcal{S})$ of size $k$. By denoting $C_i^\Omega$ and $C_j^\mathcal{S}$ the copies of $C$ associated to the copies $H_i^\Omega$ and $H_j^\mathcal{S}$ of $H$, we suggest the following set $I$ of size $k + |C|(n+m)$ as a $(f,r)$-distance identifying set of $G = \Phi^*[H,B](\Omega,\mathcal{S})$:

$$I = \{v_i^\Omega \colon u_i \in P\} \cup \bigcup_{i \in [\![n]\!]} C_i^\Omega \cup \bigcup_{j \in [\![m]\!]} C_j^\mathcal{S}.$$

Recall that by construction, $G$ is a $B_i^\Omega$-extension of $H_i^\Omega$ (respectively $B_j^\mathcal{S}$-extension of $H_j^\mathcal{S}$) for any $i \in [\![n]\!]$ (respectively $j \in [\![m]\!]$). This directly implies that $I$ is an $r$-dominating set of $G$. Indeed, the condition $(p_d)$ of Definition 19 implies that $C_i^\Omega$ (respectively $C_j^\mathcal{S}$) $r$-dominates $H_i^\Omega$ plus $v_i^\Omega$ (respectively of $H_j^\mathcal{S}$ plus $v_j^\mathcal{S}$, $\bar{v}_j^\mathcal{S}$). The remaining apex is also $r$-dominated by any $C_i^\Omega$, as it is $B_i^\Omega$-adjacent for every $i \in [\![n]\!]$.

We now have to show that $I$ $f$-distinguishes $G$. We begin with the vertices of the gadget copies because the condition $(p_h)$ implies that $C_i^\Omega \subseteq I$ $f$-distinguishes the vertices of $H_i^\Omega$ and $G$ for every $i \in [\![n]\!]$, and $I$ $f$-distinguishes the vertices of $H_j^\mathcal{S}$ and $G$ for every $j \in [\![m]\!]$. Thereby, we only have to study the vertices of the form $v_i^\Omega$, $v_j^\mathcal{S}$, $\bar{v}_j^\mathcal{S}$, and the apex $a$ (there is no vertex of the form $l_{i,j}^k$ in an apex distance identifying graph). To distinguish them, we use the condition $(p_b)$. Recall that $n > 1$. Then, for each distinct $i, i' \in [\![n]\!]$, we have:

- $v_i^\Omega$ is $B_i^\Omega$-adjacent but not $B_{i'}^\Omega$-adjacent,
- $a$ is both $B_i^\Omega$-adjacent and $B_{i'}^\Omega$-adjacent,
- a vertex of the form $v_j^\mathcal{S}$ or $\bar{v}_j^\mathcal{S}$ is neither $B_i^\Omega$-adjacent nor $B_{i'}^\Omega$-adjacent.

Enumerating the relevant $i$ and $i'$, we deduce that every couple of vertices is distinguished except when they are both of the form $v_j^\mathcal{S}$ or $\bar{v}_{j'}^\mathcal{S}$ for $j, j' \in [\![m]\!]$. But we may distinguish $v_j^\mathcal{S}$ or $\bar{v}_{j'}^\mathcal{S}$ for distinct $j, j'$ by applying $(p_b)$ on $H_j^\mathcal{S}$.

It remains to distinguish $v_j^\mathcal{S}$ and $\bar{v}_j^\mathcal{S}$ for $j \in [\![m]\!]$. We now use the fact that $P$ is a hitting set for $(\Omega, \mathcal{S})$. By definition of a hitting set, for any set $S_j \in \mathcal{S}$, there exists a vertex $u_i \in P$ such that $u_i \in S_j$. We observe that $d(v_i^\Omega, v_j^\mathcal{S}) = 1 < d(v_i^\Omega, \bar{v}_j^\mathcal{S})$ by construction of $G$ and that $v_i^\Omega \in I$ by definition of $I$. Since $f$ is 1-layered, $I$ $f$-distinguishes $v_j^\mathcal{S}$ and $\bar{v}_j^\mathcal{S}$.

In the other direction, assume that $I$ is a distance identifying set of $G$ of size $k+|C|(n+m)$. As every set of $S$ is not empty, we may define a function $\varphi : [\![m]\!] \to [\![n]\!]$ such that $u_{\varphi(j)} \in S_j$.

We suggest the following set $P$ as an hitting set of $\mathcal{S}$ of size at most $k$:

$$P = \{u_i \in \Omega \mid v_i^\Omega \in I\} \cup \{u_{\varphi(j)} \in \Omega \mid v_j^\mathcal{S} \in I \text{ or } \bar{v}_j^\mathcal{S} \in I\}$$

We claim that the only vertices that may $f$-distinguish $v_j^\mathcal{S}$ and $\bar{v}_j^\mathcal{S}$ are themselves and the vertices $v_i^\Omega$ such that $u_i \in S_j$. To prove so, we apply propriety $(\alpha)$ of Definition 5:

- the apex $a$ verifies $d(a, v_j^\mathcal{S}) = 1 = d(a, \bar{v}_j^\mathcal{S})$
- a vertex $v_i^\Omega$ such that $u_i \notin S_j$ verifies $d(v_i^\Omega, v_j^\mathcal{S}) = 3 = d(v_i^\Omega, \bar{v}_j^\mathcal{S})$
- a vertex $v$ of $H_i^\Omega$ verifies $d(v, v_j^\mathcal{S}) = 2 + d(v, B_i^\Omega) = d(v, \bar{v}_j^\mathcal{S})$
- a vertex $v$ of $H_{j'}^\mathcal{S}$ with $j \neq j'$ verifies $d(v, v_j^\mathcal{S}) = 3 + d(v, B_j^\mathcal{S}) = d(v, \bar{v}_j^\mathcal{S})$
- both $v_j^\mathcal{S}$ and $\bar{v}_j^\mathcal{S}$ are $B_j^\mathcal{S}$-adjacent, so they are at the same distance of any vertex of $H_j^\mathcal{S}$.

We deduce that $v_j^\mathcal{S}$ and $\bar{v}_j^\mathcal{S}$ are $f$-distinguished only if either one on them belongs to $I$ (in that case $u_{\varphi(j)} \in P \cap S_j$) or there exists $v_i^\Omega \in I$ such that $u_i \in S_j$ (and then $u_i \in P \cap S_j$).

It remains to show that $|P| \leq k$. By the condition $(p_s)$ of Definition 19, we know that $|I \cap V_{H_i^\Omega}| \geq |C_i^\Omega|$ and $|I \cap V_{H_j^\mathcal{S}}| \geq |C_j^\mathcal{S}|$ for any $i \in [\![n]\!]$ and $j \in [\![m]\!]$, implying

$$k = |I| - |C|(n+m) \geq \sum_{i \in [n]} |I \cap \{v_i^\Omega\}| + \sum_{j \in [m]} |I \cap \{v_j^\mathcal{S}, \bar{v}_j^\mathcal{S}\}| \geq \sum_{v_i^\Omega \in I} 1 + \sum_{I \cap \{v_j^\mathcal{S}, \bar{v}_j^\mathcal{S}\} \neq \emptyset} 1 = |P|$$

The equivalence between the first and third points is proven in [4]. ◀

## 3.2 Binary Compression of Gadgets

The Theorem 21 is a powerful tool to get reductions, in particular in the planar cases. However, the number of involved gadgets does not allow to use Theorem 1. This limitation is due to the uses of a gadget per vertex to identify in the distance identifying graphs. Using power set, we may obtain a better order: given $k$ gadgets, we may identify $2^k - 1$ vertices (we avoid to identify a vertex with the empty subset of gadgets). Thus, we will consider *binary representations* of integers as sequences of bits, with weakest bit at last position. For a positive integer $n$, we define the integer $\lg_n = 1 + \lfloor \log_2(n) \rfloor$ and introduce a new graph:

▶ **Definition 22** ($(H, B, r)$-compressed graph)**.** Let $(\Omega = \{u_i \mid i \in [\![n]\!]\}, \mathcal{S} = \{S_i \mid i \in [\![m]\!]\})$ be an instance of Hitting Set. Let $H$ be a connected graph, $B$ be a subset of its vertices, and $r$ be a positive integer. The $(H, B, r)$*-compressed graph* $\Psi[H, B, r](\Omega, \mathcal{S})$ is defined as follows. $\Psi[H, B, r](\Omega, \mathcal{S})$ contains as induced subgraphs $\lg_{n+1}$ copies of $H$ denoted $H_i^\Omega$ for $i \in [\![\lg_{n+1}]\!]$ and $\lg_m$ another copies of $H$ denoted $H_j^\mathcal{S}$ for $j \in [\![\lg_m]\!]$. Then:

- for each $j \in [\![m]\!]$, we add two non-adjacent vertices $v_j^\mathcal{S}$ and $\bar{v}_j^\mathcal{S}$. They are $B_k^\mathcal{S}$-adjacent for each $k \in [\![\lg_m]\!]$ such that the $k^{th}$ bit of the binary representation of $j$ is 1.
- for each $i \in [\![n]\!]$, we add $r$ vertices denoted $l_i^{j-1}$ with $j \in [\![r]\!]$ to form a fresh path such that $d(v_i^\Omega, l_i^{j-1}) = j - 1$ where $v_i^\Omega = l_i^0$. We make $v_i^\Omega$ $B_k^\Omega$-adjacent for each $k \in [\![\lg_{n+1}]\!]$ such that the $k^{th}$ bit of the binary representation of $i$ is 1.
- for each $S_j \in \mathcal{S}$ and each $u_i \in S_j$, we add the edge $(l_i^{r-1}, v_j^\mathcal{S})$.
- we add $r$ vertices denoted $a^{j-1}$ with $j \in [\![r]\!]$ to form a path such that $d(a^0, a^{j-1}) = j - 1$. The vertex $a^0$ is $B_k^\Omega$-adjacent for every $k \in [\![\lg_{n+1}]\!]$, and we add the edges $(a^{r-1}, v_j^\mathcal{S})$ and $(a^{r-1}, \bar{v}_j^\mathcal{S})$ for each $j \in [\![m]\!]$.

By definition of $\lg_{n+1}$, for every $i \in [\![n]\!]$, one of the last $\lg_{n+1}$ bits of the binary representation of $i$ is 0. So, $a^0$ has a distinct characterization in the power set formed by the gadgets $H_i^\Omega$.

▶ **Proposition 23.** *The graph $\Psi[H, B, r](\Omega, \mathcal{S})$ built on an instance $(\Omega, \mathcal{S})$ of* Hitting Set

- *is connected and has size at most $|H|(\lg_{n+1} + \lg_m) + r(n + 1) + 2m$, where $|\Omega| = n$, $|\mathcal{S}| = m$*
- *may be built in polynomial time in its size,*
- *is bipartite if the $B$-single extension of $H$ is bipartite.*

▶ **Theorem 24.** *Let $(\Omega, \mathcal{S})$ be an instance of* Hitting Set *such that $|\Omega| = n$, $|\mathcal{S}| = m$. Let $(H, B, C)$ be a $(f, r)$-gadget for a 1-layered identifying function $f$ and let $(H', B', C')$ be a local $(g, q)$-gadget. The following propositions are equivalent:*

- *there exists a hitting set of $\mathcal{S}$ of size $k$.*
- *there exists a $(f, r)$-distance identifying set of $\Psi[H, B, 1](\Omega, \mathcal{S})$ of size $k + |C|(\lg_{n+1} + \lg_m)$.*
- *there exists a $(q, r)$-distance identifying set of $\Psi[H', B', q](\Omega, \mathcal{S})$ of size $k + |C'|(\lg_{n+1} + \lg_m)$.*

# 4 On Providing Gadgets to Establish Generic Reductions

In this section, we finalize the reductions by furnishing some gadgets and combining them with the suitable theorems and propositions from Section 3. We directly define the gadgets.

▶ **Definition 25** (The 1-layered gadget)**.** Let $H$ be the bipartite planar graph such that:

- Its ten vertices are denoted $b$, $\bar{b}$, $u_1$, $\bar{u}_1$, $u_2$, $\bar{u}_2$, $v_1$, $\bar{v}_1$, $v_2$ and $\bar{v}_2$,
- The vertices $u_1, u_2, \bar{u}_1$ and $\bar{u}_2$ form a cycle as well as the vertices $v_1, v_2, \bar{v}_1$ and $\bar{v}_2$.
- The vertices $b$ and $\bar{b}$ are adjacent to $u_1$, $\bar{u}_1$, $v_1$ and $\bar{v}_1$.

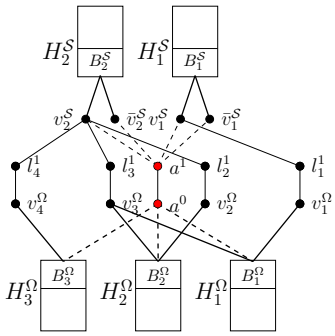



**Figure 3** A (H,B,2)-compressed graph built with $\Omega = \{1, 2, 3, 4\}$ and $\mathcal{S} = \{\{1, 2\}, \{2, 3, 4\}\}$.

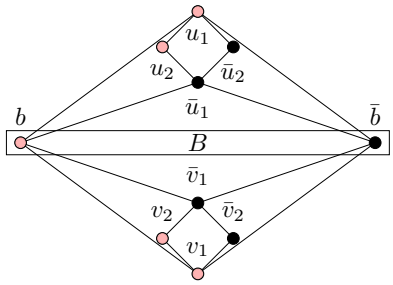



**Figure 4** The 1-layered gadget $(H, B, C)$. $C$ contains the colored vertices.

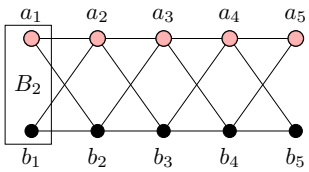



**Figure 5** The 2-local 0-layered gadget. $C_2$ contains the colored vertices.

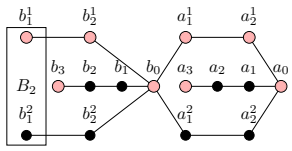



**Figure 6** The 2-IC gadget. $C_2$ contains the colored vertices.

We define the sets $B = \{b, \bar{b}\}$ and $C = \{b, u_1, u_2, v_1, v_2\}$.
The triple $(H, B, C)$ is called *the* 1*-layered gadget* (see Fig. 4).

▶ **Definition 26** (The $r$-local 0-layered gadget). Given an integer $r > 1$ (respectively $r = 1$), let $H_r$ be the bipartite planar graph of size $4r + 2$ (respectively 8) such that:

- its vertices are denoted $a_i$ and $b_i$ for $i \in [\![2r + 1]\!]$ (respectively $i \in [\![4]\!]$),
- for each $i \in [\![2r]\!]$ (respectively $i \in [\![3]\!]$), both $a_i$ and $b_i$ are adjacent to $a_{i+1}$ and $b_{i+1}$

We define the sets $B_r = \{a_1, b_1\}$ and $C_r = \{a_i \mid i \in [\![2r + 1]\!]\}$ (respectively $C_1 = \{a_1, a_2, a_3, a_4\}$). The triple $(H_r, B_r, C_r)$ is called *the* $r$*-local* 0*-layered gadget* (see Fig. 5).

For each positive integer $r$, $r$-LD and $r$-MD are $r$-local 0-layered problems, whereas $r$-IC is not 0-layered. We define a specific gadget for this remaining problem.

▶ **Definition 27** (The $r$-IC gadget). Given a positive integer $r$, let $H_r$ be the bipartite planar graph of size $6r + 4$ such that:

- its vertices are denoted $a_{i-1}$ and $b_{i-1}$ for $i \in [\![r + 2]\!]$, and $a_i^j$ and $b_i^j$ for $i \in [\![r]\!]$ and $j \in [\![2]\!]$. We also denote $a_0$ as $a_{r+1}^1$ and $a_{r+1}^2$ and we denote $b_0$ as $b_{r+1}^1$, $b_{r+1}^2$, $a_0^1$ and $a_0^2$.
- the edges are all included in the six following paths
  - from $a_0$ to $a_{r+1}$ such that $d(a_0, a_i) = i$ for $i \in [\![r + 1]\!]$.
  - from $b_0$ to $b_{r+1}$ such that $d(b_0, b_i) = i$ for $i \in [\![r + 1]\!]$.
  - from $a_0^1$ to $a_{r+1}^1$ such that $d(a_0^1, a_i^1) = i$ for $i \in [\![r + 1]\!]$.
  - from $a_0^2$ to $a_{r+1}^2$ such that $d(a_0^2, a_i^2) = i$ for $i \in [\![r + 1]\!]$.
  - from $b_1^1$ to $b_{r+1}^1$ such that $d(b_1^1, b_i^1) = i - 1$ for $i \in [\![r + 1]\!]$.
  - from $b_1^2$ to $b_{r+1}^2$ such that $d(b_1^2, b_i^2) = i - 1$ for $i \in [\![r + 1]\!]$.

We define the sets $B_r = \{b_1^1, b_1^2\}$ and $C_r = \{a_{r+1}, b_{r+1}\} \cup \bigcup_{i \in [\![r+1]\!]} \{a_i^1, b_i^1\}$.
The triple $(H_r, B_r, C_r)$ is called *the* $r$-IC *gadget* (see Fig. 6).

As expected, we have the following propositions:

▶ **Proposition 28.** *The* 1*-layered gadget is a bipartite planar* $(f, r)$*-gadget for any* 1*-layered distance identifying function* $f$ *and* $r \in [\![\infty]\!]$.

▶ **Proposition 29.** *Given a positive integer* $r$*, the* $r$*-local* 0*-layered gadget is a local bipartite planar* $(f, r)$*-gadget for every* $r$*-local* 0*-layered distance identifying function* $f$.

▶ **Proposition 30.** *Given a positive integer* $r$*, the* $r$-IC *gadget is a local bipartite planar* $(f, r)$*-gadget for the identifying function* $f$ *associated with* $r$-IC*, where* $f_G[w](u, v) = \text{true}$ *if* $w \in N_r[u] \Delta N_r[v]$ *for relevant inputs* $G$*,* $u$*,* $v$ *and* $w$.

With Propositions 28 to 30, we can now prove the Theorems 9 to 12. We focus here on 1-layered distance identifying functions and show the other reductions in [4].

**Proof of Theorems 9 and 12 for each 1-layered identifying function $f$ and $r \in [\![\infty]\!]$.** We first suggest a reduction from Planar Hitting Set to $(f, r)$-DIS based on the bipartite planar 1-layered gadget $(H, B, C)$. Let $(\Omega, \mathcal{S})$ be an instance of Planar Hitting Set with $|\Omega| = n$ and $|\mathcal{S}| = m$ such that $m = \mathcal{O}(n)$. According to Proposition 18, the bipartite apex graph $G = \Phi^*[H, B](\Omega, \mathcal{S})$ has size $n'$ linear in $n + m = \mathcal{O}(n)$ and may be built in polynomial-time in its size. Recall that $(H, B, C)$ is a $(f, r)$-gadget by Proposition 28. By Theorem 21, $G$ admits a $(f, r)$-distance identifying set of size $k' = k + |C|(n + m)$ if an only if $\mathcal{S}$ admits an hitting set of size $k$. Thus, an algorithm solving $(f, r)$-DIS in $2^{o(\sqrt{n'})}$ would solve Planar Hitting Set in time $2^{o(\sqrt{n})}$, a contradiction to Theorem 3 (assuming ETH).

We adapt the previous argumentation to get a reduction from Hitting Set to $(f, r)$-DIS, the instance $(\Omega, \mathcal{S})$ belonging now to the Hitting Set problem. According to Proposition 23, the bipartite graph $G = \Psi[H, B, 1](\Omega, \mathcal{S})$ has size $n'$ linear in $n + m = \mathcal{O}(n)$ and may also be built in polynomial-time in its size. By Theorem 24, $G$ admits a $(f, r)$-distance identifying set of size $k' = k + |C|(\lg_{n+1} + \lg_m)$ if an only if $\mathcal{S}$ admits an hitting set of size $k$. Thus, an algorithm solving $(f, r)$-DIS in $2^{o(n')}$ would solve Hitting Set in time $2^{o(n)}$, contradicting Theorem 2 when assuming ETH. Moreover, a parameterized algorithm solving $(f, r)$-DIS in $2^{\mathcal{O}(k)} \cdot {n'}^{\mathcal{O}(1)}$ would be in contradiction with Theorem 1 when assuming $\mathsf{W}[2] \neq \mathsf{FPT}$. ◀

## 5 Conclusion

In this paper, we showed generic tools to analysis identifying problems and their computational lower bounds. This study opens some new questions. First of all, we observe that our toolbox does not contain a $r$-local gadget. Does one exist? Furthermore, there is still a gap between the computational lower bound provided by Theorem 12 and the elementary upper bound from Proposition 14 in the local cases. We wonder if local problems may be solved in $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$. Notice that a polynomial kernel would imply such a complexity (but the reciprocal is not true). For non-local problems, an FPT upper bound is globally unknown. In particular, W[2]-hard problems like MD cannot admit FPT algorithms unless $\mathsf{W}[2] = \mathsf{FPT}$. Then, which non-local problem is W[2]-hard? We mention that we actually get a FPT reduction from Hitting Set to some scarce non-local problems (however including MD) proving their W[2]-hardness, but the family of involved problems is not precise nor wide. Nevertheless, we remark that most of our reductions may be generalized to the oriented version of Distance Identifying Set sometimes even for the strongly connected graphs –this is due to the fact that the paths in our distance identifying graphs and gadgets may often be seen as oriented–. Thus, we inform the community that the oriented version of MD (studied for Cayley graphs in [15]) remains W[2]-hard.

## References


1. David Auger. Minimal identifying codes in trees and planar graphs with large girth. *Eur. J. Comb.*, 31(5):1372–1384, 2010. doi:10.1016/j.ejc.2009.11.012.
2. László Babai. On the Complexity of Canonical Labeling of Strongly Regular Graphs. *SIAM J. Comput.*, 9(1):212–216, 1980. doi:10.1137/0209018.
3. Evangelos Bampas, Davide Bilò, Guido Drovandi, Luciano Gualà, Ralf Klasing, and Guido Proietti. Network verification via routing table queries. *J. Comput. Syst. Sci.*, 81(1):234–248, 2015. doi:10.1016/j.jcss.2014.06.003.
4. Florian Barbero, Lucas Isenmann, and Jocelyn Thiebaut. On the Distance Identifying Set meta-problem and applications to the complexity of identifying problems on graphs. *ArXiv e-prints*, October 2018. arXiv:1810.03868.
5. Zuzana Beerliova, Felix Eberhard, Thomas Erlebach, Alexander Hall, Michael Hoffmann, Matús Mihalák, and L. Shankar Ram. Network Discovery and Verification. *IEEE Journal on Selected Areas in Communications*, 24(12):2168–2181, 2006. doi:10.1109/JSAC.2006.884015.
6. Béla Bollobás and Alex D. Scott. On separating systems. *Eur. J. Comb.*, 28(4):1068–1071, 2007. doi:10.1016/j.ejc.2006.04.003.
7. J.A Bondy. Induced subsets. *Journal of Combinatorial Theory, Series B*, 12(2):201–202, 1972. doi:10.1016/0095-8956(72)90025-1.
8. Emmanuel Charbit, Irène Charon, Gérard D. Cohen, Olivier Hudry, and Antoine Lobstein. Discriminating codes in bipartite graphs: bounds, extremal cardinalities, complexity. *Adv. in Math. of Comm.*, 2(4):403–420, 2008. doi:10.3934/amc.2008.2.403.
9. Irène Charon, Olivier Hudry, and Antoine Lobstein. Minimizing the size of an identifying or locating-dominating code in a graph is NP-hard. *Theor. Comput. Sci.*, 290(3):2109–2120, 2003. doi:10.1016/S0304-3975(02)00536-4.
10. Gérard D. Cohen, Iiro S. Honkala, Antoine Lobstein, and Gilles Zémor. On identifying codes. In *Codes and Association Schemes, Proceedings of a DIMACS Workshop, Piscataway, New Jersey, USA, November 9-12, 1999*, pages 97–110, 1999.
11. Charles J Colbourn, Peter J Slater, and Lorna K Stewart. Locating dominating sets in series parallel networks. *Congr. Numer*, 56:135–162, 1987.
12. Josep Díaz, Olli Pottonen, Maria J. Serna, and Erik Jan van Leeuwen. On the Complexity of Metric Dimension. In *Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*, pages 419–430, 2012. doi:10.1007/978-3-642-33090-2_37.
13. Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. doi:10.1007/978-1-4612-0515-9.
14. Leah Epstein, Asaf Levin, and Gerhard J. Woeginger. The (Weighted) Metric Dimension of Graphs: Hard and Easy Cases. *Algorithmica*, 72(4):1130–1171, 2015. doi:10.1007/s00453-014-9896-2.
15. Melodie Fehr, Shonda Gosselin, and Ortrud R Oellermann. The metric dimension of Cayley digraphs. *Discrete Mathematics*, 306(1):31–41, 2006.
16. Florent Foucaud, George B. Mertzios, Reza Naserasr, Aline Parreau, and Petru Valicov. Identification, Location-Domination and Metric Dimension on Interval and Permutation Graphs. II. Algorithms and Complexity. *Algorithmica*, 78(3):914–944, 2017. doi:10.1007/s00453-016-0184-1.
17. M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
18. Frank Harary and RA Melter. On the metric dimension of a graph. *Ars Combin*, 2(191-195):1, 1976.

19 Sepp Hartung. *Exploring parameter spaces in coping with computational intractability.* PhD thesis, Fakultät Elektrotechnik und Informatik der Technischen Universität Berlin, December 2014.

20 Sepp Hartung and André Nichterlein. On the Parameterized and Approximation Hardness of Metric Dimension. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 266–276, 2013. doi:10.1109/CCC.2013.36.

21 Stefan Hoffmann and Egon Wanke. Metric Dimension for Gabriel Unit Disk Graphs Is NP-Complete. In *Algorithms for Sensor Systems, 8th International Symposium on Algorithms for Sensor Systems, Wireless Ad Hoc Networks and Autonomous Mobile Entities, ALGOSENSORS 2012, Ljubljana, Slovenia, September 13-14, 2012. Revised Selected Papers*, pages 90–92, 2012. doi:10.1007/978-3-642-36092-3_10.

22 Mark G. Karpovsky, Krishnendu Chakrabarty, and Lev B. Levitin. On a New Class of Codes for Identifying Vertices in Graphs. *IEEE Trans. Information Theory*, 44(2):599–611, 1998. doi:10.1109/18.661507.

23 Jeong Han Kim, Oleg Pikhurko, Joel H. Spencer, and Oleg Verbitsky. How complex are random graphs in first order logic? *Random Struct. Algorithms*, 26(1-2):119–145, 2005. doi:10.1002/rsa.20049.

24 Peter J Slater. Leaves of trees. *Congr. Numer*, 14(549-559):37, 1975.

25 Peter J. Slater. Domination and location in acyclic graphs. *Networks*, 17(1):55–64, 1987. doi:10.1002/net.3230170105.

26 Peter J Slater. Dominating and reference sets in a graph. *J. Math. Phys. Sci*, 22(4):445–455, 1988.

27 Simon Tippenhauer. *On Planar 3-SAT and its Variants.* PhD thesis, Freien Universität Berlin, 2016. URL: https://www.mi.fu-berlin.de/inf/groups/ag-ti/theses/download/Tippenhauer16.pdf.

28 Rachanee Ungrangsi, Ari Trachtenberg, and David Starobinski. An Implementation of Indoor Location Detection Systems Based on Identifying Codes. In *Intelligence in Communication Systems, IFIP International Conference, INTELLCOMM 2004, Bangkok, Thailand, November 23-26, 2004, Proceedings*, pages 175–189, 2004. doi:10.1007/978-3-540-30179-0_16.

# The Parameterized Complexity of Finding Point Sets with Hereditary Properties

**David Eppstein**[1]
Computer Science Department, University of California, Irvine, USA
eppstein@uci.edu

**Daniel Lokshtanov**[2]
Department of Informatics, University of Bergen, Norway
dlokshtanov@gmail.com

**Abstract**

We consider problems where the input is a set of points in the plane and an integer $k$, and the task is to find a subset $S$ of the input points of size $k$ such that $S$ satisfies some property. We focus on properties that depend only on the order type of the points and are monotone under point removals. We exhibit a property defined by three forbidden patterns for which finding a $k$-point subset with the property is W[1]-complete and (assuming the exponential time hypothesis) cannot be solved in time $n^{o(k/\log k)}$. However, we show that problems of this type are fixed-parameter tractable for all properties that include all collinear point sets, properties that exclude at least one convex polygon, and properties defined by a single forbidden pattern.

**2012 ACM Subject Classification** Theory of computation → Design and analysis of algorithms

**Keywords and phrases** parameterized complexity, fixed-parameter tractability, point set pattern matching, largest pattern-avoiding subset, order type

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.11

## 1 Introduction

In this work, we study the parameterized complexity of finding subsets of planar point sets having hereditary properties, by analogy to past work on hereditary properties of graphs.

Hereditary properties of graphs have long been a central organizing principle for such diverse topics as coloring, perfection, intersection graph theory, and the theories of claw-free and triangle-free graphs. Finding the largest induced subgraph with any hereditary property is NP-hard [11, 21]. The parameterized complexity of finding a $k$-vertex induced subgraph with a given hereditary property was resolved by Khot and Raman [19]. If all cliques and all independent sets have the property, then only graphs of bounded size can fail to include a $k$-vertex induced subgraph with the property. If there exist a clique and an independent set that do not have the property, then only bounded-size induced subgraphs can have the property. In all remaining cases the problem is W[1]-complete. For hereditary properties with finitely many minimal forbidden induced subgraphs, removing $k$ vertices so that the remaining induced subgraph has the property is always fixed-parameter tractable, as a hitting set problem [1, 10]. The parameterized complexity of problems of this type with more than finitely many obstacles, such as deletion to a planar graph ($k$-apex graph recognition) [5, 17] or to a bipartite graph (odd cycle transversal) [18, 20], also remains of interest.

[1] Supported in part by NSF grants CCF-1618301 and CCF-1616248.
[2] Supported by Pareto-Optimal Parameterized Algorithms, ERC Starting Grant 715744

© David Eppstein and Daniel Lokshtanov;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 11; pp. 11:1–11:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Point sets in the Euclidean plane can be described combinatorially by their *order type*, the specification for each ordered triple of points of whether that triple forms the vertices of a triangle in clockwise or counterclockwise order, or whether the three points are collinear. It is natural to define a hereditary property of point sets to be a property that depends only on the order type, and that remains true for every subset of a point set having the property. This concept of hereditary properties has been adopted recently as a central organizing principle in discrete geometry [8] and encompasses many famous and well-studied problems:

- The *happy ending problem* concerns conditions that force some $k$-point subset to be in convex position [9, 24]. The property of not containing a $k$-point convex subset is hereditary [8, Chapter 11]. It can be tested in polynomial time for points in the plane [6,7], but is NP-hard in three dimensions, and finding the largest 3d $k$-point convex set that does not enclose any other input points is W[1]-hard [12].
- The *no-three-in-line problem* concerns the size of the largest general-position subset of an $n \times n$ grid [15, 23]. Here, "general position" means having no three collinear points. The property of not containing a $k$-point general-position subset is hereditary [8, Chapter 9].
- The *Erdős–Ulam problem* asks whether there exist dense sets with all distances rational [25]; *Harborth's conjecture* is that every planar graph can be drawn with non-crossing straight edges of rational length [16]. The existence of a rational-distance point set with the same order type as the given point set is hereditary. If every general-position order type has a rational-distance realization then Harborth's conjecture follows, and if not then the Erdős–Ulam problem has a negative answer [8, Section 13.5].

Every hereditary property can be defined by its *forbidden patterns*, the point-minimal order types that do not have the property. The first author's recent book on these problems asked whether every problem of finding $k$ points with a hereditary property defined by finitely many forbidden patterns is fixed-parameter tractable [8, Open Problem 7.6]. We answer this question negatively, by finding a property for which (under standard complexity-theoretic assumptions) no FPT algorithm exists. However, we show that several natural classes of properties have fixed-parameter tractable algorithms. Specifically, we show:

- There exists a hereditary property $\Pi$, defined by finitely many forbidden patterns, such that finding a $k$-point subset with property $\Pi$ is W[1]-complete and (assuming the exponential time hypothesis) cannot be solved in time $n^{o(k/\log k)}$).
- Finding a $k$-point subset with a given hereditary property is fixed-parameter tractable whenever all collinear sets have the property.
- Finding a $k$-point subset with a given hereditary property is fixed-parameter tractable whenever there exists a convex polygon that does not have the property.
- Finding a $k$-point subset that avoids a single forbidden pattern is fixed-parameter tractable.

As with the analogous graph problems, the dual problem of removing $k$ points to make the remaining points have a given property is fixed-parameter tractable whenever the property has finitely many forbidden patterns, as a hitting set problem for the copies of the patterns within the point set [8, Theorem 7.9].

Our algorithmic results apply to any model of point set input that allows us to determine the orientation of a triple of points in constant (or polynomial) time. For points given by Cartesian coordinates, these orientations can be determined from the sign of a quadratic polynomial of these coordinates. Not all order types can be realized with integer coordinates, and even when this is possible a realization may need very large coordinates [13], but our algorithms could instead use inputs that specify an $n \times n \times n$ three-dimensional array of orientations. The point sets used for our hardness results and lower bounds can be constructed using explicit integer coordinates of polynomial magnitude.

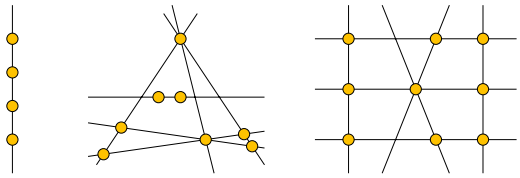


**Figure 1** The three forbidden patterns for a compliant point set.

## 2 A hard property

The property for which we will prove finding a $k$-point subset hard is defined by the three forbidden patterns in Figure 1: four points in a line (left), an eight-point pattern based on a complete quadrilateral[3] (center), and a nine-point pattern resembling a tic-tac-toe board or $3 \times 3$ grid, but with the central vertical line broken rather than straight (right). We call a set of points *compliant* if it does not contain any of these three patterns.

The hard inputs to our problem will be point sets of a special form, which we call *yards*. A yard consists of points with the following properties:

- Most of its points are arranged onto horizontal lines of at least three points per line, which we call *rows*. We call the leftmost and rightmost points of each row the *guards* of the row (or more specifically the left and right guards, respectively). We call the remaining points, that are neither leftmost nor rightmost in their row, the *inmates*.
- Some triples of points from different rows form non-horizontal lines of three points, but there are no non-horizontal lines of four points.
- Every line between two inmates of different rows passes completely above or completely below all left guards, and completely below or completely above (respectively) all right guards. Similarly, every line through two left guards or through two right guards passes completely above or completely below all inmates. (These points may determine vertical lines, which we consider to pass above and below all points.)
- The only lines of three or more points with both inmate and guard points are rows.
- No subset of nine guards forms the configuration on the right of Figure 1.
- In addition to the points on rows, the yard has six additional points, forming a complete quadrilateral. We call these points the *fence*.
- The only lines of three or more points that include points of the fence are the four lines from which the fence was defined. No other six points of the yard form a complete quadrilateral.
- The fence encloses the inmates but not the guards. Every two inmates on a single row, plus the fence, form the middle forbidden pattern of Figure 1, and every instance of the middle forbidden pattern is formed in this way.

An example of a yard is shown in Figure 2. For a yard with $r$ rows, we define a *lineup* to be a compliant subset of $k = 3r + 6$ points from the yard.

▶ **Lemma 1.** *Every lineup must consist of all six points of the fence, and two guards and one inmate from each row. Whenever three rows have the property that their three left guards and their three right guards form two collinear triples of points, the three inmates of the lineup from these rows must also be collinear.*

[3] A *complete quadrilateral* consists of four lines and six points, with each line containing three points and each point contained in two lines. It can be constructed from four Euclidean lines, no two parallel and no three crossing in a single point, by including the six crossing points of pairs of lines.

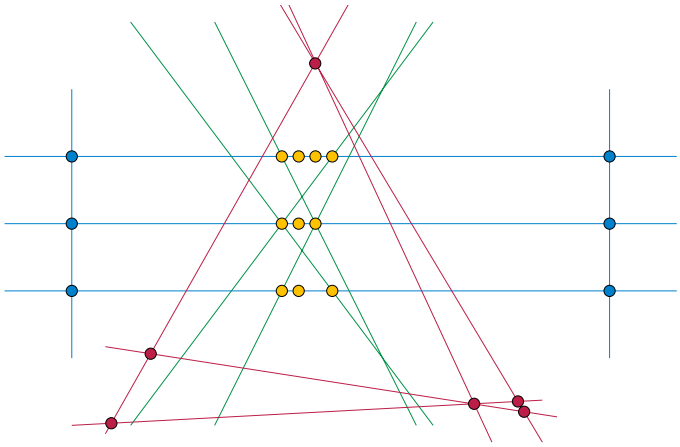


**Figure 2** A yard with three rows. The fence and its points are shown in red; the rows and their guards are blue, and the inmates on each row are yellow. The green lines are the most far from vertical among the lines through pairs of inmates that are not both on the same row as each other; they can be used to verify that all non-horizontal and non-vertical lines through pairs of inmates pass above or below the guards on each side, and that no pair of inmates on different rows from each other forms a forbidden pattern with the fence. By Lemma 1, every lineup in this yard consists of all of the blue and red points together with three collinear yellow points, one from each row. In this example all the left guards and all the right guards are collinear, but in larger examples only certain triples of guards would be collinear.

**Proof.** The forbidden four-point line forces the lineup to contain at most three points from each row, so if it contains $3r + 6$ points it must include exactly three points from each row and all six points of the fence. The forbidden pattern in the form of a complete quadrilateral plus two points forces the lineup to contain at most one inmate from each row, so in order for it to contain three points from each row it must contain both guards.

If the three left guards and the three right guards of a triple of rows are collinear (as they are in Figure 2), then the three inmates of the lineup from the same three rows must also be collinear. For otherwise, they would form the forbidden pattern on the right of Figure 1, either as shown or under a 180° rotation (which causes no change to the order type of the configuration). Therefore, every such triple of inmates must be collinear. ◀

Lemma 1 is not a necessary and sufficient condition (for instance, it is possible to have a subset of points that meets the condition of the lemma but contains a copy of the rightmost forbidden pattern, rotated by 90°) but we will see that it is necessary and sufficient for the yards constructed by the reduction that we describe next.

We prove hardness of finding lineups in yards via a parameterized reduction from *partitioned subgraph isomorphism*. This problem's input consists of two graphs, a *host graph* $H$ and a pattern graph $G$, whose sizes (numbers of edges) are $n$ and $k$ respectively. The vertices of $G$ and $H$ are colored, with all vertices of $G$ having distinct colors. The problem is to find a subgraph of $H$ that is isomorphic to $G$ and matches it in coloring. Thus, the color classes of $H$ partition its vertices. We must choose a representative vertex from each color class so that whenever two vertices in $G$ are adjacent their representatives in $H$ are also adjacent. The color partition of the vertices in $H$ is what gives rise to the name, partitioned subgraph isomorphism. In our hardness reduction, we will consider a restricted case of this problem, *cubic partitioned subgraph isomorphism*, in which we require the pattern graph $G$ to be a cubic (that is, 3-regular) graph. $H$ is not restricted. This problem is W[1]-complete and, assuming the exponential time hypothesis, cannot be solved in time $n^{o(k/\log k)}$.

Cubic partitioned subgraph isomorphism has been used for the same sort of hardness reduction previously [3, 4]. The W[1]-completeness of partitioned subgraph isomorphism appears to be folklore, but it has a straightforward proof by reduction from the W[1]-complete clique problem.[4] In turn, partitioned subgraph isomorphism on graphs of minimum degree at least three can be reduced to the 3-regular case by expanding each high-degree vertex into a 3-regular tree, as follows:

- Choose for each vertex $v$ of the pattern graph a tree $T_v$ having degree three at each internal vertex and having the neighbors of $v$ as its leaves. Let $T'_v$ be the subtree of $T_v$ consisting only of the interior nodes of $T_v$.
- Form the disjoint union of the trees $T'_v$. For each edge $uv$ of the pattern graph add an edge to this union, connecting the node in $T'_u$ to which the leaf for $v$ is attached to the node in $T'_v$ to which the leaf for $u$ is attached.
- For each vertex $w$ of the host graph, having the same color as a vertex $v$ of the pattern graph, make a tree $T'_w$ isomorphic to $T'_v$. Form a new host graph from the disjoint union of these trees $T'_w$.
- For each pair of vertices $wx$ of the host graph such that $w$ has the same color as a vertex $u$ of the pattern graph, $x$ has the same color as a vertex $v$ of the pattern graph, $u$ and $v$ are adjacent, and $w$ and $x$ are adjacent, add an edge to this union, connecting the node in $T'_w$ corresponding to the one in $T'_u$ to which the leaf for $v$ would be attached with the node in $T'_x$ corresponding to the one in $T'_v$ to which the leaf for $u$ would be attached.

The ETH-based lower bounds on cubic partitioned subgraph isomorphism come from Corollary 6.1 of Marx [22], using the fact that a cubic expander graph $G$ has treewidth $\Omega(k)$ [14].

To translate an instance of cubic partitioned subgraph isomorphism to the problem of finding lineups in yards, we construct a yard with a row for each vertex or edge of the pattern graph $G$. We will arrange the guards and inmates of the yard in such a way that collinearities of guards correspond to vertex-edge incidences in the pattern graph, and collinearities of inmates correspond to vertex-edge incidences in the host graph. In this way, every lineup of the yard will necessarily correspond to a solution to the partitioned subgraph isomorphism instance. We take some care in the construction in order to ensure that the resulting yard uses only points with integer coordinates of small magnitude.

The rows of the yard will be placed on horizontal lines, with positive integer $y$-coordinate $y_v$ or $y_{uv}$ for each vertex $v$ or edge $uv$ of the pattern graph. It will be convenient to be able to place arbitrary integer-coordinate points on two vertex rows and have the intersection of the line through these points with an edge row automatically be an integer-coordinate point. To ensure this property, we will always choose $y_{uv} = 2y_v - y_u$ (where $y_v > y_u$). However, this requires that we choose the coordinates $y_u$ and $y_v$ carefully, so that no two vertices or edges have rows with equal $y$-coordinates. We make this choice greedily, choosing the coordinates $y_v$ for the vertices in an arbitrary order. For each vertex $v$ in this order, we choose $y_v$ to be the smallest positive integer that obeys the following constraints:

- $y_v$ is unequal to any $y_u$ for an earlier vertex $u$
- $y_v$ is unequal to any $y_{uw}$ for any two adjacent earlier vertices $u$ and $w$

[4] The reduction uses the tensor product product $G \times H$ of graphs $G$ and $H$. This product has as vertices the pairs $(u, v)$ where $u$ is a vertex in $G$ and $v$ is a vertex in $H$. Pairs $(u, v)$ and $(u', v')$ are adjacent if and only if either $u = u'$ and $v$ is adjacent to $v'$, or $v = v'$ and $u$ is adjacent to $u'$. If we properly color $K_k$ and assign each vertex $(u, v)$ in $G \times K_k$ the same color as $v$, then $G$ has a $k$-clique if and only if the partitioned subgraph isomorphism instance for $G \times K_k$ and $K_k$ (with these colors) has a solution. This reduction proves the W[1]-completeness of partitioned subgraph isomorphism problem from the known completeness of the clique problem.

- $y_v$ does not make any $y_u = y_{vx}$ for any earlier vertex $u$ and earlier neighbor $x$ of $v$
- $y_v$ does not make $y_{uw} = y_{vx}$ for any three earlier vertices $u$, $w$, or $x$ with $u$ adjacent to $w$ and $x$ to $v$.

These conditions are each determined by at most one earlier vertex $u$, at most one of three neighbors $w$ of $u$, and at most one of three neighbors $x$ of $v$. For each combination of these vertices, each condition causes one integer to be unavailable as a choice for $y_v$. Therefore, $O(k)$ positive integers are unavailable. Because our greedy algorithm chooses the first available integer in each case, it chooses all coordinates $y_v$ and $y_{uv}$ to have magnitude $O(k)$. Our actual row coordinates will be scaled and translated from these values, but both of these kinds of transformations preserve the property that two integer points on vertex rows determine an integer point on an edge row.

The first points we determine in the construction are the six points of the fence. We construct a complete quadrilateral with integer coordinates, by using lines with rational coefficients and then clearing denominators. We scale it by a sufficiently large factor so that there are as many integer-coordinate horizontal lines through the fence as we need rows (given the assignment of row coordinates to vertices and edges of the pattern graph) and so that each row intersects the region inside the fence in a segment of length proportional to the diameter of the fence. We scale these fences and rows a second time, by a polynomially-large factor $\phi$, to ensure that the following steps can be carried out using integer coordinates for the remaining points. We leave $\phi$ unspecified for now, and will later state constraints on how large it needs to be for the construction to work.

Next, we choose three intervals on the $x$-coordinate axis: a left interval that will contain the $x$-coordinates of all left guards, a middle interval that will contain the $x$-coordinates of all inmates, and a right interval that will contain the $x$-coordinates of all right guards. The middle interval should be chosen so that its intersection with all row lines is inside the fence (using the fact that there is a vertical line segment inside the fence that crosses all row lines) and the left and right intervals should intersect all row lines outside the fence, on either side of it. inside the fence and the left and right intervals should be outside it, on either side of the fence. These intervals should be sufficiently narrow that all lines through pairs of points on corresponding intervals of different rows pass in the correct way above or below the points of the fence and the intervals of other types. In order to prevent an inmate from being collinear with two guards from other rows than its own, we also require that there be no line that crosses a row line in the left interval, a second row line in the middle interval, and a third row line in the right interval. In order to achieve this, it will be sufficient to choose the interval widths to be smaller than the diameter of the fence by a sufficiently large factor $\rho$. With this factor, we can choose the left and right guard intervals first, arbitrarily subject to the above constraints. Given this choice, let $C$ be the set of crossing points of the rows with non-horizontal lines through potential left and right guard points. Then $C$ is a union of a cubic number of intervals (one for each triple of rows). Each of these intervals defining $C$ is wider than the left and right intervals by a factor of $O(k)$, because the slopes of the lines through points within the left and right intervals on different rows are bounded away from zero by $\Omega(1/k)$. Therefore, the projection of $C$ onto the $x$-axis has length $O(k^4/\rho)$ relative to the diameter of the fence. For $\rho = \rho_0 k^4$ for a sufficiently large constant factor $\rho_0$, a constant fraction of the length within the fence will remain disjoint from $C$, and at least one interval within this constant fraction of the length will be long enough to use as the middle interval.

After these choices, we place the two guards on each row, with the left guard having its $x$-coordinate within the left interval and the right guard within the right interval. We choose the positions of the *vertex guards* (the guards on the rows that correspond to vertices of the

pattern graph), one at a time. As each vertex guard is placed, it may also determine the location of one or more *edge guards* (the guards on the rows that correspond to edges of the pattern graph). An edge guard's location is determined when the vertex guards for its two endpoints have been placed. Whenever this happens, we place the edge guard at the point where the line through the two vertex guards crosses the edge's row. In order to ensure that the edge guards are placed within the left and right intervals, we place each vertex guard within the middle third of its interval. In this way, for each edge $uv$ of the pattern graph we will produce two collinear triples of guards: the triple of left guards for $u$, $v$, and $uv$, and the triple of right guards for $u$, $v$, and $uv$. We will position the vertex guards in sufficiently general position so that these are the only collinearities between triples of guards. As with our choice of row coordinates, we achieve this absence of extra collinearities by a greedy algorithm, in which we place one vertex guard at a time, subject to the constraint that this choice should not create any undesired collinearities between the new vertex guard, the new edge guards whose position becomes determined by the new vertex guard, and the previously determined vertex and edge guards. Each potential undesired collinearity is determined by the choice of two previous guard positions (for which there are $O(k)$ choices) and possibly one of the three neighbors of the vertex whose guard we are trying to place. This implies that, when we are placing each vertex guard, there may be up to $O(k^2)$ positions that we should not place it, because placing it in one of those positions would cause a collinearity. Recall that, when constructing the fence, we left a scaling factor $\phi$ undetermined, and that the left and right interval each contain $\Omega(k\phi/\rho)$ integer coordinates, where $\rho = O(k^4)$ is the factor by which the intervals are narrower than the diameter of the fence and where the factor of $k$ comes from the earlier scaling used to make the fence big enough to include as many integer-coordinate rows as we need. We will choose our scaling factor $\phi$ to be large enough (at least a sufficiently large multiple of $k^5$) to ensure that the middle thirds of the left and right intervals contain more integer coordinates than the number of unavailable positions. With this choice of $\phi$, our greedy guard-placing strategy will always be able to determine a valid integer-coordinate placement for each vertex guard point. By our previous choice of the row coordinates, the edge guard points will also have integer coordinates.

Finally, we place the inmates on each row, within the middle interval. Each edge or vertex of the host graph corresponds to exactly one inmate point, which we place on the row for the corresponding edge or vertex of the pattern graph, given by the coloring of the host graph vertex or the coloring of the endpoints of the host graph edge. We choose the locations for these inmate points in exactly the same way as we chose them for the guard points, greedily one vertex at a time, with the vertex guards placed within the middle third of the middle interval. We place each edge inmate point on the intersection between each edge row and the line through the inmate points for its two endpoints, as soon as these two vertex inmate points have been placed. This will cause each triple of an edge and its two endpoints in the host graph to correspond to three collinear inmate points. We require in addition that no three inmate points on different rows are collinear if they do not form such a triple, and that no two points on the same row as each other coincide. Each potential undesired collinearity is determined by the choice of two previous guard positions (for which there are $O(n^2)$ choices, as the host graph may be dense) and possibly one of up to $O(n)$ neighbors of the vertex whose guard we are trying to place. This implies that there may be $O(n^5)$ positions in the row of the new vertex guard that would cause an unwanted collinearity or coincidence. We will choose our scaling factor $\phi$ to be large enough (at least a sufficiently large multiple of $k^3n^5$) to ensure that the middle third of the middle interval in which we place the inmate points contains more integer coordinates than this number of unavailable positions. In this way, we ensure the success of our greedy inmate-placing strategy.

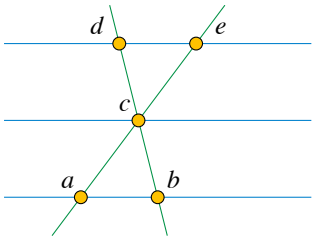



**Figure 3** Illustration for the proof of Lemma 2.

▶ **Lemma 2.** *The translation described above constructs a yard.*

**Proof.** The construction explicitly ensures that most properties of a yard are true: there is a fence, and rows of points, each having two guards outside the fence and inmates inside the fence. There are no non-horizontal lines of four points, and all non-horizontal lines defined by pairs of guards or pairs of inmates pass above or below the other points and through the fence in the proper manner. There are no lines of three points that include both guards and inmates and are not rows. It remains to verify two properties, not stated explicitly as part of the construction: there must be no complete quadrilateral other than the fence, and no subset of guards that form the forbidden pattern on the right of Figure 1.

If either of these forbidden patterns existed, but did not include three points from the same row of the configuration, then all of its three-point lines would have to be non-horizontal lines, composed only of guards or only of inmates. But an edge point can only participate in one such line, and both forbidden patterns include at least one line all three points of which participate in two three-point lines. So all three of the points on this line would have to be vertex points, an impossibility.

The only remaining possibility is a complete quadrilateral that uses at least one row as one of its lines. Let $a$ and $b$ be any two points of this row, and choose $c$, $d$, and $e$ so $acd$ and $bce$ are distinct lines of the quadrilateral. Then triples $acd$ and $ade$ must each be the crossing points of their lines with three rows that correspond in $G$ to an edge and its two endpoints. The choice of the row through $a$ and $b$, and a second row through $c$, determines the third row, which must therefore include both $d$ and $e$. But then lines $ab$ and $de$ are parallel, and cannot intersect to produce the sixth point of a complete quadrilateral (Figure 3). ◀

▶ **Lemma 3.** *An instance $(H, G, c)$ of partitioned subgraph isomorphism (where $c$ is the coloring of the instance) is solvable if and only if the yard constructed from it admits a lineup.*

**Proof.** If a lineup exists, it must include all fence and guard points, and one inmate from each row. This choice of the inmate for each row selects one vertex or edge from the host graph $H$, of the appropriate color class (or pair of endpoint color classes), for each vertex or edge in the pattern graph $G$. If the selected collection of vertices and edges in $H$ does not form an isomorph of $G$, it will include two vertices $u$ and $v$ in $H$ that are adjacent in $G$ but for which the edge $uv$ does not exist in $H$, or has not been selected by the choice of inmate for its row. In this case, the guards for the rows of $u$, $v$, and $uv$ will be collinear in triples (because the corresponding vertices in $G$ are adjacent) but the inmates will not be collinear, creating an instance of the forbidden pattern on the right of Figure 1. So when $H$ does not contain a subgraph isomorphic to $G$, the yard admits no lineup.

If $H$ does contain a subgraph isomorphic to $G$, we can choose the points corresponding to the vertices and edges of that subgraph as the inmates of a lineup. As the following case analysis shows, the resulting system of points does not include any forbidden pattern.

- It does not include four points in a line.
- Because it has only one inmate on each row and the fence is the only complete quadrilateral in the yard, it does not include the middle forbidden pattern in Figure 1.
- It does not include a copy of the right forbidden pattern in which none of its lines are rows, or in which only one of the vertical lines is a row, because in that case (as in Lemma 2) the other vertical line would be a line of three points, each of which belongs to two non-row lines of three points. This is an impossibility because the points on edge rows only belong to one non-row three-point line, and each non-row three-point line includes at least one of these edge points.
- It does not include a copy of the right forbidden pattern in which only one or two of its horizontal lines is a row, because then the two vertical lines could only be lines through three guards. This would force the remaining horizontal lines to contain a left guard, a right guard, and a point between them, and therefore be rows themselves.
- It does not include a copy of the right forbidden pattern with two rows as its two three-point vertical lines. If these rows represent two vertices $u$ and $v$ of $G$, the three points not on these lines would belong to the row for edge $uv$, as they belong to a line containing a representative for $u$ and a representative for $v$. If the rows represent a vertex $u$ and edge $uv$ of $G$, the three points not on these lines would all belong to the row for vertex $v$. In either case these points form another line, unlike the forbidden pattern.
- The only remaining possibility is that the three horizontal lines of this pattern are rows. To form this pattern, the guards of these three rows must be collinear, meaning that they represent two adjacent vertices and the edge between them in $G$. But since we have selected the inmates on each row to form a correctly-colored subgraph isomorphic to $G$, the three inmates must also be collinear, preventing the forbidden pattern from occurring.

Thus, the selected points form a lineup of the yard, as required. ◀

By combining these results with the known hardness of partitioned subgraph isomorphism we obtain our main result:

▶ **Theorem 4.** *Finding a $k$-vertex compliant subset of a given $n$-point set is* W[1]*-complete and, under the exponential time hypothesis, cannot be solved in time $n^{o(k/\log k)}$.*

## 3 Collinearity and convex subsets

We have claimed that finding a $k$-point subset with a given hereditary property is fixed-parameter tractable whenever the property is true for all collinear sets, or whenever there exists a convex polygon for which the property is not true. We split the proof into cases:

- Properties that are true for all collinear point sets and for all convex polygons.
- Properties that are neither true for all collinear point sets nor for all convex polygons.
- Properties that are true for all collinear point sets but not true for all convex polygons

However, the first two cases follow easily from known results.

The first case covers properties that are true for all collinear subsets and for all convex polygons. By the happy ending theorem, every $n$ points in the plane include $\Omega(\log n)$ points that are either all collinear or all in convex position. So, if property $\Pi$ is true for all collinear subsets and for all convex polygons (the first case), then every sufficiently large point set has a $k$-point subset with property $\Pi$. Here, "sufficiently large" means large enough that this $\Omega(\log n)$ bound is at least $k$. For smaller values of $n$ we can search by brute force all subsets of the point set in time that is fixed-parameter tractable in $k$, because (for these values) $n$ itself is bounded by a function of $k$. For larger values of $n$, we can either just answer yes (for

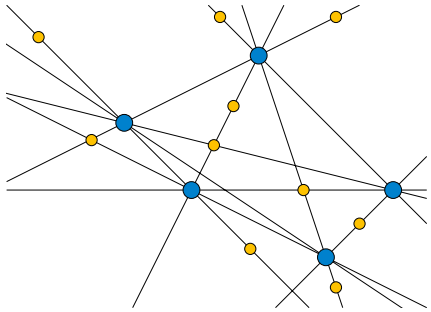


**Figure 4** If $G$ (blue) is a maximal general-position subset of $P$, the remaining points of $P$ (yellow) must be covered by lines through pairs of points in $G$, or else they could be added to $G$.

a decision problem) or reduce the input arbitrarily to the smallest number of points that ensures the existence of a solution (a number bounded by a function of $k$) and then perform a brute force search for a solution within the reduced subproblem.

The second case covers properties that exclude a collinear subset and a convex polygon. By the happy ending theorem, sets of points with the property have bounded size (at most some number $p$ independent of the parameter $k$), because all sufficiently large sets contain a collinear or convex subset without the property, and by heredity do not have the property themselves. To test whether there exists a $k$-point subset with such a property, we answer no immediately when $k > p$. Otherwise we perform a brute-force search over all subsets of the input of size $k$. The time for this search is $O(n^p)$, a polynomial with constant exponent.

We now detail a fixed-point-tractable solution for the remaining case.

▶ **Lemma 5.** *Let decidable hereditary property $\Pi$ be true for all collinear point sets but not true for all convex polygons. Then it is fixed-parameter tractable to find a $k$-point subset having property $\Pi$, among a given set of points in the plane.*

**Proof.** It is straightforward, in polynomial time, to find all lines through pairs of input points and determine whether any of these lines contains $k$ or more points. If so, we can immediately return a subset of $k$ collinear points. So for the remainder of the proof we assume without loss of generality that each line contains at most $k-1$ input points.

Let $q$ be the smallest number of vertices in a convex polygon that does not have property $\Pi$. By assumption, $q$ exists, and by the definition of hereditary properties, any set of points containing the vertices of a convex $q$-gon does not have property $\Pi$. Let $r$ be the largest number of points in a set of points in general position (no three in a line) that does not include a convex $q$-gon; by Suk's improvement to the happy ending theorem [24], $r \leq 2^{q+o(q)}$. Let $P$ be the unknown $k$-point subset (assuming it exists), and $G$ be a maximal subset of $P$ that is in general position (meaning that no other point from $P$ can be added to $G$ without creating a three-point line). Then $|G| \leq r$, or else $G$ and $P$ would contain a convex $q$-gon and $P$ would not have property $\Pi$. The pairs of points in $G$ determine $\binom{|G|}{2} \leq \binom{r}{2}$ lines, and these lines together must cover all of $P$, or else $G$ would not be maximal (Figure 4).

Therefore, we can solve the problem of finding a $k$-point subset $P$ with property $\Pi$ by guessing an $\binom{r}{2}$-tuple of lines that cover $P$, among all $\binom{n}{2}$ lines through pairs of input points, and then for each tuple of lines performing a brute-force search among the points covered by those lines. Because $r$ is a constant (depending on $\Pi$ but not on the input parameter $k$), there are a polynomial number of tuples of lines to be searched. Each tuple covers at most $(k-1)\binom{r}{2}$ points, so performing each brute-force search takes time bounded by a function of $k$, independent of the input size. Therefore, this algorithm is fixed-parameter tractable. ◀

This completes the last case of the following result:

▶ **Theorem 6.** *Let decidable hereditary property* $\Pi$ *either be true for all collinear point sets or not be true for at least one convex polygon (or both). Then it is fixed-parameter tractable to find a* $k$*-point subset having property* $\Pi$*, among a given set of points in the plane.*

Not every hereditary property is decidable: there are uncountably many hereditary properties,[5] only countably many of which are decidable. When a hereditary property is not decidable, the same technique applies, but (because of the need to test the property within each brute-force search used as a subroutine of the algorithms) it gives us a non-uniform fixed-parameter tractable algorithm.

## 4 Properties with a single obstacle

If $X$ is any fixed point set, we can define hereditary property $\Pi_X$ to be true for the point sets that do not contain $X$, and false otherwise. We are then interested in testing whether a given input includes $k$ points with property $\Pi_X$. That is, are there at least $k$ points in the largest point set that avoids $X$? If $X$ is not itself collinear, then all collinear point sets have property $\Pi_X$. Therefore, there is only one family of possible choices for $X$ that is not already covered by Theorem 6: the family of sets $X$ consisting of $q$ points on a single line. The order type of any such point set depends only on $q$. We call a point set with this order type $L_q$.

▶ **Lemma 7.** *Let* $q$ *be fixed. Then it is fixed-parameter tractable to find a* $k$*-point subset avoiding* $L_q$*, among a given set of points in the plane.*

**Proof.** We assume $q > 2$ for otherwise the problem is trivial (an $L_2$-avoiding set can only be a single point, and an $L_1$-avoiding set must be empty). We begin by finding a maximal subset $G$ of the input that is in general position (no three in a line). If $|G| \geq k$, we return any $k$ points from $G$ as our $k$-point $L_q$-avoiding set. Otherwise, as in Lemma 5 and Figure 4, the input can be covered by at most $\binom{k-1}{2}$ lines, the lines through pairs of points in $G$.

Observe that, on each of these lines, an $L_q$-avoiding set can include at most $q-1$ points from the line. Consider a single line $\ell$ and a process in which we build up a $k$-point $L_q$-avoiding set, one point at a time, starting from a subset of points that are disjoint from $\ell$ and then adding points from $\ell$ in an arbitrary order. At each step of this process, at most $q-2$ points from $\ell$ have already been chosen, and the next point must be chosen to be disjoint from these already-chosen points and from any of the lines through pairs of points (neither on $\ell$) that already contain $q-1$ points. There at most $\binom{k-1}{2}$ of these lines through pairs of already-chosen points (because we have already chosen at most $k-1$ points), and each can prevent only a single point of $\ell$ from being chosen. So, as long as $\ell$ contains $\binom{k-1}{2} + q - 1$ points, there will always remain at least one point that is free to be added until we have included exactly $q-1$ points from $\ell$. Therefore, if we reduce the input to contain any arbitrary subset of $\binom{k-1}{2} + q - 1$ points on $\ell$, keeping the points disjoint from $\ell$ unchanged, we cannot affect the existence or nonexistence of a $k$-point $L_q$-avoiding subset.

[5] This follows from the fact that there are infinite families of point sets in which no family member has the same order type as a subset of another family member. For example, the set of vertices and edge midpoints of a convex $r$-gon is not a subset of the set of vertices and edge midpoints of any other convex polygon, so choosing one such point set for each $r$ gives a family of point sets none of which have the order type of a subset of another. This family has uncountably many subfamilies, each of which forms the set of forbidden patterns for a hereditary property. See [8, Observation 5.9].

By applying this observation repeatedly we can kernelize the problem. That is, we find a polynomial-time algorithm that reduces any parameterized instance to an equivalent instance (the "kernel") whose size is bounded by a function of the parameter. To do so, we consider the lines through pairs of points of $G$, one at a time. Whenever any such line contains more than $\binom{k-1}{2} + q - 1$ points of the remaining input, we delete arbitrarily chosen points until it contains exactly $\binom{k-1}{2} + q - 1$ points. At the end of this process, the total number of remaining points is at most $\binom{k-1}{2}\left(\binom{k-1}{2} + q - 1\right) = O(k^4)$, depending only on $k$. We may then find a $k$-point $L_q$-avoiding subset by a brute force search of this kernel. ◀

This includes as a special case the problem of finding an $L_3$-avoiding subset (that is, a subset in general position), the central computational task for the no-3-in-line problem. (We already proved this special case to be NP-hard in [8, Theorem 9.3] and fixed-parameter tractable in [8, Theorem 9.5].) It completes the last case of the following result:

▶ **Theorem 8.** *Let $X$ be any fixed finite set of points. Then it is fixed-parameter tractable to find a $k$-point subset avoiding $X$, among a given set of points in the plane.*

## 5 Conclusions

We have identified a hereditary property of point sets such that finding a $k$-point subset with this property is unlikely to be fixed-parameter tractable, and identified several general classes of properties for which the corresponding problems are fixed-parameter tractable. These include properties that include all collinear sets, properties that exclude a convex polygon, and properties with a single obstacle. Several additional cases are fixed-parameter tractable:

- The largest convex subset (avoiding two forbidden patterns, one of three points on a line and the other of a triangle with an interior point) is polynomially solvable, in cubic time and linear space [6, 7].
- If finding $k$-point subsets is fixed-parameter tractable for two properties $\Pi$ and $\Pi'$, then it is also fixed-parameter tractable for their disjunction (the point sets that have at least one of the two properties), by running both algorithms and returning any $k$-point set found by either of them.
- If finding $k$-point subsets is fixed-parameter tractable for property $\Pi$, then it is also fixed-parameter tractable for the property $\Pi_q$ of being partitionable into $q$ subsets that have property $\Pi$, by color-coding [2]. Alternatively one could use a different hereditary property for each color class.

We have been unable to complete a classification of which of these problems are fixed-parameter tractable and which are not. We leave this as open for future research.

Another open problem concerns properties (such as being in convex position) that can be true only of point sets in general position. That is, these properties have a three-point line as one of their forbidden patterns. Can it be hard, for such a property, to find a $k$-point subset with the property? We suspect that the answer is yes, by a reduction that mimics the one we have given, using more complex constraints on general-position subsets of points to make them act like the collinear subsets of points in our reduction. However, we have not worked out the details of such a reduction.

## References


1. Faisal N. Abu-Khzam. A kernelization algorithm for $d$-hitting set. *J. Comput. System Sci.*, 76(7):524–531, 2010. doi:10.1016/j.jcss.2009.09.002.
2. Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.
3. Saeed Akhoondian Amiri, Stephan Kreutzer, Dániel Marx, and Roman Rabinovich. Routing with congestion in acyclic digraphs. In *41st International Symposium on Mathematical Foundations of Computer Science*, volume 58 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 7:1–7:11. Schloss Dagstuhl, Leibniz-Zent. Inform., Wadern, 2016. doi:10.4230/LIPIcs.MFCS.2016.7.
4. Édouard Bonnet, Tillmann Miltzow, and Paweł Rzążewski. Complexity of token swapping and its variants. *Algorithmica*, 2017. doi:10.1007/s00453-017-0387-0.
5. Glencora Borradaile, David Eppstein, and Pingan Zhu. Planar induced subgraphs of sparse graphs. *J. Graph Algorithms Appl.*, 19(1):281–297, 2015. doi:10.7155/jgaa.00358.
6. Václav Chvátal and Gheza T. Klincsek. Finding largest convex subsets. In *11th Southeastern Conference on Combinatorics, Graph Theory and Computing, Vol. II*, volume 29 of *Congressus Numerantium*, pages 453–460, 1980.
7. Herbert Edelsbrunner and Leonidas J. Guibas. Topologically sweeping an arrangement. *J. Comput. System Sci.*, 38(1):165–194, 1989. doi:10.1016/0022-0000(89)90038-X.
8. David Eppstein. *Forbidden Configurations in Discrete Geometry.* Cambridge University Press, 2018.
9. Paul Erdős and George Szekeres. A combinatorial problem in geometry. *Compositio Math.*, 2:463–470, 1935. URL: http://www.numdam.org/item?id=CM_1935__2__463_0.
10. Jörg Flum and Martin Grohe. 9.1 Polynomial Kernelizations. In *Parameterized Complexity Theory*, EATCS Ser. Texts in Theoretical Computer Science, pages 208–212. Springer, 2006. doi:10.1007/3-540-29953-X.
11. Michael R. Garey and David S. Johnson. GT21: Induced subgraph with property Π. In *Computers and Intractability: A Guide to the Theory of NP-completeness*, page 195. W. H. Freeman, 1979.
12. Panos Giannopoulos, Christian Knauer, and Daniel Werner. On the computational complexity of Erdős-Szekeres and related problems in $\mathbb{R}^3$. In *21st Annual European Symposium on Algorithms (ESA 2013)*, volume 8125 of *Lecture Notes in Computer Science*, pages 541–552. Springer, 2013. doi:10.1007/978-3-642-40450-4_46.
13. Jacob E. Goodman, Richard Pollack, and Bernd Sturmfels. Coordinate representation of order types requires exponential storage. In *21st Symposium on Theory of Computing (STOC '89)*, pages 405–410. ACM, 1989. doi:10.1145/73007.73046.
14. Martin Grohe and Dániel Marx. On tree width, bramble size, and expansion. *J. Combin. Theory Ser. B*, 99(1):218–228, 2009. doi:10.1016/j.jctb.2008.06.004.
15. Richard R. Hall, Terence H. Jackson, Anthony Sudbery, and Ken Wild. Some advances in the no-three-in-line problem. *J. Combin. Theory Ser. A*, 18:336–341, 1975. doi:10.1016/0097-3165(75)90043-6.
16. Heiko Harborth, Arnfried Kemnitz, Meinhard Möller, and Andreas Süssenbach. Ganzzahlige planare Darstellungen der platonischen Körper. *Elemente der Mathematik*, 42(5):118–122, 1987. URL: https://eudml.org/doc/141411.
17. Ken-ichi Kawarabayashi. Planarity allowing few error vertices in linear time. In *50th IEEE Symposium on Foundations of Computer Science (FOCS 2009)*, pages 639–648. IEEE Computer Society, Los Alamitos, CA, 2009. doi:10.1109/FOCS.2009.45.
18. Ken-ichi Kawarabayashi and Bruce Reed. An (almost) linear time algorithm for odd cycles transversal. In *21st ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, pages 365–378, Philadelphia, PA, 2010. SIAM.

19 Subhash Khot and Venkatesh Raman. Parameterized complexity of finding subgraphs with hereditary properties. *Theoret. Comput. Sci.*, 289(2):997–1008, 2002. doi:10.1016/S0304-3975(01)00414-5.

20 R. Krithika and N. S. Narayanaswamy. Another disjoint compression algorithm for odd cycle transversal. *Inform. Process. Lett.*, 113(22-24):849–851, 2013. doi:10.1016/j.ipl.2013.08.007.

21 John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *J. Comput. System Sci.*, 20(2):219–230, 1980. doi:10.1016/0022-0000(80)90060-4.

22 Dániel Marx. Can you beat treewidth? *Theory of Computing*, 6:85–112, 2010. doi:10.4086/toc.2010.v006a005.

23 Klaus F. Roth. On a problem of Heilbronn. *J. London Math. Soc.*, 26:198–204, 1951. doi:10.1112/jlms/s1-26.3.198.

24 Andrew Suk. On the Erdős–Szekeres convex polygon problem. *J. Amer. Math. Soc.*, 30:1047–1053, 2017. doi:10.1090/jams/869.

25 Terence Tao. The Erdos-Ulam problem, varieties of general type, and the Bombieri-Lang conjecture. In *What's new*. WordPress, December 20, 2014. URL: https://terrytao.wordpress.com/2014/12/20/.

# Dual Parameterization of Weighted Coloring


## Júlio Araújo
Departamento de Matemática, Universidade Federal do Ceará, Fortaleza, Brazil
julio@mat.ufc.br
https://orcid.org/0000-0001-7074-2753

## Victor A. Campos
Departamento de Computação, Universidade Federal do Ceará, Fortaleza, Brazil
campos@lia.ufc.br

## Carlos Vinícius G. C. Lima
Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil
carloslima@dcc.ufmg.br
https://orcid.org/0000-0002-6666-0533

## Vinícius Fernandes dos Santos
Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil
viniciussantos@dcc.ufmg.br
https://orcid.org/0000-0002-4608-4559

## Ignasi Sau
LIRMM, CNRS, Université de Montpellier, Montpellier, France
ignasi.sau@lirmm.fr
https://orcid.org/0000-0002-8981-9287

## Ana Silva
Departamento de Matemática, Universidade Federal do Ceará, Fortaleza, Brazil
anasilva@mat.ufc.br
https://orcid.org/0000-0001-8917-0564



## Abstract

Given a graph $G$, a *proper $k$-coloring* of $G$ is a partition $c = (S_i)_{i\in[1,k]}$ of $V(G)$ into $k$ stable sets $S_1, \ldots, S_k$. Given a weight function $w : V(G) \to \mathbb{R}^+$, the *weight of a color* $S_i$ is defined as $w(i) = \max_{v\in S_i} w(v)$ and the *weight of a coloring* $c$ as $w(c) = \sum_{i=1}^{k} w(i)$. Guan and Zhu [Inf. Process. Lett., 1997] defined the *weighted chromatic number* of a pair $(G, w)$, denoted by $\sigma(G, w)$, as the minimum weight of a proper coloring of $G$. The problem of determining $\sigma(G, w)$ has received considerable attention during the last years, and has been proved to be notoriously hard: for instance, it is NP-hard on split graphs, unsolvable on $n$-vertex trees in time $n^{o(\log n)}$ unless the ETH fails, and W[1]-hard on forests parameterized by the size of a largest tree.

We focus on the so-called *dual parameterization* of the problem: given a vertex-weighted graph $(G, w)$ and an integer $k$, is $\sigma(G, w) \leq \sum_{v\in V(G)} w(v) - k$? This parameterization has been recently considered by Escoffier [WG, 2016], who provided an FPT algorithm running in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$, and asked which kernel size can be achieved for the problem.

We provide an FPT algorithm running in time $9^k \cdot n^{\mathcal{O}(1)}$, and prove that no algorithm in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ exists under the ETH. On the other hand, we present a kernel with at most $(2^{k-1}+1)(k-1)$ vertices, and rule out the existence of polynomial kernels unless NP $\subseteq$ coNP/poly, even on split graphs with only two different weights. Finally, we identify some classes of graphs on which the problem admits a polynomial kernel, in particular interval graphs and subclasses of split graphs, and in the latter case we present lower bounds on the degrees of the polynomials.



© Júlio Araújo, Victor A. Campos, Carlos V. Lima, Vinícius F. dos Santos, Ignasi Sau, and Ana Silva; licensed under Creative Commons License CC-BY


13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 12; pp. 12:1–12:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

**2012 ACM Subject Classification** Mathematics of computing → Graph algorithms, Theory of computation → Parameterized complexity and exact algorithms

**Keywords and phrases** weighted coloring, max coloring, parameterized complexity, dual parameterization, FPT algorithms, polynomial kernels, split graphs, interval graphs

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.12

**Related Version** A full version of this article is permanently available at `https://arxiv.org/abs/1805.06699`.

**Funding** Work supported by French projects DEMOGRAPH (ANR-16-CE40-0028) and ESIGMA (ANR-17-CE23-0010), and by Brazilian projects CNPq 306262/2014-2, CNPq 311013/2015-5, CNPq Universal 421660/2016-3, CNPq Universal 401519/2016-3, FAPEMIG, Funcap PNE-0112-00061.01.00/16, and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

**Acknowledgements** We would like to thank the anonymous reviewers for carefully reading the article, in particular for spotting a flaw in the proof of Claim 5, which we rewrote completely in a simpler way. We also thank Mikko Koivisto for pointing us to reference [4].

## 1 Introduction

A *(vertex) $k$-coloring* of a graph $G = (V, E)$ is a function $c : V(G) \to \{1, \ldots, k\}$. Such coloring $c$ is *proper* if $c(u) \neq c(v)$ for every edge $\{u, v\} \in E(G)$. All the colorings we consider in this paper are proper, hence we may omit the word "proper". The *chromatic number* $\chi(G)$ of $G$ is the minimum integer $k$ such that $G$ admits a $k$-coloring. Given a graph $G$, determining $\chi(G)$ is the goal of the classical Vertex Coloring problem. If $c$ is a $k$-coloring of $G$, then $S_i = \{u \in V(G) \mid c(u) = i\}$ is a stable (or independent) set. With slight abuse of notation, we shall also call such a set $S_i$ a *color*.

In this paper we study a generalization of Vertex Coloring for vertex-weighted graphs that has been defined by Guan and Zhu [24]. Given a graph $G$ and a weight function $w : V(G) \to \mathbb{R}^+$, the *weight of a color* $S_i$ is defined as $w(i) = \max_{v \in S_i} w(v)$. Then, the *weight of a coloring* $c$ is $w(c) = \sum_{i=1}^{k} w(i)$. In the Weighted Coloring problem, the goal is to determine the *weighted chromatic number* of a pair $(G, w)$, denoted by $\sigma(G, w)$, which is the minimum weight of a coloring of $(G, w)$. A coloring $c$ of $G$ such that $w(c) = \sigma(G, w)$ is an *optimal weighted coloring*. Guan and Zhu [24] also defined, for a positive integer $r$, $\sigma(G, w; r)$ as the minimum of $w(c)$ among all $r$-colorings $c$ of $G$, or as $+\infty$ if no $r$-coloring exists. Note that $\sigma(G, w) = \min_{r \geq 1} \sigma(G, w; r)$. It is worth mentioning that the Weighted Coloring problem is also sometimes called Max-Coloring in the literature; see for instance [34, 37, 18]. Guan and Zhu defined this problem in order to study practical applications related to resource allocation, which they describe in detail in [24]. One should observe that if all the vertex weights are equal to one, then $\sigma(G, w) = \chi(G)$, for every graph $G$. Consequently, determining $\sigma(G, w)$ is an NP-hard problem on general graphs [33]. In fact, this problem has been shown to be NP-hard even on very restricted graph classes, such as split graphs with only two different weights, interval graphs, triangle-free planar graphs with bounded degree, and bipartite graphs [12, 11, 19]. On the other hand, the weighted chromatic number of cographs and of some subclasses of bipartite graphs can be found in polynomial time [12, 11].

The complexity of Weighted Coloring on trees (and forests) has attracted considerable attention in the literature. Guan and Zhu [24] left as an open problem whether Weighted Coloring is polynomial on trees and, more generally, on graphs of bounded treewidth. Escoffier et al. [19] found a polynomial-time approximation scheme to solve Weighted Coloring on bounded treewidth graphs, and Kavitha and Mestre [34] showed that the problem is in P on the class of trees where vertices with degree at least three induce a stable set. But the question of Guan and Zhu has been answered only recently, when Araújo et al. [2] showed that, unless the Exponential Time Hypothesis (ETH)[1] fails, there is no algorithm computing the weighted chromatic number of $n$-vertex trees in time $n^{o(\log n)}$. Moreover, as discussed in [2], this lower bound is tight. Very recently Araújo et al. [1] focused on the parameterized complexity of computing $\sigma(G, w)$ and $\sigma(G, w; r)$ when $G$ is a forest, and they proved that computing $\sigma(G, w)$ is W[1]-hard parameterized by the size of a largest tree of $G$, and that computing $\sigma(G, w; r)$ is W[2]-hard parameterized by $r$.

In view of the above discussion, we can conclude that Weighted Coloring is a particularly hard problem from a computational point of view, and that the positive results in the literature are quite scarce. In this paper we adopt the perspective of parameterized complexity and consider the *dual parameterization* of the problem, which we call Dual Weighted Coloring and is formally defined as follows:

| Dual Weighted Coloring |
|---|
| **Input:** A vertex-weighted graph $(G, w)$ and a positive integer $k$. |
| **Parameter:** $k$. |
| **Question:** Is $\sigma(G, w) \leq \sum_{v \in V(G)} w(v) - k$? |

Since by definition of parameterized problem (cf. [16, 10]) the parameter needs to be a non-negative integer, for the above parameterization to make sense we will assume henceforth that all vertex-weights are positive integers. We will denote by $n$ the number of vertices of the input graph of Dual Weighted Coloring.

The motivation for considering such parameterization is to take as the parameter the "savings" with respect to the trivial upper bound of $\sum_{v \in V(G)} w(v)$ on $\sigma(G, w)$. This approach has proved to be very useful for the classical Vertex Coloring problem, especially from the approximation point of view [13, 28, 26, 27, 17, 18]. From a parameterized perspective, Chor et al. [9] presented an FPT algorithm for Dual Vertex Coloring. Concerning kernelization, it is not difficult to see [10, Exercise 2.22] that the problem admits a kernel with at most $3k$ vertices, by applying the so-called *crown reduction rule* to the complement of the input graph $G$. Other results concerning dual parameterization are, for instance, FPT algorithms for the Grundy and $b$-chromatic numbers of a graph [29], the parameterized approximability of subset graph problems [8], or the existence of polynomial kernels for the Set Cover and Hitting Set problems [25, 3].

The dual parameterization of the Weighted Coloring problem, under the equivalent name of Max Coloring, has been recently considered by Escoffier [18], who provided an FPT algorithm running in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$, and asked which is the smallest kernel size that can be achieved for the problem.

**Our results.** Improving over Escoffier's algorithm [18], our first result is an FPT algorithm for Dual Weighted Coloring running in time $9^k \cdot n^{\mathcal{O}(1)}$, based on some standard dynamic programming ideas used in coloring and partition problems (see, for example, [35] and [20,

[1] The ETH states that 3-SAT cannot be solved in subexponential time; see [32, 31] for more details.

Section 3.1.2]). It is easy to see that a subexponential algorithm is unlikely to exist. Indeed, consider the 3-Coloring problem, which corresponds to the unweighted version of Dual Weighted Coloring with parameter $k = n - 3$. Since 3-Coloring cannot be solved in time $2^{o(n)}$ under the ETH [36], the existence of an algorithm for Dual Weighted Coloring running in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ would imply, in particular, an algorithm for 3-Coloring running in time $2^{o(k)} \cdot n^{\mathcal{O}(1)} = 2^{o(n)}$, contradicting the ETH.

Our main contribution concerns the existence of (polynomial) kernels for Dual Weighted Coloring. By the well-known equivalence of admitting an FPT algorithm and a kernel (cf. [16, 10]), the FPT algorithm mentioned above directly yields a kernel for Dual Weighted Coloring of size at most $9^k$. On the one hand, we considerably improve this bound by providing a kernel with at most $(2^{k-1}+1)(k-1)$ vertices, inspired by an approach attributed to Jan Arne Telle [16, Exercise 4.12.12] for obtaining a quadratic kernel for Dual Vertex Coloring. On the other hand, we complement this result by showing that, unlike the Dual Vertex Coloring problem, Dual Weighted Coloring does *not* admit a polynomial kernel unless NP $\subseteq$ coNP/poly, even on split graphs with only two different weights. We prove this result by a polynomial parameter transformation from the Set Cover problem parameterized by the size of the universe, proved not to admit polynomial kernels unless NP $\subseteq$ coNP/poly by Dom et al. [15]. Our reduction is an appropriate modification of a reduction of Demange et al. [12] to prove the NP-hardness of Weighted Coloring on split graphs. Altogether, these results answer Escoffier's question [18] about the smallest kernel size for the problem.

Motivated by the above hardness result, it is natural to identify graph classes on which the Dual Weighted Coloring problem admits polynomial kernels. We prove that this is the case of graph classes with bounded clique number and of interval graphs, for which we present a linear and a cubic kernel, respectively. Finally, we identify subclasses of split graphs admitting polynomial kernels. Namely, we prove that Dual Weighted Coloring restricted to split graphs where each vertex in the clique has at most $d$ non-neighbors in the stable set, for some constant $d \geq 1$, admits a kernel with at most $k^{d+1}$ vertices. We show that the dependency on $d$ in the exponent is necessary, by proving that for any $d \geq 2$ and $\varepsilon > 0$, a kernel with $\mathcal{O}(k^{\frac{d-3}{2}-\varepsilon})$ vertices on that graph class does not exist unless NP $\subseteq$ coNP/poly. In other words, we rule out the existence of a *uniform* kernel, that is, a kernel of size $f(d) \cdot k^{\mathcal{O}(1)}$ for any function $f$.

**Organization of the paper.** In Section 2 we present some basic preliminaries about graphs and parameterized complexity. In Section 3 we present the FPT algorithm and in Section 4 we provide the kernelization results. Finally, we conclude the article in Section 5.

## 2 Preliminaries

**Graphs.** We use standard graph-theoretic notation, and we consider simple undirected graphs without loops or multiple edges; see [14] for any undefined terminology. Given a graph $G = (V, E)$, $X \subseteq V$, and $v \in V$, we denote $N_X(v) = N(v) \cap X$, where $N(v) = \{u \in V \mid \{u, v\} \in E\}$, and $\overline{N_X}(v) = X \setminus \{N_X(v) \cup \{v\}\}$. An *antimatching* in a graph $G$ is a matching in the complement of $G$. Two vertices $u, v \in V(G)$ are *twins* if $N(u) = N(v)$. A vertex $v \in V(G)$ is *universal* if $N(v) = V(G) \setminus \{v\}$. A graph $G$ is a *split graph* if $V(G)$ can be partitioned into an independent set and a clique, and an *interval graph* if one can associate a real interval with each vertex, so that two vertices are adjacent if and only if the corresponding intervals intersect.

**Parameterized complexity.** We refer the reader to [16, 10] for basic background on parameterized complexity, and we recall here only some basic definitions, with special emphasis on tools for polynomial kernelization. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$. For an instance $I = (x, k) \in \Sigma^* \times \mathbb{N}$, $k$ is called the *parameter*. A parameterized problem is *fixed-parameter tractable* (FPT) if there exists an algorithm $\mathcal{A}$, a computable function $f$, and a constant $c$ such that given an instance $I = (x, k)$, $\mathcal{A}$ (called an FPT *algorithm*) correctly decides whether $I \in L$ in time bounded by $f(k) \cdot |I|^c$.

A fundamental concept in parameterized complexity is that of *kernelization*. A kernelization algorithm, or just *kernel*, for a parameterized problem $\Pi$ takes an instance $(x, k)$ of the problem and, in time polynomial in $|x| + k$, outputs an instance $(x', k')$ such that $|x'|, k' \leqslant g(k)$ for some function $g$, and $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$. The function $g$ is called the *size* of the kernel and may be viewed as a measure of the "compressibility" of a problem using polynomial-time preprocessing rules. A kernel is called *polynomial* (resp. *linear*) if the function $g(k)$ is a polynomial (resp. linear) function in $k$. A breakthrough result of Bodlaender et al. [5] gave the first framework for proving that certain parameterized problems do not admit polynomial kernels, by establishing so-called *composition algorithms*. Together with a result of Fortnow and Santhanam [21] this allows to exclude polynomial kernels under the assumption that $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$, otherwise implying a collapse of the polynomial hierarchy to its third level [38]. Very successful notions for proving such type of result are those of *cross-composition*, introduced by Bodlaender et al. [6], and of *polynomial parameter transformation*, introduced by Bodlaender et al. [7]. We need to define the latter. A polynomial parameter transformation from a parameterized problem $P$ to a parameterized problem $Q$ is an algorithm that, given an instance $(x, k)$ of $P$, computes in polynomial time an equivalent instance $(x', k')$ of $Q$ such that $k'$ is bounded by a polynomial depending only on $k$.

Within parameterized problems, the class W[1] may be seen as the parameterized equivalent to the class NP of classical optimization problems. Without entering into details (see [16, 10] for the formal definitions), a parameterized problem being W[1]-*hard* can be seen as a strong evidence that this problem is *not* FPT. The class W[2] of parameterized problems is a class that contains W[1], and such that the problems that are W[2]-*hard* are even more unlikely to be FPT than those that are W[1]-hard (again, see [16, 10] for the formal definitions).

## 3 FPT algorithm

In this section we present an FPT algorithm for the Dual Weighted Coloring problem.

▶ **Theorem 1.** *The* Dual Weighted Coloring *problem can be solved in time* $9^k \cdot n^{\mathcal{O}(1)}$.

**Proof.** We start by computing a maximum unweighted antimatching $\bar{M}$ in $G$ (this idea has been already used, in particular, in [12]); note that this can be done in polynomial time by computing a maximum matching in the complement of $G$. If $|\bar{M}| \geq k$, since we assume that all the vertex weights are at least 1, by putting in the same color class each pair of vertices that belong to a non-edge of $\bar{M}$ and coloring any other vertex with a new color, we obtain a coloring of $G$ with weight at most $\sum_{v \in V(G)} w(v) - k$, and we can output a constant-sized yes-instance. Thus, we assume henceforth that $|\bar{M}| \leq k - 1$.

Let $V(\bar{M}) \subseteq V(G)$ be the set of vertices that appear in the non-edges in $\bar{M}$. Since $\bar{M}$ is maximum, the set of vertices $K := V(G) \setminus V(\bar{M})$ induces a clique in $G$. Note that, in any coloring $c$ of $G$, at least $|K|$ colors will be needed. Let $K = \{v_1, \ldots, v_{|K|}\}$ and let $c(v_i) = c_i$.

Hence, it remains just to color the vertices in $V(\bar{M})$, which may be colored with some colors previously used in $K$ or with new ones.

Let $X \subseteq V(\bar{M})$ and $0 \leq i \leq |K|$. We define $T(X, i)$ as the minimum weight of a coloring of $G[K \cup X]$ such that no color $c_j$, for $j > i$, is assigned to a vertex of $X$, i.e., the colors assigned to vertices in $X$ are either from the set $\{c_1, \ldots, c_i\}$ or new colors not assigned to any vertex of $K$. Note that, by definition, $\sigma(G, w) = T(V(\bar{M}), |K|)$.

We now describe how to compute $T(X, i)$ for every $X \subseteq V(\bar{M})$ and every $0 \leq i \leq |K|$.

If $X = \emptyset$, then, for any $i$,

$$T(\emptyset, i) = \sum_{v \in K} w(v),$$

since $K$ is a clique. This can be done in linear time.

If $i = 0$, then the colors used in $K$ and in $X$ are disjoint. By applying brute force over all possible stable sets of $X$ we get

$$T(X, 0) = \min_{\substack{\emptyset \neq S \subseteq X \\ S \text{ stable}}} T(X \setminus S, 0) + w(S),$$

where $w(S) = \max_{v \in S} w(v)$. The values $T(X, 0)$, for all possible sets $X \subseteq V(\bar{M})$, can be computed in time

$$\left( \sum_{j=0}^{|V(\bar{M})|} \binom{|V(\bar{M})|}{j} \cdot 2^j \right) \cdot n^{\mathcal{O}(1)} = 3^{|V(\bar{M})|} \cdot n^{\mathcal{O}(1)},$$

corresponding to considering the sets $X$ by increasing size and choosing an arbitrary subset $S$ inside $X$, and where the polynomial factor comes from checking whether such a set $S$ is stable or not.

Now, if $i > 0$ and $X \neq \emptyset$, we have two possibilities, namely either color $c_i$ is used in $V(\bar{M})$ or not. If $c_i$ is not used, clearly we have $T(X, i) = T(X, i-1)$. Otherwise, there is a non-empty set $S \subseteq X$ of vertices colored with color $c_i$. Therefore, taking into account both cases, we can iterate over every possible set $S \subseteq X$ and compute $T(X, i)$ as follows:

$$T(X, i) = \min_{\substack{S \subseteq X \\ S \cup \{v_i\} \text{ stable}}} T(X \setminus S, i-1) + w(S \cup \{v_i\}) - w(v_i).$$

Here we have to subtract the weight of $v_i$, which was, in the partial solution $T(X \setminus S, i-1)$, the weight of color $c_i$, and replace it by $w(S \cup \{v_i\})$, which may be larger. As in the previous case, for every $i$ it is possible to compute $T(X, i)$ for every $X$ in time $3^{|V(\bar{M})|} \cdot n^{\mathcal{O}(1)}$.

Hence, since $|V(\bar{M})| \leq 2k - 2$, in time bounded by $3^{2k} \cdot n^{\mathcal{O}(1)} = 9^k \cdot n^{\mathcal{O}(1)}$ we can compute $T(V(\bar{M}), |K|) = \sigma(G, w)$ and answer whether $\sigma(G, w) \leq \sum_{v \in V(G)} w(v) - k$ or not. ◀

## 4 Kernelization results

In this section we focus on the existence of (polynomial) kernels for Dual Weighted Coloring.

▶ **Theorem 2.** *The* Dual Weighted Coloring *problem admits a kernel with at most* $(2^{k-1} + 1) \cdot (k - 1)$ *vertices.*

**Proof.** We start with the following trivial polynomial-time reduction rule.

**Rule 1.** If $G$ contains a universal vertex, delete it.

▶ **Claim 1.** *Rule 1 is safe.*

**Proof.** Since a universal vertex $u$ appears as a singleton in any proper coloring of $G$, it follows that $\sigma(G, w) \leq \sum_{v \in V(G)} w(v) - k$ if and only if $\sigma(G - \{u\}, w) \leq \sum_{v \in V(G) \setminus \{u\}} w(v) - k$. ◀

As in the proof of Theorem 1, we compute in polynomial time a maximum unweighted antimatching $\bar{M}$ in $G$. Again, if $|\bar{M}| \geq k$, we can correctly answer that we have a yes-instance, so we assume henceforth that $|\bar{M}| \leq k - 1$. Let again $V(\bar{M}) \subseteq V(G)$ be the set of vertices that appear in the non-edges in $\bar{M}$, and recall that since $\bar{M}$ is maximum, the set of vertices $K = V(G) \setminus V\left(\bar{M}\right)$ induces a clique in $G$.

We now partition $K$ into a set of equivalence classes $\mathcal{C}$ according to the neighborhood in $V\left(\bar{M}\right)$. That is, $u, v \in K$ belong to the same class in $\mathcal{C}$ if and only if $N_{V\left(\bar{M}\right)}(u) = N_{V\left(\bar{M}\right)}(v)$. Note that $\mathcal{C}$ can be constructed in polynomial time, by iteratively processing the vertices of $K$, comparing the neighborhood in $V(\bar{M})$ of the currently processed vertex $v$ with those of the already processed vertices, and creating a new class containing only $v$ if no class has exactly the set $N_{V\left(\bar{M}\right)}(v)$ as neighbors in $V\left(\bar{M}\right)$. Given an equivalence class $C \in \mathcal{C}$ and a non-edge $\bar{e} \in \bar{M}$, we denote by $N_{\bar{e}}(C)$ the set of neighbors of any vertex in $C$ in the set consisting of the two endpoints of $\bar{e}$. We proceed to analyze the number and the size of the classes in $\mathcal{C}$. We start with an easy claim.

▶ **Claim 2.** *Let $C \in \mathcal{C}$ be an equivalence class with $|C| \geq 2$. For every non-edge $\bar{e} \in \bar{M}$ it holds that $|N_{\bar{e}}(C)| \geq 1$.*

**Proof.** Suppose for contradiction that $|N_{\bar{e}}(C)| = 0$, let $x, y$ be the endvertices of $\bar{e}$, and let $u, v$ be any two vertices in $C$. Then we can obtain from $\bar{M}$ a larger antimatching $\bar{M}'$ by replacing the non-edge $\{x, y\}$ with the two non-edges $\{u, x\}$ and $\{v, y\}$, a contradiction. ◀

In the next claim we restrict further the neighborhoods of the equivalence classes in $V(\bar{C})$.

▶ **Claim 3.** *Let $C_1, C_2 \in \mathcal{C}$ be two equivalence classes and let $\bar{e} \in \bar{M}$ be a non-edge with endpoints $x$ and $y$. Then it is not possible that $x \notin N_{\bar{e}}(C_1)$ and $y \notin N_{\bar{e}}(C_2)$, or vice versa.*

**Proof.** Suppose for contradiction that $x \notin N_{\bar{e}}(C_1)$ and $y \notin N_{\bar{e}}(C_2)$, and let $u \in C_1$ and $v \in C_2$. Then we can obtain from $\bar{M}$ a larger antimatching $\bar{M}'$ by replacing the non-edge $\{x, y\}$ with the two non-edges $\{u, x\}$ and $\{v, y\}$, a contradiction. ◀

We call an equivalence class $C \in \mathcal{C}$ *special* if for some non-edge $\bar{e} \in \bar{M}$, $|N_{\bar{e}}(C)| = 0$, and we call such an $\bar{e}$ a *special* non-edge; otherwise we call an equivalence class *normal*. We call a non-edge $\bar{e} \in \bar{M}$ *normal* if for every $C \in \mathcal{C}$, $|N_{\bar{e}}(C)| \geq 1$. Let $k_{\mathsf{s}}$ and $k_{\mathsf{n}}$ be the number of special and normal non-edges in $\bar{M}$, respectively, and note that $k_{\mathsf{s}} + k_{\mathsf{n}} = |\bar{M}| \leq k - 1$.

By Claim 2, every special class contains exactly one vertex. If $\bar{e} \in \bar{M}$ is a special non-edge and $C \in \mathcal{C}$ is such that $|N_{\bar{e}}(C)| = 0$, then Claim 3 implies that for every other class $C' \in \mathcal{C}$ different from $C$, it holds that $|N_{\bar{e}}(C')| = 2$. Therefore, the number of special classes in $\mathcal{C}$ is at most the number of special non-edges in $\bar{M}$, that is, at most $k_{\mathsf{s}}$.

Let $\bar{e} \in \bar{M}$ be a normal non-edge with endpoints $x, y$, let $C \in \mathcal{C}$ be such that $|N_{\bar{e}}(C)| = 1$, and assume that $N_{\bar{e}}(C) = \{x\}$. Then Claim 3 implies that for every class $C' \in \mathcal{C}$ such that $|N_{\bar{e}}(C')| = 1$, it holds that $N_{\bar{e}}(C') = \{x\}$. Hence, every normal non-edge $\bar{e} \in \bar{M}$ has at most one endpoint that has some non-neighbor in $K$; we call such a vertex the *avoidable* vertex of $\bar{e}$. This means that for every normal class $C \in \mathcal{C}$ and every non-edge $\bar{e} \in \bar{M}$, there are *exactly two* possibilities: either $|N_{\bar{e}}(C)| = 2$, or $|N_{\bar{e}}(C)| = 1$ and $N_{\bar{e}}(C)$ consists of the

endpoint of $\bar{e}$ distinct from its avoidable vertex. Moreover, by Claim 3 the latter case can only occur if $\bar{e}$ is a normal non-edge. Therefore, the number of normal classes in $\mathcal{C}$ is at most $2^{k_n} - 1$, corresponding to all the choices of neighborhoods in the set consisting of the avoidable vertices in the normal edges in $\bar{M}$, and excluding the class $C$ with $N_{V(\bar{M})}(C) = V(\bar{M})$, since these vertices would be deleted by Rule 1.

Finally, in order to bound the size of the equivalence classes in $\mathcal{C}$, we state the following reduction rule, which says that it is enough to keep, for each equivalence class, the $|\bar{M}|$ heaviest vertices.

**Rule 2.** Suppose there exists an equivalence class $C \in \mathcal{C}$ with $|C| > |\bar{M}|$. Let $W \subseteq C$ be a subset of vertices with $|W| = |\bar{M}|$ and such that, if $u \in W$ and $v \in C \setminus W$, then $w(u) \geq w(v)$. Delete from $G$ all the vertices in $C \setminus W$.

▶ **Claim 4.** *Rule 2 is safe.*

**Proof.** We need to prove that if $(G', w, k)$ results from $(G, w, k)$ after the application of Rule 2, then $(G, w, k)$ and $(G', w, k)$ are equivalent instances of Dual Weighted Coloring.

Assume first that $\sigma(G', w) \leq \sum_{v \in V(G')} w(v) - k$, and let $c'$ be a coloring of $(G', w)$ satisfying this bound. We define a coloring $c$ of $(G, w)$ starting from $c'$ and creating a new color for each of the vertices in $C \setminus W$. Clearly, $w(c) = w(c') + \sum_{v \in C \setminus W} w(v) \leq \sum_{v \in V(G)} w(v) - k$.

Conversely, assume that $\sigma(G, w) \leq \sum_{v \in V(G)} w(v) - k$, and let $c$ be a coloring of $(G, w)$ satisfying this bound. Due to Rule 1, we can clearly assume that $|\bar{M}| \geq 1$. Hence, since Rule 2 has been applied on $C \in \mathcal{C}$, it follows that $|C| \geq 2$. Thus, by Claim 2, we have that for every non-edge $\bar{e} \in \bar{M}$, $|N_{\bar{e}}(C)| \geq 1$. This implies, together with the fact that the vertices in $C$ induce a clique, that in the coloring $c$ of $(G, w)$, at most $|\bar{M}|$ vertices of $C$ appear in colors containing vertices of $V(\bar{M})$, and every other vertex in $C$ is a singleton in its color. Let $T \subseteq C$ be this set of at most $|\bar{M}|$ vertices, and let $W \subseteq C$ be the set such that Rule 2 has deleted from $G$ the vertices in $C \setminus W$. We iteratively modify the coloring $c$ by updating the set $T$ as follows (the set $W$ remains the same). While there exists a vertex $u \in V(G)$ such that $u \in T$ and $u \notin W$, let $v \in W$ such that $v \notin T$ (note that vertex $v$ exists, since $|T| \leq |\bar{M}| = |W|$), and swap $u$ and $v$ in the coloring, that is, now $u$ is a singleton in its color and $v$ is in a color containing vertices of $V(\bar{M})$. Since $u$ and $v$ are twins, this procedure indeed creates a proper coloring of $G$, which we also call $c$ with abuse of notation. Let us now argue that the weight of the coloring has not increased. Note that since $u \notin W$ and $v \in W$, it follows that $w(v) \geq w(u)$. If we denote by $S_i$ and $S_j$ the colors of $c$ before the swapping, so that $u \in S_i$ and $S_j = \{v\}$, and by $S'_i$ and $S'_j$ the corresponding colors after the swapping, note that $w(S_i) \geq w(u)$, $w(S_j) = w(v)$, $w(S'_i) = \max\{w(S_i), w(v)\}$, and $w(S'_j) = w(u)$. Therefore,

$$w(S'_i) + w(S'_j) = \max\{w(S_i), w(v)\} + w(u) \leq w(S_i) + w(v) = w(S_i) + w(S_j),$$

where we have used that $w(v) \geq w(u)$. Thus, at the end of this procedure we obtain a coloring $c$ with $w(c) \leq \sum_{v \in V(G)} w(v) - k$ such that every vertex in $C \setminus W$ is a singleton in its color class. We define $c'$ to be the restriction of $c$ to $G'$. Since every vertex in $C \setminus W = V(G) \setminus V(G')$ is a singleton in its color class in $c$, it follows that

$$w(c') = w(c) - \sum_{v \in C \setminus W} w(v) \leq \left( \sum_{v \in V(G)} w(v) - k \right) - \sum_{v \in C \setminus W} w(v) = \sum_{v \in V(G')} w(v) - k,$$

and the claim follows. ◀

We can easily apply Rule 2 exhaustively in polynomial time to all the classes $C \in \mathcal{C}$ with $|C| > |\bar{M}|$. We call an instance *reduced* if none of Rule 1 and Rule 2 can be applied anymore. The above discussion implies that if $(G, w, k)$ is a reduced instance, then

$$|V(G)| = |V(\bar{M})| + |K| \leq 2(k-1) + k_{\mathsf{s}} + (2^{k_{\mathsf{n}}} - 1) \cdot (k-1).$$

Using that $k_{\mathsf{s}} + k_{\mathsf{n}} \leq k - 1$, from the above equation we get that

$$|V(G)| \leq 2(k-1) + (2^{k-1} - 1) \cdot (k-1) = (2^{k-1} + 1) \cdot (k-1),$$

and the theorem follows. ◀

It is worth mentioning that the analysis of the kernel size in the proof of Theorem 2 is *tight*. Indeed, let $G$ consist of an antimatching $\bar{M}$ of size $k-1$, and let $K$ consist of $2^{k-1} - 1$ equivalence classes with $k-1$ vertices, each having a distinct non-complete neighborhood in the set consisting of one (arbitrary) vertex of each non-edge in $\bar{M}$. One can easily check that $|V(G)| = (2^{k-1} + 1) \cdot (k-1)$, that none of Rule 1 and Rule 2 can be applied to $G$, and that $G$ contains no antimatching strictly larger than $k-1$.

We complement the result of Theorem 2 by showing that, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, the problem does not admit polynomial kernels.

▶ **Theorem 3.** *The* Dual Weighted Coloring *problem does not admit a polynomial kernel unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$*, even on split graphs with only two different weights.*

**Proof.** We present a polynomial parameter transformation from the Set Cover problem parameterized by the size of the universe; Dom et al. [15] proved that this problem does not admit a polynomial kernel unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. Our reduction is almost the same as the reduction of Demange et al. [12] from Set Cover to Weighted Coloring on split graphs, only the vertex weights change. Let $(U, \mathcal{S}, k, \ell)$ be an instance of Set Cover, where $\mathcal{S}$ is a family of sets of elements over a universe $U$ of size $k$, and the question is whether there exists a subset $\mathcal{S}' \subseteq \mathcal{S}$ of at most $\ell$ sets covering all the elements of $U$. We construct an instance $(G, w, k')$ of Dual Weighted Coloring as follows. The graph $G$ contains a clique $K$ on $|\mathcal{S}|$ vertices and an independent set $I$ on $k$ vertices. The vertices of $K$ and $I$ are associated, respectively, with the sets in $\mathcal{S}$ and the elements in $U$. There is an edge between a vertex in $K$ and a vertex in $I$ if and only if the corresponding set does *not* contain the corresponding element. All the vertices in $K$ have weight $\ell$, and all the vertices in $I$ have weight $\ell + 1$. Note that $G$ is indeed a split graph with only two different weights. Finally, we set $k' = k(\ell + 1) - \ell$. Since we can clearly assume that $\ell \leq k$, as otherwise the instance is trivial, it follows that $k' = \mathcal{O}(k^2)$, which is required in a polynomial parameter transformation. We claim that $(U, \mathcal{S}, k, \ell)$ is a yes-instance of Set Cover if and only if $\sigma(G, w) \leq \sum_{v \in V(G)} w(v) - k'$.

Assume first that $(U, \mathcal{S}, k, \ell)$ is a yes-instance, and let $\mathcal{S}' \subseteq \mathcal{S}$ be a solution with $|\mathcal{S}'| \leq \ell$. We define a coloring $c$ of $G$ as follows. We start with a color for each vertex in $K$ and, for every element of $U$, we include its corresponding vertex of $I$ into one of the colors corresponding to the sets in $\mathcal{S}'$ containing that element. One can easily check that $w(c) \leq |\mathcal{S}| \cdot \ell + \ell = \sum_{v \in V(G)} w(v) - k'$.

Conversely, assume that $\sigma(G, w) \leq \sum_{v \in V(G)} w(v) - k' = |\mathcal{S}| \cdot \ell + \ell$, and let $c$ be a coloring of $(G, k)$ achieving this bound. Since $\sum_{v \in K} w(v) = |\mathcal{S}| \cdot \ell$ and the vertices in $I$ have weight $\ell + 1$, all the vertices in $I$ have to be included in at most $\ell$ out of the $|\mathcal{S}|$ colors of $c$ containing the vertices of the clique $K$. By construction of $G$, this is possible only if there exists a subset of at most $\ell$ sets in $\mathcal{S}$ covering all the elements of $U$, and the theorem follows. ◀

In view of Theorem 3, in what follows we focus on identifying graph classes on which the Dual Weighted Coloring problem admits a polynomial kernel.

▶ Remark. The problem clearly admits a kernel of size $\mathcal{O}(k)$ on sparse graphs, since if there are no large cliques, then the clique $K$ defined in the proof of Theorem 3 (that is, the remaining vertices of those in a maximum antimatching) is of constant size. More formally, if $\omega$ is the maximum clique size of a graph in the class, then we get a kernel with at most $2k - 2 + \omega$ vertices.

In our next result we provide a polynomial kernel on a relevant class of dense graphs, namely that of interval graphs.

▶ **Proposition 4.** *The* Dual Weighted Coloring *problem restricted to interval graphs admits a kernel with at most $k^3 - k^2 + k - 1$ vertices.*

**Proof.** We will proceed as in the proof of Theorem 2 and show that, if the input graph $G$ is an interval graph, then the number of equivalence classes is quadratic in the parameter $k$. As before, we assume that $\bar{M}$ is a maximum antimatching of $G$, that $|\bar{M}| \leq k - 1$, and also that Rule 1 and Rule 2 have been exhaustively applied. We also consider $K = V(G) \setminus V(\bar{M})$ and, as before, note that $K$ induces a clique.

We will show that the number of maximal cliques of $G$ is bounded by a linear function of $|\bar{M}|$. For that, we will make use of a well-known result of Fulkerson and Gross [22] stating that a graph $G$ is an interval graph if and only if the 0/1 incidence matrix $\mathcal{M}$ of vertices and maximal cliques of $G$ has the *consecutive ones property*, i.e., the columns of $\mathcal{M}$ can be permuted so that the ones in each row appear consecutively. A consequence of this result is that the set $\mathcal{C}$ of maximal cliques of $G$ can be arranged as $\{C_1, \ldots, C_p\}$ in such a way that, for each vertex $v$, there are indices $\ell_v$ and $r_v$ such that $v \in C_i$ if and only if $\ell_v \leq i \leq r_v$. It is worth mentioning that this property appears implicitly in [23, proof of Theorem 2].

Note that the incidence of a vertex to the maximal cliques completely defines its neighborhood. This implies that $G$ has at most $\binom{p}{2}$ neighborhood classes. Hence, for obtaining a cubic kernel it suffices to show that $p = \mathcal{O}(k)$. In what follows we provide explicit bounds on both $p$ and the size of the kernel; we will show later that these bounds are tight.

▶ **Claim 5.** *Let $G$ be an interval graph with $p \geq 2$ maximal cliques and let $\bar{M}$ be a maximum antimatching of $G$. Then $p \leq 2|\bar{M}| + 1$.*

**Proof.** We proceed by induction on $p$. Clearly, if $p \in \{2, 3\}$, then there are at least two non-adjacent vertices, so $|\bar{M}| \geq 1$ and $p \leq 2|\bar{M}| + 1$.

Let $\{C_1, \ldots, C_p\}$ be an ordering of the maximal cliques of $G$ as defined above. Since the $C_i$'s are maximal cliques, for any two consecutive cliques $C_i$ and $C_{i+1}$ it holds that $C_i \setminus C_{i+1} \neq \emptyset$ and $C_{i+1} \setminus C_i \neq \emptyset$. In particular, together with the consecutive ones property, this implies that $C_1$ has an exclusive vertex $u$, that is, a vertex that does not belong to any maximal clique other than $C_1$. Similarly, $C_p$ has an exclusive vertex $v$. Let $G' = G[V(G) \setminus \{u, v\}]$, $\bar{M}'$ be a maximum antimatching of $G'$, and $p'$ be the number of maximal cliques of $G'$. By the induction hypothesis, $p' \leq 2|\bar{M}'| + 1$.

Note that $C_2, \ldots, C_{p-1}$ are maximal cliques in $G'$, hence $p - 2 \leq p'$. On the other hand, $|\bar{M}'| \leq |\bar{M}| - 1$, otherwise $\bar{M}' \cup \{uv\}$ would be an antimatching of cardinality greater than $|\bar{M}|$ in $G$. Putting all together,

$$p - 2 \leq p' \leq 2|\bar{M}'| + 1 \leq 2(|\bar{M}| - 1) + 1 = 2|\bar{M}| - 1,$$

and $p \leq 2|\bar{M}| + 1$. This completes the proof of the claim. ◀

Since $K$ is a clique, there is an index $i$ such that $K \subseteq C_i$. For each $v \in K$, let $\ell_v$ be the smallest index such that $v \in C_{\ell_v}$ and $r_v$ be the largest index such that $v \in C_{r_v}$. Recall that $\ell_v$ and $r_v$ completely define the neighborhood of $v$. Note that for all $v \in K$, $\ell_v \leq i \leq r_v$ and that either $\ell_v \neq 1$ or $r_v \neq p$, since otherwise $v$ would be a universal vertex, which is impossible because we applied Rule 1 exhaustively. Hence, the maximum number of distinct possible combinations of indices $\ell_v$ and $r_v$ is $i \cdot (p - i + 1) - 1$, which is maximized when $i = \lfloor \frac{p+1}{2} \rfloor$, giving a total number of

$$\left\lfloor \frac{p+1}{2} \right\rfloor \cdot \left\lceil \frac{p+1}{2} \right\rceil - 1$$

distinct equivalence classes. Using the fact that $p \leq 2|\bar{M}|+1$ by Claim 5, and that $|\bar{M}| \leq k-1$, we get

$$\left\lfloor \frac{p+1}{2} \right\rfloor \cdot \left\lceil \frac{p+1}{2} \right\rceil - 1 = k^2 - 1.$$

Since, by virtue of the application of Rule 2, each class has at most $(k-1)$ elements, we have

$$|V(G)| = |V(\bar{M})| + |V(K)| \leq 2(k-1) + (k-1) \cdot (k^2 - 1) = k^3 - k^2 + k - 1. \qquad \blacktriangleleft$$

Similarly to Theorem 2, we can show that the analysis of the kernel size in the proof of Proposition 4 is tight. Indeed, given an integer $k$, it is possible to build an interval graph with $2k-1$ maximal cliques, with $(k-1)$ vertices belonging exactly to cliques $C_i, \ldots, C_j$, for each

$$(i, j) \in [1, k] \times [k,\ 2k-1] \setminus \{(1, 2k-1)\},$$

and one exclusive vertex in each clique, except for clique $C_k$, corresponding to vertices of $V(\bar{M})$. That interval graph attains the bound in the statement of Proposition 4 and cannot be reduced by Rule 1 or Rule 2.

▶ Remark. The result of Proposition 4 cannot be generalized to chordal graphs, as split graphs are chordal, and by Theorem 3 the existence of a polynomial kernel on split graphs would imply that $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

Our last result deals with a subclass of split graphs motivated by the fact that Dual Weighted Coloring on split graphs seems to have a close relation with the Set Cover problem.

▶ **Proposition 5.** *The* Dual Weighted Coloring *problem restricted to split graphs such that each vertex in the clique has at most $d$ non-neighbors in the stable set, for some constant $d \geq 1$, admits a kernel with at most $k^{d+1}$ vertices. Furthermore, for any $d \geq 2$ and $\varepsilon > 0$, a kernel with $\mathcal{O}(k^{\frac{d-3}{2}-\varepsilon})$ vertices does not exist unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

**Proof.** For the positive result, let $(G, w, k)$ be an instance of Dual Weighted Coloring, with $G$ being a split graph such that each vertex in the clique has at most $d$ non-neighbors in the stable set, for some integer $d \geq 2$. We mimic the proof of Theorem 2, and we slightly change the analysis. Recall that $|V(\bar{M})| \leq 2(k-1)$ and that the number of special classes in $\mathcal{C}$, each containing exactly one vertex, is at most $k_{\mathsf{s}}$. Concerning the normal classes in $\mathcal{C}$, we will obtain an improved bound using that each vertex in the clique has at most $d$ non-neighbors in the stable set, and therefore in the graph $G$ itself as well. Thus, the number of distinct neighborhoods in the set consisting of the avoidable vertices in

the normal edges in $\bar{M}$, which is an upper bound on the number of normal classes, is at most $\sum_{i=0}^{d} \binom{k_{\mathsf{n}}}{i} \leq \sum_{i=0}^{d} \frac{k_{\mathsf{n}}^i}{i!} \leq \max\{2, k_{\mathsf{n}}^d\}$, where the last inequality can be easily proved by induction.

Therefore, if $(G, w, k)$ is a reduced instance, then

$$\begin{aligned} |V(G)| &= |V(\bar{M})| + |K| \leq 2(k-1) + k_{\mathsf{s}} + k_{\mathsf{n}}^d \cdot (k-1) \\ &\leq 2(k-1) + (k-1)^d \cdot (k-1) \leq k^{d+1}. \end{aligned}$$

For the negative result, we reuse the reduction of Theorem 3, but starting from the $d$-Set Cover problem, that is, the restriction of Set Cover to instances where each set contains at most $d$ elements. Hermelin and Wu [30] proved that, for any fixed $d \geq 2$, $d$-Set Cover does not admit kernels of size $\mathcal{O}(k^{d-3-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, where $k$ is the size of the solution. Nevertheless, in the hardness proof for $d$-Set Cover given in [30], the size of the universe of the constructed instance is equal to $kd$. Therefore, we can conclude that $d$-Set Cover does not admit kernels of size $\mathcal{O}(k^{d-3-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, where $k$ is the size of the universe. Moreover, the results in [30] also rule out the existence of a *bikernel*, that is, a relaxed kernelization notion where the output instance is not necessarily of the same problem.

Given an instance $(U, \mathcal{S}, k, \ell)$ of $d$-Set Cover, where $k$ is the size of the universe, we construct an instance $(G, w, k')$ of Dual Weighted Coloring as in the proof of Theorem 3. Note that $G$ is indeed a split graph such that each vertex in the clique has at most $d$ non-neighbors in the stable set, and recall that $k' = k(\ell+1) - \ell$. Since we may assume that $\ell \leq k$, it follows that $k' \leq k^2$. Assume for contradiction that Dual Weighted Coloring restricted to this type of instances admits a kernel with $\mathcal{O}(k^{\frac{d-3}{2}-\varepsilon})$ vertices, for some $\varepsilon > 0$. Then the composition of the above reduction with such a kernel would yield a *bikernel* for $d$-Set Cover of size $\mathcal{O}(k^{d-3-\varepsilon})$ for some $\varepsilon > 0$, which is impossible by the results of [30] unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. ◀

## 5 Further research

In this article we investigated the dual parameterization of the Weighted Coloring problem, and we provided several positive and negative results, especially concerning polynomial kernelization. It would be interesting to identify other classes of (dense) graphs on which the problem admits polynomial kernels. It remains to close the gap in the degree of the polynomial kernels on the subclasses of split graphs considered in Proposition 5. Another question is whether the cubic kernel on interval graphs given in Proposition 4 can be improved, even on *proper* interval graphs.

Concerning Theorem 1, using the techniques of Björklund et al. [4] it may be possible to compute the values $T(X, i)$ defined in the proof, for every $X$ and $i$, in time $2^{|V(\bar{M})|} \cdot n^{\mathcal{O}(1)} \cdot \max_{v \in V(G)} w(v)$, which would yield an overall running time of $4^k \cdot n^{\mathcal{O}(1)} \cdot \max_{v \in V(G)} w(v)$. Finally, one could try to prove lower bounds under the SETH (see [36]) on the running time of any FPT algorithm solving Dual Weighted Coloring, hopefully getting close to the running time given in Theorem 1.

## References


1 Júlio Araújo, Julien Baste, and Ignasi Sau. Ruling out FPT algorithms for Weighted Coloring on forests. *Theoretical Computer Science*, 729:11–19, 2018.

2 Júlio Araújo, Nicolas Nisse, and Stéphane Pérennes. Weighted Coloring in Trees. *SIAM Journal on Discrete Mathematics*, 28(4):2029–2041, 2014.

3 Manu Basavaraju, Mathew C. Francis, M. S. Ramanujan, and Saket Saurabh. Partially Polynomial Kernels for Set Cover and Test Cover. *SIAM Journal on Discrete Mathematics*, 30(3):1401–1423, 2016. doi:10.1137/15M1039584.

4 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set Partitioning via Inclusion-Exclusion. *SIAM Journal on Computing*, 39(2):546–563, 2009.

5 Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009.

6 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization Lower Bounds by Cross-Composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014.

7 Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theoretical Computer Science*, 412(35):4570–4578, 2011. doi:10.1016/j.tcs.2011.04.039.

8 Édouard Bonnet and Vangelis Th. Paschos. Dual parameterization and parameterized approximability of subset graph problems. *RAIRO - Operations Research*, 51(1):261–266, 2017. doi:10.1051/ro/2016018.

9 Benny Chor, Mike Fellows, and David W. Juedes. Linear Kernels in Linear Time, or How to Save $k$ Colors in $O(n^2)$ Steps. In *Proc. of the 30th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 3353 of *LNCS*, pages 257–269, 2004. doi:10.1007/978-3-540-30559-0_22.

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

11 Dominique de Werra, Marc Demange, Bruno Escoffier, Jérôme Monnot, and Vangelis Th. Paschos. Weighted coloring on planar, bipartite and split graphs: Complexity and approximation. *Discrete Applied Mathematics*, 157(4):819–832, 2009.

12 Marc Demange, Dominique de Werra, Jérôme Monnot, and Vangelis Th. Paschos. Weighted Node Coloring: When Stable Sets Are Expensive. In *Proc. of the 28th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 2573 of *LNCS*, pages 114–125. Springer, 2002.

13 Marc Demange, Pascal Grisoni, and Vangelis Th. Paschos. Approximation Results for the Minimum Graph Coloring Problem. *Information Processing Letters*, 50(1):19–23, 1994. doi:10.1016/0020-0190(94)90039-6.

14 Reinhard Diestel. *Graph Theory*, volume 173. Springer-Verlag, 4th edition, 2010.

15 Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization Lower Bounds Through Colors and IDs. *ACM Transactions on Algorithms*, 11(2):13:1–13:20, 2014. doi:10.1145/2650261.

16 R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

17 Rong-chii Duh and Martin Fürer. Approximation of $k$-Set Cover by Semi-Local Optimization. In *Proc. of the 29th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 256–264, 1997. doi:10.1145/258533.258599.

18 Bruno Escoffier. Saving Colors and Max Coloring: Some Fixed-Parameter Tractability Results. In *Proc. of the 42nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 9941 of *LNCS*, pages 50–61, 2016.

19 Bruno Escoffier, Jérôme Monnot, and Vangelis Th. Paschos. Weighted Coloring: further complexity and approximability results. *Information Processing Letters*, 97(3):98–103, 2006.

20 Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2010.

21 Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *Journal of Computer and System Sciences*, 77(1):91–106, 2011.

22 Delbert Fulkerson and Oliver Gross. Incidence matrices and interval graphs. *Pacific journal of mathematics*, 15(3):835–855, 1965.

23 P. Gilmore and A. Hoffman. A characterization of comparability graphs and of interval graphs. *Canadian Journal of Mathematics*, 16:539–548, 1964.

24 D. J. Guan and Xuding Zhu. A Coloring Problem for Weighted Graphs. *Information Processing Letters*, 61(2):77–81, 1997.

25 Gregory Z. Gutin, Mark Jones, and Anders Yeo. Kernels for below-upper-bound parameterizations of the hitting set and directed dominating set problems. *Theoretical Computer Science*, 412(41):5744–5751, 2011. doi:10.1016/j.tcs.2011.06.018.

26 Magnús M. Halldórsson. Approximating Discrete Collections via Local Improvements. In *Proc. of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 160–169, 1995. URL: http://dl.acm.org/citation.cfm?id=313651.313687.

27 Magnús M. Halldórsson. Approximating $k$-Set Cover and Complementary Graph Coloring. In *Proc. of the 5th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, volume 1084 of *LNCS*, pages 118–131, 1996. doi:10.1007/3-540-61310-2_10.

28 Refael Hassin and Shlomo Lahav. Maximizing the Number of Unused Colors in the Vertex Coloring Problem. *Information Processing Letters*, 52(2):87–90, 1994. doi:10.1016/0020-0190(94)00113-8.

29 Frédéric Havet and Leonardo Sampaio. On the Grundy and $b$-Chromatic Numbers of a Graph. *Algorithmica*, 65(4):885–899, 2013.

30 Danny Hermelin and Xi Wu. Weak compositions and their applications to polynomial lower bounds for kernelization. In *Proc. of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 104–113, 2012.

31 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of $k$-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

32 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

33 Richard M. Karp. Reducibility Among Combinatorial Problems. In *Proc. of a symposium on the Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

34 Telikepalli Kavitha and Julián Mestre. Max-coloring paths: tight bounds and extensions. *Journal of Combinatorial Optimization*, 24(1):1–14, 2012.

35 E. L. Lawler. A note on the complexity of the chromatic number problem. *Information Processing Letters*, 5(3):66–67, 1976. doi:10.1016/0020-0190(76)90065-X.

36 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the Exponential Time Hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011. URL: http://albcom.lsi.upc.edu/ojs/index.php/beatcs/article/view/96.

37 Sriram V. Pemmaraju, Sriram Penumatcha, and Rajiv Raman. Approximating interval coloring and max-coloring in chordal graphs. *ACM Journal of Experimental Algorithmics*, 10, 2005.

38 Chee-Keng Yap. Some Consequences of Non-Uniform Conditions on Uniform Classes. *Theoretical Computer Science*, 26:287–300, 1983.

# Computing Kernels in Parallel: Lower and Upper Bounds

**Max Bannach**
Institute for Theoretical Computer Science, Universität zu Lübeck, Lübeck, Germany
bannach@tcs.uni-luebeck.de
https://orcid.org/0000-0002-6475-5512

**Till Tantau**
Institute for Theoretical Computer Science, Universität zu Lübeck, Lübeck, Germany
tantau@tcs.uni-luebeck.de

**Abstract**

Parallel fixed-parameter tractability studies how parameterized problems can be solved in parallel. A surprisingly large number of parameterized problems admit a high level of parallelization, but this does not mean that we can also efficiently compute small problem kernels in parallel: known kernelization algorithms are typically highly sequential. In the present paper, we establish a number of upper and lower bounds concerning the sizes of kernels that can be computed in parallel. An intriguing finding is that there are complex trade-offs between kernel size and the depth of the circuits needed to compute them: For the vertex cover problem, an exponential kernel can be computed by $AC^0$-circuits, a quadratic kernel by $TC^0$-circuits, and a linear kernel by randomized NC-circuits with derandomization being possible only if it is also possible for the matching problem. Other natural problems for which similar (but quantitatively different) effects can be observed include tree decomposition problems parameterized by the vertex cover number, the undirected feedback vertex set problem, the matching problem, or the point line cover problem. We also present natural problems for which computing kernels is inherently sequential.

**2012 ACM Subject Classification** Theory of computation → Design and analysis of algorithms

**Keywords and phrases** parallel computation, fixed-parameter tractability, kernelization

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.13

**Related Version** `https://arxiv.org/abs/1807.03604`

## 1 Introduction

The core objective of parameterized complexity has classically been to determine which problems can be solved in "FPT time," meaning time $f(k) \cdot n^c$ for instances of size $n$, where $c$ is a constant, $f$ is an arbitrary computable function (usually at least exponential), and $k$ is a hopefully small instance parameter. Over the last 25 years, theoreticians in the field have been very successful at determining which problems admit algorithms of this kind and practitioners have been very successful at implementing them. In both cases, the focus has traditionally been on finding *sequential* algorithms, but in recent years interest in *parallel* algorithms has sparked, leading to the new field of parallel fixed parameter tractability.

In classical sequential FPT algorithms, *kernelization algorithms* play a key role. They shrink the input to a small but difficult core (called the *kernel*), leading to the following design principle of modern parameterized algorithms: Firstly, in polynomial time, a kernelization algorithm computes a kernel that is, secondly, solved using an exponential (or worse) time

© Max Bannach and Till Tantau;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 13; pp. 13:1–13:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

algorithm – yielding a total running time of the form $f(k)+n^c$. Regarding the parallelization of these two algorithmic steps, it turns out that the second one is usually the easier one: the kernel is typically processed using the search tree technique or just by "brute force," both of which allow natural parallelizations. In contrast, kernelization algorithms are typically described in a very sequential way, namely "apply these reduction rules over and over again." This means that designing parallel fixed-parameter algorithms effectively means designing parallel kernelization algorithms – which is exactly what this paper addresses.

**Our Contributions.** We start our systematic investigation of parallel kernelization by linking the parameterized analogues of the NC-hierarchy to kernel computation using NC-circuits. Such a link is already known for FPT and kernels computed in polynomial time. We establish a circuit version of the well-known result that all algorithms running in time $f(k) \cdot n^c$ can also be implemented with running time $g(k) + n^c$: We can turn any circuit family of size $f(k) \cdot n^c$ and depth $f(k) + c \log^i n$ into one of size $g(k) + n^{c'}$ and depth $c' \log^i n$ (note that we can remove the parameter dependence from the depth).

The bulk of the paper consists of a series of lower and upper bounds on the size of kernels that can be computed by circuits of certain depths. We show that for natural problems like the vertex cover problem intriguing trade-offs arise: the faster our algorithm, the worse our kernel. For $p$-VERTEX-COVER we show that a simple exponential kernel can be computed in $\mathrm{AC}^0$, a quadratic kernel can be computed in $\mathrm{TC}^0$, and a linear kernel can be computed in randomized NC. Other problems for which we establish similar results include the tree width, path width, and tree depth problems parameterized by the vertex cover number of the input graph.

On the negative side, we also establish a number of lower bounds for the parallel computation of small kernels. We show that a classical $2k$ kernel for the vertex cover problem can only be computed in parallel if the maximum matching problem for bipartite graphs is in NC, for which $\mathrm{RNC}^2$ and quasi-$\mathrm{NC}^2$ are the best known upper bounds; that classic reduction rules for feedback vertex set are P-complete (but an exponential kernel can be computed in $\mathrm{AC}^2$); that for the point line cover problem we cannot (absolutely, without any assumptions) compute any kernel in $\mathrm{AC}^0$ (but we can compute a quadratic one in $\mathrm{TC}^0$); and that kernels for generalized versions of Horn satisfiability, linear programming, and maximum flow cannot be computed in polylogarithmic time unless $\mathrm{NC} = \mathrm{P}$. The later results in fact presents three *natural* FPT-complete problems, which demonstrate the limits of fixed parameter parallelization.

Table 1 summarizes which trade-offs are established in this paper between the parallel time needed to compute kernels and their sizes.

**Related Work.** Parameterized complexity is a rapidly growing field, see [14, 15, 18] for an introduction, in which parallelization is a recent research direction. Early research in the late 1990s was done by Cai, Chen, Downey, and Fellows [9] who studied parameterized logaritmic space. A structural study of parameterized logspace and parameterized circuit classes was started around 2015 by Elberfeld et al. [16]; see also the references therein. The parameterized version of the NC-hierarchy we use in this paper was introduced in [2]. Chen and Flum studied lower bounds in this context and especially provide some details and alternative characterizations for parameterized $\mathrm{AC}^0$. There is a huge body of literature on polynomial-time algorithms for computing small kernels, but the authors are not aware of results concerning how quickly these kernels can be computed in parallel.

**Table 1** An overview of problems studied in this paper, showing which kernel size can be achieved in certain layers of the NC-hierarchy. An explicit function represents the best bound the authors are aware of, pointed out in this work or (for the P-column) in cited works; $f(k)$ corresponds to kernels originating from Theorem 2.3; and "–" means that there is no kernel of any size (either absolutely or unless $\mathrm{TC}^0 = \mathrm{L}$ for $-^1$, unless $\mathrm{TC}^0 = \mathrm{NL}$ for $-^2$, unless $\mathrm{TC}^0 = \mathrm{P}$ for $-^3$, unless $\mathrm{NC} = \mathrm{P}$ for $-^4$, unless $\mathrm{P} \subseteq \mathrm{RNC}$ for $-^5$, and unless $\mathrm{NC}^1 = \mathrm{P}$ for $-^6$). For problems parameterized by the vertex cover number, $S$ is the given vertex cover; the $\delta$ in the first column can be any fixed positive integer.

| Problem | Kernel size achievable in | | | | |
|---|---|---|---|---|---|
| | $\mathrm{AC}^0$ | $\mathrm{TC}^0$ | NC | RNC | P |
| $p$-VERTEX-COVER | $2^{\sqrt[\delta]{k}}$ | $k^2 + 2k$ | $k^2 + 2k$ | $2k$ | $2k - c \log k$ |
| $p$-MATCHING | $2^{\sqrt[\delta]{k}}$ | $6k^2$ | $6k^2$ | 1 | 1 |
| $p_{\mathrm{vc}}$-TREE-WIDTH | $2^{\sqrt[\delta]{\lvert S\rvert}}$ | $\lvert S\rvert^3$ | $\lvert S\rvert^3$ | $\lvert S\rvert^3$ | $\lvert S\rvert^3$ |
| $p_{\mathrm{vc}}$-PATH-WIDTH | $2^{\sqrt[\delta]{\lvert S\rvert}}$ | $\lvert S\rvert^3$ | $\lvert S\rvert^3$ | $\lvert S\rvert^3$ | $\lvert S\rvert^3$ |
| $p_{\mathrm{vc}}$-TREE-DEPTH | $2^{\sqrt[\delta]{\lvert S\rvert}}$ | $\lvert S\rvert^3$ | $\lvert S\rvert^3$ | $\lvert S\rvert^3$ | $\lvert S\rvert^3$ |
| $p$-POINT-LINE-COVER | – | $k^2$ | $k^2$ | $k^2$ | $k^2$ |
| $p$-FEEDBACK-VERTEX-SET | – | $-^1$ | $f(k)$ | $f(k)$ | $2k^2 + k$ |
| $p$-STRONG-BACKDOOR-2CNF-SAT | – | $-^2$ | $f(k)$ | $f(k)$ | $f(k)$ |
| $p$-STRONG-BACKDOOR-HORN-SAT | – | $-^3$ | $-^3$ | $-^5$ | $f(k)$ |
| $p$-MIXED-INTEGER-PROGRAMMING | – | $-^6$ | $-^4$ | $-^5$ | $f(k)$ |
| $p$-MAX-FLOW-QUANTITIES | – | $-^6$ | $-^4$ | $-^5$ | $f(k)$ |

**Organization of This Paper.** We review basic terminology in Section 2, where we also establish the link between parameterized parallel complexity and parallel kernel computation. Each of the following sections studies a different well-known parameterized problem and establishes trade-offs between kernel size and speed. We start with the vertex cover and the matching problem in Section 3, followed by the feedback vertex set problem in Section 4, structural parameterizations for tree width, path width, and tree depth in Section 5, the $p$-POINT-LINE-COVER problem in Section 6, and finally generalized versions of Horn satisfiability, linear programming, and maximum flow in Section 7. Due to lack of space, most proofs are given only in the full version, but we sketch some of them in the main text.

## 2 Parameterized Parallel Complexity Classes and Kernelization

We use standard terminology of parameterized complexity theory, see for instance [18]. A *parameterized problem* is a tuple $(Q, \kappa)$ consisting of a *language* $Q \subseteq \Sigma^*$ and a *parameterization* $\kappa\colon \Sigma^* \to \mathbb{N}$. The complexity of $\kappa$ should not exceed the power of the classes that we consider, and since we study small parameterized circuit classes, we require $\kappa$ to be computable by DLOGTIME-uniform constant-depth AC-circuits or, equivalently, to be first-order computable. We denote parameterized problems by a leading "$p$-" as in $p$-VERTEX-COVER, and, whenever the parameterization $\kappa$ is not clear from the context, we add it as an index as in $p_{\mathrm{vc}}$-TREE-WIDTH. A parameterized problem $(Q, \kappa)$ is *fixed-parameter tractable* (or in FPT) if there is a computable function $f\colon \mathbb{N} \to \mathbb{N}$ and a constant $c$ such that we can decide $x \in Q$ in time $f(\kappa(x)) \cdot |x|^c$ for all $x \in \Sigma^*$. In this paper we study the parallel complexity of parameterized problems, that is, the parameterized counter part of the NC-hierarchy. Formally we study the following classes, see for instance [2, 11] for a detailed discussion:

▶ **Definition 2.1.** For each $i > 0$, a parameterized problem $(Q, \kappa)$ is in DLOGTIME-uniform para-AC$^i$ if there exists a computable function $f\colon \mathbb{N} \to \mathbb{N}$, a constant $c \in \mathbb{N}$, and a family of AC-circuits $(C_{n,k})_{n,k\in\mathbb{N}}$ such that:

1. For all $x \in \Sigma^*$ we have $C_{|x|,\kappa(x)}(x) = 1 \iff x \in Q$.
2. The depth of each $C_{n,k}$ is at most $f(k) + c \log^i n$.
3. The size of each $C_{n,k}$ is at most $f(k) \cdot n^c$.
4. There is a deterministic Turing machine that on input of $\mathrm{bin}(i)\#\,\mathrm{bin}(k)\#\,\mathrm{bin}(n)$, where $\mathrm{bin}(x)$ is the binary encoding of $x$, outputs the $i$th bit of a suitable encoding of $C_{n,k}$ in at most $f(k) + c \log n$ steps.

The class para-AC$^0$ is defined as above, but with circuits of *constant depth*. Additionally, we define for all $i \geq 0$ the class para-AC$^{i\uparrow}$ with circuits of depth $f(k) \cdot \log^i n$. In particular, para-AC$^{0\uparrow}$-circuits have depth $f(k)$. Recall that AC-circuits are defined over the standard base of NOT-, OR-, and AND-gates and that the last two may have unlimited fan-in. The same definition works for NC-circuits (all gates have bounded fan-in) and TC-circuits (additional threshold gates are allowed). It is known that the parameterized classes inherit their inclusion structure from their classical counterparts [2]:

$$\text{para-AC}^0 \subsetneq \text{para-TC}^0 \subseteq \text{para-NC}^1 \subseteq \text{para-AC}^1 \subseteq \text{para-TC}^1 \subseteq \cdots \subseteq \text{para-NC} \subseteq \text{FPT}.$$

**A Parallel Analogue of "FPT = Kernels Computable in Polynomial Time".** One of the most fruitful aspects of parameterized complexity is the concept of *kernelization*. Let $f\colon \mathbb{N} \to \mathbb{N}$ be a computable function. A *kernelization* of a parameterized problem $(Q, \kappa)$ is a self-reduction $K\colon \Sigma^* \to \Sigma^*$ such that for every $x \in \Sigma^*$ we have $x \in Q \iff K(x) \in Q$ and $|K(x)| \leq f(\kappa(x))$. The images of $K$ are called *kernels* and as they later need to be processed by at least exponential-time algorithms, we are interested in kernels that are as small as possible – while they still need to be efficiently computable, meaning in polynomial time from the view point of FPT theory. The following result is well-known and gives a deep connection between parameterized complexity and kernelization:

▶ **Fact 2.2** (for instance [18]). *A decidable parameterized problem $(Q, \kappa)$ is in* FPT *if, and only if, it admits a polynomial-time computable kernelization.*

The following theorem shows that the same relation also connects the AC-hierarchy with its parameterized counterpart. Note that in the theorem the AC$^i$-circuits are really "normal AC$^i$-circuits," meaning that their size is just polynomial in the input length.

▶ **Theorem 2.3.** *A decidable parameterized problem $(Q, \kappa)$ is in* para-AC$^i$ *if, and only if, it admits a kernelization computable by a* DLOGTIME*-uniform family of* AC$^i$*-circuits.*

**Sketch of Proof.** For the reverse direction, let $(C_n)_{n\in\mathbb{N}}$ be a family computing a kernelization. Circuit $C_{n,k}$ uses $C_n$ as a first black box and reduces the input to an instance of size at most $f(\kappa(x))$. Then the circuit essentially applies naive "brute force" in the form of a big OR-gate that checks if any element of $Q$ of length at most $f(\kappa(x))$ equals the computed kernel. For the other direction let $\tilde{k} \in \mathbb{N}$ be the maximum $k$ such that $f(k) \leq c \log^i n$ and let $C_n$ consist of $\tilde{k}$ subcircuits $C_n^0, \ldots, C_n^{\tilde{k}}$ that are evaluated in parallel. The circuit $C_n^j$ first checks on input $x$ whether $\kappa(x) = j$, and, if so, uses $C_{n,j}$ to solve the problem and outputs a trivial kernel. Otherwise, it signals that it is not responsible for this instance. If any $C_n^j$ produces a kernel, then $C_n$ just presents this kernel as result. Otherwise $C_n$ can just present the input as output, as we have $\kappa(x) > \tilde{k}$ and $f(\kappa(x)) > c \log^i n$. ◀

The theorem also holds if we replace AC$^i$ with NC$^i$ or TC$^i$. The only exception is NC$^0$, as this class may not be powerful enough to compute $\kappa$.

**Application: Improve the Work of Parallel Algorithms.** When we study the performance of parallel algorithms, we usually do not only measure the time of the algorithm (as we would in the sequential case), but also its *work* (the total number of computational steps performed by the algorithm). This is important as a parallel algorithm may need polynomially many processors to reach its promised runtime: For instance, an algorithm that runs in time $O(\log n)$ with $O(n^2)$ work will need at least time $O(n^2/p)$ on a machine with $p$ processors – which is bad if there exists a linear time sequential algorithm and $p < n$. In the circuit model the parallel time of an algorithm corresponds to the depth of the circuit, and the work to its size. While the layers of the AC- and para-AC-hierarchy measure the time of parallel algorithms quite precisely, they only require the size of the circuits to be polynomial or to be bounded by $f(k) \cdot n^c$, respectively. Using Theorem 2.3, we can improve the work of any parameterized parallel algorithm from $f(k) \cdot n^c$ to $g(k) + n^{c'}$ while, at the same time, reducing the depth of the circuit from $f(k) + c \log^i n$ to $c' \log^i n$.

▶ **Lemma 2.4.** *Let $(Q, \kappa)$ be a parameterized problem with $(Q, \kappa) \in$ para-AC$^i$. Then there are a computable function $g\colon \mathbb{N} \to \mathbb{N}$ and a constant $c'$ such that there is a* DLOGTIME*-uniform family $(C'_{n,k})_{n,k\in\mathbb{N}}$ of* para-AC$^i$*-circuits that decides $(Q, \kappa)$ and in which every $C'_{n,k}$ has depth at most $c' \log^i n$ and size at most $g(k) + n^{c'}$.*

**Sketch of Proof.** Use the AC$^i$-circuit from Theorem 2.3 to compute a kernel in depth $c \log^i n$ and solve the kernel using "brute force" in constant depth afterwards. ◀

Note that the function $g$ from the lemma may grow exponentially faster then $f$, as the circuit from the lemma internally solves an instance $x'$ with $|x'| \leq f(\kappa(x))$ and $\kappa(x') \leq f(\kappa(x))$. A direct application of Lemma 2.4 is therefore only of theoretical interest. It shows, however, that we can always search for parameterized parallel algorithms that run in polylogarithmic time and whose work is polynomial plus an *additive* term depending only on the parameter.

## 3 Parallel Kernels for Vertex Cover and Matching

The parameterized vertex cover problem is a prime example used to demonstrate many different kernelization techniques, and an outrider in the race for small kernels. In this section we revisit the problem from the point of view of circuit complexity and establish a link between circuit complexity and kernel size. An early result in this context is due to Cai et al. [9] which, translated into the terminology of the present paper, implies that a kernel for $p$-VERTEX-COVER can be computed in logarithmic space and, hence, in AC$^1$. Elberfeld et al. [16] later noticed that the kernel of size $k^2 + 2k$ computed by Cai et al. can actually also be computed in TC$^0$. This result was later once more refined by showing that the same kernel can be computed in para-AC$^0$ [2]. Together with Theorem 2.3 this implies that a kernel of size $f(k)$ can be computed in AC$^0$ for some computable function $f$. In fact, we can improve the bound in this case to $2^{\sqrt[\delta]{k}}$ for any fixed $\delta > 0$:

▶ **Lemma 3.1.** *For every $\delta \in \mathbb{N}$ there is a* DLOGTIME*-uniform family of* AC$^0$*-circuits that, on input of a tuple $(G, k)$, outputs a $p$-*VERTEX-COVER *kernel with at most $2^{\sqrt[\delta]{k}}$ vertices.*

**Proof.** Let $I$ be the input instance and let $n = |I|$ be the size of its encoding. The circuit first checks if we have $k \leq \log^\delta(n)$. If not, we have $2^{\sqrt[\delta]{k}} > n$ and the instance is already the desired kernel. Otherwise the circuit can simulate threshold gates up to $k$ using standard hashing techniques, as AC$^0$-circuits can simulate polylogarithmic threshold gates [30]. Since the TC$^0$-circuit from Elberfeld et al. [16] only uses threshold gates up to $k$, it follows that the AC$^0$-circuit under construction can simulate this TC$^0$-circuit, which completes the proof. ◀

The central observation in the proof of Lemma 3.1 is that the threshold-gates in the corresponding family of $\mathrm{TC}^0$-circuits only "count up to the parameter." We will use exactly the same trick for other $\mathrm{TC}^0$-kernelizations, but will then only formulate it as corollary. Summarizing the statements from above, we can compute an exponential kernel for $p$-VERTEX-COVER in $\mathrm{AC}^0$ and a quadratic kernel in $\mathrm{TC}^0$. However, the best known kernelizations for $p$-VERTEX-COVER are able to produce *linear kernels* – and a reasonable next step is to implement them in parallel as well. Unfortunately, this is a way more challenging task, as both the classical $3k$ kernel based on crown decomposition [14] and the $2k$ kernel due to Chen et al. [10] require the computation of sufficiently large matchings. We can state this more precisely for the latter observation, by showing that the core part of the kernelization is NC-equivalent to computing maximum matchings in bipartite graphs. The kernelization of Chen et al. is based on the following fact, known as the Nemhauser–Trotter Theorem:

▶ **Fact 3.2** ([29]). *Let $G = (V, E)$ be a graph and $I = \{ x_v \mid v \in V \}$ be a set of variables. For every optimal solution $\beta\colon I \to \mathbb{R}$ for the following linear program (*LPVC*)*

$$\begin{aligned} \min \textstyle\sum_{v \in V} x_v & \\ x_u + x_v \geq 1 & \quad \textit{for all } \{u, v\} \in E \\ x_v \geq 0 & \quad \textit{for all } v \in V \end{aligned}$$

*let $V_0 = \{ v \mid \beta(x_v) < 1/2 \}$, $V_{1/2} = \{ v \mid \beta(x_v) = 1/2 \}$, $V_1 = \{ v \mid \beta(x_v) > 1/2 \}$ be a partition of $V$. There is a minimum vertex cover $S$ of $G$ that satisfies $V_1 \subseteq S \subseteq V_1 \cup V_{1/2}$.*

Chen et al. have shown that one can obtain the desired kernel from a solution of LPVC by discarding the vertices of $V_0$ and by taking the vertices of $V_1$ into the solution. The remaining $2k$ vertices of $V_{1/2}$ constitute the kernel [10]. The following theorem shows that solving LPVC is tightly linked to the maximum matching problems for bipartite graphs.

▶ **Theorem 3.3.** *Computing a solution for* LPVC *is* NC*-equivalent to computing a maximum matching in bipartite graphs.*

**Sketch of Proof.** For the first direction we construct a bipartite auxiliary graph $H$ such that an optimal vertex cover of $H$ can be turned into a half-integral solution of LPVC on $G$. This vertex cover of $H$ can be computed using a maximum matching subroutine and König's Theorem [25]. In the second direction we wish to compute a maximum matching in a bipartite graph with the help of LPVC. We first turn an optimal real solution of LPVC into an optimal half-integral solution. Afterwards, we transform this half-integral solution into an optimal integral solution, that is, into a minimal vertex cover. Applying König's Theorem again results in the desired matching. ◀

The parallel complexity of the maximum matching problem is still not fully resolved. The currently best parallel algorithms run in $\mathrm{RNC}^2$ [28] or quasi-$\mathrm{NC}^2$ [17]. From the theorem we can deduce that we can compute the Nemhauser–Trotter-based $2k$-vertex kernel for $p$-VERTEX-COVER in RNC and quasi-NC; and we can deduce that we cannot compute this kernel in NC without improving the parallel complexity of the maximum matching problem – which is a longstanding open problem.

▶ **Corollary 3.4.** *There is a* DLOGTIME*-uniform family of* NC*-circuits of polylogarithmic depth that, on input of a graph $G = (V, E)$ and an integer $k$, outputs a kernel of $p$-*VERTEX-COVER *with at most $2k$ vertices. The circuits of the family either use randomness and have size $|V|^c$, or are deterministic and of size $|V|^{c \log |V|}$.*

Note that other kernels that are based on the Nemhauser–Trotter Theorem, such as the one by Soleimanfallah and Yeo [31], or the one by Lampis [27], also do not bypass Theorem 3.3. A natural goal is, thus, to compute linear kernels for $p$-VERTEX-COVER in NC – most likely using an algorithm that does not rely on a LPVC relaxation. Table 1 summarizes the complexity of computing kernels of certain size for $p$-VERTEX-COVER.

Since $p$-MATCHING turns out to be an obstruction for parallel kernelization, it is a natural question in the light of this paper, whether or not we are able to compute polynomial kernels for the matching problem in NC. Note that the problem is in para-$\mathrm{AC}^0$, and hence we can compute a size-$f(k)$ kernel in $\mathrm{AC}^0$; and since MATCHING $\in$ RNC we can compute a size-1 kernel in RNC.

▶ **Lemma 3.5.** *There is a* DLOGTIME*-uniform family of* $\mathrm{TC}^0$*-circuits that, on input of a tuple* $(G, k)$*, outputs a* $p$-MATCHING *kernel with at most* $O(k^2)$ *vertices.*

The circuits of Lemma 3.5 need their threshold gates "only" to count up to $k$. We can thus deduce the following corollary (the proof argument is the same as for Lemma 3.1):

▶ **Corollary 3.6.** *For every* $\delta \in \mathbb{N}$ *there is a* DLOGTIME*-uniform family of* $\mathrm{AC}^0$*-circuits that, on input of a tuple* $(G, k)$*, outputs a* $p$-MATCHING *kernel with at most* $O(2^{\sqrt[\delta]{k}})$ *vertices.*

## 4 Parallel Kernels for the Feedback Vertex Set Problem

The input for $p$-FEEDBACK-VERTEX-SET $= p$-FVS is an undirected multigraph $G = (V, E)$ and an integer $k$, the question is whether it is possible to delete $k$ vertices such that the remaining graph is a forest. The problem is well-known to be fixed-parameter tractable. Concerning the parallel complexity, it is known that membership in FPT can be witnessed by a machine that uses "FPT time and XL space" [16] and the problem was recently shown to lie in para-$\mathrm{NC}^{2+\epsilon} \subseteq$ para-$\mathrm{NC}^3$ [3].

A lot of effort has been put into the design of sequential kernels for this problem, ultimately resulting in a kernel with $O(k^2)$ vertices [8, 7, 33, 22]. Much less is known concerning parallel kernels. Since the $k = 0$ slice of $p$-FVS is exactly the L-complete [13] problem whether a given graph is a forest, we get as a lower bound that no kernel of any size can be computed for $p$-FVS by any circuit class $C$ unless $\mathrm{L} \subseteq C$ and the smallest AC-class for which this is known is $\mathrm{AC}^1$. On the other hand, the mentioned membership in para-$\mathrm{NC}^{2+\epsilon}$ together with Theorem 2.3 yield an $\mathrm{NC}^{2+\epsilon}$ kernel. In summary:

▶ **Lemma 4.1.** *There is a* DLOGTIME*-uniform family of* $\mathrm{NC}^{2+\epsilon}$*-circuits that, on input of a tuple* $(G, k)$*, outputs a* $p$-FVS *kernel with at most* $f(k)$ *vertices. There is no such family of* $\mathrm{AC}^{1-\epsilon}$*-circuits, unless* $\mathrm{L} \subseteq \mathrm{AC}^{1-\epsilon}$*.*

A natural first question arising from this lemma is: Can we improve the bounds? It turns out that we can lower the upper bound from $\mathrm{NC}^{2+\epsilon}$ to $\mathrm{AC}^{1+\epsilon}$ by observing the reduction rules used in sequential kernels for $p$-FVS can, in certain cases, be applied in parallel. In detail, the known sequential kernels for $p$-FVS all repeatedly apply (at least) the below rules, whose correctness is very easily seen. We will show that each of the first three rules can individually be applied exhaustively in $\mathrm{AC}^1$. Based on this, we show $p$-FVS $\in$ para-$\mathrm{AC}^{1\uparrow}$.

**Leaf Rule** Delete a vertex $v$ of degree 1.

**Chain Rule** Contract a vertex $v$ of degree 2 to one of its neighbors.

**Loop Rule** Delete a vertex $v$ with $v \in N(v)$, reduce $k$ by 1.

**Flower Rule** Delete a vertex $v$ that appears in more then $k$ cycles that only share the vertex $v$, reduce $k$ by 1.

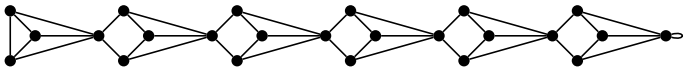


**Figure 1** A graph that is fully reduced by the Chain Rule and the Loop Rule in $k = 6$ rounds. In every round, the Chain Rule can only be applied after the Loop Rule was used exhaustively.

▶ **Lemma 4.2.** *There is a* DLOGTIME*-uniform family of* $\mathrm{AC}^1$*-circuits that, on input of a tuple* $(G, k)$*, outputs a tuple* $(G', k')$ *that results from repeatedly applying (only) the Leaf Rule as long as possible. The same holds for the Chain Rule and for the Loop Rule.*

▶ **Theorem 4.3.** $p$-FVS $\in$ para-$\mathrm{AC}^{1\uparrow}$.

**Sketch of Proof.** The circuit implements a search-tree of depth $O(k)$ that, on any layer, applies the reduction rules of Lemma 4.2 using an $\mathrm{AC}^1$-circuit. This either reduces $k$ immediately by one, or provides a small set of vertices on which the circuit can branch. ◀

▶ **Corollary 4.4.** *There is a* DLOGTIME*-uniform family of* $\mathrm{AC}^{1+\epsilon}$*-circuits that, on input of a tuple* $(G, k)$*, outputs a* $p$-FVS *kernel with at most* $f(k)$ *vertices.*

**Proof.** Follows by Theorem 2.3 and by the fact that para-$\mathrm{AC}^{i\uparrow} \subseteq$ para-$\mathrm{AC}^{i+\epsilon}$ [2]. ◀

We now have rather tight bounds (an upper bound of $\mathrm{AC}^{1+\epsilon}$ and a conditional lower bound of $\mathrm{AC}^{1-\epsilon}$) on how quickly we can compute *some* kernel for $p$-FVS in parallel. However, there is a natural second question arising from Lemma 4.1: Can we also compute a polynomial kernel in parallel?

We claim that progress towards such a kernel cannot solely be based on the presented reduction rules. In the proof of Theorem 4.3 we may need to branch after the exhaustive application of one of the rules Leaf Rule, Chain Rule, or Loop Rule. If we seek to implement a polynomial kernel for $p$-FVS in NC, we have to implement these rules without branching and have to apply the rules exhaustively *together* while they may influence each other. Figure 1 provides an intuition why this interplay is "very sequential," and Theorem 4.5 provides evidence that it is in fact very unlikely that there exists a parallel algorithm that computes the result of jointly applying all rules exhaustively.

▶ **Theorem 4.5.** *The problem of deciding whether a specific vertex of a given graph will be removed by an exhaustive application of the Leaf Rule, the Chain Rule, and the Loop Rule is* P*-hard under* $\mathrm{NC}^1$*-reduction.*

**Sketch of Proof.** We encode the monotone circuit value problem into a graph similar to the one of Figure 1. The reduction rules then simulate the propagation of truth values through the circuit. ◀

▶ **Remark 4.6.** *The proof of Theorem 4.5 shows that the problem remains* P*-hard restricted to the Chain Rule and the Loop Rule, even if they are alternatingly executed exhaustively.*

We close this section with the observation that also the last rule, the Flower Rule, is unlikely to yield a parallel algorithm.

▶ **Theorem 4.7.** *Unless* MATCHING $\in$ NC*, there is no* DLOGTIME*-uniform family of* $\mathrm{NC}^i$*-circuits for any* $i$ *that determines, give a graph* $G = (V, E)$*, an integer* $k$*, and a vertex* $v$*, whether the Flower Rule can be applied to* $v$*.*

## 5 Parallel Kernels for Structural Parameterizations

It is known that NP-hard graph parameters that are closed under taking disjoint union do not allow a polynomial kernel unless NP $\subseteq$ coNP/poly [14]. Famous problems that suffer from this result are the decision versions of tree width, path width, and tree depth, which has led to a growing body of research that considers structural parameters for these problems [6, 5, 24]. A commonly used parameter in this line of research is the vertex cover number of the input graph and in this section we extend the cited results by proving that the corresponding kernels can be computed in small circuit classes.

We use the following standard definitions: A *tree decomposition* of a graph $G = (V, E)$ is a tuple $(T, \iota)$ where $T$ is a tree and $\iota$ a mapping from the nodes of $T$ to subsets of $V$ (which we call *bags*) such that for every $u \in V$ and every $\{v, w\} \in E$ there is (1) a node $n$ with $u \in \iota(n)$, (2) a node $m$ with $\{v, w\} \subseteq \iota(m)$, and (3) the set $\{\, n \mid u \in \iota(n) \,\}$ is connected in $T$. The *width* of a tree decomposition is the maximum size of the bags minus one. For a graph $G$, its *tree width* $\mathrm{tw}(G)$ is the minimum width of all tree decompositions of $G$, its *path width* $\mathrm{pw}(G)$ is the minimum width of all tree decompositions of $G$ that are paths, and its *tree depth* $\mathrm{td}(G)$ is the minimum width of all tree decompositions $(T, \iota)$ of $G$ that can be rooted in such a way that for all $n, m \in V(T)$ we have $\iota(n) \subsetneq \iota(m)$ if $m$ is an descendant of $n$. The following two facts will be useful, where $N(v) = \{\, u \mid \{u, v\} \in E \,\}$ is the neighborhood of $v$, $N[v] = N(v) \cup \{v\}$, and where we call a vertex $v$ *simplicial* if $N(v)$ is a clique:

▶ **Fact 5.1** ([6, 5, 24]). *Let $G = (V, E)$ be a graph with tree width, path width, or tree depth at most $k$ and with $u, v \in V$, $\{u, v\} \notin E$, and $|N(u) \cap N(v)| > k$. Then adding the edge $\{u, v\}$ to $G$ will not increase the tree width, path width, or tree depth of $G$, respectively.*

▶ **Fact 5.2** ([4]). *Let $G = (V, E)$ be a graph and $v \in V$ be a simplicial vertex, then we have $\mathrm{tw}(G) \geq |N(v)|$.*

**Computing a Kernel for Tree Width.** For the problem $p_{\mathrm{vc}}$-TREE-WIDTH we are given a graph $G = (V, E)$, an integer $k$, and a vertex cover $S \subseteq V$ of $G$; the parameter is $|S|$ and the question is whether $\mathrm{tw}(G) \leq k$ holds.

▶ **Theorem 5.3.** *There is a* DLOGTIME*-uniform family of* $\mathrm{TC}^0$*-circuits that, on inputs of a triple $(G, k, S)$, outputs a* $p_{\mathrm{vc}}$-TREE-WIDTH *kernel with at most $O(|S|^3)$ vertices.*

**Sketch of Proof.** We check in parallel for every pair $u, v \in S$ with $\{u, v\} \notin E$ if they have more than $k$ common neighbors in $V \setminus S$ and, if so, add the edge $\{u, v\}$ by Fact 5.1. By Fact 5.2 simplicial vertices in $V \setminus S$ can not have high degree and we may safely remove them by standard arguments. A counting argument then provides the claimed kernel size. ◀

▶ **Corollary 5.4.** *For every $\delta \in \mathbb{N}$ there is a* DLOGTIME*-uniform family of* $\mathrm{AC}^0$*-circuits that, on input of a triple $(G, k, S)$, outputs a* $p_{\mathrm{vc}}$-TREE-WIDTH *kernel with at most $2^{\sqrt[\delta]{|S|}}$ vertices.*

**Computing a Kernel for Path Width.** We define $p_{\mathrm{vc}}$-PATH-WIDTH analogously to $p_{\mathrm{vc}}$-TREE-WIDTH and the aim of this section is to reformulate Theorem 5.3 in terms of path width. The main difference is that we cannot simply delete simplicial vertices as this would, for instance, eliminate trees completely. We can, however, use the following weaker result:

▶ **Fact 5.5** ([5]). *Let $G = (V, E)$ be a graph, $k \in \mathbb{N}$, and $v \in V$ be a simplicial vertex. If the degree $|N(v)|$ of $v$ is 1 and the neighbor of $v$ has another degree-1 neighbor, or if we have $2 \leq |N(v)| \leq k$ and for each pair $x, y \in N(v)$ there is a simplicial vertex $w \in N(x) \cap N(y)$ with $w \notin N[v]$, then $\mathrm{pw}(G) \leq k$ if, and only if, $\mathrm{pw}(G[V \setminus \{v\}]) \leq k$.*

▶ **Theorem 5.6.** *There is a* DLOGTIME*-uniform family of* $\mathrm{TC}^0$*-circuits that, on input of a triple* $(G, k, S)$*, outputs a* $p_{\mathrm{vc}}$-PATH-WIDTH *kernel with at most* $O(|S|^3)$ *vertices.*

**Sketch of Proof.** In difference to the proof of Theorem 5.3, we must now identify the vertices for which Fact 5.5 applies in constant time. This is not trivial because of dependencies between them, but a circuit can use a two-stage marking process to find them. ◀

▶ **Corollary 5.7.** *For every* $\delta \in \mathbb{N}$ *there is a* DLOGTIME*-uniform family of* $\mathrm{AC}^0$*-circuits that, on input of a triple* $(G, k, S)$*, outputs a* $p_{\mathrm{vc}}$-PATH-WIDTH *kernel with at most* $2\sqrt[\delta]{|S|}$ *vertices.*

**Computing a Kernel for Tree Depth.** The last problem we consider is tree depth, and, as for path width, we prove a version of Theorem 5.3 for it. The main problem is once more that we cannot simply remove simplicial vertices. However, by the following fact of Kobayashi and Tamaki there are still enough simplicial vertices that are safe to remove:

▶ **Fact 5.8** ([24])**.** *Let* $G = (V, E)$ *be a graph,* $k \in \mathbb{N}$*, and let* $v \in V$ *be a simplicial vertex with* $1 \leq |N(v)| \leq k$*. If every neighbor of* $v$ *has degree at least* $k + 1$*, then we have* $\mathrm{td}(G) \leq k$ *if, and only if,* $\mathrm{td}(G[V \setminus \{v\}]) \leq k$*.*

▶ **Theorem 5.9.** *There is a* DLOGTIME*-uniform family of* $\mathrm{TC}^0$*-circuits that, on input of a triple* $(G, k, S)$*, outputs a* $p_{\mathrm{vc}}$-TREE-DEPTH *kernel with at most* $O(|S|^3)$ *vertices.*

**Sketch of Proof.** Similar to the proofs of the Theorems 5.3 and 5.6, we identify vertices for which Fact 5.8 holds in parallel constant time. This time, we mark for every vertex $v \in S$ with $|N(v)| > k$ the $k + 1$ lexicographically smallest neighbors of $v$, then the circuit marks every simplicial vertex $v \in V \setminus S$ that has at least one neighbor of degree less than $k$. ◀

▶ **Corollary 5.10.** *For every* $\delta \in \mathbb{N}$ *there is a* DLOGTIME*-uniform family of* $\mathrm{AC}^0$*-circuits that, on inputs of a triple* $(G, k, S)$*, outputs a* $p_{\mathrm{vc}}$-TREE-DEPTH *kernel with at most* $2\sqrt[\delta]{|S|}$ *vertices.*

## 6 A Parallel Kernel for Point Line Cover

In this section we study a natural, well-known problem for which we can prove (unconditionally) that we *cannot* compute a kernel using $\mathrm{AC}^0$-circuits while we *can* compute polynomially-sized kernels in $\mathrm{TC}^0$. In the $p$-POINT-LINE-COVER problem we are given distinct points $p_1, \ldots, p_n \in \mathbb{Z}^d$ for some dimension $d \geq 2$ and a natural number $k \in \mathbb{N}$, the question is whether we can cover all points by at most $k$ lines. This problem is NP-hard in general (even for $d = 2$) and in FPT parameterized by $k$ [26]. There is a simple $k^2$ kernel, which is essentially optimal [26]: If any line covers at least $k + 1$ points, remove all points on this line and reduce $k$ by one. This is safe since we would require at least $k + 1$ different lines if we would not use this line. Because no set of $k + 1$ points lies on the same line after the reduction, we have at most $k^2$ points left or we deal with a no-instance.

▶ **Lemma 6.1.** *There is a* DLOGTIME*-uniform family of* $\mathrm{TC}^0$*-circuits that, on input of a dimension* $d$*, a set of distinct points* $p_1, \ldots, p_n \in \mathbb{Z}^d$*, and an integer* $k$*, outputs a* $p$-POINT-LINE-COVER *kernel with at most* $k^2$ *points.*

The lemma shows that the optimal kernel for $p$-POINT-LINE-COVER can be computed in $\mathrm{TC}^0$ and it is natural to ask if we can do the same using a $\mathrm{AC}^0$-circuit or, failing that, to at least compute *some* kernel using a $\mathrm{AC}^0$-circuit (as we could for the problems in the previous sections). We answer this question in the negative, settling the complexity of the problem to para-$\mathrm{TC}^0$:

▶ **Lemma 6.2.** *For every fixed $k$, the $k$th slice of $p$-POINT-LINE-COVER is $\mathrm{TC}^0$-complete under $\mathrm{AC}^0$-reduction.*

▶ **Corollary 6.3.** *$p$-POINT-LINE-COVER is para-$\mathrm{TC}^0$-complete under $\mathrm{AC}^0$-reduction.*

Now assume there would be a uniform family of $\mathrm{AC}^0$-circuits computing a kernel of arbitrary size for $p$-POINT-LINE-COVER. Then by Theorem 2.3 the problem is in para-$\mathrm{AC}^0$, which on the other hand implies that for every fixed $k$ the problem must be in $\mathrm{AC}^0$. This contradicts Lemma 6.2 as it is known that $\mathrm{AC}^0 \subsetneq \mathrm{TC}^0$ [19]. Therefore, no family of $\mathrm{AC}^0$-circuits can compute such a kernel.

## 7 Problems for Which Computing Kernels is Inherently Sequential

As surprisingly many problems have NC-computable, in fact often even $\mathrm{AC}^0$-computable, kernelizations, we may ask which problems do not have this property. We would like to find problems for which the computation of any kernel is P-complete or, equivalently, which are FPT-complete under $\mathrm{AC}^0$- or $\mathrm{NC}^1$-reductions. While it is easy to find artificial problems with this property – such as any P-complete problem (like CVP) with the trivial parametrization ($\kappa(x) \equiv 1$) –, no *natural* problems that are FPT-complete for sensible parametrizations can be found in the literature. We remedy this situation in the following; but must caution the reader that in all our results the hardness of the parameterized problem for FPT stems from the fact that some slice of the problem is (essentially) a known P-complete problem. Unfortunately, it is known [18] that this "cannot be helped" since all FPT-complete problems have this property. Our main contribution here lies, thus, in the assembly of a diverse body of relevant, non-trivial FPT-problems that will serve as starting points for further studies of the limits of parameterized parallelization.

**Strong Backdoors to Satisfiability.** A *strong backdoor set* of a propositional formula $\phi$ is a set of variables such that under any assignment of these variables the resulting formula $\phi'$ belongs to a certain class of formulas [20]. In the $p$-STRONG-BACKDOOR-{HORN,2CNF}-SAT problems, we are given a formula $\phi$ and an integer $k$, the question is whether $\phi$ is satisfiable and has a strong backdoor set of size $k$ to Horn- or 2CNF-formulas, respectively. Solving such problems is usually done in two phases: first *detect* the backdoor set and, second, *solve* the satisfiability problem of the formula for every assignment of the backdoor set. While the first part might seem harder in general, it is not from a parameterized point of view: (1) A strong backdoor set to Horn formulas is exactly a vertex cover of size $k$ in the positive primal graph of $\phi$, that is, the graph that has a vertex for each variable and an edge between any two variables appearing together positively in a clause; (2) strong backdoor sets to 2CNF-formulas are exactly the hitting sets of the hypergraph that has the variables of $\phi$ as vertices and that connects three vertices by a hyperedge if they appear together in a clause. Since $p$-VERTEX-COVER $\in$ para-$\mathrm{AC}^0$ and also $p$-3-HITTING-SET $\in$ para-$\mathrm{AC}^0$ [2, 12], we can conclude:

▶ **Corollary 7.1.** *There is a DLOGTIME-uniform family of para-$\mathrm{AC}^0$-circuits that, on input of a propositional formula $\phi$ and an integer $k$, either outputs a size-$k$ strong backdoor set to $\{Horn, 2\mathrm{CNF}\}$-formulas, or concludes that no such set exists.*

The second step of solving $p$-STRONG-BACKDOOR-{HORN,2CNF}-SAT is to solve the satisfiability problem for $\phi$ on every assignment to the variables of the backdoor set. While we can nicely handle all assigments in parallel, checking if the formulas are satisfiable in parallel is difficult. Indeed, it is known that, under $\mathrm{AC}^0$-reductions, the satisfiability problem is NL-complete for 2CNF-formulas, and is even P-complete for Horn formulas [1].

▶ **Corollary 7.2.** *$p$-STRONG-BACKDOOR-2CNF-SAT is para-NL-complete under $\mathrm{AC}^0$-reduction.*

▶ **Corollary 7.3.** *$p$-STRONG-BACKDOOR-HORN-SAT is FPT-complete under $\mathrm{AC}^0$-reduction.*

The last corollary implies that there is no parallel polylogarithmic time kernelization for $p$-STRONG-BACKDOOR-HORN-SAT that produces a kernel of any size, unless NC = P.

**Mixed Integer Linear Programming.** The FPT-complete problem above is an intermediate problem between a P-complete problem (HORN-SAT) and a NP-complete problem (SAT); the transition between the problems is caused by the backdoor variables. A similar intermediate problem is known for LINEAR-PROGRAMMING (another classical P-complete problem) and its integer variant (which is NP-complete). The intermediate version of these problems is called $p$-MIXED-INTEGER-PROGRAMMING, which asks, given a matrix $A \in \mathbb{Z}^{n\times n}$, vectors $b \in \mathbb{Z}^n$, $c \in \mathbb{Z}^n$, and integers $k$ and $w$, if there is a vector $x \in \mathbb{R}^n$ such that $Ax \leq b$, $c^{\mathrm{T}}x \geq w$, and such that $x[i] \in \mathbb{Z}$ for $0 \leq i < k$. A celebrated result by Lenstra states that an instance $I$ of this problem can be solved in time $2^{O(k^3)} \cdot |I|^c$ for a suitable constant $c$, that is, the problem is in FPT. Therefore, every slice of the problem is in P and, as LINEAR-PROGRAMMING trivially reduced to it, we get that $k$-MIXED-INTEGER-PROGRAMMING is P-complete for every $k$ (under $\mathrm{NC}^1$-reductions [34]).

▶ **Corollary 7.4.** *$p$-MIXED-INTEGER-PROGRAMMING is FPT-complete under $\mathrm{NC}^1$-reductions.*

**Maximum Flow with Minimum Quantities.** The last problem we review in this section is the maximum flow problem with minimum quantities: Inputs are directed graphs $G = (V, E)$ with $s, t \in V$, two weight functions $u, l\colon E \to \mathbb{N}$, an integer $w \in \mathbb{N}$, and a set of edges $B \subseteq E$; the question is whether there is a set $A \subseteq B$ such that in $G' = (V, E \setminus A)$ there is a valid $s$-$t$-flow $f$ of value at least $w$ that fulfills the flow conservation constraints and $l(e) \leq f(e) \leq u(e)$ for all $e \in E \setminus A$. For $B = \emptyset$ the problem boils down to classical maximum flow with lower bounds on the edges, which can be solved in polynomial time [23] and which is known to be P-hard under $\mathrm{NC}^1$-reduction [34]. On the other hand, for $B = E$ the problem becomes NP-complete even on serial-parallel graphs [21] and it is also NP-hard to approximate the problem within any positive factor [32]. The intermediate problem between this two cases is the parameterized problem $p$-MAX-FLOW-QUANTITIES, defined as above, where the cardinality of $B$ is the parameter.

▶ **Lemma 7.5.** *$p$-MAX-FLOW-QUANTITIES is FPT-complete under $\mathrm{NC}^1$-reduction.*

## 8 Conclusion and Outlook

Kernelization is a fundamental concept of parameterized complexity and we have studied its parallelization. Since traditional descriptions of kernelization algorithms are inherently sequential, we found it surprising how many parameterized problems lie in para-$\mathrm{AC}^0$ – the smallest robust class in parallel parameterized complexity theory. We found, furthermore, that for many problems the equation "smaller circuit class = larger kernel" holds, see Table 1 for a summary of our results.

Apart from classifying more parameterized problems in the spirit of this paper, namely according to how well small kernels can be computed by small circuits, an interesting open problem is to improve any of the $\mathrm{AC}^0$-kernelizations presented in the present paper so that they produce a *polynomially sized* kernel (which we, at best, can currently do only in $\mathrm{TC}^0$). Perhaps even more challenging seems to be the design of a framework for proving that polynomially sized kernels for these problems can*not* be computed in $\mathrm{AC}^0$.

## References


1. E. Allender, M. Bauland, N. Immerman, H. Schnoor, and H. Vollmer. The complexity of satisfiability problems: Refining Schaefer's theorem. *J. Comput. Syst. Sci.*, 75(4):245–254, 2009. doi:10.1016/j.jcss.2008.11.001.
2. M. Bannach, C. Stockhusen, and T. Tantau. Fast Parallel Fixed-parameter Algorithms via Color Coding. In *Proceedings of IPEC 2015*, pages 224–235, 2015. doi:10.4230/LIPIcs.IPEC.2015.224.
3. M. Bannach and T. Tantau. Parallel Multivariate Meta-Theorems. In *Proceedings of IPEC 2016*, pages 4:1–4:17, 2016. doi:10.4230/LIPIcs.IPEC.2016.4.
4. H. L. Bodlaender. A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996. doi:10.1137/S0097539793251219.
5. H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Kernel Bounds for Structural Parameterizations of Pathwidth. In *Proceedings of SWAT 2012*, pages 352–363, 2012. doi:10.1007/978-3-642-31155-0_31.
6. H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Preprocessing for Treewidth: A Combinatorial Analysis through Kernelization. *SIAM J. Discrete Math.*, 27(4):2108–2142, 2013. doi:10.1137/120903518.
7. H. L. Bodlaender and T. C. van Dijk. A Cubic Kernel for Feedback Vertex Set and Loop Cutset. *Theory Comput. Syst.*, 46(3):566–597, 2010. doi:10.1007/s00224-009-9234-2.
8. Kevin Burrage, Vladimir Estivill-Castro, Michael R. Fellows, Michael A. Langston, Shev Mac, and Frances A. Rosamond. The Undirected Feedback Vertex Set Problem Has a Poly($k$) Kernel. In *Proceedings of IWPEC 2006*, pages 192–202, 2006. doi:10.1007/11847250_18.
9. L. Cai, J. Chen, R. G. Downey, and M. R. Fellows. Advice Classes of Parameterized Tractability. *Ann. Pure Appl. Logic*, 84(1):119–138, 1997. doi:10.1016/S0168-0072(95)00020-8.
10. J. Chen, I. A. Kanj, and W. Jia. Vertex Cover: Further Observations and Further Improvements. *J. Algorithms*, 41(2):280–301, 2001. doi:10.1006/jagm.2001.1186.
11. Y. Chen and J. Flum. Some Lower Bounds in Parameterized $AC^0$. In *Proceedings of MFCS 2016*, pages 27:1–27:14, 2016. doi:10.4230/LIPIcs.MFCS.2016.27.
12. Yijia Chen, Jörg Flum, and Xuangui Huang. Slicewise Definability in First-Order Logic with Bounded Quantifier Rank. In Valentin Goranko and Mads Dam, editors, *Proceedings of CSL 2017*, volume 82, pages 19:1–19:16, 2017. doi:10.4230/LIPIcs.CSL.2017.19.
13. Stephen A. Cook and Pierre McKenzie. Problems Complete for Deterministic Logarithmic Space. *J. Algorithms*, 8(3):385–394, 1987. doi:10.1016/0196-6774(87)90018-6.
14. M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer Berlin Heidelberg, 2015.
15. R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.
16. M. Elberfeld, C. Stockhusen, and T. Tantau. On the Space and Circuit Complexity of Parameterized Problems: Classes and Completeness. *Algorithmica*, 71(3):661–701, 2015. doi:10.1007/s00453-014-9944-y.
17. S. A. Fenner, R. Gurjar, and T. Thierauf. Bipartite perfect matching is in quasi-NC. In *Proceedings of STOC 2016*, pages 754–763, 2016. doi:10.1145/2897518.2897564.
18. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. doi:10.1007/3-540-29953-X.
19. M. L. Furst, J. B. Saxe, and M. Sipser. Parity, Circuits, and the Polynomial-Time Hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984. doi:10.1007/BF01744431.

20 S. Gaspers and S. Szeider. Backdoors to Satisfaction. In H.L. Bodlaender, R. Downey, F. V. Fomin, and D. Marx, editors, *The Multivariate Algorithmic Revolution and Beyond*, pages 287–317. Springer, 2012. doi:10.1007/978-3-642-30891-8_15.

21 D. Haugland, M. Eleyat, and M. Lie Hetland. The Maximum Flow Problem with Minimum Lot Sizes. In *Proceedings of ICCL 2011*, pages 170–182, 2011. doi:10.1007/978-3-642-24264-9_13.

22 Yoichi Iwata. Linear-Time Kernelization for Feedback Vertex Set. In *Proceedings of ICALP 2017*, pages 68:1–68:14, 2017. doi:10.4230/LIPIcs.ICALP.2017.68.

23 Jon M. Kleinberg and Éva Tardos. *Algorithm design*. Addison-Wesley, 2006.

24 Y. Kobayashi and H. Tamaki. Treedepth Parameterized by Vertex Cover Number. In *Proceedings of IPEC 2016*, pages 18:1–18:11, 2016. doi:10.4230/LIPIcs.IPEC.2016.18.

25 D. König. Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. *Mathematische Annalen*, 77(4):453–465, 1916. doi:10.1007/BF01456961.

26 S. Kratsch, G. Philip, and S. Ray. Point Line Cover: The Easy Kernel is Essentially Tight. *ACM Trans. Algorithms*, 12(3):40:1–40:16, 2016. doi:10.1145/2832912.

27 M. Lampis. A kernel of order $2k - c \log k$ for vertex cover. *Inf. Process. Lett.*, 111(23-24):1089–1091, 2011. doi:10.1016/j.ipl.2011.09.003.

28 K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987. doi:10.1007/BF02579206.

29 G. L. Nemhauser and L. E. Trotter Jr. Properties of vertex packing and independence system polyhedra. *Math. Program.*, 6(1):48–61, 1974. doi:10.1007/BF01580222.

30 I. Newman, P. Ragde, and A. Wigderson. Perfect Hashing, Graph Entropy, and Circuit Complexity. In *Proceedings of STC 1990*, pages 91–99. IEEE Computer Society, Los Alamitos, California, 1990. doi:10.1109/SCT.1990.113958.

31 A. Soleimanfallah and A. Yeo. A kernel of order $2k - c$ for Vertex Cover. *Discrete Mathematics*, 311(10-11):892–895, 2011. doi:10.1016/j.disc.2011.02.014.

32 Clemens Thielen and Stephan Westphal. Complexity and approximability of the maximum flow problem with minimum quantities. *Networks*, 62(2):125–131, 2013. doi:10.1002/net.21502.

33 Stéphan Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Trans. Algorithms*, 6(2):32:1–32:8, 2010. doi:10.1145/1721837.1721848.

34 Jacobo Toran. P-completeness. In Alan Gibbons and Paul Spirakis, editors, *Lectures on parallel computation*, pages 177–196. Cambridge University Press, 1993.

# Exploring the Kernelization Borders for Hitting Cycles

**Akanksha Agrawal**
Institute of Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary
akanksha@sztaki.mta.hu

**Pallavi Jain**
Institute of Mathematical Sciences, HBNI, Chennai, India
pallavij@imsc.res.in

**Lawqueen Kanesh**
Institute of Mathematical Sciences, HBNI, Chennai, India
lawqueen@imsc.res.in

**Pranabendu Misra**
University of Bergen, Bergen, Norway
Pranabendu.Misra@uib.no

**Saket Saurabh**
Institute of Mathematical Sciences, HBNI, Chennai, India
saket@imsc.res.in

## Abstract

A generalization of classical cycle hitting problems, called conflict version of the problem, is defined as follows. An input is undirected graphs $G$ and $H$ on the same vertex set, and a positive integer $k$, and the objective is to decide whether there exists a vertex subset $X \subseteq V(G)$ such that it intersects all desired "cycles" (all cycles or all odd cycles or all even cycles) and $X$ is an independent set in $H$. In this paper we study the conflict version of classical Feedback Vertex Set, and Odd Cycle Transversal problems, from the view point of kernelization complexity. In particular, we obtain the following results, when the conflict graph $H$ belongs to the family of $d$-degenerate graphs.

1. CF-FVS admits a $\mathcal{O}(k^{\mathcal{O}(d)})$ kernel.
2. CF-OCT does not admit polynomial kernel (even when $H$ is 1-degenerate), unless $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$.

For our kernelization algorithm we exploit ideas developed for designing polynomial kernels for the classical Feedback Vertex Set problem, as well as, devise new reduction rules that exploit degeneracy crucially. Our main conceptual contribution here is the notion of "$k$-independence preserver". Informally, it is a set of "important" vertices for a given subset $X \subseteq V(H)$, that is enough to capture the independent set property in $H$. We show that for $d$-degenerate graph independence preserver of size $k^{\mathcal{O}(d)}$ exists, and can be used in designing polynomial kernel.

**2012 ACM Subject Classification** Theory of computation → Parameterized complexity and exact algorithms

**Keywords and phrases** Parameterized Complexity, Kernelization, Conflict-free problems, Feedback Vertex Set, Even Cycle Transversal, Odd Cycle Transversal

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.14

**Funding** This research has received funding from the European Research Council under ERC grant no. 306992 PARAPPROX, ERC grant no. 715744 PaPaALG and ERC grant no. 725978 SYSTEMATIC-GRAPH, and DST, India for SERB-NPDF fellowship [PDF/2016/003508].

© A. Agrawal, and P. Jain, and L. Kanesh, and P. Misra, and S. Saurabh;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 14; pp. 14:1–14:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Reducing the input data, in polynomial time, without altering the answer is one of the popular ways in dealing with intractable problems in practice. While such polynomial time heuristics can not solve NP-hard problems exactly, they work well on input instances arising in real-life. It is a challenging task to assess the effectiveness of such heuristics theoretically. Parameterized complexity, via kernelization, provides a natural way to quantify the performance of such algorithms. In parameterized complexity each problem instance comes with a parameter $k$ and the parameterized problem is said to admit a *polynomial kernel* if there is a polynomial time algorithm, called a *kernelization* algorithm, that reduces the input instance down to an instance with size bounded by a polynomial $p(k)$ in $k$, while preserving the answer. The reduced instance is called a $p(k)$ kernel for the problem.

The quest for designing polynomial kernels for "hitting cycles" in undirected graphs has played significant role in advancing the field of polynomial time pre-processing – kernelization. Hitting all cycles, odd cycles and even cycles correspond to well studied problems of Feedback Vertex Set (FVS), Odd Cycle Transversal (OCT) and Even Cycle Transversal (ECT), respectively. Alternatively, FVS, OCT and ECT correspond to deleting vertices such that the resulting graph is a forest, a bipartite graph and an odd cactus graph, respectively. All these problems, FVS, OCT, and ECT, have been extensively studied in parameterized algorithms and kernelization. The earliest known FPT algorithms for FVS go back to the late 80's and the early 90's [4, 11] and used the seminal Graph Minor Theory of Robertson and Seymour. On the other hand the parameterized complexity of OCT was open for long time. Only, in 2003, Reed et al. [24] gave a $3^k n^{\mathcal{O}(1)}$ time algorithm for OCT. This is also the paper which introduced the *method of iterative compression* to the field of parameterized complexity. However, the existence of polynomial kernel, for FVS and OCT were open questions for long time. For FVS, Burrage et al. [7] resolved the question in the affirmative by designing a kernel of size $\mathcal{O}(k^{11})$. Later, Bodlaender [5] reduced the kernel size to $\mathcal{O}(k^3)$, and finally Thomassé [25] designed a kernel of size $\mathcal{O}(k^2)$. The kernel of Thomassé [25] is best possible under a well known complexity theory hypothesis. It is important to emphasize that [25] popularized the method of *expansion lemma*, one of the most prominent approach in designing polynomial kernels. While, the kernelization complexity of FVS was settled in 2006, it took another 6 years and a completely new methodology to design polynomial kernel for OCT. Kratsch and Wahlström [16] resolved the question of existence of polynomial kernel for OCT by designing a randomized kernel of size $\mathcal{O}(k^{4.5})$ using matroid theory.[1] As a counterpart to OCT, Misra et al. [20] studied ECT and designed an $\mathcal{O}(k^3)$ kernel.

Fruitful and productive research on FVS and OCT have led to the study of several variants and generalizations of FVS and OCT. Some of these admit polynomial kernels and for some one can show that none can exist, unless some unlikely collapse happens in complexity theory. In this paper we study the following generalization of FVS, and OCT, from the view-point of kernelization complexity.

| Conflict Free Feedback Vertex Set (CF-FVS) | **Parameter:** $k$ |
|---|---|

**Input:** An undirected graph $G$, a conflict graph $H$ on vertex set $V(G)$ and a non-negative integer $k$.
**Question:** Does there exist $S \subseteq V(G)$, such that $|S| \leq k$, $G - S$ is a forest and $H[S]$ is edgeless?

[1] This foundational paper has been awarded the Nerode Prize for 2018.

One can similarly define Conflict Free Odd Cycle Transversal (CF-OCT).

**Motivation.** On the outset, a natural thought is "*why does one care*" about such an esoteric (or obscure) problem. We thought *exactly the same* in the beginning, till we realized the modeling power the problem provides and the rich set of questions one can ask. In the course of this paragraph we will try to explain this. First observe that, if one wants to model "independent" version of these problems (where the solution is suppose to be an independent set), then one takes conflict graph to be same as the input graph. An astute reader will figure out that the problem as stated above is W[1]-hard – a simple reduction from Multicolor Independent Set with each color class being modeled as cycle and the conflict graph being the input graph. Thus, a natural question is: *when does the problem become* FPT? To state the question formally, let $\mathcal{F}$ and $\mathcal{G}$ be two families of graphs. Then, $(\mathcal{G}, \mathcal{F})$-CF-FVS is same problem as CF-FVS, but the input graph $G$ and the conflict graph $H$ are restricted to belong to $\mathcal{G}$ and $\mathcal{H}$, respectively. It immediately brings several questions: (a) for which pairs of families the problem is FPT; (b) can we obtain some kind of dichotomy results; and (c) what could we say about the kernelization complexity of the problem. We believe that answering these questions for basic problems such as FVS, OCT, and Dominating Set will extend both the tractability as well as intractability tools in parameterized complexity and led to some fruitful and rewarding research. It is worth to note that initially we were inspired to define these problems by similar problems in computational geometry. See related results for more on this.

**Our Results and Methods.** A graph $G$ is called *$d$-degenerate* if every subgraph of $G$ has a vertex of degree at most $d$. For a fixed positive integer $d$, let $\mathcal{D}_d$ denote the set of graphs of *degeneracy* at most $d$. In this paper we study the $(\star, \mathcal{D}_d)$-CF-FVS ($\mathcal{D}_d$-CF-FVS) problem. The symbol $\star$ denotes that the input graph $G$ is arbitrary. One can similarly define $\mathcal{D}_d$-CF-OCT. In fact, we study, CF-OCT for a very restricted family of conflict graphs, a family of disjoint union of paths of length at most three and at most two star graphs. We denote this family as $\mathcal{P}^{\star\star}_{\leq 3}$ and this variant of CF-OCT as $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT. Starting point of our research is the recent study of Jain et al. [14], who studied conflict-free graph modification problems in the realm of parameterized complexity. As a part of their study they gave FPT algorithms for $\mathcal{D}_d$-CF-FVS, $\mathcal{D}_d$-CF-OCT and $\mathcal{D}_d$-CF-ECT using the independence covering families [17]. Their results also imply similar FPT algorithm when the conflict graph belongs to nowhere dense graphs. In this paper we focus on the kernelization complexity of $\mathcal{D}_d$-CF-FVS, and $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT obtain the following results.

1. $\mathcal{D}_d$-CF-FVS admits a $\mathcal{O}(k^{\mathcal{O}(d)})$ kernel.
2. $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT does not admit polynomial kernel, unless $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$.

Note that $\mathcal{D}_0$ denotes edgeless graphs and hence $\mathcal{D}_0$-CF-FVS, and $\mathcal{D}_0$-CF-OCT are essentially FVS, and OCT, respectively. Thus, any polynomial kernel for $\mathcal{D}_d$-CF-FVS, and $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT, must generalize the known kernels for these problems. We remark that the above result imply that CF-FVS admits polynomial kernels, when the conflict graph belong to several well studied graph families, such as planar graphs, graphs of bounded degree, graphs of bounded treewidth, graphs excluding some fixed graph as a minor, a topological minor and graphs of bounded expansion etc. (all these graphs classes have bounded degeneracy).

**Strategy for CF-FVS.** Our kernelization algorithm for CF-FVS consists of the following two steps. The first step of our kernelization algorithm is a structural decomposition of the

input graph $G$. This does not depend on the conflict graph $H$. In this phase of the algorithm, given an instance $(G, H, k)$ of CF-FVS we obtain an equivalent instance $(G', H', k')$ of CF-FVS such that:

- The minimum degree of $G'$ is at least 2.
- The number of vertices of degree at least 3 in $G'$ is upper bounded by $\mathcal{O}(k^3)$. Let $V_{\geq 3}$ denote the set of vertices of degree at least 3 in $G'$.
- The number of maximal degree 2 paths in $G'$ is upper bounded by $\mathcal{O}(k^3)$. That is, $G' - V_{\geq 3}$ consists of $\mathcal{O}(k^3)$ connected components where each component is a path.

We obtain this structural decomposition using reduction rules inspired by the quadratic kernel for FVS [25]. As stated earlier, this step can be performed for any graph $H$. Thus the problem reduces to designing reduction rules that bound the number of vertices of degree 2 in the reduced graph. Note that we can not do this for any arbitrary graph $H$ as the problem is W[1]-hard. Once the decomposition is obtained we can not use the known *reduction rules* for FVS. This is for a simple reason that in $G'$ the only vertices that are not bounded have degree exactly 2 in $G'$. On the other hand for FVS we can do simple "short-circuit" of degree 2 vertices (remove the vertex and add an edge between its two neighbors) and assume that there is no vertices of degree two in the graph. So our actual contributions start here.

The second step of our kernelization algorithm bounds the degree two vertices in the graph $G'$. Here we must use the properties of the graph $H$. We propose new reduction rules for bounding degree two vertices, when $H$ belongs to the family of $d$-degenerate graphs. Towards this we use the notion of $d$-degeneracy sequence, which is an ordering of the vertices in $H$ such that any vertex can have at most $d$ forward neighbors. This is used in designing a marking scheme for the degree two vertices. Broadly speaking our marking scheme associates a set with every vertex $v$. Here, set consists of " paths and cycles of $G'$ on which the forward neighbors of $v$ are". Two vertices are called similar if their associated sets are same. We show that if some vertex is not marked then we can safely contract this vertex to one of its neighbors. We then upper bound the degree two vertices by $\mathcal{O}(k^{\mathcal{O}(d)} d^{\mathcal{O}(d)})$, and thus obtain a kernel of this size for $\mathcal{D}_d$-CF-FVS.

At the heart of our kernelization algorithm is a combinatorial tool of "$k$-independence preserver". Informally, it is a set of "important" vertices for a given subset $X \subseteq V(H)$, that is enough to capture the independent set property in $H$. We show that for $d$-degenerate graph independence preserver of size $k^{\mathcal{O}(d)}$ exists, and can be used in designing polynomial kernel. This is our main conceptual contribution.

**Strategy for CF-OCT.** The kernelization lower bound is obtained by the method of cross-composition [6]. We first define a conflict version of the $s$-$t$-Cut problem, where $H$ belongs to $\mathcal{P}^{\star\star}_{\leq 3}$. Then, we show that the problem is NP-hard and cross composes to itself. Finally, we give a parameter preserving reduction from the problem to $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT, and obtain the desired kernel lower bound.

**Related Work.** In the past, the conflict free versions of some classical problems have been studied, e.g. for Shortest Path [15], Maximum Flow [21, 22], Knapsack [23], Bin Packing [12], Scheduling [13], Maximum Matching and Minimum Weight Spanning Tree [10, 9]. It is interesting to note that some of these problems are NP-hard even when their non-conflicting version is polynomial time solvable. The study of conflict free problems has also been recently initiated in computational geometry motivated by various applications (see [1, 2, 3]).

## 2 Preliminaries

Throughout the paper, we follow the following notions. Let $G$ be a graph, $V(G)$ and $E(G)$ denote the vertex set and the edge set of graph $G$, respectively. Let $n$ and $m$ denote the number of vertices and the number of edges of $G$, respectively. Let $G$ be a graph and $X \subseteq V(G)$, then $G[X]$ is the graph induced on $X$ and $G - X$ is graph $G$ induced on $V(G) \setminus X$. Let $\Delta$ denotes the maximum degree of graph $G$. We use $N_G(v)$ to denote the neighborhood of $v$ in $G$ and $N_G[v]$ to denote $N_G(v) \cup \{v\}$. Let $E'$ be subset of edges of graph $G$, by $G[E']$ we mean the graph with the vertex set $V(G)$ and the edge set $E'$. Let $X \subseteq E(G)$, then $G - X$ is a graph with the vertex set $V(G)$ and the edge set $E(G) \setminus X$. Let $Y$ be a set of edges on vertex set $V(G)$, then $G \cup Y$ is graph with the vertex set $V(G)$ and the edge set $E(G) \cup Y$. Degree of a vertex $v$ in graph $G$ is denoted by $deg_G(v)$. For an integer $\ell$, we denote the set $\{1, 2, \ldots, \ell\}$ by $[\ell]$. A *path* $P = \{v_1, \ldots, v_n\}$ is an ordered collection of vertices such that there is an edge between every consecutive vertices in $P$ and $v_1, v_n$ are *endpoints* of $P$. For a path $P$ by $V(P)$ we denote set of vertices in $P$ and by $E(P)$ we denote set of edges in $P$. A *cycle* $C = \{v_1, \ldots, v_n\}$ is a path with an edge $v_1v_n$. We define a *maximal degree two induced path in* $G$ as an induced path of maximal length such that all vertices in path are of *degree exactly two in* $G$. An *isolated cycle* in graph $G$ is defined as an induced cycle whose all the vertices are of degree exactly two in $G$. Let $G'$ and $G$ be graphs, $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$, then we say that $G'$ is a *subgraph* of G. The subscript in the notations will be omitted if it is clear from the context.

A graph $G$ has *degeneracy* $d$ if every subgraph of $G$ has a vertex of degree at most $d$. An ordering of vertices $\sigma : V(G) \to \{1, \cdots, n\}$ is is called a *$d$-degeneracy sequence* of graph $G$, if every vertex $v$ has at most $d$ neighbors $u$ with $\sigma(u) > \sigma(v)$. A graph $G$ is $d$-degenerate if and only if it has a $d$-degeneracy sequence. For a vertex $v$ in $d$-degenerate graph $G$, the neighbors of $v$ which comes *after (before)* $v$ in $d$-degeneracy sequence are called *forward (backward) neighbors* of $v$ in the graph $G$. Given a $d$-degenerate graph, we can find $d$-degeneracy sequence in linear time [18].

## 3 A Tool for Our Kernelization Algorithm

In this section, we give a tool, which we believe might be useful in obtaining kernelization algorithm for "conflict free" versions of various parameterized problems (admitting kernels), when the conflict graph belongs to the family of $d$-degenerate graphs. We particularly use this tool to obtain kernel for $\mathcal{D}_d$-CF-FVS (Section 4). For a parameterized problem $\Pi$, consider an instance $(G, H, k)$ of its conflict free variant, Conflict Free $\Pi$. Then in the kernelization step where we want to bound the number of vertices, it is seemingly useful to be able to obtain a set of "important" vertices for a given subset $X \subseteq V(H)$ that will be enough to capture the independent set property in $H$. The above intuition becomes clear when we describe the kernelization algorithm for $\mathcal{D}_d$-CF-FVS.

To formalize the notion of "important" set of vertices, we give the following definition.

▶ **Definition 1.** For a $d$-degenerate graph $H$ and a set $X \subseteq V(H)$, a *$k$-independence preserver* for $(H, X)$ is a set $X' \subseteq X$, such that for any independent set $S$ in $H$ of size at most $k$, if there is $v \in (S \cap X) \setminus X'$, then there is $v' \in X' \setminus S$, such that $(S \setminus \{v\}) \cup \{v'\}$ is an independent set in $H$.

Throughout this section, we work with a (fixed) $d$, which is the degeneracy of the input graph. The goal of this section will be to obtain an algorithm for computing a $k$-independence

preserver for $(H, X)$ of "small" size. To quantify the "small" size, we need the following definition.

▶ **Definition 2.** For each $q \in [d]$, we define an integer $n_q$ as follows.
1. If $q = 1$, then $n_q = kd + k + 1$, and
2. $n_q = kn_{q-1} + kd + k + 1$, otherwise.

Next, we formally define the problem for which we want to design a polynomial time algorithm. We call this problem $d$-Bounded Independence Preserver ($d$-BIP, for short).

| |
|---|
| $d$-Bounded Independence Preserver ($d$-BIP)<br>**Input:** A $d$-degenerate graph $H$, a set $X \subseteq V(H)$, and an integer $k$.<br>**Output:** A set $X' \subseteq X$ of size at most $n_{d+1}$, such that $X'$ is a $k$ independence preserver for $(H, X)$. |

In the following, let $(H, X, k)$ be an instance of $d$-BIP. We work with a (fixed) $d$-degeneracy sequence, $\sigma$ of $H$. We recall that such a sequence can be computed in polynomial time [18]. Forward and backward neighbors of a vertex $v$ are also defined with respect to the ordering $\sigma$. If $\sigma(u) < \sigma(v)$, then $u$ is a backward neighbor of $v$ and $v$ is a forward neighbor of $u$. By $N_H^f(v)$ ($N_H^b(v)$) we denote the set of forward (backward) neighbors of the vertex $v$ in $H$.

To design our polynomial time algorithm for $d$-BIP, we need the notion of *$q$-reducible sets*, which is formally defined below.

▶ **Definition 3.** A set $Y \subseteq V(H)$ is *$q$-reducible*, if for every set $U \subseteq Y$, for which there is a set $Z \subseteq V(H)$, such that: (i) $Z$ is of size exactly $d - q + 1$ and (ii) for each $u \in U$, we have $Z \subseteq N_H^f(u)$, it holds that $|U| \leq n_q$.

Now, we give our polynomial time algorithm for $d$-BIP in Algorithm 1.

**Algorithm 1** Algo1$(H, X)$.

**Require:** $d$-degenerate graph $H$, $X \subseteq V(H)$, and an integer $k$.
**Ensure:** $X' \subseteq X$ of size at most $n_{d+1}$, which is a $k$-independence preserver of $(H, X)$.

```
For q ∈ [d], set n_q = kd + 1, when q = 1, and n_q = kn_{q-1} + kd + k + 1, otherwise.
q = 1.
while q ≤ d do
    while X is not q-reducible do
        Find U ⊆ X of size n_q + 1, for which there is Z ⊆ V(H) of size exactly d − q + 1,
    such that for each u ∈ U, we have Z ⊆ N_H^f(u).
        Let v be an arbitrary vertex in U.
        X = X \ {v}.
    end while
    q = q + 1.
end while
while |X| > n_{d+1} do
    Let v be an arbitrary vertex in X.
    X = X \ {v}.
end while
Set X' = X.
return X'
```

To prove the correctness of our algorithm, we state an observation, the proof of which follows from the fact that any vertex can have at most $d$ forward neighbors in $H$.

▶ **Observation 4.** *Let $H$ be a $d$-degenerate graph and $S$ be an independent set of $H$ of size at most $k$. Then, for any set $U \subseteq V(H)$, such that for each vertex $u \in U$, $N_H^b(u) \cap S \neq \emptyset$, we have that $|U| \leq kd$.*

Now we are ready to prove the correctness of our algorithm (Algorithm 1) for $d$-BIP.

▶ **Lemma 5.** *Algorithm 1 is correct.*

**Proof.** Let $(H, X, k)$ be an instance of $d$-BIP, and $X'$ be the output returned by Algorithm 1 with it as the input. Clearly, $X' \subseteq X$ as we do not add any new vertex to obtain the set $X'$, and size of $X'$ is bounded by $n_{d+1}$, since at Step 10-13 of the algorithm we reduce its size to (at most) $n_{d+1}$. Therefore, it remains to show that $X'$ is a $k$-independence preserver of $(H, X)$. To this end, we consider the following cases.

**Case 1:** $X$ is $q$-reducible, for each $q \in [d]$. In this case, the algorithm arbitrarily deletes vertices (if required) from $X$ to obtain $X'$. If $X = X'$, then the claim trivially holds. Therefore, we assume that $X'$ is a strict subset of $X$. To show that $X'$ is a $k$-independence preserver for $(H, X)$, consider an independent set $S$ in $H$ of size at most $k$. Furthermore, consider a vertex $v \in (S \cap X) \setminus X'$ (again, if such a vertex does not exists, the claim follows). To prove the desired result, we want to find a replacement vertex for $v$ in $X'$ which can be added to $S$ (after removing $v$) to obtain an independent set in $H$. To this end, we mark some vertices in $X'$. Firstly, mark all the forward neighbors of each $s \in S$ in the set $X'$. That is, we let $X'_M$ to be the set $(\cup_{s \in S} N_H^f(s)) \cap X'$. Also, we add all vertices in $S \cap X'$ to the set $X'_M$. By the property of $d$-degeneracy sequence, we have that $|X'_M| \leq kd + k$ (see Observation 4). Next, we will mark some more vertices in $X'_M$ with the hope to find a replacement vertex for $v$ in $X' \setminus X'_M$ to add to $S$. Recall that by our assumption $X$ is $q$-reducible, for each $q \in [d]$, and in particular, it is $d$-reducible. Thus, for each $s \in S$, the set $X_s = \{x \in X \mid s \in N_H^f(x)\} \subseteq X$ has size at most $n_d$. Based on the above observation, we describe our second level of marking of vertices in $X'$. For each $s \in S$, we add each vertex in $X_s$ to $X'_M$. From the discussions above, we have that $|X'_M| \leq kd + k + kn_d$. Since $|X'| = n_{d+1}$, and by definition, $n_{d+1} = kn_d + kd + k + 1$, we have $X' \setminus X'_M \neq \emptyset$. Moreover, no vertex in $X'$ has a neighbor in $S \setminus \{v\}$. Therefore, for $v' \in X' \setminus X'_M$, we have that $S' = (S \setminus \{v\}) \cup \{v'\}$ is an independent set in $H$.

**Case 2:** $X$ is not $q$-reducible, for some $q \in [d]$. Let $q'$ be the smallest integer for which $X$ is not $q'$-reducible. Since $X$ is not $q'$-reducible, there is a set $U \subseteq X$ of size at least $n_q + 1$, for which there is a set $Z \subseteq V(H)$ of size exactly $d - q + 1$, such that for each $u \in U$, we have $Z \subseteq N_H^f(u)$. Consider (first) such pair of sets $U, Z$ considered by the algorithm in Step 4. Furthermore, let $v \in U$ be the vertex deleted by the algorithm in Step 6. Let $\hat{U} = U \setminus \{v\}$. To prove the claim, it is enough to show that for an independent set $S$ of size at most $k$ containing $v$ in $H$, there is $v' \in \hat{U}$ such that $(S \setminus \{v\}) \cup \{v'\}$ is an independent set in $H$. Here, we will use the fact that deleting a vertex from a set does not change a set from being $\tilde{q}$-reducible to a set which is not $\tilde{q}$-reducible, where $\tilde{q} \in [d]$. In the following, consider an independent set $S$ of size at most $k$ containing $v$ in $H$. We construct a marked set $\hat{U}_M$, of vertices in $\hat{U}$. Firstly, we add all the vertices in $(\cup_{s \in S \setminus \{v\}} N_H^f(s)) \cap \hat{U}$ to $\hat{U}_M$. Also, we add all vertices in $S \cap \hat{U}$ to $\hat{U}_M$. Notice that at the end of above marking scheme, we have $|\hat{X}_M| \leq kd + k$. We will mark some more vertices in $\hat{U}$. Before stating the second level of marking, we remark that $S \cap Z = \emptyset$. For each $s \in S \setminus \{v\}$, let $Z_s = Z \cup \{s\}$. Since $S \cap Z = \emptyset$, we have that $|Z_s| = d - (q - 1) + 1$. For $s \in S \setminus \{v\}$, let $\hat{U}_s = \{u \in \hat{U} \mid Z_s \subseteq N_H^f(u)\}$. Since $X$ is $q^*$-reducible for each $q^* < q'$, we have $|\hat{U}_s| \leq n_{q-1}$, for each $s \in S \setminus \{v\}$. Now we are ready to describe our second

level of marking. For each $s \in S \setminus \{v\}$, add all vertices in $U_s$ to the set $\hat{U}_M$. Notice that $|\hat{U}_M| \leq kd + k + kn_{q-1}$. Moreover, $|\hat{U}| \geq n_q$ and $n_q = kn_{q-1} + kd + k + 1$. Thus, there is a vertex $v' \in \hat{U} \setminus \hat{U}_M$, such that $(S \setminus \{v\}) \cup \{v'\}$ is an independent set in $H$. ◀

▶ **Lemma 6.** $(\star)$[2] *Algorithm 1 runs in time* $n^{\mathcal{O}(d)}$.

Using Lemma 5 and Lemma 6 we obtain the following theorem.

▶ **Theorem 7.** $d$-Bounded Independence Preserver *admits an algorithm running in time* $n^{\mathcal{O}(d)}$.

## 4 A Polynomial Kernel for $\mathcal{D}_d$-CF-FVS

In this section, we design a kernelization algorithm for $\mathcal{D}_d$-CF-FVS.

To design a kernelization algorithm for $\mathcal{D}_d$-CF-FVS, we define another problem called $\mathcal{D}_d$-Disjoint-CF-FVS ($\mathcal{D}_d$-DCF-FVS, for short). We first define the problem $\mathcal{D}_d$-DCF-FVS formally, and then explain its uses in our kernelization algorithm.

| $\mathcal{D}_d$-Disjoint-CF-FVS ($\mathcal{D}_d$-DCF-FVS) **Parameter:** $k$ |
|---|
| **Input:** An undirected graph $G$, a graph $H \in \mathcal{D}_d$ such that $V(G) = V(H)$, a subset $R \subseteq V(G)$, and a non-negative integer $k$. |
| **Question:** Is there a set $S \subseteq V(G) \setminus R$ of size at most $k$, such that $G - S$ does not have any cycle and $S$ is an independent set in $H$? |

Notice that $\mathcal{D}_d$-CF-FVS is a special case of $\mathcal{D}_d$-DCF-FVS, where $R = \emptyset$. Given an instance of $\mathcal{D}_d$-CF-FVS, the kernelization algorithm creates an instance of $\mathcal{D}_d$-DCF-FVS by setting $R = \emptyset$. Then it applies a kernelization algorithm for $\mathcal{D}_d$-DCF-FVS. Finally, the algorithm takes the instance returned by the kernelization algorithm for $\mathcal{D}_d$-DCF-FVS and generates an instance of $\mathcal{D}_d$-CF-FVS. Before moving forward, we note that the purpose of having set $R$ is to be able to prohibit certain vertices to belong to a solution. This is particularly useful in maintaining the independent set property of the solution, when applying reduction rules which remove vertices from the graph (with an intention of it being in a solution).

We first focus on designing a kernelization algorithm for $\mathcal{D}_d$-DCF-FVS, and then give a polynomial time linear parameter preserving reduction from $\mathcal{D}_d$-DCF-FVS to $\mathcal{D}_d$-CF-FVS. If the kernelization algorithm for $\mathcal{D}_d$-DCF-FVS returns that $(G, H, R, k)$ is a YES (NO) instance of $\mathcal{D}_d$-DCF-FVS, then conclude that $(G, H, k)$ is a YES (NO) instance of $\mathcal{D}_d$-CF-FVS. In the following, we describe a kernelization algorithm for $\mathcal{D}_d$-DCF-FVS. Let $(G, H, R, k)$ be an instance of $\mathcal{D}_d$-DCF-FVS. The algorithm starts by applying the following simple reduction rules.

▶ **Reduction Rule 1.**

**(a)** *If* $k \geq 0$ *and* $G$ *is acyclic, then return that* $(G, H, R, k)$ *is a* YES *instance of* $\mathcal{D}_d$-DCF-FVS.

**(b)** *Return that* $(G, H, R, k)$ *is a* NO *instance of* $\mathcal{D}_d$-DCF-FVS, *if one of the following conditions is satisfied:*

- **(i)** $k \leq 0$ *and* $G$ *is not acyclic,*
- **(ii)** $G$ *is not acyclic and* $V(G) \subseteq R$, *or*

[2] Due to space constraints, the proofs of results marked with $\star$ have been omitted from this extended abstract.

**(iii)** *There are more than $k$ isolated cycles in $G$.*

▶ **Reduction Rule 2.**

**(a)** *Let $v$ be a vertex of degree at most 1 in $G$. Then delete $v$ from the graphs $G, H$ and the set $R$.*

**(b)** *If there is an edge in $G$ ($H$) with multiplicity more than 2 (more than 1), then reduce its multiplicity to 2 (1).*

**(c)** *If there is a vertex $v$ with self loop in $G$. If $v \notin R$, delete $v$ from the graphs $G$ and $H$, and decrease $k$ by one. Furthermore, add all the vertices in $N_H(v)$ to the set $R$, otherwise return that $(G, H, R, k)$ is a* NO *instance of $\mathcal{D}_d$-*DCF-FVS.

**(d)** *If there are parallel edges between (distinct) vertices $u, v \in V(G)$ in $G$:*

**(i)** *If $u, v \in R$, then return that $(G, H, R, k)$ is a* NO *instance of $\mathcal{D}_d$-*DCF-FVS.

**(ii)** *If $u \in R$ ($v \in R$), delete $v$ ($u$) from the graphs $G$ and $H$, and decrease $k$ by one. Furthermore, add all the vertices in $N_H(v)$ ($N_H(u)$) to the set $R$.*

It is easy to see that the above reduction rules are correct, and can be applied in polynomial time. In the following, we define some notion and state some known results, which will be helpful in designing our next reduction rules.

▶ **Definition 8.** For a graph $G$, a vertex $v \in V(G)$, and an integer $t \in \mathbb{N}$, a *$t$-flower* at $v$ is a set of $t$ vertex disjoint cycles whose pairwise intersection is exactly $\{v\}$.

▶ **Proposition 9.** [8, 19, 25] *For a graph $G$, a vertex $v \in V(G)$ without a self-loop in $G$, and an integer $k$, the following conditions hold.*

**(i)** *There is a polynomial time algorithm, which either outputs a $(k+1)$-flower at $v$, or it correctly concludes that no such $(k+1)$-flower exists. Moreover, if there is no $(k+1)$-flower at $v$, it outputs a set $X_v \subseteq V(G) \setminus \{v\}$ of size at most $2k$, such that $X_v$ intersects every cycle passing through $v$ in $G$.*

**(ii)** *If there is no $(k+1)$-flower at $v$ in $G$ and the degree of $v$ is at least $4k + (k+2)2k$. Then using a polynomial time algorithm we can obtain a set $X_v \subseteq V(G) \setminus \{v\}$ and a set $\mathcal{C}_v$ of components of $G[V(G) \setminus (X_v \cup \{v\})]$, such that each component in $\mathcal{C}_v$ is a tree, $v$ has exactly one neighbor in $C \in \mathcal{C}_v$, and there exist at least $k+2$ components in $\mathcal{C}_v$ corresponding to each vertex $x \in X_v$ such that these components are pairwise disjoint and vertices in $X_v$ have an edge to each of their associated components.*

▶ **Reduction Rule 3.** *Consider $v \in V(G)$, such that there is a $(k+1)$-flower at $v$ in $G$. If $v \in R$, then return that $(G, H, R, k)$ is a* NO *instance of $\mathcal{D}_d$-*DCF-FVS. *Otherwise, delete $v$ from $G, H$ and decrease $k$ by one. Furthermore, add all the vertices in $N_H(v)$ to $R$.*

The correctness of the above reduction rule follows from the fact that such a vertex must be part of every solution of size at most $k$. Moreover, the applicability of it in polynomial time follows from Proposition 9 (item (i)).

▶ **Reduction Rule 4.** *Let $v \in V(G)$, $X_v \subseteq V(G) \setminus \{v\}$, and $\mathcal{C}_v$ be the set of components which satisfy the conditions in Proposition 9(ii) (in $G$), then delete edges between $v$ and the components of the set $\mathcal{C}_v$, and add parallel edges between $v$ and every vertex $x \in X_v$ in $G$.*

The polynomial time applicability of Reduction Rule 4 follows from Proposition 9. And, in the following lemma, we prove the safeness of this reduction rule.

▶ **Lemma 10.** ($\star$) *Reduction Rule 4 is safe.*

In the following, we state an easy observation, which follows from non-applicability of Reduction Rule 1 to 4.

▶ **Observation 11.** *Let $(G, H, R, k)$ be an instance of $\mathcal{D}_d$-DCF-FVS, where none of Reduction Rule 1 to 4 apply. Then the degree of each vertex in $G$ is bounded by $\mathcal{O}(k^2)$.*

**Proof.** As Reduction Rule 3 is not applicable, then there is no $k+1$-flower in $G$. Now, if there is $v \in V(G)$ with degree at least $4k + (k+2)2k$, then Reduction Rule 4 would be applicable. ◀

To design our next reduction rule, we construct an auxiliary graph $G^\star$. Intuitively speaking, $G^\star$ is obtained from $G$ by shortcutting all degree two vertices. That is, vertex set of $G^\star$ comprises of all the vertices of degree at least three in 3. From now on, vertices of degree at least 3 (in $G$) will be referred to as high degree vertices. For each $uv \in E(G)$, where $u, v$ are high degree vertices, we add the edge $uv$ in $G^\star$. Furthermore, for an induced maximal path $P_{uv}$, between $u$ and $v$ where all the internal vertices of $P_{uv}$ are degree two vertices in $G$, we add the (multi) edge $uv$ to $E(G^\star)$. Next, we will use the following result to bound the number of vertices and edges in $G^\star$.

▶ **Proposition 12.** [8] *A graph $G$ with minimum degree at least 3, maximum degree $\Delta$, and a feedback vertex set of size at most $k$ has at most $(\Delta+1)k$ vertices and $2\Delta k$ edges.*

The above result (together with the construction of $G^\star$) gives us the following (safe) reduction rule.

▶ **Reduction Rule 5.** *If $|V(G^\star)| \geq 4k^2 + 2k^2(k+2)$ or $|E(G^\star)| \geq 8k^2 + 4k^2(k+2)$, then return* NO.

▶ **Lemma 13.** *Let $(G, H, R, k)$ be an instance of $\mathcal{D}_d$-DCF-FVS, where none of the Reduction Rules 1 to 5 are applicable. Then we obtain the following bounds:*
- *The number of vertices of degree at least 3 in $G$ is bounded by $\mathcal{O}(k^3)$.*
- *The number of maximal degree two induced paths in $G$ is bounded by $\mathcal{O}(k^3)$.*

Having shown the above bounds, it remains to bound the number of degree two vertices in $G$. We start by applying the following simple reduction rule to eliminate vertices of degree two in $G$, which are also in $R$.

▶ **Reduction Rule 6.** *Let $v \in R$, $d_G(v) = 2$, and $x, y$ be the neighbors of $v$ in $G$. Delete $v$ from the graphs $G, H$ and the set $R$. Furthermore, add the edge $xy$ in $G$.*

The correctness of this reduction rule follows from the fact that vertices in $R$ can not be part of any solution and all the cycles passing through $v$ also passes through its neighbors.

In the polynomial kernel for the Feedback Vertex Set problem (with no conflict constraints), we can short-circuit degree two vertices. But in our case, we cannot perform this operation, since we also need the solution to be an independent set in the conflict graph. Thus to reduce the number of degree two vertices in $G$, we exploit the properties of a $d$-degenerate graph. To this end, we use the tool that we developed in Section 3. This immediately gives us the following reduction rule.

▶ **Reduction Rule 7.** *Let $P$ be a maximal degree two induced path in $G$. If $|V(P)| \geq n_{d+1}+1$, apply Algorithm 1 with input $(H, V(P) \setminus R)$. Let $\widehat{V}(P)$ be the set returned by Algorithm 1. Let $v \in (V(P) \setminus R) \setminus \widehat{V}(P)$, and $x, y$ be the neighbors of $v$ in $G$. Delete $v$ from the graphs $G, H$. Furthermore, add edge $xy$ in $G$.*

▶ **Lemma 14.** *Reduction Rule 7 is safe.*

**Proof.** Let $(G, H, R, k)$ be an instance of $\mathcal{D}_d$-DCF-FVS and $v$ be a vertex in a maximal degree two path $P$ with neighbors $x$ and $y$, with respect to which Reduction Rule 14 is applied. Furthermore, let $(G', H', R, k)$ be the resulting instance after application of the reduction rule. We will show that $(G, H, R, k)$ is a YES instance of $\mathcal{D}_d$-DCF-FVS if and only if $(G', H', R, k)$ is a YES instance of $\mathcal{D}_d$-DCF-FVS.

In the forward direction, let $(G, H, R, k)$ be a YES instance of $\mathcal{D}_d$-DCF-FVS and $S$ be one of its minimal solution. Consider the case when $v \notin S$. In this case, we claim that $S$ is also a solution of $\mathcal{D}_d$-DCF-FVS for $(G', H', R, k)$. Suppose not then either $S$ is not an independent set in $H'$ or $G' - S$ contains a cycle. Since, $H'$ is an induced subgraph of $H$, we have that $S'$ is also an independent set in $H'$. So we assume that $G' - S$ has a cycle, say $C$. If $C$ does not contain the edge $xy$, then $C$ is also a cycle in $G - S$. Therefore, we assume that $C$ contains the edge $xy$. But then $(C \setminus \{xy\}) \cup \{xv, vy\}$ is a cycle in $G - S$. Next, we consider the case when $v \in S$. By Lemma 5 we have a vertex $v' \in V(P) \setminus \{v\}$ such that $(S \setminus \{v\}) \cup \{v'\}$ is an independent set in $H'$. By using the fact that any cycle that passes through $v$ also contains all vertices in $P$ (together with the discussions above) imply that $(S \setminus \{v\}) \cup \{v'\}$ is a solution of $\mathcal{D}_d$-DCF-FVS for $(G', H', R, k)$.

In the reverse direction, let $(G', H', R, k)$ be a YES instance of $\mathcal{D}_d$-DCF-FVS and $S'$ be one of its minimal solution. We claim that $S'$ is also a solution of $\mathcal{D}_d$-DCF-FVS for $(G, H, R, k)$. Suppose not, then either $S$ is not an independent set in $H$ or $G - S$ contains a cycle. Since, $H'$ is an induced subgraph of $H$, we have that $S'$ is also an independent set in $H$. Next, assume that there is a cycle $C$ in $G - S$. The cycle $C$ must contain $v$, otherwise, $C$ is also a cycle in $G' - S'$. Since $v$ is a degree two vertex in $G$, therefore any cycle that contains $v$, must also contain $x$ and $y$. As observed before, $G - \{xv, vy\}$ is identical to $G' - \{xy\}$. But then, $(C \setminus \{xv, vy\}) \cup \{xy\}$ is a cycle in $G' - S'$, a contradiction. This concludes that $S'$ is a solution of $\mathcal{D}_d$-DCF-FVS for $(G, H, R, k)$. ◀

▶ **Lemma 15.** ($\star$) *Let $(G, H, R, k)$ be an instance of $\mathcal{D}_d$-DCF-FVS, where none of the Reduction Rules 1 to 7 are applicable. Then the number of vertices in a degree two induced path in $G$ is bounded by $\mathcal{O}(k^{\mathcal{O}(d)})$.*

▶ **Theorem 16.** $\mathcal{D}_d$-DCF-FVS *admits a kernel with $\mathcal{O}(k^{\mathcal{O}(d)})$ vertices.*

▶ **Lemma 17.** ($\star$) *There is a polynomial time parameter preserving reduction from $\mathcal{D}_d$-DCF-FVS to $\mathcal{D}_d$-CF-FVS.*

By Theorem 16 and Lemma 17, we obtain the following result.

▶ **Theorem 18.** $\mathcal{D}_d$-CF-FVS *admits a kernel with $\mathcal{O}(k^{\mathcal{O}(d)})$ vertices.*

## 5 Kernelization Complexity of $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT

In this section, we show that CF-OCT does not admit a polynomial kernel when the conflict graph belongs to the family $\mathcal{P}^{\star\star}_{\leq 3}$. Let $\mathcal{P}_{\leq 3}$ denotes the family of disjoint union of paths of length at most three, and $\mathcal{P}^{\star}_{\leq 3}$ denotes the family of disjoint union of paths of length at most three and a star graph. We give parameter preserving reduction from $\mathcal{P}^{\star}_{\leq 3}$-Conflict Free $s$-$t$ Cut ($\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ Cut) to $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT.

We first prove that $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ Cut is NP-hard. Then, we prove that $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ Cut does not admit a polynomial compression, unless $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$ using the method of cross-composition.

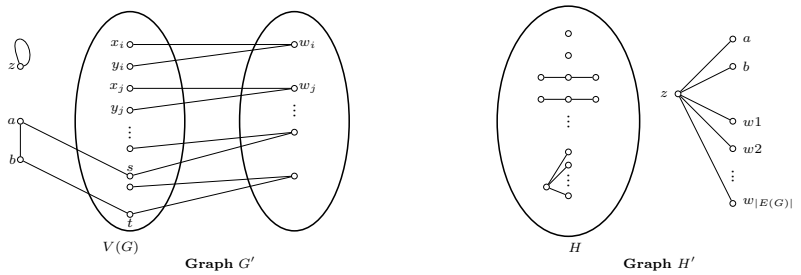



**Figure 1** An illustration of construction of graph $G'$ and $H'$ in reduction from $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ Cut to $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT.

▶ **Theorem 19** ($\star$). $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ Cut *does not admit a polynomial compression unless* $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$.

**Lower Bound for Kernel of $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT.** In this subsection, we prove the main result of this section. We show that there does not exist a polynomial kernel of $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT. Towards this we give a parameter preserving reduction from $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ Cut to $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT. Given an instance $(G, H, s, t, k)$ of $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ Cut, we construct an instance $(G', H', k+1)$ of $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT as follows. Initially, we have $V(G') = V(H') = V(G) \cup \{z, a, b\}$. Now, for each edge $e_i \in E(G)$, add a vertex $w_i$ to $V(G')$ and $V(H')$. Now, we define the edge set of $G'$. Let $x_i, y_i$ be end points of $e_i \in E(G)$. For each $e_i \in E(G)$, add edges $x_iw_i$ and $y_iw_i$ to $E(G')$. Also, add a self loop on $z$ in $G'$ and edges $sa, ab$ and $bt$ to $E(G')$. To construct the edge set of $H'$, we set $E(H') = E(H - \{s, t\})$. Additionally, we add $zs, zt, za, zt$, and $zw_i$ for each $w_i \in V(H')$ to $E(H')$. Figure 1 describes the construction of $G'$ and $H'$.

Clearly, $H'$ belongs to $\mathcal{P}^{\star\star}_{3}$ and this construction can be carried out in the polynomial time. Now, we prove the equivalence between the instances $(G, H, s, t, k)$ of $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ Cut and $(G', H', k+1)$ of $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT in the following lemma.

▶ **Lemma 20.** $(G, H, s, t, k)$ *is a yes-instance of* $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ Cut *if and only if* $(G', H', k+1)$ *is a yes-instance of* $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT.

**Proof.** In the forward direction, let $(G, H, s, t, k)$ be a yes-instance of $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ Cut and $S$ be one of its solution. We claim that $S \cup \{z\}$ is a solution to $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT in $(G', H', k+1)$. In the graph $G'$, since we subdivide each edge, all the paths from $s - t$ are of even length. Since, we subdivide each edge of $G$, $G' - \{a, b, z\}$ is a bipartite graph. Hence, an odd cycle in $G' - z$ consists of an $s - t$ path in $G' - \{a, b\}$ and edges $sa$, $ab$ and $bt$. Clearly, by the construction of $G'$, $(G' - \{a, b\}) \setminus S$ does not contain an $s - t$ path and hence $G' - z$ does not contain an odd cycle. Since, $H[S]$ is edgeless, $S \cup \{z\}$ is an independent set in $H'$. This completes the proof in the forward direction.

In the reverse direction, let $S$ be a solution to $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT in $(G', H', k+1)$. Since, $z \in S$, therefore, $s, t, a, b, w_i \notin S$ for any $w_i \in V(H')$. We claim that $S' = S \setminus \{z\}$ is a solution to $\mathcal{P}^{\star}_{\leq 3}$-CF-$s$-$t$ Cut in $(G, H, s, t, k)$. Suppose not, then there exists a $s-t$ path $(s, x_1, x_2, \cdots, x_l, t)$ in $G \setminus S'$. Correspondingly, there exists a $s-t$ path $(s, w_1, x_1, w_2, x_2, \cdots, x_l, w_{l+1}, t)$ in $G'$ of even length which results into an odd cycle $(s, w_1, x_1, w_2, x_2, \cdots, x_l, w_{l+1}, t, b, a)$ in $G' \setminus S$, a contradiction. This completes the proof. ◀

Now, we present the main result of this section in the following theorem.

▶ **Theorem 21.** $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT *does not admit a polynomial kernel. unless* $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$.

## 6 Conclusion

In this paper we studied kernelization complexity of $\mathcal{D}_d$-CF-FVS and $\mathcal{D}_d$-CF-OCT. We showed that the former admits a polynomial kernel of size $k^{\mathcal{O}(d)}$, while $\mathcal{D}_d$-CF-OCT does not admit any polynomial kernel unless $\mathsf{NP} \subseteq \frac{\mathsf{coNP}}{\mathsf{poly}}$. In fact, the later does not admit polynomial kernel even for much more specialized problem, namely $\mathcal{P}^{\star\star}_{\leq 3}$-CF-OCT. Using much more involved marking scheme we can show that $\mathcal{D}_d$-CF-ECT admits polynomial kernel of size $k^{\mathcal{O}(d)}$. Similarly, we can extend the known polynomial kernel for OCT to CF-OCT when the conflict graph $H$ has maximum degree at most one. Two most interesting questions that still remain open form our work are following: (a) does CF-FVS admit uniform polynomial kernel on graphs of bounded expansion; and (b) does CF-OCT admit a polynomial kernel when $H$ is disjoint union of paths of length at most 2.

## References


1. Esther M. Arkin, Aritra Banik, Paz Carmi, Gui Citovsky, Matthew J. Katz, Joseph S. B. Mitchell, and Marina Simakov. Choice is Hard. In *ISAAC*, pages 318–328, 2015.
2. Esther M. Arkin, Aritra Banik, Paz Carmi, Gui Citovsky, Matthew J. Katz, Joseph S. B. Mitchell, and Marina Simakov. Conflict-free Covering. In *CCCG*, pages 17–23, 2015.
3. Aritra Banik, Fahad Panolan, Venkatesh Raman, and Vibha Sahlot. Fréchet Distance Between a Line and Avatar Point Set. *FSTTCS*, pages 32:1–32:14, 2016.
4. Hans L. Bodlaender. On disjoint cycles. In *WG*, volume 570, pages 230–238, 1992.
5. Hans L. Bodlaender. A Cubic Kernel for Feedback Vertex Set. In *STACS*, volume 4393, pages 320–331, 2007.
6. Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization Lower Bounds by Cross-Composition. *J. Discrete Math.*, 28:277–305, 2014.
7. Kevin Burrage, Vladimir Estivill-Castro, Michael R. Fellows, Michael A. Langston, Shev Mac, and Frances A. Rosamond. The Undirected Feedback Vertex Set Problem Has a Poly($k$) Kernel. In *IWPEC*, volume 4169, pages 192–202, 2006.
8. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
9. Andreas Darmann, Ulrich Pferschy, and Joachim Schauer. Determining a Minimum Spanning Tree with Disjunctive Constraints. In *ADT*, volume 5783, pages 414–423, 2009.
10. Andreas Darmann, Ulrich Pferschy, Joachim Schauer, and Gerhard J. Woeginger. Paths, trees and matchings under disjunctive constraints. *Discrete Applied Mathematics*, 159(16):1726–1735, 2011.
11. Rodney G. Downey and Michael R. Fellows. Fixed Parameter Tractability and Completeness. In *Complexity Theory: Current Research*, pages 191–225, 1992.
12. Leah Epstein, Lene M. Favrholdt, and Asaf Levin. Online variable-sized bin packing with conflicts. *Discrete Optimization*, 8(2):333–343, 2011.
13. Guy Even, Magnús M. Halldórsson, Lotem Kaplan, and Dana Ron. Scheduling with conflicts: online and offline algorithms. *J. Scheduling*, 12(2):199–224, 2009.
14. Pallavi Jain, Lawqueen Kanesh, and Pranabendu Misra. Conflict Free Version of Covering Problems on Graphs: Classical and Parameterized. *CSR*, pages 194–206, 2018.
15. Viggo Kann. Polynomially bounded minimization problems which are hard to approximate. In *ICALP*, pages 52–63, 1993.
16. Stefan Kratsch and Magnus Wahlström. Compression via Matroids: A Randomized Polynomial Kernel for Odd Cycle Transversal. *ACM Trans. Algorithms*, 10(4):20:1–20:15, 2014.

17 Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Roohani Sharma, and Meirav Zehavi. Covering Small Independent Sets and Separators with Applications to Parameterized Algorithms. In *SODA*, pages 2785–2800, 2018.

18 David W. Matula and Leland L. Beck. Smallest-Last Ordering and clustering and Graph Coloring Algorithms. *J. ACM*, 30(3):417–427, 1983.

19 Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, and Saket Saurabh. On Parameterized Independent Feedback Vertex Set. *Theor. Comput. Sci.*, 461:65–75, 2012.

20 Pranabendu Misra, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Parameterized Algorithms for Even Cycle Transversal. In *WG*, volume 7551, pages 172–183, 2012.

21 Ulrich Pferschy and Joachim Schauer. The Maximum Flow Problem with Conflict and Forcing Conditions. In *INOC*, volume 6701, pages 289–294, 2011.

22 Ulrich Pferschy and Joachim Schauer. The maximum flow problem with disjunctive constraints. *J. Comb. Optim.*, 26(1):109–119, 2013.

23 Ulrich Pferschy and Joachim Schauer. Approximation of knapsack problems with conflict and forcing graphs. *J. Comb. Optim.*, 33(4):1300–1323, 2017.

24 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004.

25 Stéphan Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Trans. Algorithms*, 6(2):32:1–32:8, 2010.

# Best-Case and Worst-Case Sparsifiability of Boolean CSPs

**Hubie Chen**
Birkbeck, University of London, Malet Street, Bloomsbury, London WC1E 7HX, United Kingdom
hubie@dcs.bbk.ac.uk

**Bart M. P. Jansen**[1]
Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
b.m.p.jansen@tue.nl

**Astrid Pieterse**[2]
Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
a.pieterse@tue.nl

## Abstract

We continue the investigation of polynomial-time sparsification for NP-complete Boolean Constraint Satisfaction Problems (CSPs). The goal in sparsification is to reduce the number of constraints in a problem instance without changing the answer, such that a bound on the number of resulting constraints can be given in terms of the number of variables $n$. We investigate how the worst-case sparsification size depends on the types of constraints allowed in the problem formulation (the constraint language). Two algorithmic results are presented. The first result essentially shows that for any arity $k$, the only constraint type for which no nontrivial sparsification is possible has exactly one falsifying assignment, and corresponds to logical OR (up to negations). Our second result concerns linear sparsification, that is, a reduction to an equivalent instance with $\mathcal{O}(n)$ constraints. Using linear algebra over rings of integers modulo prime powers, we give an elegant necessary and sufficient condition for a constraint type to be captured by a degree-1 polynomial over such a ring, which yields linear sparsifications. The combination of these algorithmic results allows us to prove two characterizations that capture the optimal sparsification sizes for a range of Boolean CSPs. For NP-complete Boolean CSPs whose constraints are *symmetric* (the satisfaction depends only on the number of 1 values in the assignment, not on their positions), we give a complete characterization of which constraint languages allow for a linear sparsification. For Boolean CSPs in which every constraint has arity at most three, we characterize the optimal size of sparsifications in terms of the largest OR that can be expressed by the constraint language.

**2012 ACM Subject Classification** Computing methodologies → Symbolic and algebraic algorithms, Theory of computation → Problems, reductions and completeness, Theory of computation → Parameterized complexity and exact algorithms

**Keywords and phrases** constraint satisfaction problems, kernelization, sparsification, lower bounds

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.15

**Related Version** A full version is available at [4], `https://arxiv.org/abs/1809.06171v1`.

**Acknowledgements** We would like to thank Emil Jeřábek for the proof of Lemma 4.2.

[1] Supported by NWO Gravitation grant "Networks".
[2] Supported by NWO Gravitation grant "Networks".

© Hubie Chen, Bart M. P. Jansen, and Astrid Pieterse;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 15; pp. 15:1–15:13
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

# 1 Introduction

## Background

The framework of constraint satisfaction problems (CSPs) provides a unified way to study the computational complexity of a wide variety of combinatorial problems such as CNF-Satisfiability, Graph Coloring, and Not-All-Equal SAT. The framework uncovers algorithmic approaches that simultaneously apply to several problems, and also identifies common sources of intractability. For the purposes of this discussion, a CSP is specified using a (finite) *constraint language*, which is a set of (finite) relations; the problem is to decide the satisfiability of a set of constraints, where each constraint has a relation coming from the constraint language. The fact that many problems can be viewed as CSPs motivates the following investigation: how does the complexity of a CSP depend its constraint language? A key result in this area is Schaefer's dichotomy theorem [18], which classifies each CSP over the Boolean domain as polynomial-time solvable or NP-complete.

Continuing a recent line of investigation [10, 12, 15], we aim to understand for which NP-complete CSPs an instance can be *sparsified* in polynomial time, without changing the answer. In particular, we investigate the following questions. Can the number of constraints be reduced to a small function of the number of variables $n$? How does the sparsifiability of a CSP depend on its constraint language? We utilize the framework of kernelization [5, 8, 16], originating in parameterized complexity theory, to answer such questions.

The first results concerning polynomial-time sparsification in terms of the number $n$ of variables or vertices were mainly negative. Under the assumption that $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$ (which we tacitly assume throughout this introduction), Dell and van Melkebeek [7] proved a strong lower bound: For any integer $d \geq 3$ and positive real $\varepsilon$, there cannot be a polynomial-time algorithm that compresses any instance $\varphi$ of $d$-CNF-SAT on $n$ variables, into an equivalent SAT instance $\varphi'$ of bitsize $\mathcal{O}(n^{d-\varepsilon})$. In fact, there cannot even be an algorithm that transforms such $\varphi$ into small equivalent instances $\psi$ of an *arbitrary* decision problem. Since an instance of $d$-CNF-SAT has at most $2^d n^d \in \mathcal{O}(n^d)$ distinct clauses, it can *trivially* be sparsified to $\mathcal{O}(n^d)$ clauses by removing duplicates, and can be compressed to size $\mathcal{O}(n^d)$ by storing it as a bitstring indicating for each possible clause whether or not it is present. The cited lower bound therefore shows that the trivial sparsification for $d$-CNF-SAT cannot be significantly improved; we say that the problem does not admit nontrivial (polynomial-time) sparsification. Following these lower bounds for SAT, a number of other results were published [6, 9, 14] proving other problems do not admit nontrivial sparsification either.

This pessimistic state of affairs concerning nontrivial sparsification algorithms changed several years ago, when a subset of the authors [12] showed that the $d$-Not-All-Equal SAT problem *does* have a nontrivial sparsification. In this problem, clauses have size at most $d$ and are satisfied if the literals do not all evaluate to the same value. While there can be $\Omega(n^d)$ different clauses in an instance, there is an efficient algorithm that finds a subset of $\mathcal{O}(n^{d-1})$ clauses that preserves the answer, resulting in a compression of bitsize $\mathcal{O}(n^{d-1} \log n)$. The first proof of this result was based on an ad-hoc application of a theorem of Lovász [17]. Later, the underlying proof technique was extracted and applied to a wider range of problems [10]. This led to the following understanding: if each relation in the constraint language can be represented by a polynomial of degree at most $d$, in a certain technical sense, then this allows the number of constraints in an $n$-variable instance of such a CSP to be reduced to $\mathcal{O}(n^d)$. The sparsification for $d$-Not-All-Equal SAT is then explained by noting that such constraints can be captured by polynomials of degree $d-1$. It is therefore apparent that finding a low-degree polynomial to capture the constraints of a CSP is a powerful tool

to obtain sparsification algorithms for it. Finding such polynomials of a certain degree $d$, or determining that they do not exist, proved a challenging and time-intensive task (cf. [11]).

The polynomial-based framework [10] also resulted in some *linear* sparsifications. Since "1-in-$d$" constraints (to satisfy a clause, *exactly one* out of its $\leq d$ literals should evaluate to true) can be captured by *linear* polynomials, the 1-IN-$d$-SAT problem has a sparsification with $\mathcal{O}(n)$ constraints for each constant $d$. This prompted a detailed investigation into linear sparsifications for CSPs by Lagerkvist and Wahlström [15], who used the toolkit of universal algebra in an attempt to obtain a characterization of the Boolean CSPs with a linear sparsification. Their results give a necessary and sufficient condition on the constraint language of a CSP for having a so-called Maltsev embedding over an infinite domain. They also show that when a CSP has a Maltsev embedding over a *finite* domain, then this can be used to obtain a linear sparsification. Alas, it remains unclear whether Maltsev embeddings over infinite domains can be exploited algorithmically, and a characterization of the linearly-sparsifiable CSPs is currently not known.

## Our contributions

We analyze and demonstrate the power of the polynomial-based framework for sparsifying CSPs using universal algebra, linear algebra over rings, and relational analysis. We present two new algorithmic results. These allow us to characterize the sparsifiability of Boolean CSPs in two settings, wherein we show that the polynomial-based framework yields *optimal* sparsifications. In comparison to previous work [10], our results are much more fine-grained and based on a deeper understanding of the reasons why a certain CSP *cannot* be captured by low-degree polynomials.

**Algorithmic results.** Our first result (Section 3) shows that, contrary to the pessimistic picture that arose during the initial investigation of sparsifiability, the phenomenon of nontrivial sparsification is widespread and occurs for almost all Boolean CSPs! We prove that if $\Gamma$ is a constraint language whose largest constraint has arity $k$, then the *only* reason that CSP($\Gamma$) does not have a nontrivial sparsification, is that it contains an arity-$k$ relation that is essentially the $k$-ary OR (up to negating variables). When $R \subseteq \{0,1\}^k$ is a relation with $|\{0,1\}^k \setminus R| \neq 1$ (the number of assignments that fail to satisfy the constraint is not equal to 1), then it can be captured by a polynomial of degree $k-1$. This yields a nontrivial sparsification compared to the $\Omega(n^k)$ distinct applications of this constraint that can be in such an instance.

Our second algorithmic result (Section 4) concerns the power of the polynomial-based framework for obtaining linear sparsifications. We give a necessary and sufficient condition for a relation to be captured by a degree-1 polynomial. Say that a Boolean relation $R \subseteq \{0,1\}^k$ is *balanced* if there is *no* sequence of vectors $s_1, \ldots, s_{2n}, s_{2n+1} \in R$ for $n \geq 1$ such that $s_1 - s_2 + s_3 \ldots - s_{2n} + s_{2n+1} = u \in \{0,1\}^k \setminus R$. (The same vector may appear multiple times in this sum.) In other words: $R$ is balanced if one cannot find an odd-length sequence of vectors in $R$ for which alternating between adding and subtracting these vectors component-wise results in a 0/1-bitvector $u$ that is outside $R$. For example, the binary OR relation 2-OR $= \{0,1\}^2 \setminus \{(0,0)\}$ is *not* balanced, since $(0,1) - (1,1) + (1,0) = (0,0) \notin$ 2-OR, but the 1-in-3 relation $R_{=1} = \{(1,0,0),(0,1,0),(0,0,1)\}$ is. We prove that if a Boolean relation $R$ is balanced, then it can efficiently be captured by a degree-1 polynomial and the number of constraints that are applications of this relation can be reduced to $\mathcal{O}(n)$. Hence when *all* relations in a constraint language $\Gamma$ are balanced—we call such a constraint language *balanced*—then CSP($\Gamma$) has a sparsification with $\mathcal{O}(n)$ constraints. We also show

that, on the other hand, if a Boolean relation $R$ is not balanced, then there does not exist a degree-1 polynomial over any ring that captures $R$ in the sense required for application of the polynomial framework. The property of being balanced is (as defined) a universal-algebraic property; these results thus tightly bridge universal algebra and the polynomial framework.

**Characterizations.** The property of being balanced gives an easy way to prove that certain Boolean CSPs admit linear sparsifications. But perhaps more importantly, this characterization constructively exhibits a certain *witness* when a relation can *not* be captured by a degree-1 polynomial, in the form of the alternating sum of satisfying assignments that yield an unsatisfying assignment. In several scenarios, we can turn this witness structure against degree-1 polynomials into a lower bound proving that the problem does not have a linear sparsification. As a consequence, we can prove two fine-grained characterizations of sparsification complexity.

**Characterization of symmetric CSPs with a linear sparsification (Section 5)** We say that a Boolean relation is *symmetric* if the satisfaction of a constraint only depends on the *number* of 1-values taken by the variables (the *weight* of the assignment), but does not depend on the *positions* where these values appear. For example, "1-in-$k$"-constraints are symmetric, just as "not-all-equal"-constraints, but the relation $R_{a\to b} = \{(0,0),(0,1),(1,1)\}$ corresponding to the truth value of $a \to b$ is not. We prove that if a symmetric Boolean relation $R$ is not balanced, then it can implement (Definition 2.7) a binary OR using constants and negations but without having to introduce fresh variables. Building on this, we prove that if such an unbalanced symmetric relation $R$ occurs in a constraint language $\Gamma$ for which CSP($\Gamma$) is NP-complete, then CSP($\Gamma$) does *not* admit a sparsification of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$. Consequently, we obtain a characterization of the sparsification complexity of NP-complete Boolean CSPs whose constraint language consists of symmetric relations: there is a linear sparsification if and only if the constraint language is balanced. This yields linear sparsifications in several new scenarios that were not known before.

**Characterization of sparsification complexity for CSPs of low arity (Section 6)** By combining the linear sparsifications guaranteed by balanced constraint languages with the nontrivial sparsification when the largest-arity relations do not have exactly one falsifying assignment, we obtain an exact characterization of the optimal sparsification size for all Boolean CSPs where each relation has arity at most three. For a Boolean constraint language $\Gamma$ consisting of relations of arity at most three, we characterize the sparsification complexity of $\Gamma$ as an integer $k \in \{1,2,3\}$ that represents the largest OR that $\Gamma$ can implement using constants and negations, but without introducing fresh variables. Then we prove that CSP($\Gamma$) has a sparsification of size $\mathcal{O}(n^k)$, but no sparsification of size $\mathcal{O}(n^{k-\varepsilon})$ for any $\varepsilon > 0$, giving matching upper and lower bounds. Hence for all Boolean CSPs with constraints of arity at most three, the polynomial-based framework gives provably *optimal* sparsifications.

## 2 Preliminaries

For a positive integer $n$, define $[n] := \{1,2,\ldots,n\}$. For an integer $q$, we let $\mathbb{Z}/q\mathbb{Z}$ denote the integers modulo $q$. These form a field if $q$ is prime, and a ring otherwise. We will use $x \equiv_q y$ to denote that $x$ and $y$ are congruent modulo $q$, and $x \not\equiv_q y$ to denote that they are incongruent modulo $q$. For statements marked with a star (★), the (full) proof can be found in the full version [4].

**Parameterized complexity.** A parameterized problem $\mathcal{Q}$ is a subset of $\Sigma^* \times \mathbb{N}$, where $\Sigma$ is a finite alphabet. Let $\mathcal{Q}, \mathcal{Q}' \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems and let $h\colon \mathbb{N} \to \mathbb{N}$ be a computable function. A *generalized kernel for $\mathcal{Q}$ into $\mathcal{Q}'$ of size $h(k)$* is an algorithm that, on input $(x, k) \in \Sigma^* \times \mathbb{N}$, takes time polynomial in $|x| + k$ and outputs an instance $(x', k')$ such that: (i) $|x'|$ and $k'$ are bounded by $h(k)$, and (ii) $(x', k') \in \mathcal{Q}'$ if and only if $(x, k) \in \mathcal{Q}$. The algorithm is a *kernel* for $\mathcal{Q}$ if $\mathcal{Q}' = \mathcal{Q}$.

Since a polynomial-time reduction to an equivalent sparse instance yields a generalized kernel, lower bounds against generalized kernels can be used to prove the non-existence of such sparsification algorithms. To relate the sparsifiability of different problems to each other, the following notion is useful.

▶ **Definition 2.1.** Let $\mathcal{P}, \mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A *linear-parameter transformation* from $\mathcal{P}$ to $\mathcal{Q}$ is a polynomial-time algorithm that, given an instance $(x, k) \in \Sigma^* \times \mathbb{N}$ of $\mathcal{P}$, outputs an instance $(x', k') \in \Sigma^* \times \mathbb{N}$ of $\mathcal{Q}$ such that the following holds:

1. $(x, k) \in \mathcal{Q}$ if and only if $(x', k') \in \mathcal{P}$, and
2. $k' \in \mathcal{O}(k)$.

It is well-known [1, 2] that the existence of a linear-parameter transformation from problem $\mathcal{P}$ to $\mathcal{Q}$ implies that any generalized kernelization lower bound for $\mathcal{P}$, also holds for $\mathcal{Q}$.

**Operations, relations, and preservation.** A *Boolean operation* is a mapping from $\{0,1\}^k$ to $\{0,1\}$, where $k$, a natural number, is said to be the *arity* of the operation; we assume throughout that operations have positive arity. From here, we define a *partial Boolean operation* in the usual way, that is, it is a mapping from a subset of $\{0,1\}^k$ to $\{0,1\}$. We say that a partial Boolean operation $f$ of arity $k$ is *idempotent* if $f(0, \ldots, 0) = 0$ and $f(1, \ldots, 1) = 1$; and, *self-dual* if for all $(a_1, \ldots, a_k) \in \{0,1\}^k$, when $f(a_1, \ldots, a_k)$ is defined, it holds that $f(\neg a_1, \ldots, \neg a_k)$ is defined and $f(a_1, \ldots, a_k) = \neg f(\neg a_1, \ldots, \neg a_k)$.

▶ **Definition 2.2.** A partial Boolean operation $f\colon \{0,1\}^k \to \{0,1\}$ is *balanced* if there exist integer values $\alpha_1, \ldots, \alpha_k$, called the *coefficients* of $f$, such that

- $\sum_{i \in [k]} \alpha_i = 1$,
- $(x_1, \ldots, x_k)$ is in the domain of $f$ if and only if $\sum_{i \in [k]} \alpha_i x_i \in \{0,1\}$, and
- $f(x_1, \ldots, x_k) = \sum_{i \in [k]} \alpha_i x_i$ for all tuples in its domain.

A *relation* over the set $D$ is a subset of $D^k$; here, $k$ is a natural number called the *arity* of the relation. Throughout, we assume that each relation is over a finite set $D$. A *Boolean relation* is a relation over $\{0,1\}$.

▶ **Definition 2.3.** For each $k \geq 1$, we use $k$-OR to denote the relation $\{0,1\}^k \setminus \{(0, \ldots, 0)\}$.

A *constraint language over $D$* is a finite set of relations over $D$; a *Boolean constraint language* is a constraint language over $\{0,1\}$. For a Boolean constraint language $\Gamma$, we define $\mathsf{CSP}(\Gamma)$ as follows.

| $\mathsf{CSP}(\Gamma)$ | **Parameter:** The number of variables $\lvert V \rvert$. |
|---|---|
| **Input:** A tuple $(\mathcal{C}, V)$, where $\mathcal{C}$ is a finite set of constraints, $V$ is a finite set of variables, and each constraint is a pair $R(x_1, \ldots, x_k)$ for $R \in \Gamma$ and $x_1, \ldots, x_k \in V$. | |
| **Question:** Does there exist a *satisfying assignment*, that is, an assignment $f\colon V \to \{0,1\}$ such that for each constraint $R(x_1, \ldots, x_k) \in \mathcal{C}$ it holds that $(f(x_1), \ldots, f(x_k)) \in R$? | |

Let $f\colon \{0,1\}^k \to \{0,1\}$ be a partial Boolean operation, and let $T \subseteq \{0,1\}^n$ be a Boolean relation. We say that $T$ is *preserved* by $f$ when, for any tuples $t^1 = (t^1_1, \ldots, t^1_n), \ldots, t^k =$

$(t_1^k, \ldots, t_n^k) \in T$, if all entries of the tuple $(f(t_1^1, \ldots, t_1^k), \ldots, f(t_n^1, \ldots, t_n^k))$ are defined, then this tuple is in $T$. We say that a Boolean constraint language $\Gamma$ is *preserved* by $f$ if each relation in $\Gamma$ is preserved by $f$. We say that a Boolean relation is *balanced* if it is preserved by all balanced operations, and that a Boolean constraint language is *balanced* if each relation therein is balanced.

Define an *alternating operation* to be a balanced operation $f\colon \{0,1\}^k \to \{0,1\}$ such that $k$ is odd and the coefficients alternate between $+1$ and $-1$, so that $\alpha_1 = +1$, $\alpha_2 = -1$, $\alpha_3 = +1$, …, $\alpha_k = +1$. We have the following.

▶ **Proposition 2.4 (★).** *A Boolean relation $R$ is balanced if and only if for all odd $k \geq 1$, the relation $R$ is preserved by the alternating operation of arity $k$.*

We will use the following straightforwardly verified fact tacitly, throughout.

▶ **Observation 2.5.** *Each balanced operation is idempotent and self-dual.*

For $b \in \{0,1\}$, let $u_b\colon \{0,1\} \to \{0,1\}$ be the unary operation defined by $u_b(0) = u_b(1) = b$; let $\mathsf{major}\colon \{0,1\}^3 \to \{0,1\}$ to be the operation defined by $\mathsf{major}(x,y,z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$; and, let $\mathsf{minor}\colon \{0,1\}^3 \to \{0,1\}$ to be the operation defined by $\mathsf{minor}(x,y,z) = x \oplus y \oplus z$, where $\oplus$ denotes exclusive OR. We say that a Boolean constraint language $\Gamma$ is *tractable* if it is preserved by one of the six following operations: $u_0$, $u_1$, $\wedge$, $\vee$, $\mathsf{minor}$, $\mathsf{major}$; we say that $\Gamma$ is *intractable* otherwise. It is known that, in terms of classical complexity, the problem $\mathsf{CSP}(\Gamma)$ is polynomial-time decidable when $\Gamma$ is tractable, and that the problem $\mathsf{CSP}(\Gamma)$ is NP-complete when $\Gamma$ is intractable (see [3] for a proof; in particular, refer there to the proof of Theorem 3.21).

### Constraint Satisfaction and Definability.

▶ **Assumption 2.6.** *By default, we assume in the sequel that the operations, relations, and constraint languages under discussion are Boolean, and that said operations and relations are of positive arity. We nonetheless sometimes describe them as being Boolean, for emphasis.*

▶ **Definition 2.7.** Let us say that a Boolean relation $T$ of arity $m$ is *cone-definable* from a Boolean relation $U$ of arity $n$ if there exists a tuple $(y_1, \ldots, y_n)$ where:

- for each $j \in [n]$, it holds that $y_j$ is an element of $\{0,1\} \cup \{x_1, \ldots, x_m\} \cup \{\neg x_1, \ldots, \neg x_m\}$;
- for each $i \in [m]$, there exists $j \in [n]$ such that $y_j \in \{x_i, \neg x_i\}$; and,
- for each $f\colon \{x_1, \ldots, x_m\} \to \{0,1\}$, it holds that $(f(x_1), \ldots, f(x_m)) \in T$ if and only if $(\hat{f}(y_1), \ldots, \hat{f}(y_n)) \in U$. Here, $\hat{f}$ denotes the natural extension of $f$ where $\hat{f}(0) = 0$, $\hat{f}(1) = 1$, and $\hat{f}(\neg x_i) = \neg f(x_i)$.

(The prefix *cone* indicates the allowing of **co**nstants and **ne**gation.)

▶ **Example 2.8.** Let $R = \{(0,0),(0,1)\}$ and let $S = \{(0,1),(1,1)\}$. We have that $R$ is cone-definable from $S$ via the tuple $(\neg x_2, \neg x_1)$; also, $S$ is cone-definable from $R$ via the same tuple.

When $\Gamma$ is a constraint language over $D$, we use $\Gamma^*$ to denote the expansion of $\Gamma$ where each element of $D$ appears as a relation, that is, we define $\Gamma^*$ as $\Gamma \cup \{\{(d)\} \mid d \in D\}$.

The following is a key property of cone-definability; it states that relations that are cone-definable from a constraint language $\Gamma$ may be simulated by the constraint language, and thus used to prove hardness results for $\mathsf{CSP}(\Gamma)$.

▶ **Proposition 2.9 (★).** *Suppose that $\Gamma$ is an intractable constraint language, and that $\Delta$ is a constraint language such that each relation in $\Delta$ is cone-definable from a relation in $\Gamma$. Then, there exists a linear-parameter transformation from $\mathsf{CSP}(\Gamma^* \cup \Delta)$ to $\mathsf{CSP}(\Gamma)$.*

## 3 Trivial versus non-trivial sparsification

It is well known that $k$-CNF-SAT allows no non-trivial sparsification, for each $k \geq 3$ [7]. This means that we cannot efficiently reduce the number of clauses in such a formula to $\mathcal{O}(n^{k-\varepsilon})$. The $k$-OR relation is special, in the sense that there is exactly one $k$-tuple that is not contained in the relation. We show in this section that when considering $k$-ary relations for which there is more than one $k$-tuple not contained in the relation, a non-trivial sparsification is always possible. In particular, the number of constraints of any input can efficiently be reduced to $\mathcal{O}(n^{k-1})$. Using Lemmas 3.4 and 3.6, we will completely classify the constraint languages that allow a non-trivial sparsification as follows.

▶ **Theorem 3.1 (★).** *Let $\Gamma$ be an intractable (Boolean) constraint language. Let $k$ be the maximum arity of any relation $R \in \Gamma$. The following dichotomy holds.*

- *If for all $R \in \Gamma$ it holds that $|R| \neq 2^k - 1$, then $\mathsf{CSP}(\Gamma)$ has a kernel with $\mathcal{O}(n^{k-1})$ constraints that can be stored in $\mathcal{O}(n^{k-1} \log n)$ bits.*
- *If there exists $R \in \Gamma$ with $|R| = 2^k - 1$, then $\mathsf{CSP}(\Gamma)$ has no generalized kernel of bitsize $\mathcal{O}(n^{k-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

To obtain the kernels given in this section, we will heavily rely on the following notion for representing constraints by polynomials.

▶ **Definition 3.2.** Let $R$ be a $k$-ary Boolean relation. We say that a polynomial $p_u$ over a ring $E_u$ *captures* an unsatisfying assignment $u \in \{0,1\}^k \setminus R$ with respect to $R$, if the following two conditions hold over $E_u$.

$$p_u(x_1, \ldots, x_k) = 0 \text{ for all } (x_1, \ldots, x_k) \in R, \text{ and} \tag{1}$$

$$p_u(u_1, \ldots, u_k) \neq 0. \tag{2}$$

The following Theorem is a generalization of Theorem 16 in [13]. The main improvement is that we now allow the usage of different polynomials, over different rings, for each $u \notin R$. Previously, all polynomials had to be given over the same ring, and each constraint was captured by a single polynomial.

▶ **Theorem 3.3 (★).** *Let $R \subseteq \{0,1\}^k$ be a fixed $k$-ary relation, such that for every $u \in \{0,1\}^k \setminus R$ there exists a ring $E_u \in \{\mathbb{Q}\} \cup \{\mathbb{Z}/q_u\mathbb{Z} \mid q_u \text{ is a prime power}\}$ and polynomial $p_u$ over $E_u$ of degree at most $d$ that captures $u$ with respect to $R$. Then there exists a polynomial-time algorithm that, given a set of constraints $\mathcal{C}$ over $\{R\}$ over $n$ variables, outputs $\mathcal{C}' \subseteq \mathcal{C}$ with $|\mathcal{C}'| = \mathcal{O}(n^d)$, such that any Boolean assignment satisfies all constraints in $\mathcal{C}$ if and only if it satisfies all constraints in $\mathcal{C}'$.*

The next lemma states that any $k$-ary Boolean relation $R$ with $|R| < 2^k - 1$ admits a non-trivial sparsification. To prove the lemma, we show that such relations can be represented by polynomials of degree at most $k-1$, such that the sparsification can be obtained using Theorem 3.3. Since relations with $|R| = 2^k$ have a sparsification of size $\mathcal{O}(1)$, as constraints over such relations are satisfied by any assignment, it will follow that $k$-ary relations with $|\{0,1\}^k \setminus R| \neq 1$ always allow a non-trivial sparsification.

▶ **Lemma 3.4 (★).** *Let $R$ be a $k$-ary Boolean relation with $|R| < 2^k - 1$. Let $\mathcal{C}$ be a set of constraints over $\{R\}$, using $n$ variables. Then there exists a polynomial-time algorithm that outputs $\mathcal{C}' \subseteq \mathcal{C}$ with $|\mathcal{C}'| = \mathcal{O}(n^{k-1})$, such that a Boolean assignment satisfies all constraints in $\mathcal{C}'$ if and only if it satisfies all constraints in $\mathcal{C}$.*

**Proof sketch.** We will show that for every $u \in \{0,1\}^k \setminus R$, there exists a degree-$(k-1)$ polynomial $p_u$ over $\mathbb{Q}$ that captures $u$, such that the result follows from Theorem 3.3. We will prove the existence of such a polynomial by induction on $k$. For $k = 1$, the lemma statement implies that $R = \emptyset$. Thereby, for any $u \notin R$, we simply choose $p_u(x_1) := 1$. This polynomial satisfies the requirements, and has degree 0. Let $k > 1$ and let $u = (u_1, \ldots, u_k) \in \{0,1\}^k \setminus R$. Since $|R| < 2^k - 1$, we can choose $w = (w_1, \ldots, w_k)$ such that $w \in \{0,1\}^k \setminus R$ and $w \neq u$. We distinguish two cases, depending on whether $u$ and $w$ agree on some position.

Suppose $u_i \neq w_i$ for all $i$, and assume for concreteness that $u = (0, \ldots, 0)$ and $w = (1, \ldots, 1)$. Then the polynomial $p_u(x_1, \ldots, x_k) := \prod_{i=1}^{k-1}(i - \sum_{j=1}^{k} x_j)$ suffices: $p_u(0, \ldots, 0) = \prod_{j=1}^{k-1} j \neq 0$, while for any $(x_1, \ldots, x_k) \in R$, it holds that $\sum_{i=1}^{k} x_i \in [k-1]$ and thereby $p_u(x_1, \ldots, x_k) = 0$; the product has a 0-term. Other values of $u$ and $w$ are handled similarly.

Now suppose $u_i = w_i$ for some $i \in [k]$, and assume for concreteness that $u_1 = w_1 = 1$. Define $R' := \{(x_2, \ldots, x_k) \mid (1, x_2, \ldots, x_k) \in R\}$ and let $u' := (u_2, \ldots, u_k)$. Since $(u_2, \ldots, u_k)$ and $(w_2, \ldots, w_k)$ are distinct tuples not in $R'$, by induction there is a polynomial $p_{u'}$ of degree $k-2$ that captures $u'$ with respect to $R'$. Then the polynomial $p_u(x_1, \ldots, x_k) := x_1 \cdot p_{u'}(x_2, \ldots, x_k)$ has degree $k-1$ and captures $u$ with respect to $R$. ◀

To show the other part of the dichotomy, we will need the following theorem.

▶ **Theorem 3.5 (★).** *Let $\Gamma$ be an intractable (Boolean) constraint language, and let $k \geq 1$. If there exists $R \in \Gamma$ such that $R$ cone-defines $k$-OR, then* CSP($\Gamma$) *does not have a generalized kernel of size $\mathcal{O}(n^{k-\varepsilon})$, unless* NP $\subseteq$ coNP/poly.

The next lemma formalizes the idea that any $k$-ary relation with $|\{0,1\}^k \setminus R| = 1$ is equivalent to $k$-OR, up to negation of variables. The proof of the dichotomy given in Theorem 3.1 will follow from Lemma 3.4, together with the next lemma and Theorem 3.5.

▶ **Lemma 3.6 (★).** *Let $R$ be a $k$-ary relation with $|R| = 2^k - 1$. Then $R$ cone-defines $k$-OR.*

## 4 From balanced operations to linear sparsification

The main result of this section is the following theorem, which we prove below.

▶ **Theorem 4.1.** *Let $\Gamma$ be a balanced (Boolean) constraint language. Then* CSP($\Gamma$) *has a kernel with $\mathcal{O}(n)$ constraints that are a subset of the original constraints. The kernel can be stored using $\mathcal{O}(n \log n)$ bits.*

To prove the theorem, we will use two additional technical lemmas. To state them, we introduce some notions from linear algebra. Given a set $S = \{s_1, \ldots, s_n\}$ of $k$-ary vectors in $\mathbb{Z}^k$, we define $\text{span}_{\mathbb{Z}}(S)$ as the set of all vectors $y$ in $\mathbb{Z}^k$ for which there exist $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}$ such that $y = \sum_{i \in [n]} \alpha_i s_i$. Similarly, we define $\text{span}_q(S)$ as the set of all $k$-ary vectors $y$ over $\mathbb{Z}/q\mathbb{Z}$, such that there exist $\alpha_1, \ldots, \alpha_n$ such that $y \equiv_q \sum_{i \in [n]} \alpha_i s_i$. For an $m \times n$ matrix $S$, we use $s_i$ for $i \in [m]$ to denote the $i$'th row of $S$.

▶ **Lemma 4.2 (★).** *Let $S$ be an $m \times n$ integer matrix. Let $u \in \mathbb{Z}^n$ be a row vector. If $u \in \text{span}_q(\{s_1, \ldots, s_m\})$ for all prime powers $q$, then $u \in \text{span}_{\mathbb{Z}}(\{s_1, \ldots, s_m\})$.*

▶ **Lemma 4.3 (★).** *Let $q$ be a prime power. Let $A$ be an $m \times n$ matrix over $\mathbb{Z}/q\mathbb{Z}$. Suppose there exists no constant $c \not\equiv_q 0$ for which the system $Ax \equiv_q b$ has a solution, where $b := (0, \ldots, 0, c)^T$ is the vector with $c$ on the last position and zeros in all other positions.*

*Then $a_m \in \text{span}_q(\{a_1, \ldots, a_{m-1}\})$.*

Using these tools from linear algebra, we now prove the main sparsification result.

**Proof of Theorem 4.1.** We show that for all relations $R$ in the balanced constraint language $\Gamma$, for all $u \notin R$, there exists a linear polynomial $p_u$ over a ring $E_u \in \{\mathbb{Z}/q_u\mathbb{Z} \mid q_u \text{ is a prime power}\}$ that captures $u$ with respect to $R$. By applying Theorem 3.3 once for each relation $R \in \Gamma$, to reduce the number of constraints involving $R$ to $\mathcal{O}(n)$, we then reduce any $n$-variable instance of $\mathsf{CSP}(\Gamma)$ to an equivalent one on $|\Gamma| \cdot \mathcal{O}(n) \in \mathcal{O}(n)$ constraints.

Suppose for a contradiction that there exists $R \in \Gamma$ and $u \notin R$, such that no prime power $q$ and polynomial $p$ over $\mathbb{Z}/q\mathbb{Z}$ exist that satisfy conditions (1) and (2). We can view the process of finding such a linear polynomial, as solving a set of linear equations whose unknowns are the coefficients of the polynomial. We have a linear equation for each evaluation of the polynomial for which we want to enforce a certain value.

Let $R = \{r_1, \ldots, r_\ell\}$. By the non-existence of $p$ and $q$, the system

$$\begin{pmatrix} 1 & r_{1,1} & r_{1,2} & \ldots & r_{1,k} \\ 1 & r_{2,1} & r_{2,2} & \ldots & r_{2,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & r_{\ell,1} & r_{\ell,2} & \ldots & r_{\ell,k} \\ 1 & u_1 & u_2 & \ldots & u_k \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{pmatrix} \equiv_q \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ c \end{pmatrix}$$

has no solution for any prime power $q$ and $c \not\equiv_q 0$. Otherwise, it is easy to verify that $q$ is the desired prime power and $p(x_1, \ldots, x_k) := \alpha_0 + \sum_{i=1}^{k} \alpha_i x_i$ is the desired polynomial.

The fact that no solution exists, implies that $(1, u_1, \ldots, u_k)$ is in the span of the remaining rows of the matrix, by Lemma 4.3. But this implies that for any prime power $q$, there exist coefficients $\beta_1, \ldots, \beta_\ell$ over $\mathbb{Z}/q\mathbb{Z}$ such that $u \equiv_q \sum \beta_i r_i$. Furthermore, since the first column of the matrix is the all-ones column, we obtain that $\sum \beta_i \equiv_q 1$. By Lemma 4.2, it follows that there exist integer coefficients $\gamma_1, \ldots, \gamma_\ell$ such that $\sum \gamma_i = 1$ and furthermore $u = \sum \gamma_i r_i$. But it immediately follows that $R \in \Gamma$ is not preserved by the balanced operation given by $f(x_1, \ldots, x_\ell) := \sum \gamma_i x_i$, which contradicts the assumption that $\Gamma$ is balanced. ◀

The kernelization result above is obtained by using the fact that when $\Gamma$ is balanced, the constraints in $\mathsf{CSP}(\Gamma)$ can be replaced by linear polynomials. We show in the next theorem that this approach fails when $\Gamma$ is not balanced.

▶ **Theorem 4.4 (★).** *Let $R$ be a $k$-ary relation that is not balanced. Then there exists $u \in \{0,1\}^k \setminus R$ for which there exists no polynomial $p_u$ over any ring $E$ that captures $u$ with respect to $R$.*

## 5 Characterization of symmetric CSPs with linear sparsification

In this section, we characterize the symmetric constraint languages $\Gamma$ for which $\mathsf{CSP}(\Gamma)$ has a linear sparsification.

▶ **Definition 5.1.** We say a $k$-ary Boolean relation $R$ is *symmetric*, if there exists $S \subseteq \{0, 1, \ldots, k\}$ such that a tuple $x = (x_1, \ldots, x_k)$ is in $R$ if and only if $\text{weight}(x) \in S$. We call $S$ the set of *satisfying weights* for $R$.

We will say that a constraint language $\Gamma$ is symmetric, if it only contains symmetric relations. We will prove the following theorem at the end of this section.

▶ **Theorem 5.2.** *Let $\Gamma$ be a finite Boolean symmetric intractable constraint language.*

- *If $\Gamma$ is balanced, then* $\mathsf{CSP}(\Gamma)$ *has a kernel with $\mathcal{O}(n)$ constraints that can be stored in $\mathcal{O}(n \log n)$ bits.*
- *If $\Gamma$ is not balanced, then* $\mathsf{CSP}(\Gamma)$ *does not have a generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

To show this, we use the following lemma.

▶ **Lemma 5.3 (★).** *Let $R$ be a $k$-ary symmetric relation with satisfying weights $S \subseteq \{0, 1, \ldots, k\}$. Let $U := \{0, 1, \ldots, k\} \setminus S$. If there exist $a, b, c \in S$ and $d \in U$ such that $a - b + c = d$, then $R$ cone-defines* 2-OR.

**Proof sketch.** We will demonstrate the result in the case that $b \leq a$, $b \leq c$, and $b \leq d$; the other cases are similar. We use the following tuple to express $x_1 \vee x_2$.

$$(\underbrace{\neg x_1, \ldots, \neg x_1}_{(a-b)\text{ copies}}, \underbrace{\neg x_2, \ldots, \neg x_2}_{(c-b)\text{ copies}}, \underbrace{1, \ldots, 1}_{b\text{ copies}}, \underbrace{0, \ \ldots \ , 0}_{(k-d)\text{ copies}}).$$

Let $f \colon \{x_1, x_2\} \to \{0, 1\}$, then $(\neg f(x_1), \ldots, \neg f(x_1), \neg f(x_2), \ldots, \neg f(x_2), 1, \ldots, 1, 0, \ldots, 0)$ has weight $d \notin S$ when $f(x_1) = f(x_2) = 0$. It is easy to verify that in all other cases, the weight is one of $a, b, c \in S$ and hence the tuple belongs to $R$. The other cases are similar. ◀

We now give the main lemma that is needed to prove Theorem 5.2. It shows that if a relation is symmetric and not balanced, it must cone-define 2-OR.

▶ **Lemma 5.4.** *Let $R$ be a symmetric (Boolean) relation of arity $k$. If $R$ is not balanced, then $R$ cone-defines* 2-OR.

**Proof.** Let $f$ be a balanced operation that does not preserve $R$. Since $f$ has integer coefficients, it follows that there exist (not necessarily distinct) $r_1, \ldots, r_m \in R$, such that $r_1 - r_2 + r_3 - r_4 \cdots + r_m = u$ for some $u \in \{0, 1\}^k \setminus R$ and odd $m \geq 3$. Thereby, $\text{weight}(r_1) - \text{weight}(r_2) + \text{weight}(r_3) - \text{weight}(r_4) \cdots + \text{weight}(r_m) = \text{weight}(u)$. Let $S$ be the set of satisfying weights for $R$ and let $U := \{0, \ldots, k\} \setminus S$. Define $s_i := \text{weight}(r_i)$ for $i \in [m]$, and $t = \text{weight}(u)$, such that $s_1 - s_2 + s_3 - s_4 \ldots + s_m = t$, and furthermore $s_i \in S$ for all $i$, and $t \in U$. We show that there exist $a, b, c \in S$ and $d \in U$ such that $a - b + c = d$, such that the result follows from Lemma 5.3. We do this by induction on the length of the alternating sum.

If $m = 3$, we have that $s_1 - s_2 + s_3 = t$ and define $a := s_1$, $b := s_2$, $c := s_3$, and $d := t$.

If $m > 3$, we will use the following claim.

▶ **Claim 5.5 (★).** *Let $s_1, \ldots, s_m \in S$ and $t \in U$ such that $s_1 - s_2 + s_3 - s_4 \cdots + s_m = t$. There exist distinct $i, j, \ell \in [m]$ with $i, j$ odd and $\ell$ even, such that $s_i - s_\ell + s_j \in \{0, \ldots, k\}$.*

Use Claim 5.5 to find $i, j, \ell$ such that $s_i - s_\ell + s_j \in \{0, \ldots, k\}$. We consider two options. If $s_i - s_\ell + s_j \in U$, then define $d := s_i - s_\ell + s_j$, $a := s_i$, $b := s_\ell$, and $c := s_j$ and we are done. The other option is that $s_i - s_\ell + s_j = s \in S$. Replacing $s_i - s_\ell + s_j$ by $s$ in $s_1 - s_2 + s_3 - s_4 \cdots + s_m$ gives a shorter alternating sum with result $t$. We obtain $a, b, c$, and $d$ by the induction hypothesis.

Thereby, we have obtained $a, b, c \in S$, $d \in U$ such that $a - b + c = d$. It now follows from Lemma 5.3 that $R$ cone-defines 2-OR. ◀

Using the lemma above, we can now prove Theorem 5.2.

**Proof of Theorem 5.2.** If $\Gamma$ is balanced, it follows from Theorem 4.1 that $\mathsf{CSP}(\Gamma)$ has a kernel with $\mathcal{O}(n)$ constraints that can be stored in $\mathcal{O}(n \log n)$ bits. Note that the assumption that $\Gamma$ is symmetric is not needed in this case.

If the symmetric constraint language $\Gamma$ is not balanced, then $\Gamma$ contains a symmetric relation $R$ that is not balanced. It follows from Lemma 5.4 that $R$ cone-defines the 2-OR relation. Thereby, we obtain from Theorem 3.5 that $\mathsf{CSP}(\Gamma)$ has no generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. ◀

## 6 Low-arity classification

In this section, we will give a full classification of the sparsifiability for constraint languages that consist only of low-arity relations. The next results will show that in this case, if the constraint language is not balanced, it can cone-define the 2-OR relation.

▶ **Observation 6.1.** *Each relation of arity* 1 *is balanced.*

▶ **Theorem 6.2 (★).** *A relation of arity* 2 *is balanced if and only if it is not cone-interdefinable with the* 2-OR *relation.*

▶ **Theorem 6.3 (★).** *Suppose that $U \subseteq \{0,1\}^3$ is an arity* 3 *Boolean relation that is not balanced. Then, the* 2-OR *relation is cone-definable from $U$.*

Combining the results in this section with the results in previous sections, allows us to give a full classification of the sparsifiability of constraint languages that only contain relations of arity at most three. Observe that any $k$-ary relation $R$ such that $R \neq \emptyset$ and $\{0,1\}^k \setminus R \neq \emptyset$ cone-defines the 1-OR relation. Since we assume that $\Gamma$ is intractable in the next theorem, it follows that $k$ is always defined and $k \in \{1,2,3\}$.

▶ **Theorem 6.4.** *Let $\Gamma$ be an intractable Boolean constraint language such that each relation therein has arity $\leq 3$. Let $k \in \mathbb{N}$ be the largest value for which $k$-OR can be cone-defined from a relation in $\Gamma$. Then $\mathsf{CSP}(\Gamma)$ has a kernel with $\mathcal{O}(n^k)$ constraints that can be encoded in $\mathcal{O}(n^k \log k)$ bits, but for any $\varepsilon > 0$ there is no kernel of size $\mathcal{O}(n^{k-\varepsilon})$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

**Proof.** To show that there is a kernel with $\mathcal{O}(n^k)$ constraints, we do a case distinction on $k$.

- ($k = 1$) If $k = 1$, there is no relation in $\Gamma$ that cone-defines the 2-OR relation. It follows from Observation 6.1 and Theorems 6.2 and 6.3 that thereby, $\Gamma$ is balanced. It now follows from Theorem 4.1 that $\mathsf{CSP}(\Gamma)$ has a kernel with $\mathcal{O}(n)$ constraints that can be stored in $\mathcal{O}(n \log n)$ bits.
- ($k = 2$) If $k = 2$, there is no relation $R \in \Gamma$ with $|R| = 2^3 - 1 = 7$, as otherwise by Lemma 3.6 such a relation $R$ would cone-define 3-OR which is a contradiction. Thereby, it follows from Theorem 3.1 that $\mathsf{CSP}(\Gamma)$ has a sparsification with $\mathcal{O}(n^{3-1}) = \mathcal{O}(n^2)$ constraints that can be encoded in $\mathcal{O}(n^2 \log n)$ bits.
- ($k = 3$) Given an instance $(\mathcal{C}, V)$, it is easy to obtain a kernel of with $\mathcal{O}(n^3)$ constraints by simply removing duplicate constraints. This kernel can be stored in $\mathcal{O}(n^3)$ bits, by storing for each relation $R \in \Gamma$ and for each tuple $(x_1, x_2, x_3) \in V^3$ whether $R(x_1, x_2, x_3) \in \mathcal{C}$. Since $|\Gamma|$ is constant and there are $\mathcal{O}(n^3)$ such tuples, this results in using $\mathcal{O}(n^3)$ bits.

It remains to prove the lower bound. By definition, there exists $R \in \Gamma$ such that $R$ cone-defines the $k$-OR relation. Thereby, the result follows immediately from Theorem 3.5. Thus, $\mathsf{CSP}(\Gamma)$ has no kernel of size $\mathcal{O}(n^{k-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. ◀

## 7 Conclusion

In this paper we analyzed the best-case and worst-case sparsifiability of $\mathsf{CSP}(\Gamma)$ for intractable finite Boolean constraint languages $\Gamma$. First of all, we characterized those Boolean CSPs for which a nontrivial sparsification is possible, based on the number of non-satisfying assignments. Then we presented our key structural contribution: the notion of balanced constraint languages. We have shown that $\mathsf{CSP}(\Gamma)$ allows a sparsification with $\mathcal{O}(n)$ constraints whenever $\Gamma$ is balanced. The constructive proof of this statement can be transformed into an effective algorithm to find a series of low-degree polynomials to capture the constraints, which earlier had to be done by hand. By combining the resulting upper and lower bound framework, we fully classified the symmetric constraint languages for which $\mathsf{CSP}(\Gamma)$ allows a linear sparsification. Furthermore, we fully classified the sparsifiability of $\mathsf{CSP}(\Gamma)$ when $\Gamma$ contains relations of arity at most three, based on the arity of the largest OR that can be cone-defined from $\Gamma$. It follows from results of Lagerkvist and Wahlström [15] that for constraint languages of arbitrary arity, the exponent of the best sparsification size does not always match the arity of the largest OR cone-definable from $\Gamma$. (This will be described in more detail in the upcoming journal version of this work.) Hence the type of characterization we presented is inherently limited to low-arity constraint languages. It may be possible to extend our characterization to languages of arity at most four, however.

The ultimate goal of this line of research is to fully classify the sparsifiability of $\mathsf{CSP}(\Gamma)$, depending on $\Gamma$. In particular, we would like to classify those $\Gamma$ for which $\mathcal{O}(n)$ sparsifiability is possible. In this paper, we have shown that $\Gamma$ being balanced is a sufficient condition to obtain a linear sparsification; it is tempting to conjecture that this condition is also necessary.

We conclude with a brief discussion on the relation between our polynomial-based framework for linear compression and the framework of Lagerkvist and Wahlström [15]. They used a different method for sparsification, based on embedding a Boolean constraint language $\Gamma$ into a constraint language $\Gamma'$ defined over a larger domain $D$, such that $\Gamma'$ is preserved by a Maltsev operation. This latter condition ensures that $\mathrm{CSP}(\Gamma')$ is polynomial-time solvable, which allows $\mathrm{CSP}(\Gamma)$ to be sparsified to $\mathcal{O}(n)$ constraints when $D$ is finite. It turns out that the Maltsev-based linear sparsification is more general than the polynomial-based linear sparsification presented here: all finite Boolean constraint languages $\Gamma$ that are balanced, admit a Maltsev embedding over a finite domain (the direct sum of the rings $\mathbb{Z}/q_u\mathbb{Z}$ over which the capturing polynomials are defined) and can therefore be linearly sparsified using the algorithm of Lagerkvist and Wahlström. Despite the fact that our polynomial-based framework is not more general than the Maltsev-based approach, it has two distinct advantages. First of all, there is a straight-forward decision procedure to determine whether a constraint can be captured by degree-1 polynomials, which follows from the proof of Theorem 4.1. To the best of our knowledge, no decision procedure is known to determine whether a Boolean constraint language admits a Maltsev embedding over a finite domain. The second advantage of our method is that when the polynomial framework for linear compression does not apply, this is witnessed by a relation in $\Gamma$ that is violated by a balanced operation. As we have shown, in several scenarios this violation can be used to construct a sparsification lower bound to give provably optimal bounds.

It would be interesting to determine whether the Maltsev-based framework for sparsification is strictly more general than the polynomial-based framework. We are not aware of any concrete Boolean constraint language $\Gamma$ for which $\mathrm{CSP}(\Gamma)$ admits a Maltsev embedding over a finite domain, yet is not balanced.

## References


1. Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization Lower Bounds by Cross-Composition. *SIAM J. Discrete Math.*, 28(1):277–305, 2014. doi:10.1137/120880240.
2. Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.*, 412(35):4570–4578, 2011. doi:10.1016/j.tcs.2011.04.039.
3. Hubie Chen. A Rendezvous of Logic, Complexity, and Algebra. *ACM Computing Surveys*, 42(1), 2009. doi:10.1145/1189056.1189076.
4. Hubie Chen, Bart M. P. Jansen, and Astrid Pieterse. Best-case and Worst-case Sparsifiability of Boolean CSPs. *CoRR*, abs/1809.06171v1, 2018. arXiv:1809.06171v1.
5. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
6. Holger Dell and Dániel Marx. Kernelization of packing problems. In *Proc. 23rd SODA*, pages 68–81, 2012. doi:10.1137/1.9781611973099.6.
7. Holger Dell and Dieter van Melkebeek. Satisfiability Allows No Nontrivial Sparsification unless the Polynomial-Time Hierarchy Collapses. *J. ACM*, 61(4):23:1–23:27, 2014. doi:10.1145/2629620.
8. Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.
9. Bart M. P. Jansen. On Sparsification for Computing Treewidth. *Algorithmica*, 71(3):605–635, 2015. doi:10.1007/s00453-014-9924-2.
10. Bart M. P. Jansen and Astrid Pieterse. Optimal Sparsification for Some Binary CSPs Using Low-Degree Polynomials. In *Proc. 41st MFCS*, pages 71:1–71:14, 2016. doi:10.4230/LIPIcs.MFCS.2016.71.
11. Bart M. P. Jansen and Astrid Pieterse. Optimal Data Reduction for Graph Coloring Using Low-Degree Polynomials. In *Proc. 12th IPEC*, pages 22:1–22:12, 2017. doi:10.4230/LIPIcs.IPEC.2017.22.
12. Bart M. P. Jansen and Astrid Pieterse. Sparsification Upper and Lower Bounds for Graph Problems and Not-All-Equal SAT. *Algorithmica*, 79(1):3–28, 2017. doi:10.1007/s00453-016-0189-9.
13. Bart M. P. Jansen and Astrid Pieterse. Optimal Sparsification for Some Binary CSPs Using Low-Degree Polynomials. *CoRR*, abs/1606.03233v2, 2018. arXiv:1606.03233v2.
14. Stefan Kratsch, Geevarghese Philip, and Saurabh Ray. Point Line Cover: The Easy Kernel is Essentially Tight. *ACM Trans. Algorithms*, 12(3):40:1–40:16, 2016. doi:10.1145/2832912.
15. Victor Lagerkvist and Magnus Wahlström. Kernelization of Constraint Satisfaction Problems: A Study Through Universal Algebra. In *Proc. 23rd CP*, pages 157–171, 2017. doi:10.1007/978-3-319-66158-2_11.
16. Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Kernelization - Preprocessing with a Guarantee. In *The Multivariate Algorithmic Revolution and Beyond*, pages 129–161, 2012. doi:10.1007/978-3-642-30891-8_10.
17. Lásló Lovász. Chromatic number of hypergraphs and linear algebra. In *Studia Scientiarum Mathematicarum Hungarica 11*, pages 113–114, 1976.
18. Thomas J. Schaefer. The Complexity of Satisfiability Problems. In *Proc. 10th STOC*, pages 216–226, 1978. doi:10.1145/800133.804350.

# Parameterized Leaf Power Recognition via Embedding into Graph Products

**David Eppstein**[1]
Computer Science Department, University of California, Irvine, USA
eppstein@uci.edu

**Elham Havvaei**
Computer Science Department, University of California, Irvine, USA
ehavvaei@uci.edu

**Abstract**

The $k$-leaf power graph $G$ of a tree $T$ is a graph whose vertices are the leaves of $T$ and whose edges connect pairs of leaves at unweighted distance at most $k$ in $T$. Recognition of the $k$-leaf power graphs for $k \geq 6$ is still an open problem. In this paper, we provide an algorithm for this problem for sparse leaf power graphs. Our result shows that the problem of recognizing these graphs is fixed-parameter tractable when parameterized both by $k$ and by the degeneracy of the given graph. To prove this, we describe how to embed the leaf root of a leaf power graph into a product of the graph with a cycle graph. We bound the treewidth of the resulting product in terms of $k$ and the degeneracy of $G$. As a result, we can use methods based on monadic second-order logic ($\mathrm{MSO}_2$) to recognize the existence of a leaf power as a subgraph of the product graph.

**2012 ACM Subject Classification** Theory of computation → Design and analysis of algorithms, Theory of computation → Fixed parameter tractability

**Keywords and phrases** leaf power, phylogenetic tree, monadic second-order logic, Courcelle's theorem, strong product of graphs, fixed-parameter tractability

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.16

## 1 Introduction

*Leaf powers* are a class of graphs that were introduced in 2002 by Nishimura, Ragde and Thilikos [29], extending the notion of graph powers. For a graph $G$, the $k$th power graph $G^k$ has the same set of vertices as $G$ but a different notion of adjacency: two vertices are adjacent in $G^k$ if there is a path of at most $k$ edges between them in $G$. The leaf powers are defined in the same way from trees, but only including the leaves of the trees as vertices. The $k$th leaf power of a tree $T$ has the leaves of $T$ as its vertices, with two vertices adjacent in the leaf power if there is a path of at most $k$ edges between them in $T$. A given graph $G$ is a $k$-leaf-power graph when there exists a tree $T$ for which $G$ is the $k$th leaf power. In this case, $T$ is called the *$k$-leaf root* of $G$. For example, Figure 1 shows a 3-leaf power alongside its 3-leaf root. Nishimura et al., further, derived the first polynomial-time algorithms to recognize $k$-leaf powers for $k = 3$ and $k = 4$ [29].

The problem of recognizing leaf powers arises as a formalization of a problem in computational biology, the reconstruction of evolutionary history and evolutionary trees from information about the similarity between species [12, 21]. In this problem, the common ancestry of different species can be represented by an evolutionary or phylogenetic tree, in

[1] Supported in part by NSF grants CCF-1618301 and CCF-1616248.

© David Eppstein and Elham Havvaei;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 16; pp. 16:1–16:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

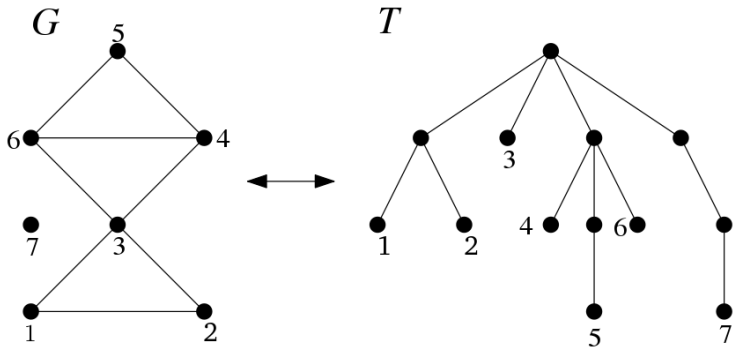



**Figure 1** A 3-leaf power graph $G$ and its 3-leaf root $T$.

which each vertex represents a species and each edge represents a direct ancestry relation between two species. We only have full access to living species, the species at the leaves of the tree; the other species in the tree are typically long-extinct, and may be represented physically only through fossils or not at all. If we suppose that we can infer, from observations of living species, which ones are close together (within some number $k$ of steps in this tree) and which others are not, then we could use an algorithm for leaf power recognition to infer a phylogenetic tree consistent with this data.

## 1.1 New Results

In this paper, we prove that the $k$-leaf powers of degeneracy $d$ can be recognized in time that is fixed-parameter tractable when parameterized by $k$ and $d$. Here, the degeneracy of a graph is the maximum, over its subgraphs, of the minimum degree of any subgraph. Our result provides an algorithm whose running time is polynomial (in fact linear) in the size of its input graph, multiplied by a factor that depends non-polynomially on $k$ and $d$.

It is known that the $k$-leaf powers have bounded clique-width for bounded $k$ [24]. A wide class of graph problems (those expressible in a version of monadic second order logic quantifying over only vertex sets, $\mathrm{MSO}_1$) can be solved in fixed-parameter time for graphs of bounded clique-width, via Courcelle's theorem [16]. However we have been unable to express the recognition of leaf powers in $\mathrm{MSO}_1$. Instead, our algorithm uses a more powerful version of monadic second order logic allowing quantification over edge sets, $\mathrm{MSO}_2$. The graphs of bounded clique-width and bounded degeneracy have bounded treewidth, allowing us to apply a form of Courcelle's theorem for $\mathrm{MSO}_2$ and for graphs of bounded treewidth.

However, there is an additional complication that makes it tricky to apply these methods to leaf power recognition. The tree that we wish to find, for which our given input graph is a leaf power, will in general include vertices and edges that are not part of the input, but $\mathrm{MSO}_2$ can only quantify over subsets of the existing vertices and edges of a graph, not over sets of vertices and edges that are not subsets of the input. To work around this problem, we apply Courcelle's theorem not to the given graph $G$ itself, but to a *product graph* $G \boxtimes C_k$ where $C_k$ is a $k$-vertex cycle graph. We prove that the leaf root (the tree for which $G$ is a leaf power, if it is one) can be embedded as a subgraph of this product, that it can be recognized by an $\mathrm{MSO}_2$ formula applied to this product, and that this product has bounded treewidth whenever $G$ is a $k$-leaf power of bounded degeneracy. In this way we can recognize $G$ as a leaf power, not by applying Courcelle's theorem to $G$, but by applying it to the product graph.

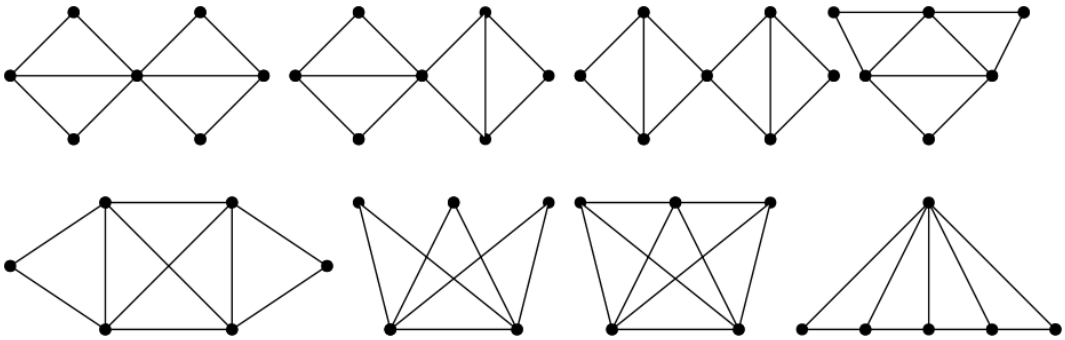


**Figure 2** A graph is a 4-leaf power if and only if it is chordal and does not contain any of the graphs above as a subgraph.

Thus, our algorithm combines the following ingredients:

- Our embedding of the $k$-leaf root as a subgraph of the product graph $G \boxtimes C_k$.
- Our logical representation of $k$-leaf roots as subgraphs of product graphs.
- Courcelle's theorem, which provides general-purpose algorithms for testing $\mathrm{MSO}_2$ formulas on graphs of bounded treewidth.
- The fact that the $k$-leaf powers, for bounded values of $k$, have bounded clique-width.
- The fact that graphs of bounded clique-width and bounded degeneracy also have bounded treewidth.
- The fact that, by taking a product with a graph of bounded size, we preserve the bounded treewidth of the product.

Our algorithm runs in fixed-parameter tractable time when parameterized by $k$ and the degeneracy $d$ of the given input graph. In particular, it runs in linear-time when $k$ and $d$ are both constant.

Our result provides the first known efficient algorithm for recognizing $k$-leaf powers for $k \geq 6$, for graphs of bounded degeneracy. More generally, our method of embedding into graph products appears likely to apply to other graph problems involving network design (the addition of edges to an existing graph, rather than the identification of a special subgraph of the input). In the case we apply this method to leaf power recognition, we expect that it should be possible to translate our $\mathrm{MSO}_2$ formula over the product graph into a significantly more complicated $\mathrm{MSO}_2$ formula over the input graph, but the method of embedding into graph products considerably simplifies our task of designing a logical formula for our problem.

## 1.2 Related Work

Polynomial-time algorithms are known for recognizing $k$-leaf powers for $k \geq 5$.

- A graph is a 2-leaf power if it is a disjoint union of cliques, so this class of graphs is trivial to recognize.
- There exist various ways to characterize 3-leaf powers [29, 7, 18, 30], some of which lead to efficient algorithms. For instance, one way to determine if a graph is a 3-leaf power is to check whether it is bull-, dart- and gem-free and chordal [18]. The chordal graphs have a known recognition algorithm, and testing for the existence of any of the other forbidden induced subgraphs is polynomial, because they all have bounded size.
- Similarly, there are various known ways to characterize 4-leaf powers [29, 30, 19, 9]. One is that graph is a 4-leaf power if and only if it is chordal and does not contain any

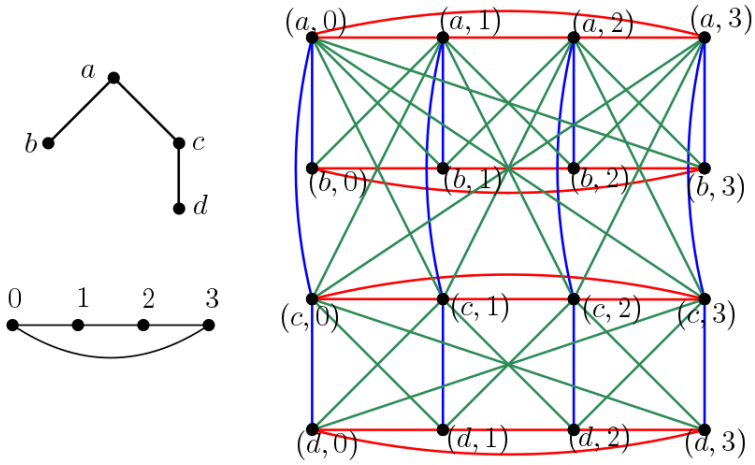



**Figure 3** The graph on the right is the strong product of a four-vertex path graph (top left) and a four-vertex cycle graph (bottom left). The colors indicate the partition of the edges into vertical, horizontal, and diagonal subsets.

of the graphs depicted in Figure 2 as induced subgraphs [30]. Again, this leads to a polynomial-time recognition algorithm, because all of these graphs have bounded size.

- $k$-leaf powers can be recognized in polynomial time if the $(k-2)$- Steiner root problem can be solved in polynomial time. Chang and Ko in 2007, provided a linear-time algorithm recognition for 3-Steiner root problem [11]. This implies that 5-leaf powers can be recognized in linear time.

Polynomial-time structural characterization of $k$-leaf powers for $k \geq 6$ is still an open problem. Nonetheless, Brandstädt et al. provided a characterization for the distance-hereditary 5-leaf powers [8].

Throughout the literature, there exist many structural characterizations of leaf powers which provide potentially useful insight into this class of graphs. It is known, for instance, that all leaf powers are strongly chordal, but the converse is not always true. Further, Kennedy et al. showed that strictly chordal graphs are always $k$-leaf powers for $k \geq 4$; these are the chordal graphs that are also, dart- and gem-free. They provided a linear-time algorithm to construct $k$-leaf roots of strictly chordal graphs [27].

For all $k \geq 2$, every $k$-leaf power is also a $(k+2)$-leaf power. A $(k+2)$-leaf root of any $k$-leaf-power can be obtained from its $k$-leaf root, by subdividing all edges incident to leaves. However, the problems of recognizing $k$-leaf powers for different values of $k$ do not collapse: for all $k \geq 4$, there exists a $k$-leaf power which is not a $(k+1)$-leaf power [10].

# 2 Preliminaries

## 2.1 Definitions

Throughout this paper, we let $G(V, E)$ denote a simple undirected graph (typically, the input to the leaf power recognition problem). If $u$ and $v$ are two vertices in $V$ that are adjacent in $G$, we let $e(u, v)$ denote the edge connecting them.

The strong product of graphs $G_1$ and $G_2$, denoted as $G_1 \boxtimes G_2$, is a graph whose vertices are ordered pairs of a vertex from $G_1$ and a vertex from $G_2$. In it, two distinct vertices $(u_1, u_2)$ and $(v_1, v_2)$ are adjacent if and only if for all $i \in \{1, 2\}$, $u_i = v_i$ or $u_i$ and $v_i$ are adjacent in $G_i$. Figure 3 shows an example, the strong product of a four-vertex path graph

with a four-vertex cycle graph. When we construct a strong product, we will classify the edges of the product into three subsets:

- We call an edge from $(u_1, u_2)$ to $(v_1, v_2)$ a *vertical edge* if $u_2 = v_2$. The edges of this type form $|V(G_2)|$ disjoint copies of $G_1$ as subgraphs of the product.
- We call an edge from $(u_1, u_2)$ to $(v_1, v_2)$ a *horizontal edge* if $u_1 = v_1$. The edges of this type form $|V(G_1)|$ disjoint copies of $G_2$ as subgraphs of the product.
- We call the remaining edges, for which $u_1 \neq v_1$ and $u_2 \neq v_2$, *diagonal edges*. The subgraph composed of the diagonal edges forms a different kind of graph product, the *tensor product* $G_1 \times G_2$.

We may think of these three edge sets as forming an (improper) edge coloring of the product graph. In Figure 3 these edge sets are colored blue, red and green, respectively.

## 2.2 Graph Parameters

One of the simplest ways of parameterizing sparse graphs is by their *degeneracy*. The degeneracy $d(G)$ of a graph $G$ is the smallest number such that every nonempty subgraph of $G$ contains at least one vertex of degree at most $d(G)$. It may be computed in linear-time by a greedy algorithm that repeatedly removes the minimum-degree vertex and records the largest degree seen among the vertices at the time they are removed [28].

The notion of *treewidth*, a more complicated graph sparsity parameter, was first introduced by Bertelé and Brioschi [3] and Halin [25] and later rediscovered by Robertson and Seymour [31]. One way to define treewidth is to use the concept of tree decomposition. A tree decomposition of graph $G$ consists of a tree $T$ where each vertex $X_i \in T$ (called a bag) is a subset of vertices of $G$ . This tree and its bags are required to satisfy the following properties:

- For each edge $e(u, v)$ in $G$, there exists a bag in $T$ containing both $u$ and $v$; and
- For each vertex $v$ in $G$, the bags containing $v$ form a nonempty connected subtree of $T$.

The width of a tree decomposition is the size of its largest bag, minus one. The treewidth of a graph is defined as the minimum width achieved over all tree decompositions of the graph. Bounded treewidth graphs are especially interesting from an algorithmic point of view. Many well-known NP-complete problems have linear-time algorithms on graphs of bounded treewidth [4].

Another related graph parameter, *clique-width*, was introduced by Courcelle et al. to characterize the structural complexity of graphs [16]. The clique-width of a graph $G$ is the minimum number of labels necessary to construct $G$ by means of four graph operations: creation of a new vertex with a label, vertex disjoint union of labeled graphs, insertion of an edge between two vertices with specified labels and relabeling of vertices. Relevantly for us, Courcelle et al. showed that unit interval graphs are of unbounded clique-width. A graph is an interval graph if and only if all its vertices can be mapped into intervals on a straight line such that two vertices are adjacent when the corresponding intervals intersect each other. In the unit interval graphs, each interval has a unit length. As shown by Brandstädt et al., unit interval graphs belong to the class of leaf powers, which implies that leaf powers also have unbounded clique-width [5, 6]. However, Gurski and Wanke proved that for a fixed $k$, $k$-leaf powers have a clique-width of at most $k + \max(\lfloor \frac{k}{2} \rfloor - 2, 0)$ [24]. Therefore, when seeking the complexity of recognizing leaf powers, parameterized by $k$, we may restrict our attention to graphs of bounded clique-width.

These three properties are defined differently to each other, and may have significantly different values. For instance, the complete bipartite graph $K_{n,n}$ has clique-width two but treewidth and degeneracy $n$, and the $n \times n$ grid graph has degeneracy two but clique-width and treewidth $\Omega(n)$. Nevertheless, there is an important relation between them: in the

graphs of bounded degeneracy and bounded clique-width, the treewidth is also bounded. More specifically, if a graph has clique-width $k$ and contains no complete bipartite subgraph $K_{t,t}$, then it has treewidth at most $3k(t-1)-1$. Since such a subgraph would force the degeneracy to be at least $t$, it follows that the graphs of degeneracy $d$ and clique-width $k$ have treewidth at most $3kd-1$ [23].

## 2.3 Courcelle's Theorem

By considering graphs as logical structures, their properties can be expressed in first-order and second-order logic. In first-order logic, graph properties are expressed as logical formulas wherein the variables range over vertices and the predicates include equality and adjacency relations. Second-order logic is an extension of first-order logic with the power to quantify over relations. Particularly, many natural graph properties can be described in monadic second-order logic, which is a restriction of second-order logic in which only unary relations (sets of vertices or edges) are allowed [15].

There exist two variations of monadic second-order logic: $\mathrm{MSO}_1$ and $\mathrm{MSO}_2$. In $\mathrm{MSO}_1$, quantification is allowed only over sets of vertices, while $\mathrm{MSO}_2$ allows quantification over both sets of vertices and sets of edges. $\mathrm{MSO}_2$ is strictly more expressive; there are some properties, such as Hamiltonicity [14], which are expressible in $\mathrm{MSO}_2$ but not in $\mathrm{MSO}_1$. A graph property is $\mathrm{MSO}_2$-*expressible* if there exists an $\mathrm{MSO}_2$ formula to express it, in which case the corresponding class of graphs becomes $\mathrm{MSO}_2$-*definable*.

The connection between treewidth and monadic second-order logic is given by Courcelle's theorem, according to which every property definable in monadic second-order logic can be tested in linear time on graphs of bounded treewidth [13]. Later, Courcelle et al. extended this theorem to the class of graphs with bounded clique-width when the underlying property is $\mathrm{MSO}_1$-definable [17]. In our application of Courcelle's theorem, we will use an $\mathrm{MSO}_2$ formula with a free variable HORIZONTAL, an edge set, which we will use to pass to the formula certain information about the structural decomposition of the graph it is operating on. This extension of Courcelle's theorem to formulas with a constant number of additional free variables, whose values are assigned through some extra-logical process prior to applying the theorem, is non-problematic and standard.

However, even in $\mathrm{MSO}_2$, it is only possible to quantify over subsets of vertices and edges that belong to the graph to which the logical formula is applied. Much of the difficulty of the leaf power problem rests in this restriction. If we could quantify over edges and vertices that were not already present, we could construct a formula that asserts the existence of sets of vertices and edges forming a leaf root of a given graph, and then add clauses to the formula that ensure that the quantified sets describe a valid leaf root. However, we are not allowed such quantification, because in general the leaf root has vertices and edges that do not belong to our input graph. To apply Courcelle's theorem to leaf power recognition, we must instead find a way to express the property of being a leaf power using only quantification over subsets of vertices and edges of the graph to which we apply the theorem. For this reason, the problem of leaf power recognition forms an important test case for the ability to express graph problems in MSO logic.

A problem is *fixed-parameter tractable* with respect to a parameter $x$ of the input if the problem can be solved in time $f(x)n^{O(1)}$ where $n$ is the size of the input, $f$ is a computable function of $x$ (independent of $n$), and the exponent of $n$ in the $O(1)$ term is independent of $x$. Courcelle's theorem is the foundation of many fixed-parameter tractable algorithms [2, 22, 20, 26], as it proves that properties expressible in $\mathrm{MSO}_1$ or $\mathrm{MSO}_2$ are fixed-parameter tractable with respect to the clique-width or treewidth (respectively) of the input graph.

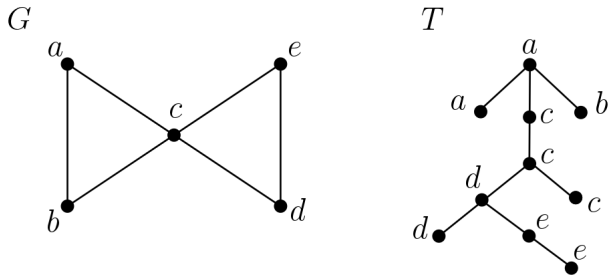



**Figure 4** A 4-leaf power graph $G$ (left), and its leaf root $T$ (right). Each leaf of $T$ is labeled by the vertex of $G$ that it represents, and each internal node of $T$ is labeled by its closest leaf node. When there are ties at a node (as for instance at the root of $T$) the choice of label is made arbitrarily among the closest leaf nodes whose labels appear among the children of the node.

## 3 Embedding Leaf Roots Into Graph Products

Let $G$ be a $k$-leaf power graph, and $T$ be a $k$-leaf root of $G$. If $G$ is not connected, we can handle each of its connected components independently; in this way, we can assume from now on, without loss of generality that $G$ is a connected graph with at least three vertices, and that $T$ is a leaf root chosen arbitrarily among the possible $k$-leaf roots of $T$. It follows from these assumptions that $T$ is a tree, because every edge in $G$ must be represented by a path in $T$. Because $T$ has at least three leaves, it has at least one interior node; we choose one of these nodes arbitrarily to be the root of $T$. Additionally, every vertex or edge of $T$ participates in a path of length at most $k$ between two leaves, representing an edge of $G$. For, if some vertices and edge do not participate in these paths, removing all non-participating vertices and edges from $T$ would produce a smaller leaf root, without creating any new leaves. But this removal would disconnect pairs of leaves on the opposite sides of any removed edge, contradicting the assumption that $G$ is connected.

As the first step of the embedding, we label the vertices of $T$ with the names of vertices in $G$. Each vertex of $T$ will get a label in this way; some labels will be used more than once. In particular, we label each leaf of $T$ by the vertex of $G$ represented by that leaf. Then, as shown in Figure 4, we give each non-leaf node of $T$ the same label as its closest leaf. If there are two or more closest leaves, we choose which one to use as the label arbitrarily among the labels already applied to the child of the given interior node. In this way, when the same label appears more than once, the tree nodes having that label form a connected path in $T$.

As we now show, these labels, together with the depths of the nodes modulo $k$, can be used to embed the $k$-leaf root $T$ into the strong product $G \boxtimes C_k$, where $C_k$ denotes a $k$-vertex cycle graph.

▶ **Lemma 1.** *If $G$ is a connected $k$-leaf power graph on three or more vertices, and $T$ is any $k$-leaf root of $G$, then $T$ can be embedded as a subgraph of the strong product $G \boxtimes C_k$. Additionally, the embedding can be chosen in such a way that each horizontal cycle in the strong product (the product of a vertex $v$ of $G$ with $C_k$) contains exactly one leaf of the embedded copy of $T$, the leaf representing $v$.*

**Proof.** We map a vertex $u$ of $T$ to the pair $(v, i)$ where $v$ is the label assigned to $u$ (the name of a vertex in $G$) and $i$ is the depth of $u$ (its distance from the root of $T$), taken modulo $k$. This pair is one of the vertices of the strong product, so we have mapped vertices of $T$ into vertices of the strong product. An example of such embedding can be seen in Figure

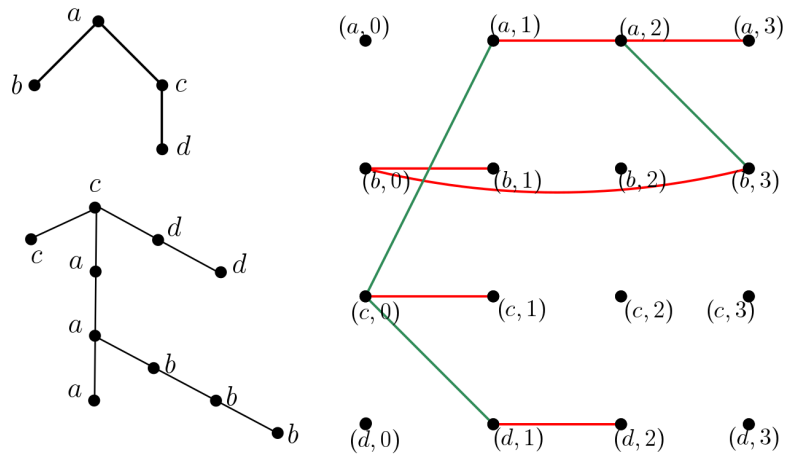



**Figure 5** The graph on the bottom left is a 4-leaf root $T$ of graph $G$ (top left). $T$ can be embedded in the strong product $G \boxtimes C_4$ (right), by mapping each vertex $u$ of $T$ to the pair $(v, i)$ where $v$ is the label of $u$ and $i$ is the depth of $u$ (modulo $k$).

Figure 5. Because $G$ is assumed to be connected, each node of $T$ participates in at least one path of length at most $k$ between two leaves of $T$, representing an adjacency of $G$; it follows that the label for each node of $T$ is at most $k-1$ steps away from the node, and that each path of same-labeled nodes in $T$ has length at most $k-1$. As a consequence, when we take depths modulo $k$, none of these paths can wrap around the cycle and cover the same vertex of the product graph more than once. That is, our mapping from $T$ to $G \boxtimes C_k$ is one-to-one. Because each leaf of $T$ is labeled with the vertex of $G$ that it represents, this mapping has the property described in the lemma, that each horizontal cycle in the strong product contains exactly one leaf of the embedded copy of $T$, the leaf representing the vertex whose product with $C_k$ forms that particular horizontal cycle.

We must also show that this mapping from $T$ to $G \boxtimes C_k$ maps each pair of vertices that are adjacent in $T$ into a pair of vertices that are adjacent in $G \boxtimes C_k$. Recall that adjacency in $G \boxtimes C_k$ is the conjunction of two conditions: two vertices in the product are adjacent if their first coordinates are equal or adjacent in $G$ and their second coordinates are equal or adjacent in $C_k$. Because every two adjacent vertices in $T$ have depths that differ by one, the second coordinates of their images in the product will always be adjacent in $C_k$. It remains to show that, when two vertices are adjacent in $T$, their images in the product have first coordinates that are equal or adjacent in $G$. That is, the labels of the two adjacent vertices in $T$ should be equal or adjacent.

Rephrasing what we still need to show, it is the following: whenever two adjacent vertices in $T$ have different labels, those labels represent adjacent vertices in $G$.

To see that this is true, consider two adjacent vertices $u_1$ and its parent $u_2$ in $T$, labeled by two different vertices $v_1$ and $v_2$ in $G$. As we already stated at the start of this section, the assumption of the lemma that $G$ is connected implies that edge $u_1u_2$ in $T$ participates in at least one path $P$ of length at most $k$ between two leaves, corresponding to an adjacency in $G$. But because $v_1$ and $v_2$ are represented by the closest leaves to $u_1$ and $u_2$ (respectively) the length of the path in $T$ between the leaves representing $v_1$ and $v_2$ must be at most equal to the length of $P$. Therefore, there is a path of length at most $k$ between the leaves representing $v_1$ and $v_2$, so $v_1$ and $v_2$ are adjacent in the $k$-leaf power $G$, as required. ◀

Based on this embedding, we can prove the following characterization of leaf powers, which we will use in our application of Courcelle's theorem to the problem. It is important, for this characterization, that we express everything intrinsically in terms of the properties of the product graph $G \boxtimes C_k$, its edge coloring, and its subgraphs, without reference to the given graph $G$.

▶ **Lemma 2.** *A given connected graph $G$ on three or more vertices is a $k$-leaf power if and only if the product $G \boxtimes C_k$ has a subgraph $T$ with the following properties:*

1. *$T$ is 1-degenerate (i.e., a forest).*
2. *Every vertex of $G \boxtimes C_k$ is connected by horizontal edges of the product to exactly one leaf of $T$.*
3. *Two vertices of $G \boxtimes C_k$ are the endpoints of a non-horizontal edge of the product if and only if the corresponding leaves of $T$ (given according to Property 2) are the distinct endpoints of a path of length at most $k$ in $T$.*

**Proof.** A subgraph obeying these properties is a forest (Property 1), whose leaves can be placed into one-to-one correspondence with the vertices of $G$ (Property 2, using the fact that the horizontal cycles of the product correspond one-for-one with vertices of $G$). It has a path of length at most $k$ between two leaves if and only if the corresponding vertices of $G$ are adjacent (Property 3). So if it exists, it is a $k$-leaf root of $G$ and $G$ is a $k$-leaf power.

In the other direction, if $G$ is a connected $k$-leaf power, let $T$ be a $k$-leaf root of $G$, embedded according to Lemma 1. Then $T$ is a subtree of the product graph (Property 1), with exactly one leaf for each horizontal cycle (Property 2), that forms a $k$-leaf root of $G$ (Property 3). So when $G$ is a $k$-leaf power, a subgraph $T$ obeying the properties of the lemma exists. ◀

## 4 Logical Expression

In this section, we describe how to express the components of Lemma 2, our characterization of the products $G \boxtimes C_k$ that contain a $k$-leaf root of $G$, in monadic second-order logic. Our logical formula will involve a free variable HORIZONTAL, the subset of edges of the given graph (assumed to be of the form $G \boxtimes C_k$) that are horizontal in the product (that is, edges that connect two copies of the same vertex in $G$). We will also assume that $V$ and $E$ refer to the vertices and edges of the graph $G \boxtimes C_k$. In our logical formulas, we will express the type of each quantified variable (whether it is a vertex, edge, set of vertices, or set of edges) by annotating its quantifier with a membership or subset relation. For instance, "$\forall x \in V : \ldots$" quantifies $x$ as a vertex variable. We will express the incidence predicate between an edge $e$ and a vertex $v$ (true if $v$ is an endpoint of $e$, false otherwise) by $e \multimap v$. Because our formulas will also use equality as a predicate, we will express the equality between names of formulas and their explicit logical formulation using a different symbol, $\equiv$. In our formulas, predicates (equality, incidence, and adjacence) will be considered to bind more tightly than logical connectives, allowing us to omit parentheses in many cases.

A subgraph of the given graph may be represented by its set $S$ of edges. In this representation, adjacency between two vertices $a$ and $b$ may be expressed by the formula

$$\textsc{adjacent}(a, b, S) \equiv \exists e \in S : (e \multimap a \wedge e \multimap b).$$

The following formula expresses the property that the neighbors of vertex $\ell$ in subgraph $S$ include at most one vertex from a set $X$:

$$\textsc{leaf}(\ell, X, S) \equiv \forall c, d \in X : \Big(\big(\textsc{adjacent}(\ell, c, S) \wedge \textsc{adjacent}(\ell, d, S)\big) \to c = d\Big).$$

This allows us to express the acyclicity of a subgraph $S$ in terms of 1-degeneracy: every nonempty subset $Y$ of vertices contains a leaf.

$$\textsc{acyclic}(S) \equiv \forall X \subset V : (\exists x \in X) \to \exists \ell \in X : \textsc{leaf}(\ell, X, S).$$

This already allows us to express the first condition of Lemma 2.

We will also use a predicate for whether two vertices $p$ and $q$ are connected by horizontal edges. This is true if there is no cut $(C, V \setminus C)$ that separates them.

$$\textsc{aligned}(p, q) \equiv \forall C \subset V : \big(p \in C \wedge \neg(q \in C)\big) \to \\ \exists h \in \textsc{horizontal} : \exists y, z \in V : \ \big(y \in C \wedge \neg(z \in C) \wedge h \multimap y \wedge h \multimap z\big).$$

This allows us to express a predicate for the property that vertex $\ell$ is a leaf of subgraph $S$ on the same horizontal level as another vertex $v$ (that is, $\ell$ is the representative leaf for $v$'s level):

$$\textsc{representative}(v, \ell, S) \equiv \textsc{leaf}(\ell, V, S) \wedge \textsc{aligned}(v, \ell).$$

The second part of Lemma 2 is that every level has exactly one representative leaf:

$$\textsc{represented}(S) \equiv \big(\forall v \in V : \exists \ell \in V : \textsc{representative}(v, \ell, S)\big) \wedge \\ \Big(\forall v, \ell_1, \ell_2 \in V : \big(\textsc{representative}(v, \ell_1, S) \wedge \textsc{representative}(v, \ell_2, S)\big) \to \ell_1 = \ell_2\Big)$$

Unlike for the previous formulas, there is no way of expressing the existence of a path of length $k$ from $u$ to $v$ in subgraph $S$, for a variable $k$, in $\mathrm{MSO}_2$. We need a different formula $\textsc{path}_k$ for each $k$. We do not require these paths to be simple, as this would only complicate the formula without simplifying our use of it. However it is essential for our application to the third condition of Lemma 2 that we require our paths to have distinct endpoints.

$$\textsc{path}_k(u, v, S) \equiv \exists w_1, w_2, \ldots w_{k-1} \in V : \exists e_1, e_2, \ldots e_k \in S : \\ \neg(u = v) \wedge e_1 \multimap u \wedge e_1 \multimap w_1 \wedge e_2 \multimap w_1 \wedge \cdots \wedge e_k \multimap w_{k-1} \wedge e_k \multimap v.$$

Other than the inequality of the two endpoints, this formula allows repetitions of vertices and edges within each path. In particular, it allows $w_i$ and $w_{i+1}$ to be equal to each other, repeating one endpoint of an edge twice and omitting the other endpoint. Because we allow repetitions in this way, this formulation of the $\textsc{path}$ predicate has the following convenient property:

▶ **Lemma 3.** *For all $k \geq 1$ and all $u$, $v$, and $S$, we have that*

$$\textsc{path}_k(u, v, S) \to \textsc{path}_{k+1}(u, v, S).$$

**Proof.** Let $w_1, \ldots w_{k-1}$ and $e_1, \ldots e_k$ be the vertices and edges witnessing the truth of $\textsc{path}_k(u, v, S)$, let $w_k = v$, and let $e_{k+1} = e_k$. Then $w_1, \ldots, w_k$ and $e_1, \ldots, e_{k+1}$ witness the truth of $\textsc{path}_{k+1}(u, v, S)$. ◀

▶ **Corollary 4.** *Two vertices $u$ and $v$ of a subgraph $S$ of a given graph obey the predicate $\textsc{path}_k(u, v, S)$ if and only if they are distinct and their distance in $S$ is at most $k$.*

This allows us to express the final part of Lemma 2, the requirement that each two vertices are connected by a non-horizontal edge if and only if their representatives are connected by a short path:

$$\textsc{root}_k(S) \equiv \forall u, v \in V : \Big(\big(\exists e \in E : e \multimap u \wedge e \multimap v \wedge \neg(e \in \textsc{horizontal})\big) \longleftrightarrow \\ \exists x, y \in V : \big(\textsc{representative}(u, x, S) \wedge \textsc{representative}(v, y, S) \wedge \textsc{path}_k(x, y, S)\big)\Big).$$

▶ **Lemma 5.** *There exists an* $\mathrm{MSO}_2$ *formula that is modeled by a graph* $G \boxtimes C_k$ *and its set* HORIZONTAL *of horizontal edges exactly when* $G \boxtimes C_k$ *meets the conditions of Lemma 2.*

**Proof.** The formula is

$$\exists S : \big(\textsc{acyclic}(S) \wedge \textsc{represented}(S) \wedge \textsc{root}_k(S)\big).$$

A subgraph defined by a set $S$ of its edges meets the first condition of the lemma if $\textsc{acyclic}(S)$ is true, it meets the second condition of the lemma if $\textsc{represented}(S)$ is true, and it meets the third condition of the lemma if $\textsc{root}_k(S)$ is true. ◀

▶ **Corollary 6.** *The property of a graph* $G$ *being* $k$*-leaf power can be expressed as an* $\mathrm{MSO}_2$ *formula of* $G \boxtimes C_k$ *and of the set* HORIZONTAL *of horizontal edges of this product graph.*

## 5 Fixed-Parameter Tractability of Leaf Powers

In order to apply Courcelle's theorem to the product graph $G \boxtimes C_k$ we need to bound its treewidth.

▶ **Lemma 7.** *If* $G$ *has treewidth* $t$ *and* $H$ *has a bounded number of vertices* $s$ *then* $G \boxtimes H$ *has treewidth at most* $s(t+1)-1$.

**Proof.** Given any tree-decomposition of $G$ with width $t$, we can form a decomposition of $G \boxtimes H$ by using the same tree, and placing each vertex $(v, w)$ of $G \boxtimes H$ (where $v$ and $w$ are vertices of $G$ and $H$ respectively) into the same bag as vertex $v$ of $G$. The size of the largest bag of the tree-decomposition of $G$ is $t+1$, so the size of the largest bag of the resulting tree-decomposition of the product graph is $s(t+1)$. The treewidth is one less than the size of the largest bag. ◀

▶ **Corollary 8.** *If* $G$ *has a bounded treewidth and* $k$ *is bounded, then* $G \boxtimes C_k$ *also has bounded treewidth.*

This gives us our main theorem:

▶ **Theorem 9.** *For fixed constants* $k$ *and* $d$*, it is possible to recognize in linear time (with fixed-parameter tractable dependence on* $k$ *and* $d$*) whether a graph of degeneracy at most* $d$ *is a* $k$*-leaf power.*

**Proof.** If we consider the class of graphs with bounded degeneracy $d$, trivially these graphs do not contain the complete bipartite graph $K_{d+1,d+1}$ as a subgraph. Gurski and Wanke proved that $k$-leaf powers with fixed $k$ have bounded clique-width [24]. In separate work, they also proved that every class of graphs with bounded clique-width and no $K_{a,a}$ subgraph has bounded treewidth [23]. These results immediately imply that $k$-leaf powers with bounded degeneracy have bounded treewidth, and it follows from Corollary 8 that $G \boxtimes C_k$ also has bounded treewidth. Therefore, by applying Courcelle's theorem to the $\mathrm{MSO}_2$ formula of Corollary 6 we obtain the result. ◀

## 6 Conclusion

We have provided a fixed-parameter algorithm to recognize $k$-leaf powers for graphs of bounded degeneracy. Our method embeds a $k$-leaf root of a $k$-leaf power graph in the product graph of the input graph and a $k$-vertex cycle $C_k$, finds a logical characterization of the

leaf roots that are embedded in this way, and applies Courcelle's theorem to determine the existence of a subgraph of the product graph that meets our characterization.

It is perhaps of interest to observe that, although ACYCLIC and ALIGNED involve quantification over vertex sets, the only quantification over edge sets that we use is in the formula of Lemma 5. If we could eliminate all edge sets from our formula (including HORIZONTAL), and obtain an $\mathrm{MSO}_1$ formula for $k$-leaf powers in place of the $\mathrm{MSO}_2$ formula that we use, it might be possible to eliminate the dependence on degeneracy from our parameterized analysis.

Our methods of using low-treewidth supergraphs to represent vertices and edges that are not part of the input graph, and of using graph products to find these supergraphs, may be useful in other graph problems. For instance, this method would have greatly simplified the application of Courcelle's theorem in our recent work on planar split thickness [20]: a graph $G$ has planar split thickness $k$ if and only if $G \boxtimes K_k$ has a planar subgraph $S$ such that, for each non-horizontal edge of the product, the endpoints of the edge are aligned with the endpoints of an edge in $S$. In reducing the logical complexity of problems such as these, our method also makes it more likely that faster model checkers for restricted fragments of MSO logic [1] can be applied to our problem. Additionally, and more speculatively, the reduced logical complexity of our method may help guide future efforts to solve the problem directly using dynamic programming rather than by applying Courcelle's theorem.

## References


1. Max Bannach and Sebastian Berndt. Practical Access to Dynamic Programming on Tree Decompositions. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:13, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ESA.2018.6.
2. Michael J Bannister and David Eppstein. Crossing minimization for 1-page and 2-page drawings of graphs with bounded treewidth. In *International Symposium on Graph Drawing*, pages 210–221. Springer, 2014.
3. Umberto Bertelé and Francesco Brioschi. *Nonserial Dynamic Programming.* Academic Press, 1972.
4. Hans L. Bodlaender. A tourist guide through treewidth. *Acta Cybernet.*, 11(1-2):1–21, 1993.
5. Andreas Brandstädt and Christian Hundt. Ptolemaic graphs and interval graphs are leaf powers. In *Latin American Symposium on Theoretical Informatics*, pages 479–491. Springer, 2008.
6. Andreas Brandstädt, Christian Hundt, Federico Mancini, and Peter Wagner. Rooted directed path graphs are leaf powers. *Discrete Math.*, 310(4):897–910, 2010. doi:10.1016/j.disc.2009.10.006.
7. Andreas Brandstädt and Van Bang Le. Structure and linear time recognition of 3-leaf powers. *Inform. Process. Lett.*, 98(4):133–138, 2006. doi:10.1016/j.ipl.2006.01.004.
8. Andreas Brandstädt, Van Bang Le, and Dieter Rautenbach. A forbidden induced subgraph characterization of distance-hereditary 5-leaf powers. *Discrete Math.*, 309(12):3843–3852, 2009. doi:10.1016/j.disc.2008.10.025.
9. Andreas Brandstädt, Van Bang Le, and R. Sritharan. Structure and linear-time recognition of 4-leaf powers. *ACM Trans. Algorithms*, 5(1):A11:1–A11:22, 2009. doi:10.1145/1435375.1435386.

**10** Andreas Brandstädt and Peter Wagner. On k-versus (k+ 1)-leaf powers. In *International Conference on Combinatorial Optimization and Applications*, pages 171–179. Springer, 2008.

**11** Maw-Shang Chang and Ming-Tat Ko. The 3-Steiner root problem. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 109–120. Springer, 2007.

**12** Zhi-Zhong Chen, Tao Jiang, and Guohui Lin. Computing phylogenetic roots with bounded degrees and errors. *SIAM J. Comput.*, 32(4):864–879, 2003. doi:10.1137/S0097539701389154.

**13** Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inform. and Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

**14** Bruno Courcelle. On the expression of graph properties in some fragments of monadic second-order logic. In Neil Immerman and Phokion G. Kolaitis, editors, *Descriptive Complexity and Finite Models: Proceedings of a DIMACS Workshop, January 14–17, 1996, Princeton University*, volume 31 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 33–62. American Mathematical Society, Providence, RI, 1997.

**15** Bruno Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In *Handbook of graph grammars and computing by graph transformation, Vol. 1*, pages 313–400. World Scientific, River Edge, NJ, 1997. doi:10.1142/9789812384720_0005.

**16** Bruno Courcelle, Joost Engelfriet, and Grzegorz Rozenberg. Handle-rewriting hypergraph grammars. *J. Comput. System Sci.*, 46(2):218–270, 1993. doi:10.1016/0022-0000(93)90004-G.

**17** Bruno Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000. doi:10.1007/s002249910009.

**18** Michael Dom, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Error compensation in leaf root problems. In *International Symposium on Algorithms and Computation*, pages 389–401. Springer, 2004.

**19** Michael Dom, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Extending the tractability border for closest leaf powers. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 397–408. Springer, 2005.

**20** David Eppstein, Philipp Kindermann, Stephen Kobourov, Giuseppe Liotta, Anna Lubiw, Aude Maignan, Debajyoti Mondal, Hamideh Vosoughpour, Sue Whitesides, and Stephen Wismath. On the planar split thickness of graphs. *Algorithmica*, 80(3):977–994, 2018. doi:10.1007/s00453-017-0328-y.

**21** Walter M. Fitch and Emanuel Margoliash. Construction of phylogenetic trees. *Science*, 155(3760):279–284, 1967. doi:10.1126/science.155.3760.279.

**22** Martin Grohe. Computing crossing numbers in quadratic time. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pages 231–236, New York, 2001. ACM. doi:10.1145/380752.380805.

**23** Frank Gurski and Egon Wanke. The tree-width of clique-width bounded graphs without $K_{n,n}$. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 196–205. Springer, 2000.

**24** Frank Gurski and Egon Wanke. The clique-width of tree-power and leaf-power graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 76–85. Springer, 2007.

**25** Rudolf Halin. $S$-functions for graphs. *J. Geometry*, 8(1-2):171–186, 1976. doi:10.1007/BF01917434.

**26** Petr Hliněný. Branch-width, parse trees, and monadic second-order logic for matroids. *J. Combin. Theory Ser. B*, 96(3):325–351, 2006. doi:10.1016/j.jctb.2005.08.005.

27 William Kennedy, Guohui Lin, and Guiying Yan. Strictly chordal graphs are leaf powers. *J. Discrete Algorithms*, 4(4):511–525, 2006. doi:10.1016/j.jda.2005.06.005.

28 David W. Matula and Leland L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM*, 30(3):417–427, 1983. doi:10.1145/2402.322385.

29 Naomi Nishimura, Prabhakar Ragde, and Dimitrios M. Thilikos. On graph powers for leaf-labeled trees. *J. Algorithms*, 42(1):69–108, 2002. doi:10.1006/jagm.2001.1195.

30 Dieter Rautenbach. Some remarks about leaf roots. *Discrete Math.*, 306(13):1456–1461, 2006. doi:10.1016/j.disc.2006.03.030.

31 Neil Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986. doi:10.1016/0196-6774(86)90023-4.

# Parameterized Complexity of Independent Set in H-Free Graphs


## Édouard Bonnet

Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France

## Nicolas Bousquet

CNRS, G-SCOP laboratory, Grenoble-INP, France

## Pierre Charbit

Université Paris Diderot, IRIF, France

## Stéphan Thomassé

Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France
Institut Universitaire de France

## Rémi Watrigant

Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France



### Abstract

In this paper, we investigate the complexity of Maximum Independent Set (MIS) in the class of $H$-free graphs, that is, graphs excluding a fixed graph as an induced subgraph. Given that the problem remains $NP$-hard for most graphs $H$, we study its fixed-parameter tractability and make progress towards a dichotomy between $FPT$ and $W[1]$-hard cases. We first show that MIS remains $W[1]$-hard in graphs forbidding simultaneously $K_{1,4}$, any finite set of cycles of length at least 4, and any finite set of trees with at least two branching vertices. In particular, this answers an open question of Dabrowski *et al.* concerning $C_4$-free graphs. Then we extend the polynomial algorithm of Alekseev when $H$ is a disjoint union of edges to an $FPT$ algorithm when $H$ is a disjoint union of cliques. We also provide a framework for solving several other cases, which is a generalization of the concept of *iterative expansion* accompanied by the extraction of a particular structure using Ramsey's theorem. Iterative expansion is a maximization version of the so-called *iterative compression*. We believe that our framework can be of independent interest for solving other similar graph problems. Finally, we present positive and negative results on the existence of polynomial (Turing) kernels for several graphs $H$.



**2012 ACM Subject Classification** Theory of computation → Fixed parameter tractability

**Keywords and phrases** Parameterized Algorithms, Independent Set, H-Free Graphs

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.17

**Related Version** A full version of the paper is available at [4], `https://arxiv.org/abs/1810.04620`.

**Funding** É. B. is supported by the LABEX MILYON (ANR-10- LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR). N. B. and P. C. are supported by the ANR Project DISTANCIA (ANR-17-CE40-0015) operated by the French National Research Agency (ANR).



© É. Bonnet, N. Bousquet, P. Charbit, S. Thomassé, and R. Watrigant;
licensed under Creative Commons License CC-BY



13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 17; pp. 17:1–17:13
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Given a simple graph $G$, a set of vertices $S \subseteq V(G)$ is an *independent set* if the vertices of this set are all pairwise non-adjacent. Finding an independent set with maximum cardinality is a fundamental problem in algorithmic graph theory, and is known as the MIS problem (MIS, for short) [12]. In general graphs, it is not only $NP$-hard, but also not approximable within $O(n^{1-\epsilon})$ for any $\epsilon > 0$ unless $P = NP$ [20], and $W[1]$-hard [10] (unless otherwise stated, $n$ always denotes the number of vertices of the input graph). Thus, it seems natural to study the complexity of MIS in restricted graph classes. One natural way to obtain such a restricted graph class is to forbid some given pattern to appear in the input. For a fixed graph $H$, we say that a graph is *$H$-free* if it does not contain $H$ as an induced subgraph. Unfortunately, it turns out that for most graphs $H$, MIS in $H$-free graphs remains $NP$-hard, as shown by a very simple reduction first observed by Alekseev:

▶ **Theorem 1** ([1]). *Let $H$ be a connected graph which is neither a path nor a subdivision of the claw. Then* MIS *is NP-hard in $H$-free graphs.*

On the positive side, the case of $P_t$-free graphs has attracted a lot of attention during the last decade. While it is still open whether there exists $t \in \mathbb{N}$ for which MIS is $NP$-hard in $P_t$-free graphs, quite involved polynomial-time algorithms were discovered for $P_5$-free graphs [17], and very recently for $P_6$-free graphs [13]. In addition, we can also mention the recent following result: MIS admits a subexponential algorithm running in time $2^{O(\sqrt{tn \log n})}$ in $P_t$-free graphs for every $t \in \mathbb{N}$ [3]. The second open question concerns the subdivision of the claw. Let $S_{i,j,j}$ be a tree with exactly three vertices of degree one, being at distance $i$, $j$ and $k$ from the unique vertex of degree three. The complexity of MIS is still open in $S_{1,2,2}$-free graphs and $S_{1,1,3}$-free graphs. In this direction, the only positive results concern some subcases: it is polynomial-time solvable in $(S_{1,2,2}, S_{1,1,3}, dart)$-free graphs [15], $(S_{1,1,3}, banner)$-free graphs and $(S_{1,1,3}, bull)$-free graphs [16], where *dart*, *banner* and *bull* are particular graphs on five vertices. Given the large number of graphs $H$ for which the problem remains $NP$-hard, it seems natural to investigate the existence of parameterized algorithms[1], that is, determining the existence of an independent set of size $k$ in a graph with $n$ vertices in time $O(f(k)n^c)$ for some computable function $f$ and constant $c$. A very simple case concerns $K_r$-free graphs, that is, graphs excluding a clique of size $r$. In that case, Ramsey's theorem implies that every such graph $G$ admits an independent set of size $\Omega(n^{\frac{1}{r-1}})$, where $n = |V(G)|$. In the $FPT$ vocabulary, it implies that MIS in $K_r$-free graphs has a kernel with $O(k^{r-1})$ vertices.

To the best of our knowledge, the first step towards an extension of this observation within the $FPT$ framework is the work of Dabrowski *et al.* [8] (see also Dabrowski's PhD manuscript [7]) who showed, among others, that for any positive integer $r$, Max Weighted Independent Set is FPT in $H$-free graphs when $H$ is a clique of size $r$ minus an edge. In the same paper, they settle the parameterized complexity of MIS on almost all the remaining cases of $H$-free graphs when $H$ has at most four vertices. The conclusion is that the problem is $FPT$ on those classes, except for $H = C_4$ which is left open. We answer this question by showing that MIS remains $W[1]$-hard in a subclass of $C_4$-free graphs. On the negative side, it was proved that MIS remains $W[1]$-hard in $K_{1,4}$-free graphs [14].

Finally, we can also mention the case where $H$ is the *bull* graph, which is a triangle with a pending vertex attached to two different vertices. For that case, a polynomial Turing kernel was obtained [19] then improved [11].

[1] For the sake of simplicity, "MIS" will denote the optimisation, decision and parameterized version of the problem (in the latter case, the parameter is the size of the solution), the correct use being clear from the context.

## 1.1 Our results

In Section 2, we present three reductions proving $W[1]$-hardness of MIS in graph excluding several graphs as induced subgraphs, such as $K_{1,4}$, any fixed cycle of length at least four, and any fixed tree with two branching vertices. In Section 3, we extend the polynomial algorithm of Alekseev when $H$ is a disjoint union of edges to an $FPT$ algorithm when $H$ is a disjoint union of cliques. In Section 4, we present a general framework extending the technique of *iterative expansion*, which itself is the maximization version of the well-known iterative compression technique. We apply this framework to provide $FPT$ algorithms when $H$ is a clique minus a complete bipartite graph, or when $H$ is a clique minus a triangle. Finally, in Section 5, we focus on the existence of polynomial (Turing) kernels. We first strenghten some results of the previous section by providing polynomial (Turing) kernels in the case where $H$ is a clique minus a claw. Then, we prove that for many $H$, MIS on $H$-free graphs does not admit a polynomial kernel, unless $NP \subseteq coNP/poly$. Our results allows to obtain the complete dichotomy polynomial/polynomial kernel (PK)/no PK but polynomial Turing kernel/$W[1]$-hard for all possible graphs on four vertices, while only five graphs on five vertices remain open for the $FPT/W[1]$-hard dichotomy.

Due to space restrictions, proofs marked with a ($\star$) were omitted, and can be found in the long version of the paper [4]. This long version also contains additional figures, and two variants of the reduction presented in the next section, together with a discussion.

## 1.2 Notation

For classical notation related to graph theory or fixed-parameter tractable algorithms, we refer the reader to the monographs [9] and [10], respectively. For an integer $r \geq 2$ and a graph $H$ with vertex set $V(H) = \{v_1, \ldots, v_{n_H}\}$ with $n_H \leq r$, we denote by $K_r \setminus H$ the graph with vertex set $\{1, \ldots, r\}$ and edge set $\{ab : 1 \leq a, b \leq r \text{ such that } v_a v_b \notin E(H)\}$. For $X \subseteq V(G)$, we write $G \setminus X$ to denote $G[V(G) \setminus X]$. For two graphs $G$ and $H$, we denote by $G \uplus H$ the *disjoint union* operation, that is, the graph with vertex set $V(G) \cup V(H)$ and edge set $E(G) \cup E(H)$. We denote by $G + H$ the *join* operation of $G$ and $H$, that is, the graph with vertex set $V(G) \cup V(H)$ and edge set $E(G) \cup E(H) \cup \{uv : u \in V(G), v \in V(H)\}$. For two integers $r, k$, we denote by $Ram(r, k)$ the Ramsey number of $r$ and $k$, *i.e.* the minimum order of a graph to contain either a clique of size $r$ or an independent set of size $k$. We write for short $Ram(k) = Ram(k, k)$. Finally, for $\ell, k > 0$, we denote by $Ram_\ell(k)$ the minimum order of a complete graph whose edges are colored with $\ell$ colors to contain a monochromatic clique of size $k$.

# 2 $W[1]$-hardness

▶ **Theorem 2.** *For any $p_1 \geq 4$ and $p_2 \geq 1$,* MIS *remains $W[1]$-hard in graphs excluding simultaneously the following graphs as induced subgraphs: $K_{1,4}$, $C_4$, ..., $C_{p_1}$ and any tree $T$ with two branching vertices*[2] *at distance at most $p_2$.*

**Proof.** Let $p = \max\{p_1, p_2\}$. We reduce from Grid Tiling, where the input is composed of $k^2$ sets $S_{i,j} \subseteq [m] \times [m]$ $(0 \leq i, j \leq k-1)$, called *tiles*, each composed of $n$ elements. The objective of Grid Tiling is to find an element $s^*_{i,j} \in S_{i,j}$ for each $0 \leq i, j \leq k-1$, such that $s^*_{i,j}$ agrees in the first coordinate with $s^*_{i,j+1}$, and agrees in the second coordinate with $s^*_{i+1,j}$,

[2] A branching vertex in a tree is a vertex of degree at least 3.

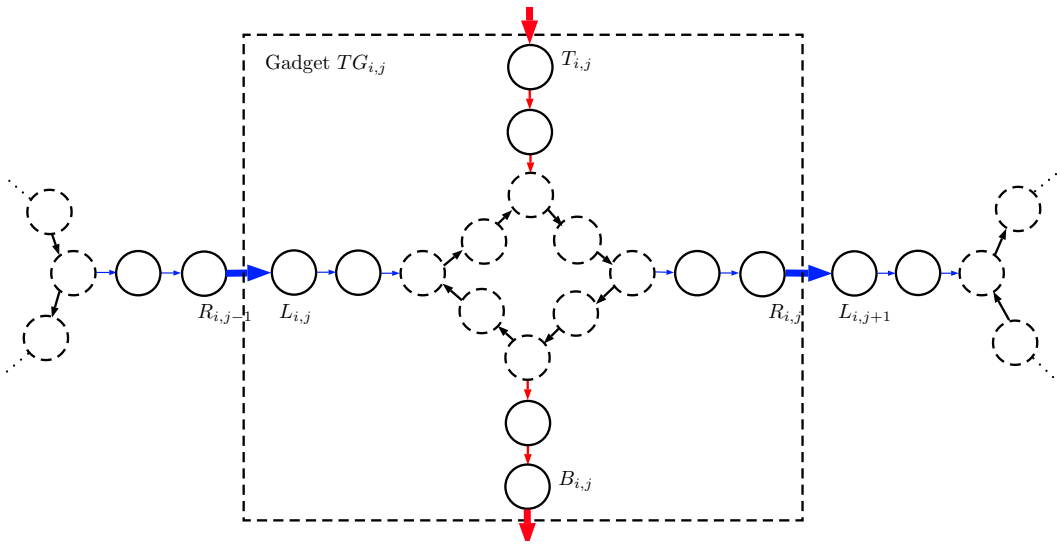



**Figure 1** Gadget $TG^{i,j}$ representing a tile and its adjacencies with $TG_{i,j-1}$ and $TG_{i,j+1}$, for $p = 1$. Each circle is a clique on $n$ vertices (dashed cliques are the cycle cliques). Black, blue and red arrows represent respectively type $T_h$, $T_r$ and $T_c$ edges (bold arrows are between two gadgets).

for every $0 \leq i, j \leq k-1$ (incrementations of $i$ and $j$ are done modulo $k$). In such case, we say that $\{s^*_{i,j},\ 0 \leq i, j \leq k-1\}$ is a *feasible solution* of the instance. It is known that Grid Tiling is $W[1]$-hard parameterized by $k$ [6].

Before describing formally the reduction, let us give some definitions and ideas. Given $s = (a, b)$ and $s' = (a', b')$, we say that $s$ is *row-compatible* (resp. *column-compatible*) with $s'$ if $a \geq a'$ (resp. $b \geq b'$)[3]. Observe that a solution $\{s^*_{i,j},\ 0 \leq i, j \leq k-1\}$ is feasible if and only if $s^*_{i,j}$ is row-compatible with $s^*_{i,j+1}$ and column-compatible with $s^*_{i+1,j}$ for every $0 \leq i, j \leq k-1$ (incrementations of $i$ and $j$ are done modulo $k$). Informally, the main idea of the reduction is that, when representing a tile by a clique, the row-compatibility (resp. column-compatibility) relation (as well at its complement) forms a $C_4$-free graph when considering two consecutive tiles, and a claw-free graph when considering three consecutive tiles. The main difficulty is to forbid the desired graphs to appear in the "branchings" of tiles. We now describe the reduction.

For every tile $S_{i,j} = \{s^{i,j}_1, \ldots, s^{i,j}_n\}$, we construct a *tile gadget* $TG_{i,j}$, depicted in Figure 1. Notice that this gadget shares some ideas with the $W[1]$-hardness of the problem in $K_{1,4}$-free graphs by Hermelin *et al.* [14]. To define this gadget, we first describe an oriented graph with three types of arcs (type $T_h$, $T_r$ and $T_c$, which respectively stands for *half graph*, *row* and *column*, this meaning will become clearer later), and then explain how to represent the vertices and arcs of this graph to get the concrete gadget. Consider first a directed cycle on $4p+4$ vertices $c_1, \ldots, c_{4p+4}$ with arcs of type $T_h$. Then consider four oriented paths on $p+1$ vertices: $P_1$, $P_2$, $P_3$ and $P_4$. $P_1$ and $P_3$ are composed of arcs of type $T_c$, while $P_2$ and $P_4$ are composed of arcs of type $T_r$. Put an arc of type $T_c$ between the last vertex of $P_1$ and $c_1$, an arc of type $T_c$ between $c_{2p+3}$ and the first vertex of $P_3$, an arc of type $T_r$ between $c_{p+2}$ and the first vertex of $P_2$, and an arc of type $T_r$ between the last vertex of $P_4$ and $c_{3p+4}$.

[3] Notice that the row-compatibility (resp. column-compatibility) relation is not symmetrical.

Now, replace every vertex of this oriented graph by a clique on $n$ vertices, and fix an arbitrary ordering on the vertices of each clique. For each arc of type $T_h$ between $c$ and $c'$, add a half graph[4] between the corresponding cliques: connect the $a^{th}$ vertex of the clique representing $c$ with the $b^{th}$ vertex of the clique representing $c'$ iff $a > b$. For every arc of type $T_r$ from a vertex $c$ to a vertex $c'$, connect the $a^{th}$ vertex of the clique representing $c$ with the $b^{th}$ vertex of the clique representing $c'$ iff $s_a^{i,j}$ is *not* row-compatible with $s_b^{i,j}$. Similarly, for every arc of type $T_c$ from a vertex $c$ to a vertex $c'$, connect the $a^{th}$ vertex of the clique representing $C$ with the $b^{th}$ vertex of the clique representing $c'$ iff $s_a^{i,j}$ is *not* column-compatible with $s_b^{i,j}$. The cliques corresponding to vertices of this gadget are called the *main cliques* of $TG_{i,j}$, and the cliques corresponding to the central cycle on $4p+4$ vertices are called the *cycle cliques*. The main cliques which are not cycle cliques are called *path cliques*. The cycle cliques adjacent to one path clique are called *branching cliques*. Finally, the clique corresponding to the vertex of degree one in the path attached to $c_1$ (resp. $c_{p+2}$, $c_{2p+3}$, $c_{3p+4}$) is called the *top* (resp. *right*, *bottom*, *left*) clique of $TG_{i,j}$, denoted by $T_{i,j}$ (resp. $R_{i,j}$, $B_{i,j}$, $L_{i,j}$). Let $T_{i,j} = \{t_1^{i,j}, \dots, t_n^{i,j}\}$, $R_{i,j} = \{r_1^{i,j}, \dots, r_n^{i,j}\}$, $B_{i,j} = \{b_1^{i,j}, \dots, b_n^{i,j}\}$, and $L_{i,j} = \{\ell_1^{i,j}, \dots, \ell_n^{i,j}\}$. For the sake of readability, we might omit the superscripts $i, j$ when it is clear from the context.

▶ **Lemma 3.** *($\star$) Let $K$ be an independent set of size $8(p+1)$ in $TG_{i,j}$. Then:*

**(a)** *$K$ intersects all the cycle cliques on the same index $x$;*

**(b)** *if $K \cap T_{i,j} = \{t_{x_t}\}$, $K \cap R_{i,j} = \{r_{x_r}\}$, $K \cap B_{i,j} = \{b_{x_b}\}$, and $K \cap L_{i,j} = \{\ell_{x_\ell}\}$. Then:*
- *$s_{x_\ell}^{i,j}$ is row-compatible with $s_x^{i,j}$ which is row-compatible with $s_{x_r}^{i,j}$, and*
- *$s_{x_t}^{i,j}$ is column-compatible with $s_x^{i,j}$ which is column-compatible with $s_{x_b}^{i,j}$.*

For $i, j \in \{0, \dots, k-1\}$, we connect the right clique of $TG_{i,j}$ with the left clique of $TG_{i,j+1}$ in a "type $T_r$ spirit": for every $x, y \in [n]$, connect $r_x^{i,j} \in R_{i,j}$ with $\ell_y^{i,j+1} \in L_{i,j+1}$ iff $s_x^{i,j}$ is *not* row-compatible with $s_y^{i,j+1}$. Similarly, we connect the bottom clique of $TG_{i,j}$ with the top clique of $TG_{i+1,j}$ in a "type $T_c$ spirit": for every $x, y \in [n]$, connect $b_x^{i,j} \in B_{i,j}$ with $t_y^{i+1,j} \in T_{i+1,j}$ iff $s_x^{i,j}$ is *not* column-compatible with $s_y^{i+1,j}$ (all incrementations of $i$ and $j$ are done modulo $k$). This terminates the construction of the graph $G$.

▶ **Lemma 4.** *($\star$) The input instance of* Grid Tiling *is positive if and only if $G$ has an independent set of size $k' = 8(p+1)k^2$.*

Let us now prove that $G$ does not contain the graphs mentionned in the statement as an induced subgraph:

**(i)** $K_{1,4}$: we first prove that for every $0 \le i, j \le k-1$, the graph induced by the cycle cliques of $TG_{i,j}$ is claw-free. For the sake of contradiction, suppose that there exist three consecutive cycle cliques $A$, $B$ and $C$ containing a claw. W.l.o.g. we may assume that $b_x \in B$ is the center of the claw, and $a_\alpha \in A$, $b_\beta \in B$ and $c_\gamma \in C$ are the three endpoints. By construction of the gadgets (there is a half graph between $A$ and $B$ and between $B$ and $C$), we must have $\alpha < x < \gamma$. Now, observe that if $x < \beta$ then $a_\alpha$ must be adjacent to $b_\beta$, and if $\beta < x$, then $b_\beta$ must be adjacent to $c_\gamma$, but both case are impossible since $\{a_\alpha, b_\beta, c_\gamma\}$ is supposed to be an independent set. Similarly, we can prove that the graph induced by each path of size $2(p+1)$ linking two consecutive gadgets is claw-free. Hence, the only way for $K_{1,4}$ to appear in $G$ would be that

[4] Notice that our definition of half graph slighly differs from the usual one, in the sense that we do not put edges relying two vertices of the same index. Hence, our construction can actually be seen as the complement of a half graph (which is consistent with the fact that usually, both parts of a half graph are independent sets, while they are cliques in our gadgets).

the center appears in the cycle clique attached to a path, for instance in the clique represented by the vertex $c_1$ in the cycle. However, it can easily be seen that in this case, a claw must lie either in the graph induced by the cycle cliques of the gadget, or in the path linking $TG_{i,j}$ with $TG_{i-1,j}$, which is impossible.

**(ii)** $C_4, \ldots, C_{p_1}$. The main argument is that the graph induced by any two main cliques does not contain any of these cycles. Then, we show that such a cycle cannot lie entirely in the cycle cliques of a single gadget $TG_{i,j}$. Indeed, if this cycle uses at most one vertex per main clique, then it must be of length at least $4p+4$. If it intersects a clique $C$ on two vertices, then either it also intersect all the cycle cliques of the gadget, in which case it is of length $4p+5$, or it intersects an adjacent clique of $C$ on two vertices, in which case these two cliques induce a $C_4$, which is impossible. Similarly, such a cycle cannot lie entirely in a path between the main cliques of two gadgets. Finally, the main cliques of two gadgets are at distance $2(p+1)$, hence such a cycle cannot intersect the main cliques of two gadgets.

**(iii)** any tree $T$ with two branching vertices at distance at most $p_2$. Using the same argument as for the $K_{1,4}$ case, observe that the claws contained in $G$ can only appear in the cycle cliques where the paths are attached. However, observe that these cliques are at distance $2(p+1) > p_2$, thus, such a tree $T$ cannot appear in $G$. ◀

## 3 Positive results I: disjoint union of cliques

For $r, q \geq 1$, let $K_r^q$ be the disjoint union of $q$ copies of $K_r$. The following proof is inspired by the case $r = 2$ by Alekseev [2].

▶ **Theorem 5.** Maximum Independent Set *is FPT in $K_r^q$-free graphs.*

**Proof.** We will prove by induction on $q$ that a $K_r^q$-free graph has an independent set of size $k$ or has at most $(Ram(r,k)+1)^{qk} n^{qr}$ independent sets. This will give the desired FPT-algorithm, as the proof shows how to construct this collection of independent sets. Note that the case $q = 1$ is trivial by Ramsey's theorem.

Let $G$ be a $K_r^q$-free graph and let $<$ be any fixed total ordering of $V(G)$ such that the largest vertex in this ordering belongs to a clique of size $r$ (the case where $G$ is $K_r$-free is trivial by Ramsey's theorem). For any vertex $x$, define $x^+ = \{y,\ x < y\}$ and $x^- = V(G) \setminus x^+$ (hence, $x \in x^-$).

Let $C$ be a fixed clique of size $r$ in $G$ and let $c$ be the largest vertex of $C$ with respect to $<$. Let $V_1$ be the set of vertices of $c^+$ which have no neighbor in $C$. Note that $V_1$ induces a $K_r^{q-1}$-free graph, so by induction either it contains an independent set of size $k$, and so does $G$, or it has at most $(Ram(r,k)+1)^{(q-1)k} n^{(q-1)r}$ independent sets. In the latter case, let $\mathcal{S}_1$ be the set of all independent sets of $G[V_1]$.

Now in a second phase we define an initially empty set $\mathcal{S}_C$ and do the following. For each independent set $S_1$ in $\mathcal{S}_1$ (including the empty set), we denote by $V_2$ the set of vertices in $c^-$ that have no neighbor in $S_1$ (notice that $c \in V_2$). For every choice of a vertex $x$ amongst the largest $(Ram(r,k)+1)$ vertices of $V_2$ in the order, we add $x$ to $S_1$ and modify $V_2$ in order to keep only vertices that are smaller than $x$ (with respect to $<$) and not adjacent to $x$. We repeat this operation $k$ times (or less if $V_2$ becomes empty) and, at the end, we either find an independent set of size $k$ or add $S_1$ to $\mathcal{S}_C$. By doing so we construct a family of at most $(Ram(r,k)+1)^k$ independent sets for each $S_1$, so in total we get indeed at most $(Ram(r,k)+1)^{kq} n^{(q-1)r}$ independent sets for each clique $C$. Finally we define $\mathcal{S}$ as the union over all $r$-cliques $C$ of the sets $\mathcal{S}_C$, so that $\mathcal{S}$ has size at most the desired number.

We claim that if $G$ does not contain an independent set of size $k$, then $\mathcal{S}$ contains all independent sets of $G$. It suffices to prove that for every independent set $S$, there exists a clique $C$ for which $S \in \mathcal{S}_C$. Let $S$ be an independent set, and define $C$ to be a clique of size $r$ such that its largest vertex $c$ (with respect to $<$) satisfies the conditions:

- no vertex of $C$ is adjacent to a vertex of $S \cap c^+$, and
- $c$ is the smallest vertex such that a clique $C$ satisfying the first item exists.

First remark that such a clique always exist, since we assumed that the largest vertex $c_{last}$ of $<$ is contained in a clique of size $r$, which means that $S \cap c^+_{last}$ is empty and thus the first item is vacuously satisfied. Secondly, note that several cliques $C$ might satisfy the two previous conditions. In that case, pick one such clique arbitrarily. This definition of $C$ and $c$ ensures that $S \cap c^+$ is an independent set in the set $V_1$ defined in the construction above (it might be empty, but we also consider this case). Thus, it will be picked in the second phase as some $S_1$ in $\mathcal{S}_1$ and for this choice, each time $V_2$ is considered, the fact that $C$ is chosen to minimize its largest element $c$ guarantees that there must be a vertex of $S$ in the $(Ram(r,k)+1)$ largest vertices in $V_2$: either $c \in S$ and we are done, or $S$ must intersect one of the $Ram(r,k)$ largest elements of $V_2 \setminus \{c\}$, otherwise there would be an an $r$-clique contradicting the choice of $C$. This shows that $S \in \mathcal{S}_C$, which concludes our proof. ◀

# 4 Positive results II

## 4.1 Key ingredient: Iterative expansion and Ramsey extraction

In this section, we present the main idea of our algorithms. It is a generalization of iterative expansion, which itself is the maximization version of the well-known iterative compression technique. Iterative compression is a useful tool for designing parameterized algorithms for subset problems (*i.e.* problems where a solution is a subset of some set of elements: vertices of a graph, variables of a logic formula...*etc.*) [6, 18]. Although it has been mainly used for minimization problems, iterative compression has been successfully applied for maximization problems as well, under the name *iterative expansion* [5]. Roughly speaking, when the problem consists in finding a solution of size at least $k$, the iterative expansion technique consists in solving the problem where a solution $S$ of size $k-1$ is given in the input, in the hope that this solution will imply some structure in the instance. In the following, we consider an extension of this approach where, instead of a single smaller solution, one is given a set of $f(k)$ smaller solutions $S_1, \ldots, S_{f(k)}$. As we will see later, we can further add more constraints on the sets $S_1, \ldots, S_{f(k)}$. Notice that all the results presented in this sub-section (Lemmas 7 and 10 in particular) hold for any hereditary graph class (including the class of all graphs). The use of properties inherited from particular graphs (namely, $H$-free graphs in our case) will only appear in Sections 4.2 and 4.3.

▶ **Definition 6.** For a function $f : \mathbb{N} \to \mathbb{N}$, the $f$-Iterative Expansion MIS takes as input a graph $G$, an integer $k$, and a set of $f(k)$ independent sets $S_1, \ldots, S_{f(k)}$, each of size $k-1$. The objective is to find an independent set of size $k$ in $G$, or to decide that such an independent set does not exist.

▶ **Lemma 7.** *($\star$) Let $\mathcal{G}$ be a hereditary graph class.* MIS *is FPT in $\mathcal{G}$ iff $f$-*Iterative Expansion MIS *is FPT in $\mathcal{G}$ for some computable function $f : \mathbb{N} \to \mathbb{N}$.*

We will actually prove a stronger version of this result, by adding more constraints on the input sets $S_1, \ldots, S_{f(k)}$, and show that solving the expansion version on this particular kind of input is enough to obtain the result for MIS.

▶ **Definition 8.** Given a graph $G$ and a set of $k-1$ vertex-disjoint cliques of $G$, $\mathcal{C} = \{C_1, \ldots, C_{k-1}\}$, each of size $q$, we say that $\mathcal{C}$ is a set of *Ramsey-extracted cliques of size* $q$ if the conditions below hold. Let $C_r = \{c_j^r : j \in \{1, \ldots, q\}\}$ for every $r \in \{1, \ldots, k-1\}$.

- For every $j \in [q]$, the set $\{c_j^r : r \in \{1, \ldots, k-1\}\}$ is an independent set of $G$ of size $k-1$.
- For any $r \neq r' \in \{1, \ldots, k-1\}$, one of the four following case can happen:
  - **(i)** for every $j, j' \in [q]$, $c_j^r c_{j'}^{r'} \notin E(G)$
  - **(ii)** for every $j, j' \in [q]$, $c_j^r c_{j'}^{r'} \in E(G)$ iff $j \neq j'$
  - **(iii)** for every $j, j' \in [q]$, $c_j^r c_{j'}^{r'} \in E(G)$ iff $j < j'$
  - **(iv)** for every $j, j' \in [q]$, $c_j^r c_{j'}^{r'} \in E(G)$ iff $j > j'$

  In the case $(i)$ (resp. $(ii)$), we say that the relation between $C_r$ and $C_{r'}$ is *empty* (resp. *full*[5]). In case $(iii)$ or $(iv)$, we say the relation is *semi-full*.

Observe, in particular, that a set $\mathcal{C}$ of $k-1$ Ramsey-extracted cliques of size $q$ can be partitionned into $q$ independent sets of size $k-1$. As we will see later, these cliques will allow us to obtain more structure with the remaining vertices if the graph is $H$-free. Roughly speaking, if $q$ is large, we will be able to extract from $\mathcal{C}$ another set $\mathcal{C}'$ of $k-1$ Ramsey-extracted cliques of size $q' < q$, such that every clique is a module[6] with respect to the solution $x_1^*, \ldots, x_k^*$ we are looking for. Then, by guessing the structure of the adjacencies between $\mathcal{C}'$ and the solution, we will be able to identify from the remaining vertices $k$ sets $X_1, \ldots, X_k$, where each $X_i$ has the same neighborhood as $x_i^*$ *w.r.t.* $\mathcal{C}'$, and plays the role of "candidates" for this vertex. For a function $f : \mathbb{N} \to \mathbb{N}$, we define the following problem:

▶ **Definition 9.** The $f$-Ramsey-extracted Iterative Expansion MIS problem takes as input an integer $k$ and a graph $G$ whose vertices are partitionned into non-empty sets $X_1 \cup \cdots \cup X_k \cup C_1 \cup \cdots \cup C_{k-1}$, where:

- $\{C_1, \ldots, C_{k-1}\}$ is a set of $k-1$ Ramsey-extracted cliques of size $f(k)$
- any independent set of size $k$ in $G$ is contained in $X_1 \cup \cdots \cup X_k$
- $\forall i \in \{1, \ldots, k\}$, $\forall v, w \in X_i$ and $\forall j \in \{1, \ldots, k-1\}$, $N(v) \cap C_j = N(w) \cap C_j = \emptyset$ or $N(v) \cap C_j = N(w) \cap C_j = C_j$
- the following bipartite graph $\mathcal{B}$ is connected: $V(\mathcal{B}) = B_1 \cup B_2$, $B_1 = \{b_1^1, \ldots, b_k^1\}$, $B_2 = \{b_1^2, \ldots, b_{k-1}^2\}$ and $b_j^1 b_r^2 \in E(\mathcal{B})$ iff $X_j$ and $C_r$ are adjacent.

The objective is to find an independent set $S$ in $G$ of size at least $k$, or to decide that $G$ does not contain an independent set $S$ such that $S \cap X_i \neq \emptyset$ for all $i \in \{1, \ldots, k\}$.

▶ **Lemma 10.** *Let $\mathcal{G}$ be a hereditary graph class. If there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ such that $f$-*Ramsey-extracted Iterative Expansion MIS *is FPT in $\mathcal{G}$, then $g$-*Iterative Expansion MIS *is FPT in $\mathcal{G}$, where $g(x) = Ram_\ell(f(x)2^{x(x-1)})$ $\forall x \in \mathbb{N}$, with $\ell_x = 2^{(x-1)^2}$.*

**Proof.** Let $f : \mathbb{N} \to \mathbb{N}$ be such a function, and let $G$, $k$ and $\mathcal{S} = \{S_1, \ldots, S_{g(k)}\}$ be an input of $g$-Iterative Expansion MIS. Recall that the objective is to find an independent set of size $k$ in $G$, or to decide that such an independent set does not exist. If $G$ contains an independent set of size $k$, then either there is one intersecting some set of $\mathcal{S}$, or every independent set of size $k$ avoids the sets in $\mathcal{S}$. In order to capture the first case, we branch on every vertex $v$ of the sets in $\mathcal{S}$, and make a recursive call with parameter $G \setminus N[v]$, $k-1$.

[5] Remark that in this case, the graph induced by $C_r \cup C_{r'}$ is the complement of a perfect matching.
[6] A set of vertices $M$ is a module if every vertex $v \notin M$ is adjacent to either all vertices of $M$, or none.

In the remainder of the algorithm, we thus assume that any independent set of size $k$ in $G$ avoids every set of $\mathcal{S}$.

We choose an arbitrary ordering of the vertices of each $S_j$. Let us denote by $s_j^r$ the $r^{th}$ vertex of $S_j$. Notice that given an ordered pair of sets of $k-1$ vertices $(A, B)$, there are $\ell_k = 2^{(k-1)^2}$ possible sets of edges between these two sets. Let us denote by $c_1, \ldots, c_{2^{(k-1)^2}}$ the possible sets of edges, called *types*. We define an auxiliary edge-colored graph $H$ whose vertices are in one-to-one correspondence with $S_1, \ldots, S_{g(k)}$, and, for $i < j$, there is an edge between $S_i$ and $S_j$ of color $\gamma$ iff the type of $(S_i, S_j)$ is $\gamma$. By Ramsey's theorem, since $H$ has $Ram_{\ell_k}(f(k)2^{k(k-1)})$ vertices, it must admit a monochromatic clique of size at least $h(k) = f(k)2^{k(k-1)}$. *W.l.o.g.*, the vertex set of this clique corresponds to $S_1, \ldots, S_{h(k)}$. For $p \in \{1, \ldots, k-1\}$, let $C_p = \{s_j^p, \ldots, s_{h(k)}^p\}$. Observe that the Ramsey extraction ensures that each $C_p$ is either a clique or an independent set. If $C_p$ is an independent set for some $r$, then we can immediately conclude, since $h(k) \geq k$. Hence, we suppose that $C_p$ is a clique for every $p \in \{1, \ldots, k-1\}$. We now prove that $C_1, \ldots, C_{k-1}$ are Ramsey-extracted cliques of size $h(k)$. First, by construction, for every $j \in \{1, \ldots, h(k)\}$, the set $\{s_j^p : p = 1, \ldots, k-1\}$ is an independent set. Then, let $c$ be the type of the clique obtained previously, represented by the adjacencies between two sets $(A, B)$, each of size $k-1$. For every $p \in \{1, \ldots, k-1\}$, let $a_p$ (resp. $b_p$) be the $a^{th}$ vertex of $A$ (resp. $B$). Let $p, q \in \{1, \ldots, t\}$, $p \neq q$. If any of $a_p b_q$ and $a_q b_p$ are edges in type $c$, then there is no edge between $C_p$ and $C_q$, and their relation is thus empty. If both edges $a_p b_q$ and $a_q b_p$ exist in $c$, then the relation between $C_p$ and $C_q$ is full. Finally if exactly one edge among $a_p b_q$ and $a_q b_p$ exists in $c$, then the relation between $C_p$ and $C_q$ is semi-full. This concludes the fact that $\mathcal{C} = \{C_1, \ldots, C_{k-1}\}$ are Ramsey-extracted cliques of size $h(k)$.

Suppose that $G$ has an independent set $X^* = \{x_1^*, \ldots, x_k^*\}$. Recall that we assumed previously that $X^*$ is contained in $V(G) \setminus (C_1 \cup \cdots \cup C_{k-1})$. The next step of the algorithm consists in branching on every subset of $f(k)$ indices $J \subseteq \{1, \ldots, h(k)\}$, and restrict every set $C_p$ to $\{s_j^p : j \in J\}$. For the sake of readability, we keep the notation $C_p$ to denote $\{s_j^p : j \in J\}$ (the non-selected vertices are put back in the set of remaining vertices of the graph, *i.e.* we do not delete them). Since $h(k) = f(k)2^{k(k-1)}$, there must exist a branching where the chosen indices are such that for every $i \in \{1, \ldots, k\}$ and every $p \in \{1, \ldots, k-1\}$, $x_i^*$ is either adjacent to all vertices of $C_p$ or none of them. In the remainder, we may thus assume that such a branching has been made, with respect to the considered solution $X^* = \{x_1^*, \ldots, x_k^*\}$. Now, for every $v \in V(G) \setminus (C_1, \ldots, C_{k-1})$, if there exists $p \in \{1, \ldots, k-1\}$ such that $N(v) \cap C_p \neq \emptyset$ and $N(v) \cap C_p \neq C_p$ , then we can remove this vertex, as we know that it cannot correspond to any $x_i^*$. Thus, we know that all the remaining vertices $v$ are such that for every $p \in \{1, \ldots, k-1\}$, $v$ is either adjacent to all vertices of $C_p$, or none of them.

In the following, we perform a color coding-based step on the remaining vertices. Informally, this color coding will allow us to identify, for every vertex $x_i^*$ of the optimal solution, a set $X_i$ of candidates, with the property that all vertices in $X_i$ have the same neighborhood with respect to sets $C_1, \ldots, C_{k-1}$. We thus color uniformly at random the remaining vertices $V(G) \setminus (C_1, \ldots, C_{k-1})$ using $k$ colors. The probability that the elements of $X^*$ are colored with pairwise distinct colors is at least $e^{-k}$. We are thus reduced to the case of finding a *colorful*[7] independent set of size $k$. For every $i \in \{1, \ldots, k\}$, let $X_i$ be the vertices of $V(G) \setminus (C_1, \ldots, C_{k-1})$ colored with color $i$. We now partition every set $X_i$ into at most $2^{k-1}$ subsets $X_i^1, \ldots, X_i^{2^{k-1}}$, such that for every $j \in \{1, \ldots, 2^{k-1}\}$, all vertices of $X_i^j$ have the same neighborhood with respect to the sets $C_1, \ldots, C_{k-1}$ (recall that every vertex of

[7] A set of vertices is called *colorful* if it is colored with pairwise distinct colors.

$V(G) \setminus (C_1, \ldots, C_{k-1})$ is adjacent to all vertices of $C_p$ or none, for each $p \in \{1, \ldots, k-1\}$). We branch on every tuple $(j_1, \ldots, j_k) \in \{1, \ldots, 2^{k-1}\}$. Clearly the number of branchings is bounded by a function of $k$ only and, moreover, one branching $(j_1, \ldots, j_k)$ is such that $x_i^*$ has the same neighborhood in $C_1 \cup \cdots \cup C_{k-1}$ as vertices of $X_i^{j_i}$ for every $i \in \{1, \ldots, k\}$. We assume in the following that such a branching has been made. For every $i \in \{1, \ldots, k\}$, we can thus remove vertices of $X_i^j$ for every $j \neq j_i$. For the sake of readability, we rename $X_i^{j_i}$ as $X_i$. Let $\mathcal{B}$ be the bipartite graph with vertex bipartition $(B_1, B_2)$, $B_1 = \{b_1^1, \ldots, b_k^1\}$, $B_2 = \{b_1^2, \ldots, b_{k-1}^2\}$, and $b_i^1 b_p^2 \in E(\mathcal{B})$ iff $x_i^*$ is adjacent to $C_p$. Since every $x_i^*$ has the same neighborhood as $X_i$ with respect to $C_1, \ldots, C_{k-1}$, this bipartite graph actually corresponds to the one described in Definition 9 representing the adjacencies between $X_i$'s and $C_p$'s. We now prove that it is connected. Suppose it is not. Then, since $|B_1| = k$ and $|B_2| = k-1$, there must be a component with as many vertices from $B_1$ as vertices from $B_2$. However, in this case, using the fixed solution $X^*$ on one side and an independent set of size $k-1$ in $C_1 \cup \cdots \cup C_{k-1}$ on the other side, it implies that there is an independent set of size $k$ intersecting $\cup_{p=1}^{k-1} C_p$, a contradiction.

Hence, all conditions of Definition 9 are now fulfilled. It now remains to find an independent set of size $k$ disjoint from the sets $\mathcal{C}$, and having a non-empty intersection with $X_i$, for every $i \in \{1, \ldots, k\}$. We thus run an algorithm solving $f$-Ramsey-extracted Iterative Expansion MIS on this input, which concludes the algorithm. ◀

The proof of the following result is immediate, by using successively Lemmas 7 and 10.

▶ **Theorem 11.** *Let $\mathcal{G}$ be a hereditary graph class. If $f$-*Ramsey-extracted Iterative Expansion MIS *is FPT in $\mathcal{G}$ for some computable function $f$, then* MIS *is FPT in $\mathcal{G}$.*

We now apply this framework to two families of graphs $H$.

## 4.2 Clique minus a smaller clique

▶ **Theorem 12.** *($\star$) For any $r \geq 2$ and $s < r$,* MIS *in $(K_r \setminus K_s)$-free graphs is FPT if $s \leq 3$, and $W[1]$-hard otherwise.*

## 4.3 Clique minus a complete bipartite graph

For every three positive integers $r$, $s_1$, $s_2$ with $s_1 + s_2 < r$, we consider the graph $K_r \setminus K_{s_1,s_2}$. Another way to see $K_r \setminus K_{s_1,s_2}$ is as a $P_3$ *of cliques* of size $s_1$, $r - s_1 - s_2$, and $s_2$. More formally, every graph $K_r \setminus K_{s_1,s_2}$ can be obtained from a $P_3$ by adding $s_1 - 1$ false twins of the first vertex, $r - s_1 - s_2 - 1$, for the second, and $s_2 - 1$, for the third.

▶ **Theorem 13.** *$\forall r \geq 2$ and $s_1 \leq s_2$ s.t. $s_1 + s_2 < r$,* MIS *in $K_r \setminus K_{s_1,s_2}$-free graphs is FPT.*

**Proof.** It is more convenient to prove the result for $K_{3r} \setminus K_{r,r}$-free graphs, for any positive integer $r$. It implies the theorem by choosing this new $r$ to be larger than $s_1$, $s_2$, and $r - s_1 - s_2$. We will show that for $f(x) := 3r$ for every $x \in \mathbb{N}$, $f$-Ramsey-extracted Iterative Expansion MIS in $K_{3r} \setminus K_{r,r}$-free graphs is FPT. By Theorem 11, this implies that MIS is FPT in this class. Let $C_1, \ldots, C_{k-1}$ (whose union is denoted by $\mathcal{C}$) be the Ramsey-extracted cliques of size $3r$, which can be partitionned, as in Definition 9, into $3r$ independent sets $S_1, \ldots, S_{3r}$, each of size $k-1$. Let $\mathcal{X} = \bigcup_{i=1}^{k} X_i$ be the set in which we are looking for an independent set of size $k$. We recall that between any $X_i$ and any $C_j$ there are either all the edges or none. Hence, the whole interaction between $\mathcal{X}$ and $\mathcal{C}$ can be described by the bipartite graph $\mathcal{B}$ described in Definition 9. Firstly, we can assume that each $X_i$ is of

size at least $Ram(r,k)$, otherwise we can branch on $Ram(r,k)$ choices to find one vertex in an optimum solution. By Ramsey's theorem, we can assume that each $X_i$ contains a clique of size $r$ (if it contains an independent set of size $k$, we are done). Our general strategy is to leverage the fact that the input graph is $(K_{3r} \setminus K_{r,r})$-free to describe the structure of $\mathcal{X}$. Hopefully, this structure will be sufficient to solve our problem in FPT time.

We define an auxiliary graph $Y$ with $k-1$ vertices. The vertices $y_1, \ldots, y_{k-1}$ of $Y$ represent the Ramsey-extracted cliques of $\mathcal{C}$ and two vertices $y_i$ and $y_j$ are adjacent iff the relation between $C_i$ and $C_j$ is not empty (equivalently the relation is full or semi-full). It might seem peculiar that we concentrate the structure of $\mathcal{C}$, when we will eventually discard it from the graph. It is an indirect move: the simple structure of $\mathcal{C}$ will imply that the interaction between $\mathcal{X}$ and $\mathcal{C}$ is simple, which in turn, will severely restrict the subgraph induced by $\mathcal{X}$. More concretely, in the rest of the proof, we will (1) show that $Y$ is a clique, (2) deduce that $\mathcal{B}$ is a complete bipartite graph, (3) conclude that $\mathcal{X}$ cannot contain an induced $K_r^2 = K_r \uplus K_r$ and run the algorithm of Theorem 5.

Suppose that there is $y_{i_1}y_{i_2}y_{i_3}$ an induced $P_3$ in $Y$, and consider $C_{i_1}$, $C_{i_2}$, $C_{i_3}$ the corresponding Ramsey-extracted cliques. For $s < t \in [3r]$, let $C_i^{s\to t} := C_i \cap \bigcup_{s\leqslant j\leqslant t} S_j$. In other words, $C_i^{s\to t}$ contains the elements of $C_i$ having indices between $s$ and $t$. Since $|C_i| = 3r$, each $C_i$ can be partitionned into three sets, of $r$ elements each: $C_i^{1\to r}$, $C_i^{r+1\to 2r}$ and $C_i^{2r+1\to 3r}$. Recall that the relation between $C_{i_1}$ and $C_{i_2}$ (resp. $C_{i_2}$ and $C_{i_3}$) is either full or semi-full, while the relation between $C_{i_1}$ and $C_{i_3}$ is empty. This implies that at least one of the four following sets induces a graph isomorphic to $K_{3r} \setminus K_{r,r}$:

- $C_{i_1}^{1\to r} \cup C_{i_2}^{r+1\to 2r} \cup C_{i_3}^{1\to r}$
- $C_{i_1}^{1\to r} \cup C_{i_2}^{r+1\to 2r} \cup C_{i_3}^{2r+1\to 3r}$
- $C_{i_1}^{2r+1\to 3r} \cup C_{i_2}^{r+1\to 2r} \cup C_{i_3}^{1\to r}$
- $C_{i_1}^{2r+1\to 3r} \cup C_{i_2}^{r+1\to 2r} \cup C_{i_3}^{2r+1\to 3r}$

Hence, $Y$ is a disjoint union of cliques. Let us assume that $Y$ is the union of at least two (maximal) cliques.

Recall that the bipartite graph $\mathcal{B}$ is connected. Thus there is $b_h^1 \in B_1$ (corresponding to $X_h$) adjacent to $b_i^2 \in B_2$ and $b_j^2 \in B_2$ (corresponding to $C_i$ and $C_j$, respectively), such that $y_i$ and $y_j$ lie in two different connected components of $Y$ (in particular, the relation between $C_i$ and $C_j$ is empty). Recall that $X_h$ contains a clique of size at least $r$. This clique induces, together with any $r$ vertices in $C_i$ and any $r$ vertices in $C_j$, a graph isomorphic to $K_{3r} \setminus K_{r,r}$; a contradiction. Hence, $Y$ is a clique.

Now, we can show that $\mathcal{B}$ is a complete bipartite graph. Each $X_h$ has to be adjacent to at least one $C_i$ (otherwise this trivially contradicts the connectedness of $\mathcal{B}$). If $X_h$ is not linked to $C_j$ for some $j \in \{1, \ldots, k-1\}$, then a clique of size $r$ in $X_h$ (which always exists) induces, together with $C_i^{1\to r} \cup C_j^{2r+1\to 3r}$ or with $C_i^{2r+1\to 3r} \cup C_j^{1\to r}$, a graph isomorphic to $K_{3r} \setminus K_{r,r}$.

Since $\mathcal{B}$ is a complete bipartite graph, every vertex of $C_1$ dominates all vertices of $\mathcal{X}$ In particular, $\mathcal{X}$ is in the intersection of the neighborhood of the vertices of some clique of size $r$. This implies that the subgraph induced by $\mathcal{X}$ is $(K_r \uplus K_r)$-free. Hence, we can run the FPT algorithm of Theorem 5 on this graph. ◀

## 5 Polynomial (Turing) kernels

In this section we investigate some special cases of Section 4.3, in particular when $H$ is a clique of size $r$ minus a claw with $s$ branches, for $s < r$. Although Theorem 13 proves that MIS is FPT for every possible values of $r$ and $s$, we show that when $s \geq r-2$, the problem

admits a polynomial Turing kernel, while for $s \leq 2$, it admits a polynomial kernel. Notice that the latter result is somehow tight, as Corollary 18 shows that MIS cannot admit a polynomial kernel in $(K_r \setminus K_{1,s})$-free graphs whenever $s \geq 3$.

▶ **Theorem 14.** *($\star$) $\forall r \geq 2$,* MIS *in $(K_r \setminus K_{1,r-2})$-free graphs has a polynomial Turing kernel.*

▶ **Theorem 15.** *($\star$) $\forall r \geq 3$,* MIS *in $(K_r \setminus K_{1,2})$-free graphs has a kernel with $O(k^{r-1})$ vertices.*

Observe that a $(K_r \setminus K_2)$-free graph is $(K_{r+1} \setminus K_{1,2})$-free, hence, thus the previous result also applies to $(K_r \setminus K_2)$-free graphs, which answers a question of [8].

We now focus on kernel lower bounds.

▶ **Definition 16.** Given the graphs $H$, $H_1, \ldots, H_p$, we say that $(H_1, \ldots, H_p)$ is a multipartite decomposition of $H$ if $H$ is isomorphic to $H_1 + \cdots + H_p$. We say that $(H_1, \ldots, H_p)$ is maximal if, for every multipartite decomposition $(H'_1, \ldots, H'_q)$ of $H$, we have $p > q$.

It can easily be seen that for every graph $H$, a maximal multipartite decomposition of $H$ is unique. We have the following:

▶ **Theorem 17.** *($\star$) Let $H$ be any fixed graph, and let $H = H_1 + \cdots + H_p$ be the maximal multipartite decomposition of $H$. If, for some $i \in [p]$,* MIS *is NP-hard in $H_i$-free graphs, then* MIS *does not admit a polynomial kernel in $H$-free graphs unless NP $\subseteq$ coNP/poly.*

The next results shows that the polynomial kernel obtained in the previous section for $(K_r \setminus K_{1,s})$-free graphs, $s \leq 2$, is somehow tight.

▶ **Corollary 18.** *($\star$) For $r \geq 4$, and every $3 \leq s \leq r-1$,* MIS *in $(K_r \setminus K_{1,s})$-free graphs does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.*

## 6 Conclusion and open problems

We started to unravel the FPT/$W[1]$-hard dichotomy for MIS in $H$-free graphs, for a fixed graph $H$. At the cost of one reduction, we showed that it is $W[1]$-hard as soon as $H$ is not chordal, even if we simultaneously forbid induced $K_{1,4}$ and trees with at least two branching vertices. Tuning this construction, it is also possible to show that if a connected $H$ is not roughly a "path of cliques" or a "subdivided claw of cliques", then MIS is $W[1]$-hard.

An interesting open problem is the case when $H$ is the *cricket*, that is a triangle with two pending vertices, each attached to a different vertex

For disconnected graphs $H$, we obtained an FPT algorithm when $H$ is a cluster (*i.e.*, a disjoint union of cliques). We conjecture that, more generally, the disjoint union of two easy cases is an easy case; formally, *if* MIS *is FPT in $G$-free graphs and in $H$-free graphs, then it is FPT in $G \uplus H$-free graphs.* A more anecdotal conclusion is the fact that the parameterized complexity of the problem on $H$-free graphs is now complete for every graph $H$ on four vertices, including concerning the polynomial kernel question, whereas the $FPT/W[1]$-hard question remains open for only five graphs $H$ on five vertices.

## References


1. V. Alekseev. The Effect of Local Constraints on the Complexity of Determination of the Graph Independence Number. *Combinatorial-Algebraic Methods in Applied Mathematics*, pages 3–13, 1982. in Russian.
2. V. E. Alekseev. On the number of maximal independent sets in graphs from hereditary classes. *Combinatorial-Algebraic Methods in Discrete Optimization*, pages 5–8, 1991. (In Russian).
3. G. Bacsó, D. Lokshtanov, D. Marx, M. Pilipczuk, Z. Tuza, and E. Jan van Leeuwen. Subexponential-time Algorithms for Maximum Independent Set in $P_t$-free and Broom-free Graphs. *CoRR*, abs/1804.04077, 2018.
4. É. Bonnet, N. Bousquet, P. Charbit, S. Thomassé, and R. Watrigant. Parameterized Complexity of Independent Set in H-Free Graphs. *CoRR*, 2018. `arXiv:1810.04620`.
5. J. Chen, Y. L., S. Lu, S. S., and F. Zhang. Iterative Expansion and Color Coding: An Improved Algorithm for 3D-Matching. *ACM Trans. Algorithms*, 8(1):6:1–6:22, 2012.
6. M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.
7. K. Dabrowski. *Structural Solutions to Maximum Independent Set and Related Problems*. PhD thesis, University of Warwick, 2012.
8. K. Dabrowski, V. V. Lozin, H. Müller, and D. Rautenbach. Parameterized complexity of the weighted independent set problem beyond graphs of bounded clique number. *J. Discrete Algorithms*, 14:207–213, 2012.
9. R. Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.
10. R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.
11. H. Perret du Cray and I. Sau. Improved FPT algorithms for weighted independent set in bull-free graphs. *Discrete Mathematics*, 341(2):451–462, 2018.
12. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
13. A. Grzesik, T. Klimosova, M. Pilipczuk, and M. Pilipczuk. Polynomial-time algorithm for Maximum Weight Independent Set on $P_6$-free graphs. *CoRR*, abs/1707.05491, 2017.
14. D. Hermelin, M. Mnich, and E. J. van Leeuwen. Parameterized Complexity of Induced Graph Matching on Claw-Free Graphs. *Algorithmica*, 70(3):513–560, 2014.
15. T. Karthick. Independent Sets in Some Classes of $S_{i,j,k}$-free Graphs. *J. Comb. Optim.*, 34(2):612–630, August 2017.
16. T. Karthick and F. Maffray. Maximum weight independent sets in classes related to claw-free graphs. *Discrete Applied Mathematics*, 216:233–239, 2017.
17. D. Lokshtanov, M. Vatshelle, and Y. Villanger. Independent Set in $P_5$-Free Graphs in Polynomial Time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014*, pages 570–581, 2014.
18. B. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.
19. S. Thomassé, N. Trotignon, and K. Vuskovic. A Polynomial Turing-Kernel for Weighted Independent Set in Bull-Free Graphs. *Algorithmica*, 77(3):619–641, 2017.
20. D. Zuckerman. Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number. *Theory of Computing*, 3(1):103–128, 2007.

# Multi-Budgeted Directed Cuts

**Stefan Kratsch**
Institut für Informatik, Humboldt-Universität zu Berlin
kratsch@informatik.hu-berlin.de

**Shaohua Li**
Institute of Informatics, University of Warsaw
Shaohua.Li@mimuw.edu.pl

**Dániel Marx**
Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)
dmarx@cs.bme.hu

**Marcin Pilipczuk**
Institute of Informatics, University of Warsaw
malcin@mimuw.edu.pl

**Magnus Wahlström**[1]
Royal Holloway, University of London
Magnus.Wahlstrom@rhul.ac.uk

## Abstract

In this paper, we study multi-budgeted variants of the classic minimum cut problem and graph separation problems that turned out to be important in parameterized complexity: Skew Multicut and Directed Feedback Arc Set. In our generalization, we assign colors $1, 2, ..., \ell$ to some edges and give separate budgets $k_1, k_2, ..., k_\ell$ for colors $1, 2, ..., \ell$. For every color $i \in \{1, ..., \ell\}$, let $E_i$ be the set of edges of color $i$. The solution $C$ for the multi-budgeted variant of a graph separation problem not only needs to satisfy the usual separation requirements (i.e., be a cut, a skew multicut, or a directed feedback arc set, respectively), but also needs to satisfy that $|C \cap E_i| \leq k_i$ for every $i \in \{1, ..., \ell\}$.

Contrary to the classic minimum cut problem, the multi-budgeted variant turns out to be NP-hard even for $\ell = 2$. We propose FPT algorithms parameterized by $k = k_1 + ... + k_\ell$ for all three problems. To this end, we develop a branching procedure for the multi-budgeted minimum cut problem that measures the progress of the algorithm not by reducing $k$ as usual, by but elevating the capacity of some edges and thus increasing the size of maximum source-to-sink flow. Using the fact that a similar strategy is used to enumerate all important separators of a given size, we merge this process with the flow-guided branching and show an FPT bound on the number of (appropriately defined) important multi-budgeted separators. This allows us to extend our algorithm to the Skew Multicut and Directed Feedback Arc Set problems.

Furthermore, we show connections of the multi-budgeted variants with weighted variants of the directed cut problems and the Chain $\ell$-SAT problem, whose parameterized complexity remains an open problem. We show that these problems admit a bounded-in-parameter number of "maximally pushed" solutions (in a similar spirit as important separators are maximally pushed), giving somewhat weak evidence towards their tractability.

**2012 ACM Subject Classification** Theory of computation → Fixed parameter tractability

**Keywords and phrases** important separators, multi-budgeted cuts, Directed Feedback Vertex Set, fixed-parameter tractability, minimum cut

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.18

[1] Supported by EPSRC grant EP/P007228/1

© Stefan Kratsch, Shaohua Li, Dániel Marx, Marcin Pilipczuk, and Magnus Wahlström;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 18; pp. 18:1–18:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

**Related Version** A full version of the paper is available at https://arxiv.org/abs/1810.06848.

**Funding** This research is a part of projects that have received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreements No 714704 (S. Li and M. Pilipczuk), 280152 (D. Marx), and 725978 (D. Marx).

## 1 Introduction

Graph separation problems are important topics in both theoretical area and applications. Although the famous minimum cut problem is known to be polynomial-time solvable, many well-known variants are NP-hard, which are intensively studied from the point of view of approximation [1, 2, 11, 13, 14, 18] and, what is more relevant for this work, parameterized complexity.

The notion of important separators, introduced by Marx [22], turned out to be fundamental for a number of graph separation problems such as Multiway Cut [22], Directed Feedback Vertex Set [4], or Almost 2-CNF SAT [27]. Further work, concerning mostly undirected graphs, resulted in a wide range of involved algorithmic techniques: applications of matroid techniques [19, 20], shadow removal [8, 25], randomized contractions [5], LP-guided branching [10, 15, 16, 17], and treewidth reduction [24], among others.

From the above techniques, only the notion of important separators and the related technique of shadow removal generalizes to directed graphs, giving FPT algorithms for Directed Feedback Arc Set [4], Directed Multiway Cut [8], and Directed Subset Feedback Vertex Set [7]. As a result, the parameterized complexity of a number of important graph separation problems in directed graphs remains open, and the quest to investigate them has been put on by the third author in a survey from 2012 [23]. Since the publication of this survey, two negative answers have been obtained. Two authors of this work showed that Directed Multicut is W[1]-hard even for four terminal pairs (leaving the case of three terminal pairs open) [26], while Lokshtanov et al. [21] showed intractability of Directed Odd Cycle Transversal.

During an open problem session at Recent Advancements in Parameterized Complexity school (December 2017) [12], Saurabh posed the question of parameterized complexity of a weighted variant of Directed Feedback Arc Set, where given a directed edge-weighted graph $G$, an integer $k$, and a target weight $w$, the goal is to find a set $X \subseteq E(G)$ such that $G - X$ is acyclic and $X$ is of cardinality at most $k$ and weight at most $w$. Consider a similar problem Weighted $st$-cut: given a directed graph $G$ with positive edge weights and two distinguished vertices $s, t \in V(G)$, an integer $k$, and a target weight $w$, decide if $G$ admits an $st$-cut of cardinality at most $k$ and weight at most $w$. The parameterized complexity of this problem parameterized by $k$ is open even if $G$ is restricted to be acyclic, while with this restriction the problem can easily be reduced to Directed Feedback Arc Set (add an arc $(t, s)$ of prohibitively large weight).

The Weighted $st$-cut problem becomes similar to another directed graph cut problem, identified in [6], namely Chain $\ell$-SAT. While this problem is originally formulated in CSP language, the graph formulation is as follows: given a directed graph $G$ with a partition of edge set $E(G) = P_1 \uplus P_2 \uplus \ldots \uplus P_m$ such that each $P_i$ is an edge set of a simple path of length at most $\ell$ (the input paths could have common nodes), an integer $k$, and two vertices $s, t \in V(G)$, find an $st$-cut $C \subseteq E(G)$ such that $|\{i | C \cap P_i \neq \emptyset\}| \leq k$. This problem can easily be seen to be equivalent to minimum $st$-cut problem (and thus polynomial-time solvable) for $\ell \leq 2$, but is NP-hard for $\ell \geq 3$ and its parameterized complexity (with $k$ as a parameter) remains an open problem.

In this paper we make progress towards resolving the question of parameterized complexity of the two aforementioned problems: weighted $st$-cut problem (in general digraphs, not necessary acyclic ones) and Chain $\ell$-SAT. Our contribution is twofold.

### Multi-budgeted variant

We define a *multi-budgeted* variant of a number of cut problems (including the minimum cut problem) and show its fixed-parameter tractability. In this variant, the edges of the graph are colored with $\ell$ colors, and the input specifies separate budgets for each color. More formally, we primarily consider the following problem.

Multi-budgeted cut
**Input:** A directed graph $G$, two disjoint sets of vertices $X, Y \subseteq V(G)$, an integer $\ell$, and for every $i \in \{1, 2, \ldots, \ell\}$ a set $E_i \subseteq E(G)$ and an integer $k_i \geq 1$.
**Question:** Is there a set of arcs $C \subseteq \bigcup_{i=1}^{\ell} E_i$ such that there is no directed $X - Y$ path in $G \setminus C$ and for every $i \in [\ell]$, $|C \cap E_i| \leq k_i$.

Similarly we can define multi-budgeted variants of Directed Feedback Arc Set and Skew Multicut.

We observe that Multi-budgeted cut for $\ell = 2$ reduces to Weighted $st$-cut as follows. Let $(G, X, Y, E_1, E_2, k_1, k_2)$ be a Multi-budgeted cut instance for $\ell = 2$. First, observe that we may assume that $E_1 \cap E_2 = \emptyset$, as we can replace every edge $e \in E_1 \cap E_2$ with two copies $e_1 \in E_1 \setminus E_2$ and $e_2 \in E_2 \setminus E_1$. Second, construct an equivalent Weighted $st$-cut instance $(G', s, t, k, w)$ as follows. To construct $G'$, first add two vertices $s, t$ to $G$ and edges $\{(s, x) | x \in X\}$ and $\{(y, t) | y \in Y\}$ of prohibitively large weight. Assign also prohibitively large weight to every edge $e \in E(G) \setminus (E_1 \cup E_2)$. Assign weight $(k_1 + 1)k_2 + 1$ to every edge $e \in E_1$. For every edge $e \in E_2$, add $k_1 + 1$ copies of $e$ to $G'$ of weight 1 each. Finally, set $k := (k_1 + 1) \cdot k_2 + k_1$ as the cardinality bound and $w := k_1((k_1 + 1)k_2 + 1) + (k_1 + 1)k_2$ as the target weight. The equivalence of the instances follows from the fact that the cardinality bound allows to pick in the solution at most $k_2$ bundles of $k_1 + 1$ copies of an edge of $E_2$, while the weight bound allows to pick only $k_1$ edges of $E_1$.

Thus, Multi-budgeted cut for $\ell = 2$ corresponds to the case of Weighted $st$-cut where the weights are integral and both target cardinality and weight are bounded in parameter.[2] This connection was our primary motivation to study the multi-budgeted variants of the cut problems.

Contrary to the classic minimum cut problem, we note that Multi-budgeted Cut becomes NP-hard for $\ell \geq 2$ by a simple reduction from constrained minimum vertex cover problem on bipartite graphs [3].[3] We show that Multi-budgeted Cut is FPT when parameterized by $k = k_1 + ... + k_\ell$. For this problem, our branching strategy is as follows. First, note that in the problem definition we assume that each $k_i$ is positive, and thus $\ell \leq k$. A standard application of the Ford-Fulkerson algorithm gives a minimum $XY$-cut $C$ of size $\lambda$ and $\lambda$ edge-disjoint $X - Y$ paths $P_1, P_2, \ldots, P_\lambda$. If $C$ is a solution, then we are done. Similarly, if $\lambda > k$, then there is no solution. Otherwise, we branch which colors of the sought

[2] For a reduction in the other direction, replace every arc $e$ of weight $\omega(e)$ with one copy of color 1 and $\omega(e)$ copies of color 2, and set budgets $k_1 = k$ and $k_2 = w$.

[3] We believe this problem must have been formulated already before and proven to be NP-hard. However, we were not able to find it in the literature. Our own reduction will be available in the full version of the paper.

solution should appear on each paths $P_j$; that is, for every $i \in [\ell]$ and $j \in [\lambda]$, we guess if $P_j \cap E_i$ contains an edge of the sought solution, and in each guess assign infinite capacities to the edges of wrong color. If this change increased the size of a maximum flow from $X$ to $Y$, then we can charge the branching step to this increase, as the size of the flow cannot exceed $k$. The critical insight is that if the size of the minimum flow does not increase (i.e., $P_1, \ldots, P_\lambda$ remains a maximum flow), then a corresponding minimum cut is necessarily a solution. As a result, we obtain the following.

▶ **Theorem 1.** Multi-budgeted Cut *admits an FPT algorithm with running time bound* $\mathcal{O}(2^{k^2\ell} \cdot k \cdot (|V(G)| + |E(G)|))$ *where* $k = \sum_{i=1}^{\ell} k_i$.

The charging of the branching step to a flow increase appears also in the classic argument for bound of the number of important separators [4] (see also [9, Chapter 8]). We observe that our branching algorithm can be merged with this procedure, yielding a bound (as a function of $k$) and enumeration procedure of naturally defined multi-budgeted important separators. This in turn allows us to generalize our FPT algorithm to Multi-budgeted Skew Multicut and Multi-budgeted Directed Feedback Arc Set.

▶ **Theorem 2.** Multi-budgeted Skew Multicut *and* Multi-budgeted Directed Feedback Arc Set *admit FPT algorithms with running time bound* $2^{\mathcal{O}(k^3 \log k)}(|V(G)| + |E(G)|)$ *where* $k = \sum_{i=1}^{\ell} k_i$.

### Bound on the number of pushed solutions

While we are not able to establish fixed-parameter tractability of the weighted variant of the minimum cut problem (even in acyclic graphs) nor of Chain $\ell$-SAT, we show the following graph-theoretic statement. Consider a directed graph $G$ with two distinguished vertices $s, t \in V(G)$. For two (inclusion-wise) minimal $st$-cuts $C_1$, $C_2$ we say that $C_1$ *is closer to* $t$ than $C_2$ if every vertex reachable from $s$ in $G - C_2$ is also reachable from $s$ in $G - C_1$. A classic submodularity argument implies that there is exactly one closest to $t$ minimum $st$-cut, while the essence of the notion of important separators is the observation that there is bounded-in-$k$ number of minimal separators of cardinality at most $k$ that are closest to $t$. In Section 5 we show a similar existential statement for the two discussed problems.

▶ **Theorem 3.** *For every integer $k$ there exists an integer $g$ such that the following holds. Let $G$ be a directed graph with positive edge weights and two distinguished vertices $s, t \in V(G)$. Let $\mathcal{F}$ be a family of all $st$-cuts that are of minimum weight among all (inclusion-wise) minimal $st$-cuts of cardinality at most $k$. Let $\mathcal{G} \subseteq \mathcal{F}$ be the family of those cuts $C$ such that no other cut of $\mathcal{F}$ is closer to $t$. Then $|\mathcal{G}| \leq g$.*

▶ **Theorem 4.** *For every integers $k, \ell$ there exists an integer $g'$ such that the following holds. Let $I := (G, s, t, (P_i)_{i=1}^m, k)$ be a* Chain $\ell$-SAT *instance that is a yes-instance but $(G, s, t, (P_i)_{i=1}^m, k-1)$ is a no-instance. Let $\mathcal{F}$ be a family of all (inclusion-wise) minimal solutions to $I$ and let $\mathcal{G} \subseteq \mathcal{F}$ be the family of those cuts $C$ such that no other cut of $\mathcal{F}$ is closer to $t$. Then $|\mathcal{G}| \leq g'$.*

Unfortunately, our proof is purely existential, and does not yield an enumeration procedure of the "closest to $t$" solutions.

### Organization

In this extended abstract, we prove Theorem 1 in Section 3, present the multi-budgeted extension of the notion of important separators (needed for Theorem 2) in Section 4, and sketch the proofs of Theorems 3 and 4 in Section 5.

## 2 Preliminaries

For an integer $n$, we denote $[n] = \{1, 2, \ldots, n\}$. For a directed graph $G$, we use $V(G)$ to represent the set of vertices of $G$ and $E(G)$ to represent the set of directed edges of $G$. In all multi-budgeted problems, the directed graph $G$ comes with sets $E_i \subseteq E(G)$ for $i \in [\ell]$ which we refer as *colors*. That is, an edge $e$ is *of color* $i$ if $e \in E_i$, and of *no color* if $e \in E(G) \setminus \bigcup_{i=1}^{\ell} E_i$. Note that an edge may have many colors, as we do not insist on the sets $E_i$ being pairwise disjoint.

Let $X$ and $Y$ be two disjoint vertex sets in a directed graph $G$, an $XY$-*cut* of $G$ is a set of edges $C$ such that every directed path from a vertex in $X$ to a vertex in $Y$ contains an edge of $C$. A cut $C$ is *minimal* if no proper subset of $C$ is an $XY$-cut, and *minimum* if $C$ is of minimum possible cardinality. Let $C$ be an $XY$-cut and let $R$ be the set of vertices reachable from $X$ in $G \setminus C$. We define $\delta^+(R) = \{(u, v) \in E(G) | u \in R \text{ and } v \notin R\}$ and note that if $C$ is minimal, then $\delta^+(R) = C$.

Let $(G, X, Y, \ell, (E_i, k_i)_{i=1}^{\ell})$ be a Multi-budgeted cut instance and let $C$ be an $XY$-cut. We say that $C$ is *budget-respecting* if $C \subseteq \bigcup_{i=1}^{\ell} E_i$ and $|C \cap E_i| \leq k_i$ for every $i \in [\ell]$. For a set $Z \subseteq E(G)$ we say that $C$ is $Z$-*respecting* if $C \subseteq Z$. In such contexts, we often call $Z$ the *set of deletable edges*. An $XY$-cut $C$ is a *minimum* $Z$-*respecting cut* if it is a $Z$-respecting $XY$-cut of minimum possible cardinality among all $Z$-respecting $XY$-cuts.

Our FPT algorithms start with $Z = \bigcup_{i=1}^{\ell} E_i$ and in branching steps shrink the set $Z$ to reduce the search space. We encapsulate our use of the classic Ford-Fulkerson algorithm in the following statement.

▶ **Theorem 5.** *Given a directed graph $G$, two disjoint sets $X, Y \subseteq V(G)$, a set $Z \subseteq E(G)$, and an integer $k$, one can in $\mathcal{O}(k(|V(G)| + |E(G)|))$ time either find the following objects:*

- *$\lambda$ paths $P_1, P_2, \ldots, P_\lambda$ such that every $P_i$ starts in $X$ and ends in $Y$, and every edge $e \in Z$ appears on at most one path $P_i$;*
- *a set $B \subseteq Z$ consisting of all edges of $G$ that participate in some minimum $Z$-respecting $XY$-cut;*
- *a minimum $Z$-respecting $XY$-cut $C$ of size $\lambda$ that is closest to $Y$ among all minimum $Z$-respecting $XY$-cuts;*

*or correctly conclude that there is no $Z$-respecting $XY$-cut of cardinality at most $k$.*

## 3 Multi-budgeted cut

We now give an FPT algorithm parameterized by $k = \Sigma_{i=1}^{\ell} k_i$ for the Multi-budgeted cut problem. We follow a branching strategy that recursively reduces a set $Z$ of deletable edges. That is, we start with $Z = \bigcup_{i=1}^{\ell} E_i$ (so that every solution is initially $Z$-respecting) and in each recursive step, we look for a $Z$-respecting solution and reduce the set $Z$ in a branching step.

Consider a recursive call where we look for a $Z$-respecting solution to the input Multi-budgeted cut instance $(G, X, Y, \ell, (E_i, k_i)_{i=1}^{\ell})$. That is, we look for a $Z$-respecting budget-respecting cut. We apply Theorem 5 to it. If it returns that there is no $Z$-respecting $XY$-cut of size at most $k$, we terminate the current branch, as there is no solution. Otherwise, we obtain the paths $P_1, P_2, \ldots, P_\lambda$, the set $B$ (which we will not use in this section), and the cut $C$.

If $C$ is budget-respecting, then it is a solution and we can return it. Otherwise, we perform the following branching step. We iterate over all tuples $(A_1, ..., A_\ell)$ such that for every $i \in [\ell]$, $A_i \subseteq [\lambda]$ and $|A_i| \leq k_i$. $A_i$ represents the subset of paths $P_1, ..., P_\lambda$ on which at

**MultiBudgetedCut**$(G, X, Y, \ell, (E_i, k_i)_{i=1}^{\ell})$
Input: A directed graph $G$, two disjoint set of vertices $X, Y \subseteq V(G)$, an integer $\ell$, for every $i \in [\ell]$ a set $E_i \subseteq E(G)$ and an integer $k$.
Output: an $XY$ cut $C \subseteq \bigcup_{i=1}^{\ell} E_i$ such that for every $i \in [\ell]$, $|C \cap E_i| \leq k_i$ if it exists, otherwise return NO.

1. $Z := \bigcup_{i=1}^{\ell} E_i$;
2. **return Solve**$(Z)$;

**Solve**$(Z)$
a. apply Theorem 5 to $(G, X, Y, k, Z)$ where $k = \sum_{i=1}^{\ell} k_i$, obtaining objects $(P_i)_{i=1}^{\lambda}$, $B$, and $C$, or an answer NO;
b. **if** the answer NO is obtained, **then return** NO;
c. **if** $C$ is budget-respecting, **then return** $C$;
d. **for each** $(A_1, ..., A_\ell)$ such that for every $i$ in $[\ell]$, $A_i \subseteq [\lambda]$ and $|A_i| \leq k_i$ **do**
d.1 $\widehat{Z} := Z$;
d.2 **for each** $i \in [\ell]$ **do**
  **for each** $j \in [\lambda] \setminus A_i$ **do**
    $\widehat{Z} := \widehat{Z} \setminus (E_i \cap E(P_j))$;
d.3 $D =$ **Solve**$(\widehat{Z})$;
d.4 **if** $D \neq$NO **then return** $D$;
e. **return** NO;

**Figure 1** FPT algorithm for Multi-budgeted cut.

least one edge of color $i$ is in the solution for each $i \in [\ell]$. For those edges of color $i$ which are on the paths not indicated by $A_i$, they are not in the solution. Thus we can safely delete them from $Z$. More formally, for every $i \in [\ell]$ and $j \in [\lambda] \setminus A_i$, we remove from $Z$ all edges of $E(P_j) \cap E_i$. We recurse on the reduced set $Z$. A pseudocode is available in Figure 1.

▶ **Theorem 6.** *The algorithm in Figure 1 for* Multi-budgeted cut *is correct and runs in time* $O(2^{\ell k^2} \cdot k \cdot (|V(G)| + |E(G)|))$ *where* $k = \Sigma_{i=1}^{\ell} k_i$.

**Proof.** We prove the correctness of the algorithm by showing that it returns a solution if and only if the input instance is a yes-instance. The "only if" direction is obvious, as the algorithm returns only $Z$-respecting budget-respecting $XY$-cuts and $Z \subseteq \bigcup_{i=1}^{\ell} E_i$ in each recursive call.

We prove the correctness for the "if" direction. Let $C_0$ be a solution, that is, a budget-respecting $XY$-cut. In the initial call to **Solve**, $C_0$ is $Z$-respecting. It suffices to inductively show that in each call to **Solve** such that $C_0$ is $Z$-respecting, either the call returns a solution, or $C_0$ is $\widehat{Z}$-respecting for at least one of the subcalls. Since $C_0$ is $Z$-respecting, the application of Theorem 5 returns objects $(P_i)_{i=1}^{\lambda}$, $B$, and $C$. If $C$ is budget-respecting, then the algorithm returns it and we are done. Otherwise, consider the branch $(A_1, A_2, \ldots, A_\ell)$ where $A_i = \{j | E(P_j) \cap C_0 \neq \emptyset\}$. Since $C_0$ is budget-respecting, $C_0 \subseteq Z$, and no edge of $Z$ appears on more than one path $P_j$, we have $|A_i| \leq k_i$ for every $i \in [\ell]$. Thus, $(A_1, A_2, \ldots, A_\ell)$ is a branch considered by the algorithm. In this branch, the algorithm refines the set $Z$ to $\widehat{Z}$. By the definition of $A_i$, for every $i \in [\ell]$ and $j \in [\lambda] \setminus A_i$, we have $C_0 \cap E_i \cap E(P_j) = \emptyset$. Consequently, $C_0$ is $\widehat{Z}$-respecting and we are done.

For the time bound, the following observation is crucial.

▶ **Claim 7.** *Consider one recursive call* ***Solve***$(Z)$ *where the application of Theorem 5 in line (a) returned objects* $(P_i)_{i=1}^{\lambda}$, $B$ *and* $C$. *Assume that in some recursive subcall* ***Solve***$(\widehat{Z})$ *invoked in line (d.3) (Figure 1), the subsequent application of Theorem 5 in line (a) of the*

*subcall returned a cut of the same size, that is, the algorithm of Theorem 5 returned a cut $\widehat{C}$ of size $\widehat{\lambda} = \lambda$. Then the cut $\widehat{C}$ is budget-respecting and, consequently, is returned in line (c) of the subcall.*

**Proof.** Since $|\widehat{C}| = \lambda$ is a $\widehat{Z}$-respecting $XY$-cut, $\widehat{Z} \subseteq Z$, and every edge $e \in Z$ appears on at most one path $P_i$, we have that $\widehat{C}$ consists of exactly one edge of $\widehat{Z}$ on every path $P_i$, that is, $\widehat{C} = \{e_1, e_2, \ldots, e_\lambda\}$ and $e_j \in E(P_j) \cap \widehat{Z}$ for every $j \in [\lambda]$. In other words, the paths $(P_j)_{j=1}^{\lambda}$ still correspond to a maximum flow from $X$ to $Y$ with edges of $\widehat{Z}$ being of unit capacity and edges outside $\widehat{Z}$ of infinite capacity because $(P_j)_{j=1}^{\lambda}$ are paths satisfying that any two of them are disjoint on $\widehat{Z} \subseteq Z$ and $\lambda$ is still equal to the size of the maximum flow. If $e_j \in E_i$ for some $j \in [\lambda]$ and $i \in [\ell]$, then by the construction of set $\widehat{Z}$, we have $j \in A_i$. Consequently, $|\{j | e_j \in E_i\}| \leq |A_i| \leq k_i$ for every $i \in [\ell]$, and thus $\widehat{C}$ is budget-respecting. ⌟

Claim 7 implies that the depth of the search tree is bounded by $k$, as the algorithm terminates when $\lambda$ exceeds $k$. At every step, there are at most $(2^\lambda)^\ell \leq (2^k)^\ell$ different tuples $(A_1, ..., A_\ell)$ to consider. Consequently, there are $\mathcal{O}(2^{(k-1)k\ell})$ nodes of the search tree that enter the loop in line (d) and $\mathcal{O}(2^{k^2\ell})$ nodes that invoke the algorithm of Theorem 5. As a result, the running time of the algorithm is $\mathcal{O}(2^{\ell k^2} \cdot k \cdot (|V(G)| + |E(G)|))$. ◀

## 4 Multi-budgeted important separators with applications

Similar to the concept of important separators proposed by Marx [22] (see also [9, Chapter 8]), we define *multi-budgeted important separators* as follows.

▶ **Definition 8.** Let $(G, X, Y, \ell, (E_i, k_i)_{i=1}^{\ell})$ be a Multi-budgeted cut instance and let $Z \subseteq \bigcup_{i=1}^{\ell} E_i$ be a set of deletable edges. Let $C_1, C_2$ be two minimal $Z$-respecting budget-respecting $XY$-cuts. We say that $C_1$ *dominates* $C_2$ if

1. every vertex reachable from $X$ in $G - C_2$ is also reachable from $X$ in $G - C_1$;
2. for every $i \in [\ell]$, $|C_1 \cap E_i| \leq |C_2 \cap E_i|$.

We say that $\widehat{C}$ is an *important $Z$-respecting budget-respecting $XY$-cut* if $\widehat{C}$ is a minimal $Z$-respecting budget-respecting $XY$-cut and no other minimal $Z$-respecting budget-respecting $XY$-cut dominates $\widehat{C}$. $\widehat{C}$ is an *important budget-respecting $XY$-cut* if it is an *important $Z$-respecting budget-respecting $XY$-cut* for $Z = \bigcup_{i=1}^{\ell} E_i$.

Chen et al. [4] showed an enumeration procedure for (classic) important separators using similar charging scheme as the one of the previous section. Our main result in this section is a merge of the arguments from the previous section with the arguments of Chen et al. Theorem 2 follows from Theorem 9 via an analogous arguments as in [4].

▶ **Theorem 9.** *Let $(G, X, Y, \ell, (E_i, k_i)_{i=1}^{\ell})$ be a* Multi-budgeted cut *instance, let $Z \subseteq \bigcup_{i=1}^{\ell} E_i$ be a set of deletable edges, and denote $k = \sum_{i=1}^{\ell} k_i$. Then one can in $2^{\mathcal{O}(k^2 \log k)}(|V(G)|+|E(G)|)$ time enumerate a family of minimal $Z$-respecting budget-respecting $XY$-cuts of size $2^{\mathcal{O}(k^2 \log k)}$ that contains all important ones.*

**Proof.** Consider the recursive algorithm presented in Figure 2. The recursive procedure **ImportantCut** takes as an input a Multi-budgeted Cut instance $I = (G, X, Y, \ell, (E_i, k_i)_{i=1}^{\ell})$ and a set $Z \subseteq \bigcup_{i=1}^{\ell} E_i$, with the goal to enumerate all important $Z$-respecting budget-respecting $XY$-cuts. Note that the procedure may output some more $Z$-respecting budget-respecting $XY$-cuts; we need only to ensure that

1. it outputs all important ones,
2. it outputs $2^{\mathcal{O}(k^2 \ell \log k)}$ cuts, and
3. it runs within the desired time.

The procedure first invokes the algorithm of Theorem 5 on $(G, X, Y, k, Z)$, where $k = \sum_{i=1}^{\ell} k_i$. If the call returned that there is no $Z$-respecting $XY$-cut of size at most $k$, we can return an empty set. Otherwise, let $(P_j)_{j=1}^{\lambda}$, $B$, and $C$ be the computed objects. We perform a branching step, with each branch labeled with a tuple $(A_1, A_2, \ldots, A_\ell)$ where $A_i \subseteq [\lambda]$ and $|A_i| \leq k_i$ for every $i \in [\ell]$. A branch $(A_1, A_2, \ldots, A_\ell)$ is supposed to capture important cuts $C_0$ with $\{j | C_0 \cap B \cap E(P_j) \cap E_i \neq \emptyset\} \subseteq A_i$ for every $i \in [\ell]$; that is, for every $i \in [\ell]$ and $j \in [\lambda]$ we guess if $C_0$ contains a *bottleneck* edge of color $i$ on path $P_j$. All this information (i.e., paths $P_j$, the set $B$, the cut $C$, and the sets $A_i$) are passed to an auxiliary procedure **Enum**.

The procedure **Enum** shrinks the set $Z$ according to sets $A_i$. More formally, for every $i \in [\ell]$ and $j \in [\lambda] \setminus A_i$ we delete from $Z$ all edges from $B \cap E_i \cap E(P_j)$, obtaining a set $\widehat{Z} \subseteq Z$. At this point, we check if the reduction of the set $Z$ to $\widehat{Z}$ increased the size of minimum $Z$-respecting $XY$-cut by invoking Theorem 5 on $(G, X, Y, k, \widehat{Z})$ and obtaining objects $(\widehat{P}_j)_{j=1}^{\widehat{\lambda}}, \widehat{B}, \widehat{C}$ or a negative answer. If the size of the minimum cut increased, that is, $\widehat{\lambda} > \lambda$, we recurse with the original procedure **ImportantCut**. Otherwise, we add one cut to $\mathcal{S}$, namely $\widehat{C}$. Furthermore, we try to shrink one of the sets $A_i$ by one and recurse; that is, for every $i \in [\ell]$ and every $j \in A_i$, we recurse with the procedure **Enum** on sets $A'_{i'}$ where $A'_i = A_i \setminus \{j\}$ and $A'_{i'} = A_{i'}$ for every $i' \in [\ell] \setminus \{i\}$.

Let us first analyze the size of the search tree. A call to **ImportantCut** invokes at most $\binom{\lambda\ell}{\leq k} \leq (k\ell + 1)^k$ calls to **Enum**. Each call to **Enum** either falls back to **ImportantCut** if $\widehat{\lambda} > \lambda$ or branches into $\sum_{i=1}^{\ell} |A_i| \leq k\ell$ recursive calls to itself. In each recursive call, the sum $\sum_{i=1}^{\ell} |A_i|$ decreases by one. Consequently, the initial call to **Enum** results in at most $(k\ell)^k$ recursive calls, each potentially falling back to **ImportantCut**. Since each recursive call to **ImportantCut** uses strictly larger value of $\lambda$, which cannot grow larger than $k$, and $\ell \leq k$, the total size of the recursion tree is $2^{\mathcal{O}(k^2 \log k)}$. Each recursive call to **Enum** adds at most one set to $\mathcal{S}$, while each recursive call to **ImportantCut** and **Enum** runs in time $\mathcal{O}(2^{k\ell} \cdot k \cdot (|V(G)| + |E(G)|))$. The promised size of the family $\mathcal{S}$ and the running time bound follows. It remains to show correctness, that is, that every important $Z$-respecting budget-respecting $XY$-cut is contained in $\mathcal{S}$ returned by a call to **ImportantCut**$(I, Z)$.

We prove by induction on the size of the recursion tree that (1) every call to **ImportantCut**$(I, Z)$ enumerates all important $Z$-respecting budget-respecting $XY$-cuts, and (2) every call to **Enum**$(I, Z, (P_j)_{j=1}^{\lambda}, B, C, (A_i)_{i=1}^{\ell})$ enumerates all important $Z$-respecting budget-respecting $XY$-cuts $C_0$ with the property that $\{j | E_i \cap E(P_j) \cap B \cap C_0 \neq \emptyset\} \subseteq A_i$ for every $i \in [\ell]$.

The inductive step for a call **ImportantCut**$(I, Z)$ is straightforward. Let us fix an arbitrary important $Z$-respecting budget-respecting $XY$-cut $C_0$. Since $C_0$ is budget-respecting, $C_0$ is a $Z$-respecting cut of size at most $k$, and thus the initial call to Theorem 5 cannot return NO. Consider the tuple $(A_1, A_2, \ldots, A_\ell)$ where for every $i \in [\ell]$, $\{j | E(P_j) \cap E_i \cap B \cap C_0\} = A_i$. Since $C_0$ is budget-respecting and the paths $P_j$ do not share an edge of $Z$, we have that $|A_i| \leq k_i$ for every $i \in [\ell]$ and the algorithm considers this tuple in one of the branches. Then, from the inductive hypothesis, the corresponding call to **Enum** returns a set containing $C_0$.

Consider now a call to **Enum**$(I, Z, (P_j)_{j=1}^{\lambda}, B, C, (A_i)_{i=1}^{\ell})$ and an important $Z$-respecting budget-respecting $XY$-cuts $C_0$ with the property that $\{j | E_i \cap E(P_j) \cap B \cap C_0 \neq \emptyset\} \subseteq A_i$ for every $i \in [\ell]$. By the construction of $\widehat{Z}$ and the above assumption, $C_0$ is $\widehat{Z}$-respecting. In particular, the call to the algorithm of Theorem 5 cannot return NO. Hence, in the case when $\widehat{\lambda} > \lambda$, $C_0$ is enumerated by the recursive call to **ImportantCut** and we are done. Assume then $\widehat{\lambda} = \lambda$.

**ImportantCut**$(I, Z)$
Input: A Multi-budgeted cut instance $I = (G, X, Y, \ell, (E_i, k_i)_{i=1}^{\ell})$ and a set $Z \subseteq \bigcup_{i=1}^{\ell} E_i$.
Output: a family $\mathcal{S}$ of minimal $Z$-respecting budget-respecting $XY$-cuts that contains all important ones.
1. $\mathcal{S} := \emptyset$;
2. apply the algorithm of Theorem 5 to $(G, X, Y, k, Z)$ with $k = \sum_{i=1}^{\ell} k_i$, obtaining either objects $(P_i)_{i=1}^{\lambda}$, $B$, and $C$, or an answer NO;
3. **if** an answer NO is obtained, **then return** $\mathcal{S}$;
4. **for each** $(A_1, ..., A_\ell)$ such that for every $i$ in $[\ell]$, $A_i \subseteq [\lambda]$ and $|A_i| \leq k_i$ **do**
4.1 $\mathcal{S} := \mathcal{S} \cup$ **Enum**$(I, Z, (P_j)_{j=1}^{\lambda}, B, C, (A_i)_{i=1}^{\ell})$
5. **return** $\mathcal{S}$

**Enum**$(I, Z, (P_j)_{j=1}^{\lambda}, B, C, (A_i)_{i=1}^{\ell})$
Input: A Multi-budgeted cut instance $I = (G, X, Y, \ell, (E_i, k_i)_{i=1}^{\ell})$, a set $Z \subseteq \bigcup_{i=1}^{\ell} E_i$, a family $(P_j)_{j=1}^{\lambda}$ of paths from $X$ to $Y$ such that every edge of $Z$ appears on at most one path $P_j$, a set $B$ consisting of all edges that participate in some minimum $Z$-respecting $XY$-cut, a minimum $Z$-respecting $XY$-cut $C$ closest to $Y$, and sets $A_i \subseteq [\lambda]$ of size at most $k_i$ for every $i \in [\ell]$
Output: a family $\mathcal{S}$ of minimal $Z$-respecting budget-respecting $XY$-cuts that contains all cuts $C_0$ that are important $Z$-respecting budget respecting $XY$-cuts and satisfy $\{j | E(P_j) \cap B \cap C_0 \cap E_i \neq \emptyset\} \subseteq A_i$ for every $i \in [\ell]$.
a. $\widehat{Z} := Z$;
b. **for each** $i \in [\ell]$ **do**
  **for each** $j \in [\lambda] \setminus A_i$ **do**
    $\widehat{Z} := \widehat{Z} \setminus (B \cap E_i \cap E(P_j))$;
c. apply the algorithm of Theorem 5 to $(G, X, Y, k, \widehat{Z})$, obtaining either objects $(\widehat{P}_i)_{i=1}^{\widehat{\lambda}}$, $\widehat{B}$, and $\widehat{C}$ or an answer NO;
d. **if** $\widehat{\lambda}$ exists and $\widehat{\lambda} > \lambda$, **then**
d.1 $\mathcal{S} := \mathcal{S} \cup$ **ImportantCut**$(I, \widehat{Z})$;
e. **else if** $\widehat{\lambda}$ exists and equals $\lambda$, **then**
e.1 $\mathcal{S} := \mathcal{S} \cup \{\widehat{C}\}$;
e.2 **for each** $i \in [\ell]$ **do**
  **for each** $j \in A_i$ **do**
    $A'_i := A_i \setminus \{j\}$ and $A'_{i'} := A_{i'}$ for every $i' \in [\ell] \setminus \{i\}$
    $\mathcal{S} := \mathcal{S} \cup$ **Enum**$(I, \widehat{Z}, (P_j)_{j=1}^{\lambda}, \widehat{B}, \widehat{C}, (A'_i)_{i=1}^{\ell})$.
f. **return** $\mathcal{S}$

**Figure 2** FPT algorithm for enumerating important multi-budgeted $Z$-respecting $XY$-cuts.

For $i \in [\ell]$, let $\widehat{A}_i = \{j | E_i \cap E(P_j) \cap \widehat{B} \cap C_0 \neq \emptyset\}$. Since $\widehat{Z} \subseteq Z$ but the sizes of minimum $Z$-respecting and $\widehat{Z}$-respecting $XY$-cuts are the same, we have $\widehat{B} \subseteq B$. Consequently, $\widehat{A}_i \subseteq A_i$ for every $i \in [\ell]$.

Assume there exists $i \in [\ell]$ such that $\widehat{A}_i \subsetneq A_i$ and let $j \in A_i \setminus \widehat{A}_i$. Consider then the branch $(i, j)$ of the **Enum** procedure, that is, the recursive call with $A'_i = A_i \setminus \{j\}$ and $A'_{i'} = A_{i'}$ for $i' \in [\ell] \setminus \{i\}$. Observe that we have $\{j | E_{i'} \cap E(P_j) \cap \widehat{B} \cap C_0 \neq \emptyset\} \subseteq A'_{i'}$ for every $i' \in [\ell]$ and, by the inductive hypothesis, the corresponding call to **Enum** enumerates $C_0$. Hence, we are left only with the case $\widehat{A}_i = A_i$, that is, $A_i = \{j | E_i \cap E(P_j) \cap \widehat{B} \cap C_0 \neq \emptyset\}$ for every $i \in [\ell]$.

We claim that in this case $C_0 = \widehat{C}$. Assume otherwise. Since $|\widehat{C}| = \widehat{\lambda} = \lambda$ and $\widehat{Z} \subseteq Z$, $\widehat{C}$ contains exactly one edge on every path $P_j$. Also, $\widehat{C} \subseteq \widehat{B}$ by the definition of the set $\widehat{B}$. Since $\widehat{C}$ is the minimum $\widehat{Z}$-respecting $XY$-cut that is closest to $Y$, $\widehat{C} = \{e_1, e_2, \ldots, e_\lambda\}$ where $e_j$ is the last (closest to $Y$) edge of $\widehat{B}$ on the path $P_j$ for every $j \in [\lambda]$.

Let $R_0$ and $\widehat{R}$ be the set of vertices reachable from $X$ in $G - C_0$ and $G - \widehat{C}$, respectively.

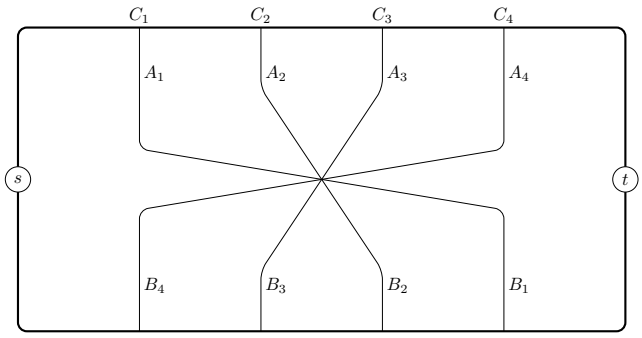



**Figure 3** A schematic picture of a 4-bowtie.

Let $D$ be a minimal $XY$-cut contained in $\delta^+(R_0 \cup \widehat{R})$. (Note that $\delta^+(R_0 \cup \widehat{R})$ is an $XY$-cut because $X \subseteq R_0 \cup \widehat{R}$ and $Y \cap (R_0 \cup \widehat{R}) = \emptyset$.) Then since $D \subseteq C_0 \cup \widehat{C} \subseteq Z$, $D$ is $Z$-respecting. By definition, every vertex reachable from $X$ in $G - R_0$ is also reachable from $X$ in $G - D$.

We claim that $D$ is budget-respecting and, furthermore, dominates $C_0$. Fix a color $i \in [\ell]$; our goal is to prove that $|D \cap E_i| \leq |C_0 \cap E_i|$. To this end, we charge every edge of color $i$ in $D \setminus C_0$ to a distinct edge of color $i$ in $C_0 \setminus D$. Since $D \subseteq C_0 \cup \widehat{C}$, we have that $D \setminus C_0 \subseteq \widehat{C}$, that is, an edge of $D \setminus C_0$ of color $i$ is an edge $e_j$ for some $j \in [\lambda]$ with $e_j \in E_i$ and $e_j \in D \setminus C_0$.

Recall that we are working in the case $A_i = \{j | E_i \cap E(P_j) \cap \widehat{B} \cap C_0 \neq \emptyset\}$. Since $e_j \in \widehat{C} \subseteq \widehat{Z}$, we have that $j \in A_i$. Hence, there exists $e'_j \in E_i \cap E(P_j) \cap \widehat{B} \cap C_0$. By the definition of $\widehat{C}$, $e_j$ is the last (closest to $Y$) edge of $\widehat{B}$ on $P_j$. Since $e_j \notin C_0$, $e'_j \neq e_j$ and $e'_j$ lies on the subpath of $P_j$ between $X$ and the tail of $e_j$. This entire subpath is contained in $\widehat{R}$ and, hence, $e'_j \notin D$.

We charge $e_j$ to $e'_j$. Since $e'_j \in E(P_j) \cap E_i \cap \widehat{B} \cap (C_0 \setminus D)$, for distinct $j$, the edges $e'_j$ are distinct as the paths $P_j$ do not share an edge belonging to $Z$ and $\widehat{B} \subseteq \widehat{Z} \subseteq Z$. Consequently, $|D \cap E_i| \leq |C_0 \cap E_i|$. This finishes the proof that $D$ dominates $C_0$.

Since $C_0$ is important, we have $D = C_0$. In particular, $\widehat{R} \subseteq R_0$. On the other hand, for every $j \in [\lambda]$ we have that $e_j \in \widehat{C} \subseteq \widehat{Z} \subseteq Z \subseteq \bigcup_{i=1}^{\ell} E_i$. In particular, there exists $i \in [\ell]$ such that $e_j \in E_i$ and $j \in A_i$. Hence, we also have $E_i \cap E(P_j) \cap \widehat{B} \cap C_0 \neq \emptyset$. But the entire subpath of $P_j$ from $X$ to the tail of $e_j$ lies in $\widehat{R} \subseteq R_0$, while $e_j$ is the last edge of $\widehat{B}$ on $P_j$. Hence, $e_j \in C_0$. Since the choice of $j$ is arbitrary, $\widehat{C} \subseteq C_0$. Since $\widehat{C}$ is an $XY$-cut and $C_0$ is minimal, $\widehat{C} = C_0$ as claimed.

This finishes the proof of Theorem 9. ◀

## 5 Bound on the number of solutions closest to $t$

In this section we sketch the proofs of Theorems 3 and 4. The central definition of this section is the following (see also Figure 3).

▶ **Definition 10.** Let $G$ be a directed graph with distinguished vertices $s$ and $t$ and let $k$ be an integer. An *$a$-bowtie* is a sequence $C_1, C_2, \ldots, C_a$ of pairwise disjoint minimal $st$-cuts of size $k$ each such that each cut $C_i$ can be partitioned $C_i = A_i \uplus B_i$ such that for every $1 \leq i < j \leq a$, the set $A_i$ is exactly the set of edges of $C_i$ reachable from $s$ in $G - C_j$ and $B_j$ is exactly the set of edges of $C_j$ reachable from $s$ in $G - C_i$.

Our main graph-theoretic result is the following. The proof is deferred to the full version of the paper.

▶ **Theorem 11.** *For every integers $a, k \geq 1$ there exists an integer $g$ such that for every directed graph $G$ with distinguished $s, t \in V(G)$, and a family $\mathcal{U}$ of pairwise disjoint minimal $st$-cuts of size $k$ each, if $|\mathcal{U}| \geq g$, then $\mathcal{U}$ contains an $a$-bowtie.*

The next two lemmata are key observations to prove Theorems 3 and 4, respectively, with the help of Theorem 11.

▶ **Lemma 12.** *Let $k$, $g$, $G$, $s$, $t$, $\mathcal{F}$, and $\mathcal{G}$ be as in the statement of Theorem 3. Then $\mathcal{G}$ does not contain an $a$-bowtie for $a > \binom{k+2}{2}$.*

**Proof.** Assume the contrary, let $(C_i, A_i, B_i)_{i=1}^a$ be such a bowtie. Since $a > \binom{k+2}{2}$, there exists $i < j$ with $|A_i| = |A_j|$ and $|B_i| = |B_j|$ (there are $\binom{k+2}{2}$ choices for $(|A_i|, |B_i|)$). However, then $A_i \cup B_j$ and $A_j \cup B_i$ have also cardinality $k$, are $st$-cuts, and have together twice the minimum weight. Furthermore, the set of vertices reachable from $s$ in $G - (A_j \cup B_i)$ is a strict superset of the set of vertices reachable from $s$ in $G - C_i$ and $G - C_j$. This contradicts the fact that $C_i, C_j \in \mathcal{G}$. ◀

▶ **Lemma 13.** *Let $k$, $\ell$, $I = (G, s, t, (P_i)_{i=1}^m, k)$, $\mathcal{F}$, and $\mathcal{G}$ be as in the statement of Theorem 4. Then $\mathcal{G}$ does not contain a 4-bowtie $(C_i, A_i, B_i)_{i=1}^4$ in which the edge set of every path $P_j$ intersects at most one cut $C_i$.*

**Proof.** Assume the contrary Let $(C_i, A_i, B_i)_{i=1}^4$ be such a 4-bowtie. Consider $i \in \{2, 3\}$ and two edges $e \in A_i$ and $f \in B_i$. In $G - C_4$, the edge $e$ is reachable from $s$ while $f$ is not; consequently, $e$ and $f$ cannot appear on the same input path with $e$ being earlier (by assumption, $C_4$ is disjoint from the input path in question). A similar reasoning for $G - C_1$ shows that $e$ and $f$ cannot appear on the same input path with $f$ being earlier than $e$.

Hence, $e$ and $f$ cannot appear together on a single path $P_j$. For a set of edges $D$, by the *cost* of $D$ we denote $|\{j | D \cap P_j \neq \emptyset\}|$. Since the choice of $e$ and $f$ was arbitrary, we infer that the sum of costs of $A_2 \cup B_3$ and of $A_3 \cup B_2$ equals the sum of costs of $C_2$ and of $C_3$. Hence, both these $st$-cuts have minimum cost. However, $A_2 \cup B_3$ is closer to $t$ than $C_2$, a contradiction. ◀

**Proof of Theorem 3.** Assume $|\mathcal{G}| > g$ for some sufficiently large $g$ to be fixed later. For $i \in [k]$, let $\mathcal{G}^i$ be the set of $u \in \mathcal{G}$ of cardinality $i$. We apply the Sunflower Lemma to the largest set $\mathcal{G}^i$: If $g > k \cdot k! g_1^k$ for some integer $g_1$ to be chosen later, there exists $\mathcal{G}_1 \subseteq \mathcal{G}$ with $|\mathcal{G}_1| > g_1$, every element of $\mathcal{G}_1$ being of the same size $k'$, and a set $c$ such that $u \cap v = c$ for every distinct $u, v \in \mathcal{G}_1$.

Let $\widehat{k} = k' - |c|$, $\widehat{u} = u \setminus c$ for every $u \in \mathcal{G}_1$, $\widehat{\mathcal{G}}_1 = \{\widehat{u} \mid u \in \mathcal{G}_1\}$ and $\widehat{G} = G - c$. Since every $u \in \mathcal{U}$ is a minimal $st$-cut of size $k'$ in $G$, every $\widehat{u} \in \widehat{\mathcal{G}}_1$ is a minimal $st$-cut of size $\widehat{k}$ in $\widehat{G}$. Furthermore, every $\widehat{u} \in \widehat{\mathcal{G}}_1$ is a minimal $st$-cut of size at most $\widehat{k}$ in $\widehat{G}$ of minimum possible weight: if there existed an $st$-cut $\widehat{x}$ of smaller weight and cardinality at most $\widehat{k}$, then $x = \widehat{x} \cup c$ would be an $st$-cut in $G$ of cardinality at most $k$ and weight smaller than every element of $\mathcal{G}_1$. Similarly, if there were a minimal $st$-cut $\widehat{x}$ in $\widehat{G}$ of minimum weight and cardinality at most $\widehat{k}$ that is closer to $t$ than $\widehat{u}$ for some $\widehat{u} \in \widehat{\mathcal{G}}_1$, then $\widehat{x} \cup c$ would be an $st$-cut in $G$ of cardinality at most $k$ and minimum weight that is closer to $t$ than $u$, a contradiction. By construction, the elements of $\widehat{\mathcal{G}}_1$ are pairwise disjoint.

Lemma 12 bounds the maximum possible size of a bowtie in $\widehat{\mathcal{G}}_1$. Hence, Theorem 11 asserts that $\widehat{\mathcal{G}}_1$ has size bounded by a function of $k$. This finishes the proof of the theorem. ◀

**Proof of Theorem 4.** We proceed similarly as in the proof of Theorem 3, but we need to be a bit more careful with the paths $P_j$. Assume $|\mathcal{G}| > g$ for some sufficiently large integer $g$.

As before, we partition $\mathcal{G}$ according to the sizes of elements: for every $i \in [k\ell]$, let $\mathcal{G}^i = \{u \in \mathcal{G} \mid |u| = i\}$. Let $i \in [k\ell]$ be such that $|\mathcal{G}^i| > g/(k\ell)$. For $u \in \mathcal{G}^i$, let $J(u) = \{j \mid u \cap P_j \neq \emptyset\}$. By the assumptions of the theorem, every set $J(u)$ is of cardinality exactly $k$. We apply the Sunflower Lemma to $\{J(u) \mid u \in \mathcal{G}^i\}$: If $g > (k\ell) \cdot k! \cdot g_1^k$ for some integer $g_1$ to be fixed later, then there exists $\mathcal{G}_1 \subseteq \mathcal{G}^i$ of size larger than $g_1$ and a set $I \subseteq [m]$ such that for every distinct $u, v \in \mathcal{G}_1$ we have $J(u) \cap J(v) = I$. For every $u \in \mathcal{G}_1$, let $u_I = u \cap \bigcup_{j \in I} P_j$. Since $|I| \leq k$, there are at most $2^{k\ell}$ choices for $u_I$ among elements $u \in \mathcal{G}_1$. Consequently, there exists $\mathcal{G}_2 \subseteq \mathcal{G}_1$ of cardinality larger than $g_2 := g_1/2^{k\ell}$ such that $u_I = v_I$ for every $u, v \in \mathcal{G}_2$. Denote $c = u_I$ for any $u \in \mathcal{G}_2$.

Let $\widehat{u} := u - c$ for every $u \in \mathcal{G}_2$. Let $\widehat{\mathcal{G}}_2 = \{\widehat{u} \mid u \in \mathcal{G}_2\}$.

Define now $\widehat{G} = G - c$ and define a partition $\widehat{\mathcal{P}}$ of $E(\widehat{G})$ into paths of length at most $\ell$ as follows: we take all paths $P_i$ for $i \notin I$ and, for every $i \in I$, each edge of $P_i \setminus c$ as a length-1 path. Furthermore, denote $\widehat{k} = k - |I|$. Note that $(\widehat{G}, s, t, \widehat{\mathcal{P}}, \widehat{k})$ is a Chain $\ell$-SAT instance for which every $\widehat{u} \in \widehat{\mathcal{G}}_2$ is a solution. Furthermore, $(\widehat{G}, s, t, \widehat{\mathcal{P}}, \widehat{k} - 1)$ is a no-instance, as if $\widehat{x}$ were its solution, then $\widehat{x} \cup c$ would be a solution to $(G, s, t, (P_i)_{i=1}^m, k - 1)$, a contradiction. Similarly, if there were a solution $\widehat{x}$ to $(\widehat{G}, s, t, \widehat{\mathcal{P}}, \widehat{k})$ that is closer to $t$ than $\widehat{u}$ for some $\widehat{u} \in \widehat{\mathcal{G}}_2$, then $\widehat{x} \cup c$ would be a solution to $(G, s, t, (P_i)_{i=1}^m, k)$ that is closer to $t$ than $u$, a contradiction. Furthermore, by construction, the elements of $\widehat{\mathcal{G}}_2$ are pairwise disjoint and no path of $\widehat{\mathcal{P}}$ intersects more than one element of $\widehat{\mathcal{G}}_2$.

Lemma 13 bounds the maximum possible size of a bowtie in $\widehat{\mathcal{G}}_2$. Hence, Theorem 11 asserts that $\widehat{\mathcal{G}}_2$ has size bounded by a function of $k$ and $\ell$. This finishes the proof of the theorem. ◀

## 6 Conclusion

We would like to conclude with a discussion on future research directions. First, our upper bound of $2^{\mathcal{O}(k^2 \log k)}$ on the number of multi-budgeted important separators (Theorem 9) is far from the $4^k$ bound for the classic important separators. As pointed out by an anonymous reviewer at IPEC 2018, there is an easy lower bound of $k!$ for the number of multi-budgeted important separators: Let $\ell = k$, $k_i = 1$ for every $i \in [\ell]$, and let $G$ consist of $k$ paths from $s$ to $t$, each path consisting of $\ell$ edges of different colors. Then there are exactly $k!$ distinct multi-budgeted important separators, as we can freely choose a different color $i \in [\ell]$ to cut on each path. We are not aware of any better lower bound, leaving a significant gap between the lower and upper bounds.

Second, our existential statement of Theorems 3 and 4 can be treated as a weak support of tractability of Chain $\ell$-SAT and Weighted $st$-cut. Are they really FPT when parameterized by the cardinality of the cut?

## References


1 Amit Agarwal, Noga Alon, and Moses Charikar. Improved approximation for directed cut problems. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 671–680. ACM, 2007. doi:10.1145/1250790.1250888.

2 Chandra Chekuri, Sudipto Guha, and Joseph Naor. The Steiner $k$-Cut Problem. *SIAM J. Discrete Math.*, 20(1):261–271, 2006. doi:10.1137/S0895480104445095.

3 Jianer Chen and Iyad A. Kanj. Constrained minimum vertex cover in bipartite graphs: complexity and parameterized algorithms. *J. Comput. Syst. Sci.*, 67(4):833–847, 2003. doi:10.1016/j.jcss.2003.09.003.

4 Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM*, 55(5):21:1–21:19, 2008. doi:10.1145/1411509.1411511.

5 Rajesh Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michal Pilipczuk. Designing FPT Algorithms for Cut Problems Using Randomized Contractions. *SIAM J. Comput.*, 45(4):1171–1229, 2016. doi:10.1137/15M1032077.

6 Rajesh Chitnis, László Egri, and Dániel Marx. List H-Coloring a Graph by Removing Few Vertices. *Algorithmica*, 78(1):110–146, 2017. doi:10.1007/s00453-016-0139-6.

7 Rajesh Hemant Chitnis, Marek Cygan, Mohammad Taghi Hajiaghayi, and Dániel Marx. Directed Subset Feedback Vertex Set Is Fixed-Parameter Tractable. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *ICALP (1)*, volume 7391 of *Lecture Notes in Computer Science*, pages 230–241. Springer, 2012. doi:10.1007/978-3-642-31594-7_20.

8 Rajesh Hemant Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. Fixed-Parameter Tractability of Directed Multiway Cut Parameterized by the Size of the Cutset. *SIAM J. Comput.*, 42(4):1674–1696, 2013. doi:10.1137/12086217X.

9 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

10 Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. On multiway cut parameterized above lower bounds. *TOCT*, 5(1):3, 2013. doi:10.1145/2462896.2462899.

11 Guy Even, Joseph Naor, Baruch Schieber, and Madhu Sudan. Approximating Minimum Feedback Sets and Multicuts in Directed Graphs. *Algorithmica*, 20(2):151–174, 1998. doi:10.1007/PL00009191.

12 Piotr Faliszewski, Fedor V. Fomin, Daniel Lokshtanov, Dániel Marx, Shmuel Onn, Marcin Pilipczuk, Michał Pilipczuk, Saket Saurabh, and Meirav Zehavi. What's next? Future directions in Parameterized Complexity. *Recent Advances in Parameterized Complexity, Tel-Aviv, Israel*, 2017. Accessed 18.09.2018. URL: https://rapctelaviv.weebly.com/uploads/1/0/5/3/105379375/future.pdf.

13 Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Approximate Max-Flow Min-(Multi)Cut Theorems and Their Applications. *SIAM J. Comput.*, 25(2):235–251, 1996. doi:10.1137/S0097539793243016.

14 Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Multiway cuts in node weighted graphs. *J. Algorithms*, 50(1):49–61, 2004. doi:10.1016/S0196-6774(03)00111-1.

15 Sylvain Guillemot. FPT algorithms for path-transversal and cycle-transversal problems. *Discrete Optimization*, 8(1):61–71, 2011. doi:10.1016/j.disopt.2010.05.003.

16 Yoichi Iwata. Linear-Time Kernelization for Feedback Vertex Set. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 68:1–68:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.68.

17 Yoichi Iwata, Magnus Wahlström, and Yuichi Yoshida. Half-integrality, LP-branching, and FPT Algorithms. *SIAM J. Comput.*, 45(4):1377–1411, 2016. doi:10.1137/140962838.

18 David R. Karger, Philip N. Klein, Clifford Stein, Mikkel Thorup, and Neal E. Young. Rounding Algorithms for a Geometric Embedding of Minimum Multiway Cut. *Math. Oper. Res.*, 29(3):436–461, 2004. doi:10.1287/moor.1030.0086.

19 Stefan Kratsch and Magnus Wahlström. Representative Sets and Irrelevant Vertices: New Tools for Kernelization. In *53rd Annual IEEE Symposium on Foundations of Computer*

*Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 450–459. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.46.

20 Stefan Kratsch and Magnus Wahlström. Compression via Matroids: A Randomized Polynomial Kernel for Odd Cycle Transversal. *ACM Transactions on Algorithms*, 10(4):20, 2014. doi:10.1145/2635810.

21 Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Parameterized Complexity and Approximability of Directed Odd Cycle Transversal. *CoRR*, abs/1704.04249, 2017. arXiv:1704.04249.

22 Dániel Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006. doi:10.1016/j.tcs.2005.10.007.

23 Dániel Marx. What's Next? Future Directions in Parameterized Complexity. In Hans L. Bodlaender, Rod Downey, Fedor V. Fomin, and Dániel Marx, editors, *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *Lecture Notes in Computer Science*, pages 469–496. Springer, 2012. doi:10.1007/978-3-642-30891-8_20.

24 Dániel Marx, Barry O'Sullivan, and Igor Razgon. Finding small separators in linear time via treewidth reduction. *ACM Transactions on Algorithms*, 9(4):30, 2013. doi:10.1145/2500119.

25 Dániel Marx and Igor Razgon. Fixed-Parameter Tractability of Multicut Parameterized by the Size of the Cutset. *SIAM J. Comput.*, 43(2):355–388, 2014. doi:10.1137/110855247.

26 Marcin Pilipczuk and Magnus Wahlström. Directed multicut is $W[1]$-hard, even for four terminal pairs. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1167–1178. SIAM, 2016. doi:10.1137/1.9781611974331.ch81.

27 Igor Razgon and Barry O'Sullivan. Almost 2-SAT is fixed-parameter tractable. *J. Comput. Syst. Sci.*, 75(8):435–450, 2009. doi:10.1016/j.jcss.2009.04.002.

# Matching Cut: Kernelization, Single-Exponential Time FPT, and Exact Exponential Algorithms

**Christian Komusiewicz**
Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, Marburg, Germany
komusiewicz@informatik.uni-marburg.de
https://orcid.org/0000-0003-0829-7032

**Dieter Kratsch**
Laboratoire de Génie Informatique, de Production et de Maintenance, Université de Lorraine, Metz, France
dieter.kratsch@univ-lorraine.fr

**Van Bang Le**
Universität Rostock, Institut für Informatik, Rostock, Germany
van-bang.le@uni-rostock.de

## Abstract

In a graph, a matching cut is an edge cut that is a matching. Matching Cut, which is known to be NP-complete, is the problem of deciding whether or not a given graph $G$ has a matching cut. In this paper we show that Matching Cut admits a quadratic-vertex kernel for the parameter distance to cluster and a linear-vertex kernel for the parameter distance to clique. We further provide an $O^*(2^{\mathrm{dc}(G)})$ time and an $O^*(2^{\mathrm{d}\overline{\mathrm{c}}(G)})$ time FPT algorithm for Matching Cut, where $\mathrm{dc}(G)$ and $\mathrm{d}\overline{\mathrm{c}}(G)$ are the distance to cluster and distance to co-cluster, respectively. We also improve the running time of the best known branching algorithm to solve Matching Cut from $O^*(1.4143^n)$ to $O^*(1.3803^n)$. Moreover, we point out that, unless NP $\subseteq$ coNP/poly, Matching Cut does not admit a polynomial kernel when parameterized by treewidth.

**2012 ACM Subject Classification** Mathematics of computing → Graph theory, Theory of computation → Graph algorithms analysis, Theory of computation → Parameterized complexity and exact algorithms

**Keywords and phrases** matching cut, decomposable graph, graph algorithm

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.19

## 1 Introduction

In a graph $G = (V, E)$, a *cut* is a partition $V = A \dot\cup B$ of the vertex set into disjoint, nonempty sets $A$ and $B$, written $(A, B)$. The set of all edges in $G$ having an endvertex in $A$ and the other endvertex in $B$, also written $(A, B)$, is called the *edge cut* of the cut $(A, B)$. A *matching cut* is an (possibly empty) edge cut that is a matching. Note that, by our definition, a matching whose removal disconnects the graph need not be a matching cut.

Another way to define matching cuts is as follows ([13, 7]). A partition $V = A \dot\cup B$ of the vertex set of the graph $G = (V, E)$ into disjoint, nonempty sets $A$ and $B$, is a matching cut if and only if each vertex in $A$ has at most one neighbor in $B$ and each vertex in $B$ has at most one neighbor in $A$. Not every graph has a matching cut; the Matching Cut problem is the problem of deciding whether or not a given graph has a matching cut:

| Matching Cut | |
|---|---|
| *Instance:* | A graph $G = (V, E)$. |
| *Question:* | Does $G$ have a matching cut? |

© Christian Komusiewicz, Dieter Kratsch, and Van Bang Le;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 19; pp. 19:1–19:13
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Farley and Proskurowski [11] studied matching cuts in graphs in the context of network applications. Patrignani and Pizzonia [18] pointed out an application of matching cuts in graph drawing. Graphs having no matching cut were first discussed by Graham in [13] under the name *indecomposable graphs*, and have been recently used by Araújo et al. [1] in the context of WDM (Wavelength Division Multiplexing) networks.

**Previous results.** Chvátal [7] showed that Matching Cut is NP-complete, even when restricted to graphs of maximum degree four, and polynomially solvable for graphs of maximum degree three. These results triggered a lot of research on the computational complexity of Matching Cut in graphs with additional structural assumptions [4, 6, 14, 16, 15, 17, 18]. In particular, the NP-hardness of Matching Cut has been further strengthened to planar graphs of maximum degree four [4] and bipartite graphs of maximum degree four [16]. As noted previously [14], the NP-hardness reduction by Chvátal [7] also shows that Matching Cut cannot be solved in $2^{o(n)}$ time, where $n$ is the number of vertices of the input graph, if the Exponential Time Hypothesis (ETH) is true.

Exact exponential algorithms for Matching Cut on graphs without any restriction have been recently considered by Kratsch and Le [14] who provided the first exact branching algorithm for Matching Cut running in time $O^*(1.4143^n)$[1], and a single-exponential algorithm of running time $2^{\tau(G)}O(n^2)$, where $\tau(G)$ is the vertex cover number. We note that Matching Cut can be expressed in MSOL, see for example [4]; hence Matching Cut is fixed-parameter tractable when parameterized by $\mathrm{tw}(G)$, the treewidth of $G$. Very recently, Aravind et al. [2] presented a tree-decomposition-based dynamic programming algorithm that solves Matching Cut in $O^*(12^{\mathrm{tw}(G)})$ time and fixed-parameter algorithms for further parameters describing the structure of the input graph. For example, Matching Cut can be solved in $O^*(2^{\mathrm{tc}(G)})$ time where $\mathrm{tc}(G) \leq \tau(G)$ is the size of a smallest twin cover of $G$ [2].

**Our contributions.** We give the first polynomial kernels for Matching Cut by showing that Matching Cut admits a quadratic-vertex kernel for the parameter distance to cluster and a linear-vertex kernel for the parameter distance to clique. Second, we show that Matching Cut can be solved by single-exponential algorithms running in time $2^{\mathrm{dc}(G)}O(n^2)$ and $2^{\mathrm{d\overline{c}}(G)}O(nm)$, respectively, where $\mathrm{dc}(G)$ is the distance to cluster and $\mathrm{d\overline{c}}(G)$ is the distance to co-cluster. This improves upon the FPT algorithms for Matching Cut with running time $2^{\tau(G)}O(n^2)$ [14], where $\tau(G) \geq \max\{\mathrm{dc}(G), \mathrm{d\overline{c}}(G)\}$ is the vertex cover number of $G$ which can be much larger than $\mathrm{dc}(G)$ and $\mathrm{d\overline{c}}(G)$. Similarly, this improves upon the FPT algorithm with $O^*(2^{\mathrm{tc}(G)})$ [2] since $\mathrm{tc}(G) \geq \mathrm{dc}(G)$. Third, we provide an exact branching algorithm for Matching Cut that has time complexity $O^*(1.3803^n)$. This result improves upon the first exact branching algorithm for Matching Cut that has time complexity $O^*(1.4143^n)$ [14].

**Notation and terminology.** Let $G = (V, E)$ be a graph with vertex set $V(G) := V$ and edge set $E(G) := E$. We assume that a (input) graph has $n$ vertices and $m$ edges. A *stable set* (a *clique*) in $G$ is a set of pairwise non-adjacent (adjacent) vertices. The neighborhood of a vertex $v$ in $G$, denoted by $N_G(v)$, is the set of all vertices in $G$ adjacent to $v$; if the context is clear, we simply write $N(v)$. Set $\deg(v) := |N(v)|$, the degree of the vertex $v$. For a subset $W \subseteq V$, $G[W]$ is the subgraph of $G$ induced by $W$, and $G - W$ stands for $G[V \setminus W]$.

[1] Throughout the paper we use the $O^*$ notation which suppresses polynomial factors.

We write $N_W(v)$ for $N(v) \cap W$ and call the vertices in $N(v) \cap W$ the *$W$-neighbors* of $v$. A graph is a *cluster graph* if it is a vertex disjoint union of cliques. The maximal cliques of a cluster graph are called *clusters*. A graph is a *co-cluster graph* if it is a complete multipartite graph or, equivalently, the complement graph of a cluster graph. Observe that a clique is a cluster graph and a co-cluster graph.

A *vertex cover* of $G$ is a subset $C \subseteq V$ such that every edge of $G$ has at least one endvertex in $C$, i.e., $V \setminus C$ is a stable set in $G$. The vertex cover number of $G$, denoted by $\tau(G)$, is the smallest size of a vertex cover of $G$. More generally, given a graph property $\mathcal{P}$, a *distance to $\mathcal{P}$ set* of a graph $G$ is a subset $U \subseteq V$ such that $G - U$ has the property $\mathcal{P}$. The *distance to $\mathcal{P}$* is the smallest size of a distance to $\mathcal{P}$ set. This number is called *distance to cluster*, denoted by $\mathrm{dc}(G)$, in case $\mathcal{P}$ is the set of cluster graphs, it is called *distance to co-cluster*, denoted by $\mathrm{d\overline{c}}(G)$, in case $\mathcal{P}$ is the set of co-cluster graphs, and it is called *distance to clique*, denoted by $\mathrm{dq}(G)$, in case $\mathcal{P}$ is the set of cliques. By the definition of cluster graphs and co-cluster graphs, we have $\tau(G) \geq \max\{\mathrm{dc}(G), \mathrm{d\overline{c}}(G)\}$ and $\mathrm{dq}(G) \geq \max\{\mathrm{dc}(G), \mathrm{d\overline{c}}(G)\}$ for any graph $G$.

Throughout the paper we use the concept of monochromatic vertex subsets and induced subgraphs. Let $G = (V, E)$ be a graph and $U \subseteq V$. Then we call $U$ *monochromatic* in $G$ if for every matching cut $(A, B)$ of $G$, either $U \subseteq A$ or $U \subseteq B$; slightly abusing notation we shall sometimes also call $G[U]$ monochromatic in $G$. Being monochromatic is hereditary: if $G[U]$ is monochromatic, then so is $G[U']$ for every $U' \subseteq U$. Note that a complete subgraph $K_n$ is monochromatic if $n = 1$ or $n \geq 3$, and that a complete bipartite subgraph $K_{n,m}$ is monochromatic if $n \geq 3$ and $m \geq 2$ or vice versa. Moreover, disconnected graphs and graphs having a vertex of degree at most one admit a matching cut. Hence, we may assume that all graphs considered are connected and have minimum degree at least two.

When an algorithm branches on the current instance of size $n$ into $r$ subproblems of sizes at most $n - t_1, n - t_2, \ldots, n - t_r$, then $(t_1, t_2, \ldots, t_r)$ is called the *branching vector* of this branching, and the unique positive root of $x^n - x^{n-t_1} - x^{n-t_2} - \cdots - x^{n-t_r} = 0$, denoted by $\tau(t_1, t_2, \ldots, t_r)$, is called its *branching number*. The running time of a branching algorithm is $O^*(\alpha^n)$, where $\alpha = \max_i \alpha_i$ and $\alpha_i$ is the branching number of branching rule $i$, and the maximum is taken over all branching rules. We refer to [12] for more details on exact branching algorithms. For the basic notions of parameterized complexity we refer to [8].

## 2 A Polynomial Kernel for the Distance to Cluster

In this section we present a polynomial kernel for the parameter $\mathrm{dc}(G)$, the distance of $G$ to a cluster graph. First, however, we motivate the study of the parameter $\mathrm{dc}(G)$ by a negative result. Recall that there is an FPT algorithm for Matching Cut when parameterized by treewidth [2, 4]. Hence, a natural question is whether Matching Cut admits a polynomial kernel for this parameter. A further candidate parameter for a polynomial kernel is the minimum number $k$ of edges crossing any matching cut; this could be considered the standard solution size parameter for Matching Cut. Finally, a common parameter in kernelizations is the maximum degree of the input graph. We rule out polynomial kernels for all three parameters.

▶ **Proposition 1.** Matching Cut *does not admit polynomial kernel with respect to the sum of the treewidth of $G$, the minimum number of edges crossing any matching cut of $G$, and the maximum degree in $G$ unless $NP \subseteq coNP/poly$.*

The proof of Proposition 1 uses cross-composition and is deferred to the full version of this work.

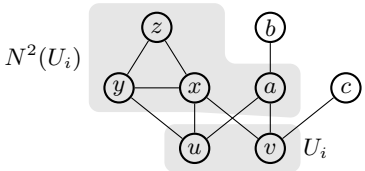



**Figure 1** An example for the definition of $N^2(U_i)$ with $U_i = \{u, v\}$. We have $\{x, a\} \subseteq N^2(U_i)$ since $x$ and $a$ have two neighbors in $U_i$. Moreover, $\{y, z\} \subseteq N^2(U_i)$ since $y$ and $z$ are in a cluster of size at least three with $x$. Alternatively, $\{x, y, z\} \subseteq N^2(U_i)$ since $u \in U_i$ is adjacent to $x$ and $y$. Finally, $b \notin N^2(U_i)$ even though $a \in N^2(U_i)$, since the cluster $\{a, b\}$ has only size two.

This excludes many natural candidate parameters for kernelization and motivates our study of kernelization for $\mathrm{dc}(G)$, the distance of $G$ to a cluster graph. Let $U = \{u_1, \ldots, u_{|U|}\}$ denote a vertex set such that $G - U$ is a cluster graph. We may assume that $|U| \leq 3\mathrm{dc}(G)$ since a 3-approximation of Cluster Vertex Deletion can be computed in time $O(\mathrm{dc}(G)(n+m))$ based on the observation that a graph is a cluster graph if and only if it does not contain an induced path on three vertices. During the kernelization, we maintain a partition of $U$ into $U_1, \ldots, U_\ell$ such that each $U_i$ is monochromatic. The initial partition contains one set for each vertex of $U$, that is, $U_i := \{u_i\}$, $1 \leq i \leq |U|$. We call the sets of the partition the *monochromatic parts* of $U$.

During the kernelization, we may *merge* two sets $U_i$ and $U_j$, $i \neq j$, which is to remove $U_i$ and $U_j$ from the partition and to add $U_i \cup U_j$. We say that merging $U_i$ and $U_j$ is *safe* if $U_i \cup U_j$ is monochromatic in $G$.

The main idea of the kernelization is to find opportunities to merge monochromatic parts of $U$ or to transform distinct clusters into a single cluster because we infer that they form a larger monochromatic set. Intuitively, an opportunity for merging arises when there are many vertices in $V \setminus U$ with at least two neighbors in $U$. The first step handles some clusters that have no such vertex.

▶ **Reduction Rule 1.** *If $V \setminus U$ contains a degree-one vertex or a cluster $C$ such that $(V \setminus C, C)$ is a matching cut, then output "yes".*

The correctness of the rule is obvious. After its application, every cluster $C$ contains either a vertex $v$ with two neighbors in $U$ or there is a vertex in $u \in U$ with two neighbors in $C$.

To identify clusters that form monochromatic sets together with some monochromatic parts of $U$, we introduce the following notation. For each monochromatic part $U_i$ of $U$, we let $N^2(U_i)$ denote the set of vertices $v \in V \setminus U$ such that at least one of the following holds:

- $v$ has two neighbors in $U_i$,
- $v$ is in a cluster of size at least three in $G - U$ that contains a vertex that has two neighbors in $U_i$, or
- $v$ is in a cluster $C$ in $G - U$ and some vertex in $U_i$ has two neighbors in $C$.

▶ **Proposition 1.** *$U_i \cup N^2(U_i)$ is monochromatic.*

**Proof.** In the first case, $v$ has two neighbors in the monochromatic set $U_i$ and thus $U_i \cup \{v\}$ is monochromatic. In the second case, the cluster $C$ containing $v$ is monochromatic and contains a vertex $w$ such that $U_i \cup \{w\}$ is monochromatic. Hence, $U_i \cup C$ is monochromatic. In the third case, the cluster $C$ contains two vertices $x$ and $y$ that form a triangle with some vertex from $U_i$ and thus $U_i \cup \{x, y\}$ is monochromatic. If $|C| = 2$, then $x = v$ or $y = v$, if $|C| > 3$, then $C$ is monochromatic and thus $U_i \cup C$ is monochromatic. ◀

The next two rules identify monochromatic parts that can be merged because they belong to overlapping monochromatic sets.

▶ **Reduction Rule 2.** *If there is a vertex $v$ that is contained in $N^2(U_i)$ and $N^2(U_j)$ for $i \neq j$, then merge $U_i$ and $U_j$.*

**Proof of safeness.** By Proposition 1, $\{v\} \cup U_i$ and $\{v\} \cup U_j$ are monochromatic in $G$. Thus, $U_i \cup U_j \cup \{v\}$ is monochromatic. ◀

▶ **Reduction Rule 3.** *If there are three vertices $v_1$, $v_2$, $v_3$ in $V$ that have two common neighbors $u \in U_i$ and $u' \in U_j$, $i \neq j$, then merge $U_i$ and $U_j$.*

**Proof of safeness.** The proof is based on the fact that a $K_{n,m}$ is monochromatic for $n \geq 3$ and $m \geq 2$: Assume that $u$ and $u'$ are not in the same part of a matching cut $(A, B)$. Then, at most one vertex of $\{v_1, v_2, v_3\}$ is in $A$ and at most one is in $B$. This is absurd and, thus, $\{u, u'\}$ is monochromatic which makes $U_i \cup U_j$ monochromatic. ◀

In the following, a cluster consisting of two vertices is an *edge cluster*, all other clusters are *nonedge* clusters. As we will show, after application of the above rules, we have essentially reached a situation in which there is a bounded number of nonedge clusters that are not contained in some $N^2(U_i)$, we call these clusters *ambiguous*. More precisely, we say that a vertex in $V \setminus U$ is *ambiguous* if it has neighbors in $U_i$ and $U_j$ where $i \neq j$. A cluster is *ambiguous* if it contains at least one ambiguous vertex. In contrast, we call a cluster *fixed* if it is contained in $N^2(U_i)$ for some $U_i$.

▶ **Proposition 2.** *If $G$ is reduced with respect to Reduction Rule 1, then every nonedge cluster in $G$ is ambiguous or fixed.*

**Proof.** Since $G$ is reduced with respect to Reduction Rule 1, every cluster $C$ contains at least one vertex $v$ that has two neighbors in $U$ or there is a vertex $u$ from some monochromatic part $U_i$ that has two neighbors in $C$. In the latter case, $C$ is contained in $N^2(U_i)$ and thus fixed. In the first case, if $v$ has two neighbors in the same part $U_i$, then $C \subseteq N^2(U_i)$ and $C$ is fixed. Otherwise, $v$ is ambiguous which means that $C$ is ambiguous. ◀

Observe that according to this definition, a nonedge cluster may be ambiguous and fixed at the same time. We will decrease the number of fixed clusters with the following rule.

▶ **Reduction Rule 4.** *If there are two clusters $C_1$ and $C_2$ that are contained in $N^2(U_i)$, then add all edges between these clusters.*

**Proof of safeness.** Let $G$ denote the graph to which the rule is applied and let $G'$ denote the resulting graph. If $G'$ has a matching cut, then so does $G$, because $G$ is a subgraph of $G'$ on the same vertex set.

For the converse, consider the following: $C_1 \cup C_2$ is monochromatic since $C_1$ and $C_2$ are contained in $N^2(U_i)$. Thus, if $G$ has a matching cut, then so does $G'$ because adding edges between vertices of a monochromatic set does not destroy the matching cut property. ◀

The following is obvious by the pigeonhole principle and the fact that the number of monochromatic parts is at most $|U|$.

▶ **Lemma 1.** *Let $G$ be an instance of* Matching Cut *with cluster vertex deletion set $U$ that is reduced with respect to Reduction Rule 4. Then $G$ has $O(|U|)$ fixed clusters.*

The next rules will help in bounding the number of vertices in the clusters.

▶ **Reduction Rule 5.** *If there is a cluster $C$ with more than three vertices that contains a vertex $v$ with no neighbors in $U$, then remove $v$.*

**Proof of safeness.** Let $G$ denote the original graph and let $G'$ be the graph obtained from the application of the reduction rule. Since $|C| \geq 4$, $C$ is monochromatic in $G$ and $C - v$ is monochromatic in $G'$. This and the fact that $v$ has only neighbors in $C$ immediately implies that $G$ and $G'$ are equivalent. ◀

After application of Rule 5, every vertex in a cluster of size at least three has a neighbor in $U$. The next rule removes unnecessary edges between monochromatic parts and clusters.

▶ **Reduction Rule 6.** *If there is a cluster $C$ with at least three vertices and a monochromatic set $U_i$ such that $C \subseteq N^2(U_i)$, then remove all edges between $C$ and $U_i$ from $G$, choose an arbitrary vertex $u \in U_i$ and two vertices $v_1, v_2 \in C$, and add two edges $\{u, v_1\}$ and $\{u, v_2\}$. If $|U_i| = 2$, then add an edge between $u' \in U_i \setminus \{u\}$ and $v_3 \in C \setminus \{v_1, v_2\}$. If $|U_i| > 2$, then make $U_i$ a clique.*

**Proof of safeness.** Let $G$ denote the original graph and let $G'$ be the graph obtained from the application of the reduction rule. Since $C \subseteq N^2(U_i)$, we have, by Proposition 1, that $U_i \cup C$ is monochromatic. Thus, if $G$ has a matching cut $(A, B)$, then without loss of generality we have $U_i \cup C \subseteq A$. This implies that $(A, B)$ is a matching cut of $G'$ since $G'$ is obtained from $G$ by removing and adding certain edges between vertices in $U_i \cup C \subseteq A$.

Conversely, assume that $G'$ has a matching cut. In $G'$, $U_i$ is monochromatic and thus the cluster $C$ is contained in $N^2(U_i)$. By Proposition 1, $U_i \cup C$ is monochromatic in $G'$. As above, since $G$ can be obtained from $G'$ by removing and adding certain edges between vertices in $U_i \cup C$, $G$ also has a matching cut. ◀

After application of these rules, the size of the instance is, with exception of the edge clusters, already bounded by a polynomial function of $|U|$ as we show in the following.

▶ **Lemma 2.** *Let $G$ be an instance of* Matching Cut *with cluster vertex deletion set $U$ that is reduced with respect to Reduction Rules 1–6. Then $G$ has*

- *$O(|U|^2)$ ambiguous vertices, and*
- *$O(|U|^2)$ nonedge clusters, each containing $O(|U|)$ vertices.*

The proof of Lemma 2 is deferred to the full version of this work.

To obtain a first bound on the instance size, it remains to reduce the overall number of edge clusters. To this end, we consider for each edge cluster $\{u, v\}$ the neighborhoods of $u$ and $v$ in $U$. First, observe that, assuming $G$ is reduced with respect to Reduction Rule 1, $u$ and $v$ have neighbors in $U$. Now we call an edge cluster $\{u, v\}$ *simple* if $u$ has only neighbors in $U_i$ and $v$ has only neighbors in $U_j$ (possibly $i = j$). Observe that the number of non-simple edge clusters is already bounded: Each such cluster is ambiguous because at least one of its vertices is ambiguous. By Lemma 2, there are $O(|U|^2)$ such vertices and thus $O(|U|^2)$ clusters containing them. To obtain a bound on the overall number of edge clusters, we show that all simple edge clusters can be removed.

▶ **Reduction Rule 7.** *If there is a simple edge cluster $\{u, v\}$, then remove $u$ and $v$ from $G$.*

The proof of the safeness of this rule is deferred to the full version of this work.

Thus, with the above rules we obtain a kernel with $O(\mathrm{dc}(G)^3)$ vertices: a reduced instance has $O(|U|^2) = O(\mathrm{dc}(G)^2)$ clusters, each containing $O(|U|) = O(\mathrm{dc}(G))$ vertices. To obtain a kernel with an overall quadratic number of vertices, we observe first that after applications

of Reduction Rule 6, every vertex in cluster $C$, where $|C| \geq 3$ and $C \subseteq N^2(U_i)$ for some $i$, has at most one neighbor in $U_i$.

It remains to bound the number of vertices in $V \setminus U$ with only one neighbor in $U$. First, we find monochromatic parts of $U$ that can be merged not because they have many common neighbors but, instead, because they have common neighbors in several nonedge clusters.

▶ **Reduction Rule 8.** *If there are two vertices $u \in U_i$ and $u' \in U_j$, $i \neq j$, and three distinct nonedge clusters $C_1, C_2, C_3$ such that $u$ and $u'$ have at least one neighbor in each of them, then merge $U_i$ and $U_j$.*

**Proof of safeness.** We show that $U_i \cup U_j$ is monochromatic; by definition this implies that merging $U_i$ and $U_j$ is safe. Let $(A, B)$ be any matching cut of $G$. Each of the three clusters is monochromatic because they are nonedge clusters. Hence, we can assume without loss of generality, that $A$ contains $C_1$ and $C_2$. Since $u$ has neighbors in $C_1$ and in $C_2$, it has two neighbors in $A$ and thus $U_i \subseteq A$. Similarly, $U_j \subseteq A$. Hence, $U_i \cup U_j$ is in the same part of the cut. This holds for all cuts, making $U_i \cup U_j$ monochromatic. ◀

▶ **Lemma 3.** *After exhaustive application of Reduction Rules 1–8, there are $O(|U^2|)$ vertices in $V \setminus U$ that are in nonedge clusters and have only one neighbor in $U$.*

**Proof.** First, by Lemma 1, there are $O(|U|)$ fixed nonedge clusters. By Lemma 2, these clusters contain $O(|U|)$ vertices each. Thus, the number of vertices in fixed nonedge clusters that have only one neighbor in $U$ is $O(|U|^2)$.

Hence, it remains to bound the number of vertices that have only one neighbor in $U$ and are contained in a nonfixed cluster $C$. By Proposition 2, these clusters are ambiguous. Each ambiguous cluster $C$ contains an ambiguous vertex with neighbors in $U_i$ and $U_j$, where $i \neq j$. Thus, for each vertex of $C$ with only one neighbor $u \in U$, there is at least one other vertex $u' \in U$ such that $u'$ has at least one neighbor in $C$, and $u$ and $u'$ are not from the same monochromatic part $U_\ell$ (because $u$ is in at most one of $U_i$ and $U_j$). Now, for each $u \in U$ and $u' \in U$ that are in distinct monochromatic parts of $U$, let $N(u, u')$ denote the number of vertex pairs $\{v, v'\}$ such that there is an ambiguous cluster $C$ containing $v$ and $v'$, one of $v$ and $v'$ is adjacent to $u$, and the other is adjacent to $u'$. By the above discussion, any vertex in an ambiguous cluster $C$ with exactly one neighbor in $U$ increases the number $N(u, u')$ for at least one pair of vertices $u$ and $u'$. By pigeonhole principle, if there are more than $3 \cdot \binom{|U|}{2}$ vertices in ambiguous nonedge clusters that have only one neighbor in $U$, then there is some pair of vertices $u$ and $u'$ such that $N(u, u') \geq 3$. Since $u$ and $u'$ each have at most one neighbor in every ambiguous cluster, this means that there are three distinct clusters $C_1$, $C_2$, $C_3$ which contain neighbors of $u$ and $u'$. Because $u$ and $u'$ are from distinct monochromatic parts, Reduction Rule 8 applies, which contradicts the fact that $G$ is reduced with respect to this rule. Consequently, the number of vertices in ambiguous nonedge clusters that have only one neighbor in $U$ is $O(|U|^2)$. ◀

▶ **Theorem 4.** Matching Cut *admits a problem kernel with $O(\mathrm{dc}(G)^2)$ vertices that can be computed in $O(\mathrm{dc}(G)^3 \cdot (n^2 + nm))$ time.*

**Proof.** Note that every instance contains $O(\mathrm{dc}(G))$ vertices in $U$ because we may assume that $|U| \leq 3\mathrm{dc}(G)$. Furthermore, the number of special vertices is also $O(\mathrm{dc}(G))$ since there is a constant number of them for every monochromatic part. To obtain the kernel, we need to reduce the size of $V \setminus U$. To this end, we first apply exhaustively Rules 1–8. Afterwards, $V \setminus U$ has $O(\mathrm{dc}(G)^2)$ vertices: By Lemma 2, $V \setminus U$ has $O(\mathrm{dc}(G)^2)$ ambiguous vertices. Moreover, since $G$ is reduced with respect to Rule 7, we have that every edge cluster

contains an ambiguous vertex. Hence, the number of edge clusters, and therefore the number of vertices in edge clusters, is $O(\mathrm{dc}(G)^2)$. It remains to bound the number of vertices in nonedge clusters that are not ambiguous. Each of these vertices has only one neighbor in $U$ because $G$ is reduced with respect to Reduction Rule 6. By Lemma 3, $V \setminus U$ has $O(\mathrm{dc}(G)^2)$ vertices that are in nonedge clusters and have only one neighbor in $U$.

Finally, the number of vertices that have no neighbors in any set $U_i$ is $O(1)$ for each cluster because $G$ is reduced with respect to Rule 5 and thus $O(\mathrm{dc}(G)^2)$ overall.

It remains to bound the running time for the application of the rules. Rule 1 can be applied in time $O(n+m)$ and applies only once. For the remaining rules, we need to maintain the set $N^2(U_i)$ for each $U_i$. These sets can be computed in $O(\mathrm{dc}(G)(n+m))$ time. Afterwards, the applicability of each rule can be tested in $O(\mathrm{dc}(G)^2(n+m))$ with the bottleneck being Rule 8. Moreover, each rule can be applied in $O(n)$ time, with the exception of Rule 4 which may take $\Theta(n^2)$ time, because it may add $\Theta(n^2)$ many edges. To make this rule more efficient, we can however store the edges in each cluster only implicitly, giving a running time bound of $O(n)$ also for this rule; we omit the details. Thus, to obtain the claimed bound on the running time it is sufficient to bound the number of applications of the rules by $O(\mathrm{dc}(G) \cdot n)$: All rules that merge monochromatic parts can be performed $O(\mathrm{dc}(G))$ times overall. For the remaining rules, we have that Rule 4 can be performed $O(n)$ times, because it decreases the number of clusters in $G - U$ by one, Rules 5 and 7 can be performed $O(n)$ times, because they remove at least one vertex from $G$. ◀

It is worth noting that if $G - U$ consists of only one cluster, i.e., $G - U$ is a clique $C$, Lemma 2 shows that $C$ can be reduced to contain $O(|U|)$ many vertices.

▶ **Corollary 5.** Matching Cut *admits a linear kernel when parameterized by* $\mathrm{dq}(G)$*, the distance to clique.*

## 3 Single-exponential FPT Algorithms

In this section, we consider Matching Cut parameterized by the distance to cluster and by the distance to co-cluster. Recall that co-cluster graphs are precisely the complete multipartite graphs. We show that Matching Cut can be solved in time $2^{\mathrm{dc}(G)}O(n^2)$ and in time $2^{\mathrm{d\bar{c}}(G)}O(nm)$. Recall that we may assume that all graphs considered are connected, have minimum degree at least two and that a clique $Q$ is monochromatic if $|Q| \neq 2$. Finally, observe that minimum distance to cluster sets and minimum distance to co-cluster sets can be computed in $O(1.92^{\mathrm{dc}(G)} \cdot n^2)$ time and $O(1.92^{\mathrm{d\bar{c}}(G)} \cdot n^2)$ time, respectively [5].

**Distance to Cluster.** Next, we provide an FPT algorithm solving Matching Cut running in single-exponential time $2^{\mathrm{dc}(G)}O(n^2)$.

▶ **Lemma 6.** *Let $U \subset V(G)$ such that $F = V(G) \setminus U$ induces a cluster graph. Given a partition $(A, B)$ of $G[U]$, it can be decided in time $O(n^2)$ if $G$ has a matching cut $(X, Y)$ such that $A \subseteq X$ and $B \subseteq Y$.*

**Proof.** We first consider the case $A$ or $B$ is empty, say $B = \emptyset$. Thus, $A = U$ and we are searching for a matching cut $(X, Y)$ such that $U \subseteq X$. If $G[F]$ has some connected component $Q$ such that $(Q, V(G) \setminus Q)$ is a matching cut, then we are done. Otherwise, consider some matching cut $(X, Y)$ such that $U \subseteq X$. For each connected component $Q$ of $G[F]$ we have $Q \subseteq X$ or $Q \subseteq Y$: This is obviously true for the connected components of size at least three and for those of size one because they are monochromatic in $G$. For each

connected component $\{u, v\}$ of size two observe the following. Since $(V(G) \setminus \{u, v\}, \{u, v\})$ is not a matching cut, either

- $u$ and $v$ have a common neighbor in $U$, or
- $|N(u) \cap U| \geq 2$ or $|N(v) \cap U| \geq 2$.

In the first case, $\{u, v\}$ is monochromatic, in the second case $u$ and $v$ are in the same part of the cut as $U$, and thus we have $\{u, v\} \subseteq X$. Summarizing, $U \subseteq X$ and for each connected component $Q$ of $G[F]$ we have $Q \subseteq X$ or $Q \subseteq Y$. Since there is no matching cut between $U$ and a connected component $Q$ of $G[F]$, this implies $Q \subseteq X$. Hence, $F \cup U \subseteq X$ and thus $Y = \emptyset$. Therefore, $G$ has no matching cut $(X, Y)$ such that $U \subseteq X$.

Now assume that $A$ and $B$ are nonempty. Then the algorithm first applies Reduction Rules (R1) – (R4) given in [14]; the correctness of these rules is easy to see.

**(R1)** If an $A$-vertex has two $B$-neighbors, or a $B$-vertex has two $A$-neighbors then STOP: "$G$ has no matching cut separating $A$, $B$".

**(R2)** If $v \in F$, $|N(v) \cap A| \geq 2$ and $|N(v) \cap B| \geq 2$ then STOP: "$G$ has no matching cut separating $A$, $B$".
If $v \in F$ and $|N(v) \cap A| \geq 2$ then $A := A \cup \{v\}$.
If $v \in F$ and $|N(v) \cap B| \geq 2$ then $B := B \cup \{v\}$.

**(R3)** If $v \in A$ has two adjacent $F$-neighbors $w_1, w_2$ then $A := A \cup \{w_1, w_2\}$.
If $v \in B$ has two adjacent $F$-neighbors $w_3, w_4$ then $B := B \cup \{w_3, w_4\}$.

**(R4)** If there is an edge $xy$ in $G$ such that $x \in A$ and $y \in B$ and $N(x) \cap N(y) \cap F \neq \emptyset$ then STOP: "$G$ has no matching cut separating $A$, $B$".
If there is an edge $xy$ in $G$ such that $x \in A$ and $y \in B$ then add $N(x) \cap F$ to $A$, and add $N(y) \cap F$ to $B$.

If none of these reduction rules can be applied then

- the $A, B$-edges of $G$ form a matching cut in $G[A \cup B] = G - F$ due to (R1),
- every $F$-vertex is adjacent to at most one $A$- and at most one $B$-vertex due to (R2),
- the $F$-neighbors of any $A$-vertex and the $F$-neighbors of any $B$-vertex form an independent set due to (R3), and
- every $A$-vertex adjacent to a $B$-vertex has no $F$-neighbor and every $B$-vertex adjacent to an $A$-vertex has no $F$-neighbor.

Clearly these properties hold for the instance $(G, A, B)$ if none of the Rules (R1) – (R4) can be applied. Note that $G[F]$ is a cluster graph. Let $\mathcal{Q}$ be the set of all monochromatic connected components in $G[F]$ and let $R := F \setminus \bigcup_{Q \in \mathcal{Q}} V(Q)$. That is, each member in $\mathcal{Q}$ is a trivial clique or a clique with at least three vertices, and each vertex in $R$ belongs to a 2-vertex connected component in $G[F]$. Now, create a boolean formula $\phi$ as follows:

- For each $Q \in \mathcal{Q}$ we have two boolean variables $Q_A$ and $Q_B$ (indicating all vertices of $Q$ should be added to $A$, respectively, to $B$).
- For each vertex $u \in R$ we have two boolean variables $u_A, u_B$ (indicating $u$ should be added to $A$, respectively, to $B$).

The clauses of $\phi$ are as follows:

**(c1)** For each $Q \in \mathcal{Q}$: $(Q_A \vee Q_B)$, $(\neg Q_A \vee \neg Q_B)$. These clauses ensure that $Q$ will be moved to $A$ or else to $B$.

**(c2)** For each vertex $u \in R$: $(u_A \vee u_B)$, $(\neg u_A \vee \neg u_B)$. These clauses ensure that $u$ will be moved to $A$ or else to $B$.

**(c3)** For each two adjacent vertices $u, v \in R$:

**(c3.1)** If $u$ has neighbors in $A$ and $B$, if $v$ has neighbors in $A$ and $B$, if $u$ and $v$ have neighbors in $A$, or if $u$ and $v$ have neighbors in $B$: $(u_A \leftrightarrow v_A)$, $(u_B \leftrightarrow v_B)$. These clauses ensure that either both $u$ and $v$ must be moved to $A$, or both must be moved to $B$.

**(c3.2)** If $|N(u) \cap A| = |N(v) \cap B| = 1$ and $N(u) \cap B = N(v) \cap A = \emptyset$: $(\neg u_B \vee \neg v_A)$. This clause ensures that in case $u$ goes to $B$, $v$ must also go to $B$, and in case $v$ goes to $A$, $u$ must also go to $A$.

**(c3.3)** If $|N(u) \cap B| = |N(v) \cap A| = 1$ and $N(u) \cap A = N(v) \cap B = \emptyset$: $(\neg u_A \vee \neg v_B)$. This clause ensures that in case $u$ goes to $A$, $v$ must also go to $A$, and in case $v$ goes to $B$, $u$ must also go to $B$.

**(c4)** For $z, z' \in \mathcal{Q} \cup R$:

**(c4.1)** If $z, z'$ have a common neighbor in $A$: $(\neg z_B \vee \neg z'_B)$. This clause ensures that in this case, $z$ or $z'$ must go to $A$.

**(c4.2)** If $z, z'$ have a common neighbor in $B$: $(\neg z_A \vee \neg z'_A)$. This clause ensures that in this case, $z$ or $z'$ must go to $B$.

Then $\phi$ is the conjunction of all these clauses over all $Q \in \mathcal{Q}$ and all $u \in R$.

▶ **Proposition 3.** *$G$ has a matching cut $(X, Y)$ with $A \subseteq X$ and $B \subseteq Y$ if and only if $\phi$ is satisfiable.*

The proof of Proposition 3 is deferred to the full version of this work.

Obviously, the length of the formula $\phi$ is $O(n^2)$. Since 2-Sat can be solved in linear time (cf. [3, 9, 10]), the above discussion yields an $O(n^2)$-time algorithm for deciding if $G$ has a matching cut $(X, Y)$ such that $A \subseteq X$ and $B \subseteq Y$. ◀

Running the algorithm of Lemma 6 for all partitions $(A, B)$ of $G[U]$, where $U$ is a minimum distance to cluster set of the input graph $G$, one obtains

▶ **Theorem 7.** Matching Cut *can be solved in time* $2^{\mathrm{dc}(G)}O(n^2)$.

**Distance to Co-cluster.** We now provide an FPT algorithm solving Matching Cut running in single-exponential time $2^{\mathrm{d\overline{c}}(G)}O(nm)$.

▶ **Lemma 8.** *Let $U \subset V(G)$ such that $F = V(G) \setminus U$ induces a co-cluster graph. Given a partition $(A, B)$ of $G[U]$, it can be decided in time $O(nm)$ if $G$ has a matching cut $(X, Y)$ such that $A \subseteq X$ and $B \subseteq Y$.*

The proof of Lemma 8 is deferred to the full version of this work.

Running the algorithm of Lemma 8 for all partitions $(A, B)$ of $G[U]$, where $U$ is a minimum distance to co-cluster set of the input graph $G$, one obtains

▶ **Theorem 9.** Matching Cut *can be solved in time* $2^{\mathrm{d\overline{c}}(G)}O(nm)$.

## 4 An Improved Exact Exponential Algorithm

Our algorithm takes as input a graph $G = (V, E)$ and decides whether or not there is an edge set $M \subseteq E$ such that $M$ is a matching cut of $G$. As above, we may assume that $G$ is connected and has minimum degree at least two. The idea is to compute a partition of the vertex set into subsets $A$ and $B$ such that $A$ and $B$ are nonempty unions of components of $G - M$ and all $M$-edges have one endvertex in $A$ and the other in $B$. Our algorithm consists of reduction rules and branching rules that label the vertices of the input graph by either $A$

or $B$ but never change the graph $G$. Finally we provide a termination lemma stating that if neither a reduction rule nor a branching rule can be applied then there is a matching cut in the graph $G$, respecting the current partial partition into $A$ and $B$.

The branching algorithm below will be executed for all possible pairs $a, b \in V$, hence $O(n^2)$ times. To do this set $A := \{a\}$, $B := \{b\}$, and $F := V \setminus \{a, b\}$ and call the branching algorithm. At each stage of the algorithm, $A$ and/or $B$ will be extended or it will be determined that there is no matching cut that separates $A$ from $B$.

We describe our algorithm by a list of reduction and branching rules given in preference order, i.e., in an execution of the algorithm on any instance of a subproblem one always applies the first rule applicable to the instance, which could be a reduction or a branching rule. A reduction rule produces one instance/subproblem while a branching rule results in at least two instances/subproblems, with different extensions of $A$ and $B$. Note that $G$ has a matching cut that separates $A$ from $B$ if and only if in at least one recursive branch, extensions $A'$ of $A$ and $B'$ of $B$ are obtained such that $G$ has a matching cut that separates $A'$ from $B'$. Typically a rule assigns one or more free vertices, vertices of $F$, either to $A$ or to $B$ and removes them from $F$, that is, we always have $F := V \setminus (A \cup B)$.

First our algorithm applies Reduction Rules (R1) – (R4) mentioned in Section 3 to the current instance (in the order of the rules). In addition, we need two new reduction rules.

**(R5)** If there are vertices $u, v \in F$ such that $N(u) = N(v) = \{x, y\}$ with $x \in A, y \in B$, then $A := A \cup \{u\}$, $B := B \cup \{v\}$.

▶ **Proposition 4.** *Reduction Rule (R5) is safe: $G$ has a matching cut separating $A$ and $B$ if and only if $G$ has a matching cut separating $A \cup \{u\}$ and $B \cup \{v\}$.*

The proof of Proposition 4 is deferred to the full version of this work.

**(R6)** If there are vertices $u, v \in F$ such that $N(u) = N(v) = \{x, y\}$ with $x \in A, y \in F$ then $A := A \cup \{u\}$.

If there are vertices $u, v \in F$ such that $N(u) = N(v) = \{x, y\}$ with $x \in F, y \in B$ then $B := B \cup \{v\}$.

▶ **Proposition 5.** *Reduction Rule (R6) is safe:*

**(i)** *Let $x \in A$ and $y \in F$. Then $G$ has a matching cut separating $A$ and $B$ if and only if $G$ has a matching cut separating $A \cup \{u\}$ and $B$.*

**(ii)** *Let $x \in F$ and $y \in B$. Then $G$ has a matching cut separating $A$ and $B$ if and only if $G$ has a matching cut separating $A$ and $B \cup \{v\}$.*

The proof of Proposition 5 is deferred to the full version of this work.

Our algorithm consists of six branching rules dealing with small configurations (connected subgraphs with at most eight vertices some of them may already belong to $A$ or $B$.) See Fig. 2. The Branching Rules (B1) and (B2) are new, the last four have been used in [14]. Moreover, compared to [14] we do a better branching on configuration (B5).

To determine the branching vectors which correspond to our branching rules, we set the size of an instance $(G, A, B)$ as its number of free vertices, i.e., $|V(G)| - |A| - |B|$.

**(B1)** We branch into two subproblems. First, add $v_1$ to $B$. Then $v_2$ has to be added to $B$ and $v_3$ has to be added to $A$. Second, add $v_1$ to $A$. Then $v_2$ has to be added to $A$ too. Hence the branching vector is $(3, 2)$.

**(B2)** We branch into two subproblems. First, add $v_2$ to $B$. Then $v_1$ has to be added to $A$ and $v_3$ has to be added to $B$. Second, add $v_2$ to $A$. Then $v_3$ has to be added to $A$ too. Hence the branching vector is $(3, 2)$.

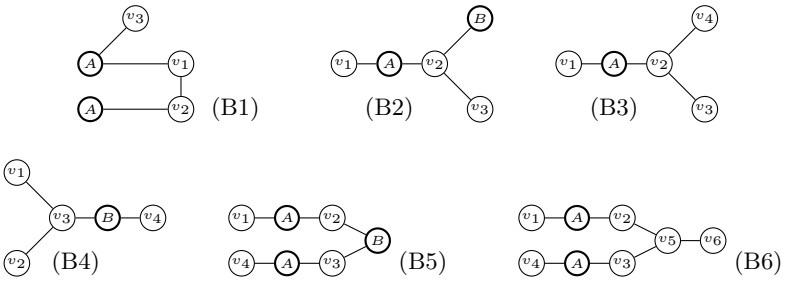



**Figure 2** Branching configurations: Vertices with label $A$ and $B$ belong to $A$, respectively, to $B$; vertices with labels $v_i$ are in $F$.

**(B3)** First, add $v_2$ to $B$. This implies that $v_1$ has to be added to $A$, and that $v_3$ and $v_4$ have to be added to $B$. Second, add $v_2$ to $A$. Hence the branching vector is $(4, 1)$.

**(B4)** On this configuration we branch into two subproblems, similar to (B3). First, add $v_3$ to $A$. This implies that $v_4$ has to be added to $B$, and that $v_1$ and $v_2$ have to be added to $A$. Second, add $v_3$ to $B$. Hence the branching vector is $(4, 1)$.

**(B5)** We branch into two subproblems. First, add $v_2$ to $A$. This implies that $v_3$ has to be added to $B$ and then $v_4$ has to be added to $A$. Second, add $v_2$ to $B$. Then $v_1$ must be added to $A$. Hence the branching vector is $(3, 2)$.

**(B6)** We branch into four subproblems. First, add $v_5$ to $A$. This implies that $v_2$ and $v_3$ have to be added to $A$. Next, add $v_5$ to $B$. There are three choices to label $v_2$ and $v_3$: $AB, BA, BB$. In the first two choices, $v_6$ has to be added to $B$ and $v_1$ or $v_4$ has to be added to $A$. In the last choice, $v_1$ and $v_4$ have to be added to $A$. Hence the branching vector is $(3, 5, 5, 5)$.

The branching numbers of the branching vectors of our algorithm are 1.3803 (B3, B4), 1.3734 (B6) and 1.3248 (B1, B2, B5). Consequently, the running time of our algorithm is $O^*(1.3803^n)$.

It remains to show that if none of the reduction and branching rules is applicable to $(G, A, B)$ then $G$ has a matching cut $(X, Y)$ with $A \subseteq X$ and $B \subseteq Y$. The proof of this fact is deferred to the full version of this work. In summary, we obtain the following result.

▶ **Theorem 10.** Matching Cut *can be solved in time* $O^*(1.3803^n)$.

## 5 Conclusions

We provided three algorithms for Matching Cut: an exact exponential algorithm of running time $O^*(1.3803^n)$, a fixed-parameter algorithm of running time $2^{\mathrm{dc}(G)}O(n^2)$ where $\mathrm{dc}(G)$ is the distance to cluster number, and a fixed-parameter algorithm of running time $2^{\mathrm{d\bar{c}}(G)}O(nm)$ where $\mathrm{d\bar{c}}(G)$ is the distance to co-cluster number. Our results improved the $O^*(1.4143^n)$ time exact algorithm and the $2^{\tau(G)}O(n^2)$ time algorithm previously given in [14], where $\tau(G) \geq \max\{\mathrm{dc}(G), \mathrm{d\bar{c}}(G)\}$ is the vertex cover number. Moreover, we found a quadratic vertex-kernel for Matching Cut for the distance to cluster, and a linear vertex-kernel for the distance to clique.

There are many possible directions for future research. Does Matching Cut admit a linear vertex-kernel for the distance to cluster? Even a linear vertex-kernel for the parameter vertex cover number $\tau(G)$ is open. Moreover, it is open whether the problem admits a polynomial kernel for the feedback vertex set number of $G$. Finally, it is natural to ask whether the running time of our $O^*(1.3803^n)$ branching algorithm can be improved.

## References


1 Júlio Araújo, Nathann Cohen, Frédéric Giroire, and Frédéric Havet. Good edge-labelling of graphs. *Discr. Appl. Math.*, 160(18):2502–2513, 2012.

2 N. R. Aravind, Subrahmanyam Kalyanasundaram, and Anjeneya Swami Kare. On Structural Parameterizations of the Matching Cut Problem. In *Proceedings of the 11th International Conference on Combinatorial Optimization and Applications (COCOA '17)*, volume 10628 of *LNCS*, pages 475–482. Springer, 2017.

3 Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A Linear-Time Algorithm for Testing the Truth of Certain Quantified Boolean Formulas. *Inf. Process. Lett.*, 8(3):121–123, 1979.

4 Paul S. Bonsma. The complexity of the matching-cut problem for planar graphs and other graph classes. *J. Graph Theory*, 62(2):109–126, 2009.

5 Anudhyan Boral, Marek Cygan, Tomasz Kociumaka, and Marcin Pilipczuk. A Fast Branching Algorithm for Cluster Vertex Deletion. *Theory Comput. Syst.*, 58(2):357–376, 2016.

6 Mieczyslaw Borowiecki and Katarzyna Jesse-Józefczyk. Matching cutsets in graphs of diameter 2. *Theor. Comput. Sci.*, 407(1-3):574–582, 2008.

7 Vasek Chvátal. Recognizing decomposable graphs. *J. Graph Theory*, 8(1):51–53, 1984.

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

9 Martin Davis and Hilary Putnam. A Computing Procedure for Quantification Theory. *J. ACM*, 7(3):201–215, 1960.

10 Shimon Even, Alon Itai, and Adi Shamir. On the Complexity of Timetable and Multicommodity Flow Problems. *SIAM J. Comput.*, 5(4):691–703, 1976.

11 Arthur M. Farley and Andrzej Proskurowski. Networks immune to isolated line failures. *Networks*, 12(4):393–403, 1982.

12 Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Springer, 2010.

13 Ron L. Graham. On primitive graphs and optimal vertex assignments. *Ann. N. Y. Acad. Sci.*, 175(1):170–186, 1970.

14 Dieter Kratsch and Van Bang Le. Algorithms solving the Matching Cut problem. *Theor. Comput. Sci.*, 609:328–335, 2016.

15 Hoàng-Oanh Le and Van Bang Le. On the Complexity of Matching Cut in Graphs of Fixed Diameter. In *Proceedings of the 27th International Symposium on Algorithms and Computation (ISAAC '16)*, volume 64 of *LIPIcs*, pages 50:1–50:12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

16 Van Bang Le and Bert Randerath. On stable cutsets in line graphs. *Theor. Comput. Sci.*, 301(1-3):463–475, 2003.

17 Augustine M. Moshi. Matching cutsets in graphs. *J. Graph Theory*, 13(5):527–536, 1989.

18 Maurizio Patrignani and Maurizio Pizzonia. The Complexity of the Matching-Cut Problem. In *Proceedings of the 27th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '01)*, volume 2204 of *LNCS*, pages 284–295. Springer, 2001.

# Subset Feedback Vertex Set on Graphs of Bounded Independent Set Size


## Charis Papadopoulos

Department of Mathematics, University of Ioannina, Greece
charis@cs.uoi.gr

## Spyridon Tzimas

Department of Mathematics, University of Ioannina, Greece
roytzimas@hotmail.com



### Abstract

The (Weighted) Subset Feedback Vertex Set problem is a generalization of the classical Feedback Vertex Set problem and asks for a vertex set of minimum (weight) size that intersects all cycles containing a vertex of a predescribed set of vertices. Although the two problems exhibit different computational complexity on split graphs, no similar characterization is known on other classes of graphs. Towards the understanding of the complexity difference between the two problems, it is natural to study the importance of a structural graph parameter. Here we consider graphs of bounded independent set number for which it is known that Weighted Feedback Vertex Set can be solved in polynomial time. We provide a dichotomy result with respect to the size of a maximum independent set. In particular we show that Weighted Subset Feedback Vertex Set can be solved in polynomial time for graphs of independent set number at most three, whereas we prove that the problem remains NP-hard for graphs of independent set number four. Moreover, we show that the (unweighted) Subset Feedback Vertex Set problem can be solved in polynomial time on graphs of bounded independent set number by giving an algorithm with running time $n^{\mathcal{O}(d)}$, where $d$ is the size of a maximum independent set of the input graph. To complement our results, we demonstrate how our ideas can be extended to other terminal set problems on graphs of bounded independent set size. Based on our findings for Subset Feedback Vertex Set, we settle the complexity of Node Multiway Cut, a terminal set problem that asks for a vertex set of minimum size that intersects all paths connecting any two terminals, as well as its variants where nodes are weighted and/or the terminals are deletable, for every value of the given independent set number.



**2012 ACM Subject Classification** Mathematics of computing → Graph algorithms

**Keywords and phrases** Subset Feedback Vertex Set, Node Multiway Cut, Terminal Set problem, polynomial-time algorithm, NP-completeness, W[1]-hardness, graphs of bounded independent set size

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.20

**Related Version** A full version of the paper is available at `https://arxiv.org/abs/1805.07141`, [32].

**Funding** This research has been financially supported by General Secretariat for Research and Technology (GSRT) and the Hellenic Foundation for Research and Innovation (HFRI) (Scholarship Code: 82220).



© Charis Papadopoulos and Spyridon Tzimas;
licensed under Creative Commons License CC-BY


13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 20; pp. 20:1–20:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Given a (vertex-weighted) graph $G = (V, E)$ and a set $S \subseteq V$, the (Weighted) Subset Feedback Vertex Set problem, introduced by Even et al. [13], asks for a vertex set of minimum (weight) size that intersects all cycles containing a vertex of $S$. Cygan et al. [10] and Kawarabayashi and Kobayashi [25] independently showed that Subset Feedback Vertex Set is fixed-parameter tractable (FPT) parameterized by the solution size, while Hols and Kratsch provided a randomized polynomial kernel for the problem [21]. There has been a considerable amount of work to obtain faster, still exponential-time, algorithms even when restricted to particular graph classes [5, 16, 15, 19]. As a generalization of the classical Feedback Vertex Set for which $S = V$, the problem remains NP-hard on bipartite graphs [36] and planar graphs [17]. On the positive side, Weighted Subset Feedback Vertex Set can be solved in polynomial time on interval graphs, permutation graphs, and cobipartite graphs [31], the latter being a subclass of graphs of independent set size at most two. However a notable difference between the two problems regarding their complexity status is the class of split graphs: Feedback Vertex Set is known to be polynomial-time solvable on split graphs [7, 34], whereas Subset Feedback Vertex Set remains NP-hard on split graphs [16].

In order to obtain further (in)tractability results for Subset Feedback Vertex Set, it is reasonable to consider structural parameters of graphs that may lend themselves to provide a unified approach. In terms of parameterized complexity Feedback Vertex Set is known to be FPT, when parameterized by tree-width [8] and clique-width [2] which implies that Feedback Vertex Set can be solved in polynomial time on graphs of bounded such parameters. Although Feedback Vertex Set is W[1]-hard parameterized by the size of the independent set, it can be solved in polynomial time on graphs of bounded maximum induced matching (i.e., Feedback Vertex Set belongs in XP parameterized by the size of a maximum induced matching) [24]. Only very recently, Jaffke et al. proposed an algorithm that solves Weighted Feedback Vertex Set in time $n^{\mathcal{O}(w)}$ where $w$ is the *maximum induced matching width* of the given graph [23]. Despite their relevant name, graphs of bounded maximum induced matching (or graphs of bounded independent set number) are not related to graphs of bounded maximum induced matching width as indicated in [35].

The approach of [23] provides a powerful mechanism, as it unifies polynomial-time algorithms for Weighted Feedback Vertex Set on several graph classes such as interval graphs, permutation graphs, circular-arc graphs, and Dilworth-$k$ graphs for fixed $k$, among others. Such a mechanism raises the question of whether the algorithm given in [23] can be extended to the more general setting of Weighted Subset Feedback Vertex Set. However the proposed algorithm is based on the crucial fact that the forest of a solution has bounded number of internal nodes which is not necessarily true for the $S$-forest of Weighted Subset Feedback Vertex Set. Thus it seems difficult to control the size of the solution whenever $S \subset V$. As this observation does not rule out any positive answer, here we develop the first step towards such an approach by considering graphs of bounded independent set number which form candidate relevant graphs. Although Weighted Feedback Vertex Set can be solved in time $n^{\mathcal{O}(p)}$ on graphs of maximum induced matching at most $p$ [24], Subset Feedback Vertex Set is already NP-complete on graphs of maximum induced matching equal to one (i.e., split graphs) [16].

In this work we show that the complexity behaviour of the weighted version of the problem is completely different from the behaviour of the unweighted variant on graphs with bounded $\alpha(G)$, where $\alpha(G)$ is the size of a maximum independent set in a graph $G$.

- We show that Weighted Subset Feedback Vertex Set can be solved in polynomial time on graphs with $\alpha(G) \leq 3$.

Such graphs consist of the complements of triangle-free graphs; recall that for triangle-free graphs Feedback Vertex Set remains NP-hard [36]. We solve Weighted Subset Feedback Vertex Set on such graphs, by exploiting a structural characterization of the solution with respect to the vertices that are *close* to $S$.

- We further provide a dichotomy result showing that Weighted Subset Feedback Vertex Set remains NP-complete on graphs with $\alpha(G) = 4$.

Thus we enlarge our knowledge on the complexity difference of the two problems with respect to a structural graph parameter.

- In order to complement our results we show that Subset Feedback Vertex Set can be solved in time $n^{\mathcal{O}(d)}$, where $\alpha(G) \leq d$.

Our findings concerning Subset Feedback Vertex Set are summarized in Table 1.

Moreover, we demonstrate how our ideas can be extended to other *terminal set* problems on graphs of bounded independent set size. In the (unweighted) Node Multiway Cut problem, we are given a graph $G = (V, E)$, a terminal set $T \subseteq V$, and a nonnegative integer $k$ and the goal is to find a set $X \subseteq V \setminus T$ of size at most $k$ such that any path between two different terminals intersects $X$. Node Multiway Cut is known to be in FPT parameterized by the solution size [4, 29] and even above guaranteed value [9]. For further results on variants of Node Multiway Cut we refer to [3, 18, 27]. We completely characterize the complexity of Node Multiway Cut with respect to the size of the maximum independent set.

- In particular, we show that for $\alpha(G) \leq 2$ Node Multiway Cut can be solved in polynomial time, whereas for $\alpha(G) = 3$ it remains NP-complete by adopting the reduction for Weighted Subset Feedback Vertex Set with $\alpha(G) = 4$.

We further consider a relaxed variation of Node Multiway Cut in which we are allowed to remove terminal vertices, called Node Multiway Cut with Deletable Terminals (also known as Unrestricted Node Multiway Cut).

- We show that the (unweighted) Node Multiway Cut with Deletable Terminals problem can be solved in polynomial time on graphs of bounded independent set number, using an idea similar to the polynomial-time algorithm for the Subset Feedback Vertex Set problem.
- Regarding its node-weighted variation, we provide a complexity dichotomy result showing that Weighted Node Multiway Cut with Deletable Terminals can be solved in polynomial time on graphs with $\alpha(G) \leq 2$, whereas it becomes NP-complete on graphs with $\alpha(G) = 3$.

We note that the polynomial-time algorithm for the weighted variation is obtained by invoking our algorithm for Weighted Subset Feedback Vertex Set on graphs with $\alpha(G) \leq 3$.

## 2 Preliminaries

We refer to [1, 11, 20] for our standard graph terminology. For $X \subseteq V$, $N_G(X) = \bigcup_{v \in X} N_G(v) \setminus X$ and $N_G[X] = N_G(X) \cup X$. A *weighted graph* $G = (V, E)$ is a graph, where each vertex $v \in V$ is assigned a *weight* that is a positive integer number. We denote by $w(v)$ the weight of each vertex $v \in V$. For a vertex set $A \subset V$, the weight of $A$, denoted by $w(A)$, is $\sum_{v \in A} w(v)$.

Given a graph $G$, the *independent set number*, denoted by $\alpha(G)$, is the size of the maximum independent set in $G$. In terms of forbidden subgraph characterization, note that $\alpha(G) \leq d$ if and only if $G$ does not contain $(d+1)K_1$ as an induced subgraph. We say that

**Table 1** Computational complexity results for Feedback Vertex Set (FVS) and Subset Feedback Vertex Set (SFVS) on graphs of bounded independent set number and graphs of bounded maximum induced matching. Note that every graph of independent set number $d$ has maximum induced matching of size at most $d$, while the converse is not necessarily true.

| | Bounded Structural Parameter | | | |
|---|---|---|---|---|
| | Max. Independent Set ($d$) | | | Max. Induced Matching ($p$) |
| Weighted FVS | | | | $n^{\mathcal{O}(p)}$ [24] |
| Weighted SFVS | $d \leq 3$ | $n^{\mathcal{O}(1)}$ | Theorem 6 | |
| | $d = 4$ | NP-complete | Theorem 7 | |
| Unweighted SFVS | | $n^{\mathcal{O}(d)}$ | Theorem 8 | NP-complete [16] |

a graph $G$ has *bounded independent set size* if there exists a positive integer $d$ such that $\alpha(G) \leq d$. The *clique cover number* of $G$, denoted by $\kappa(G)$, is the smallest number of cliques needed to partition $V(G)$ into $S_1, \ldots, S_k$ such that $G[S_i]$ is a clique. A *vertex cover* is a set of vertices such that every edge of $G$ is incident to at least one vertex of the set. A *matching* is a set of edges having no common endpoint. An *induced matching*, denoted by $pK_2$, is a matching $M$ of $p$ edges such that $G[V(M)]$ is isomorphic to $pK_2$. The *maximum induced matching* number, denoted by $p(G)$, is the largest number of edges in any induced matching of $G$. It is not difficult to see that for any graph $G$, $\kappa(G) \geq \alpha(G) \geq p(G)$ holds.

The (Weighted) Subset Feedback Vertex Set (SFVS) problem asks for a given (vertex-weighted) graph $G = (V, E)$, a set $S \subseteq V$, and an integer $k$, whether there exists a set $X$ with $|X| \leq k$ ($w(X) \leq k$) such that no cycle in $G - X$ contains a vertex of $S$. As remarked, we distinguish between the weighted and the unweighted version of the problem. In the unweighted version of the problem note that all weights are equal and positive. The classical Feedback Vertex Set (FVS) problem is a special case of Subset Feedback Vertex Set with $S = V$. A vertex of $S$ is simply called *$S$-vertex*. An induced cycle of $G$ is called *$S$-cycle* if an $S$-vertex is contained in the cycle. We define an *$S$-forest* $F = (V_F, E_F)$ to be the subgraph of $G$ induced by the vertex set $V_F \subseteq V$ for which no cycle in $G[V_F]$ is an $S$-cycle. It is not difficult to see that the problem of computing a minimum weighted subset feedback vertex set is equivalent to the problem of computing a maximum weighted $S$-forest.

Let us give a couple of observations on the nature of Subset Feedback Vertex Set on graphs of bounded independent set size. Let $G$ be a graph and let $d$ be a positive integer such that every independent set of $G$ has at most $d$ vertices. Firstly note that the bounded-size independent set is a hereditary property, meaning that for every induced subgraph $H$ of $G$, we have $\alpha(H) \leq d$. Moreover for any clique $C$ of $G$, any $S$-forest of $G$ contains at most two vertices of $S \cap C$.

▶ Observation 1. Let $G$ be a graph with $\alpha(G) \leq d$ and let $S \subseteq V$.

**(1)** For any set $X$ of $2d + 1$ vertices, there is a cycle in $G[X]$.

**(2)** Any $S$-forest of $G$ has at most $2d$ vertices from $S$.

We note that Observation 1 directly implies that any $2d + 1$ vertices of $G[S]$ induce an $S$-cycle, which allows us to construct by brute force all possible subsets of $S$ belonging to any $S$-forest in time $n^{\mathcal{O}(d)}$.

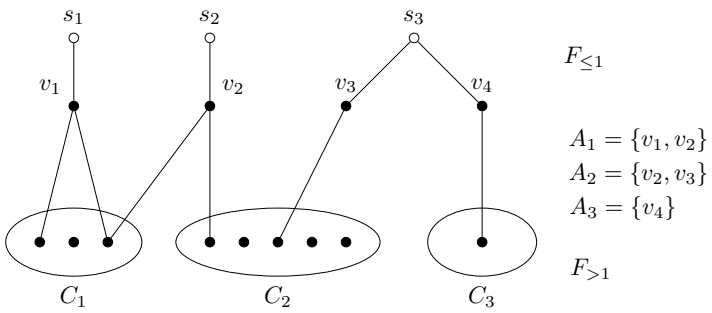



**Figure 1** Illustrating an $S$-distance partition $(F_{\leq 1}, F_{>1})$ of an $S$-forest $F$ with $S = \{s_1, s_2, s_3\}$ that shows the connected components $C_1, C_2, C_3$ of $F_{>1}$. The edges inside $F_{>1}$ are not drawn in order to highlight that the cut satisfies the given tuple $(A_1, A_2, A_3)$.

## 3 Weighted SFVS on Graphs of Bounded Independent Set

Here we consider the Weighted Subset Feedback Vertex Set and we show a dichotomy result with respect to the size of the maximum independent set. We first provide a polynomial-time algorithm on graphs of independent set size at most three and then we show that Weighted Subset Feedback Vertex Set is NP-complete on graphs of independent set size equal to four.

Let $(G, S, k)$ be an instance of Weighted Subset Feedback Vertex Set for which $G$ is a graph of independent set size at most $d$. In the forthcoming arguments, instead of directly computing a solution for Weighted Subset Feedback Vertex Set, we consider the equivalent problem of computing an $S$-forest of $G$ having weight at least $w(V) - k$.

Let $F = (V_F, E_F)$ be an induced subgraph of $G$. Recall that an $S$-forest is an induced subgraph of $G$. We partition the graph $F$ into two induced subgraphs $F_{\leq 1}$ and $F_{>1}$ as follows:

- $F_{\leq 1}$ is the subgraph of $F$ induced by the vertices of $N[S \cap V_F]$; the vertices of $F_{\leq 1}$ are at distance at most one from $S \cap V_F$ and are denoted by $S_{\leq 1}$.
- $F_{>1}$ is the graph $F - S_{\leq 1}$ and contains vertices that are at distance at least two from $S \cap V_F$.

Such a partition is called the *$S$-distance partition* of $F$, denoted by $(F_{\leq 1}, F_{>1})$. The set of edges of $F$ having one endpoint in $F_{\leq 1}$ and the other in $F_{>1}$ are called *the cut* with respect to $F_{\leq 1}$ and $F_{>1}$. Notice that a vertex of $F_{\leq 1}$ that is adjacent to a vertex of $F_{>1}$ belongs to $S_{\leq 1} \setminus S$.

Let $\{C_1, \ldots, C_{d'}\}$ be the partition of the vertices of $F_{>1}$ such that each $C_i$, $1 \leq i \leq d'$, induces a connected component in $F_{>1}$. Because $F_{>1}$ is an induced subgraph of $G$, it is clear that $d' \leq d$. Let $(A_1, \ldots, A_{d'})$ be a tuple of $d'$ subsets of $S_{\leq 1} \setminus S$, i.e., each $A_i \subseteq (S_{\leq 1} \setminus S)$ holds. We say that *the cut satisfies* the tuple $(A_1, \ldots, A_{d'})$ if for any vertex $v \in C_i$, we have $(N_G(v) \cap S_{\leq 1}) \subseteq A_i$. The notion of an $S$-distance partition of $F$ with the corresponding cut is illustrated in Figure 1.

We now utilize the $S$-distance partition of $F$ in order to construct an algorithm that solves Weighted Subset Feedback Vertex Set on graphs of independent set size at most $d$ and subsequently show that this algorithm is efficient for $d \leq 3$. Our general approach relies on the following facts:

- By Observation 1 (2) we try all subsets $S'$ of $S$ with at most $2d$ vertices and keep those sets that induce a forest. This step is used in constructing the graph $F_{\leq 1}$. In particular,

for each such set $S'$, we construct all $F_{\leq 1}$ such that $S \cap V(F_{\leq 1}) = S'$. We will show that the number of such subsets produced is bounded by $n^{\mathcal{O}(d)}$.

- For each of the potential subsets $S'$ constructed in the previous step, and for each $d' \leq d$, we determine all possible tuples $(A_1, \ldots, A_{d'})$ in $F_{\leq 1}$ with $A_i \subseteq (N[S'] \setminus S)$ that are satisfied by cuts of $S$-distance partitions. We show why considering only these tuples is sufficient in Lemma 2.
- Up to that point we can show that all steps can be executed in polynomial time regardless of $d \leq 3$. However for the next and final step we can only achieve polynomial running time if we restrict ourselves to $d \leq 3$ due to the number of connected components of $F_{>1}$. For each tuple computed in the previous step, we find connected components $C_1, \ldots, C_{d'}$ of maximum weight such that the cut of $(G[S'], G[C_1 \cup \cdots \cup C_{d'}])$ satisfies the tuple. For doing so, we take advantage of the small number of connected components ($d' < 3$) and an efficient way of computing a vertex-cut between such components.

We begin by showing that the $S$-distance partition of $F$ provides a useful tool towards computing a maximum $S$-forest. Given a set of vertices $X \subseteq N[S]$ and $d'$ subsets $A_i$ of $X \setminus S$, we construct the graph $\widehat{G}$ that is obtained from $G[X]$ by adding $d'$ vertices $w_1, \ldots, w_{d'}$ such that every vertex $w_i$ is adjacent to all the vertices of $A_i$. In what follows, we always assume that $G$ is a graph having independent set size at most $d$.

▶ **Lemma 2.** *Let $F$ be an $S$-forest of $G$ with $S$-distance partition $(F_{\leq 1}, F_{>1})$ such that $S_{\leq 1} \cap S \neq \varnothing$. Then for $d' \leq d$, there is a tuple $(A_1, \ldots, A_{d'})$ with $A_i \subseteq (S_{\leq 1} \setminus S)$ such that*

**(i)** *the cut of $(F_{\leq 1}, F_{>1})$ satisfies $(A_1, \ldots, A_{d'})$ and*

**(ii)** *every induced subgraph $H$ of $G$ with $S$-distance partition $(H[S_{\leq 1}], H - S_{\leq 1})$ that satisfies $(A_1, \ldots, A_{d'})$ is an $S$-forest.*

**Proof.** Let $\{C_1, \ldots, C_{d'}\}$ be the partition of the vertices of $F_{>1}$ such that every $C_i$ induces a connected component in $F_{>1}$. We define a tuple $(A_1, \ldots, A_{d'})$ in which every $A_i = N(C_i) \cap S_{\leq 1}$, for $1 \leq i \leq d'$. Clearly $A_i \subseteq (S_{\leq 1} \setminus S)$ since every vertex $F_{>1}$ is at distance at least two from $S_{\leq 1} \cap S$. Thus, by construction, the cut of $(F_{\leq 1}, F_{>1})$ satisfies the tuple $(A_1, \ldots, A_{d'})$.

For the next claim, we first show that $\widehat{G}$ with respect to $S_{\leq 1}$ and the tuple $(A_1, \ldots, A_{d'})$ is an $S$-forest. Assume for contradiction that there is an $S$-cycle $\widehat{C}$ in $\widehat{G}$. Since $F_{\leq 1}$ does not contain any $S$-cycle, $\widehat{C}$ contains a vertex $w_i$ and at least two vertices $u_i, v_i$ from $A_i$, $1 \leq i \leq d'$. By the fact that $A_i = N(C_i) \cap S_{\leq 1}$, there is a vertex $x$ in $C_i$ of $F_{>1}$ that is adjacent to $u_i$ and there is a vertex $y$ in $C_i$ of $F_{>1}$ that is adjacent to $v_i$. Together with a path between $x$ and $y$ in the connected component $C_i$, we construct a path in $G$ with endvertices $u_i$ and $v_i$ that is completely contained in $C_i$. This means that if we replace every vertex $w_i$ of $\widehat{C}$ by a path with internal vertices of $C_i$ then we obtain an $S$-cycle in $F$, leading to a contradiction. Thus, $\widehat{G}$ is an $S$-forest.

Let $H$ be an induced subgraph of $G$ with $S$-distance partition $(H[S_{\leq 1}], H - S_{\leq 1})$ that satisfies $(A_1, \ldots, A_{d'})$. Observe that $H[S_{\leq 1}] = F_{\leq 1}$ as they are induced subgraphs of the same vertex set of $G$. Thus $H[S_{\leq 1}]$ does not contain any $S$-cycle, because $F$ is an $S$-forest. Since the cut of $(H[S_{\leq 1}], H - S_{\leq 1})$ satisfies $(A_1, \ldots, A_{d'})$, there is a partition $\{T_1, \ldots, T_{d'}\}$ in $H - S_{\leq 1}$ such that $T_i$ is a connected component of $H - S_{\leq 1}$ and $N(T_i) \subseteq A_i$, for $1 \leq i \leq d'$. We show that $H$ is indeed an $S$-forest. For contradiction, assume an $S$-cycle $C$ in $H$. There are no $S$-cycles in $H[S_{\leq 1}]$ which implies that $C \cap T_i \neq \varnothing$ for some $1 \leq i \leq d'$. For every such set we replace the part $C \cap T_i$ by a vertex $w'_i$. Denote by $H'$ the resulting graph. Notice that $H'[C]$ is a subgraph of $\widehat{G}[C]$ because $N_{H'}(w'_i) \subseteq N_{\widehat{G}}(w_i)$. This, however, implies an $S$-cycle in $\widehat{G}$ which gives the desired contradiction. Therefore, $H$ is an $S$-forest. ◀

Notice that $G - S$ is trivially an $S$-forest of $G$. Thus, if $F$ is an $S$-forest of $G$ such that $S_{\leq 1} \cap S = \varnothing$ then $F$ does not contain any vertex of $S$ and $F = G - S$. Next, we assume that $S_{\leq 1} \cap S \neq \varnothing$ and show how to bound the vertex set $S_{\leq 1}$ of $F_{\leq 1}$.

▶ **Lemma 3.** *Let $F$ be an $S$-forest of $G$ with $S$-distance partition $(F_{\leq 1}, F_{>1})$ such that $S_{\leq 1} \cap S \neq \varnothing$.*
1. *If $|S_{\leq 1} \cap S| \leq 2d - 2$ then $|S_{\leq 1}| \leq 4d - 2$.*
2. *If $|S_{\leq 1} \cap S| \geq 2d - 1$ then $|S_{\leq 1}| \leq 2d$.*

**Proof.** Let $F$ be such an $S$-forest of $G$ with $|S_{\leq 1} \cap S| \geq 1$. By Observation 1 (2), we know that $|S_{\leq 1} \cap S| \leq 2d$. To ease the presentation, we let $S' = S_{\leq 1} \setminus S$. We consider separately the two cases of the claim.

**Case 1.** Let $1 \leq |S_{\leq 1} \cap S| \leq 2d-2$. Assume for contradiction that $|S'| > 4d - |S_{\leq 1} \cap S| - 2$. We show that $F[S']$ contains a matching with at least $d$ edges. Observe that $|S'| + |S_{\leq 1} \cap S| > 4d - 2$. Applying Observation 1 (1) shows that there is a cycle $C$ in $F[S_{\leq 1}]$. Since $F$ is an $S$-forest, this is not an $S$-cycle, so all vertices contained in $C$ are vertices of $S'$. Iteratively removing the two endpoints of an edge from $C$, as long as $|S'| + |S_{\leq 1} \cap S| > 2d$, constructs $d$ edges of $S'$ having no common endpoints by Observation 1 (1). Thus, $F[S']$ contains a matching $M$ with at least $d$ edges.
Let $C_1, \ldots, C_{d'}$ be the connected components of $F[S_{\leq 1} \cap S]$. Notice that $d' \leq d$ because $F[S_{\leq 1} \cap S]$ is an induced subgraph of a graph with maximum independent set size at most $d$. By construction, every vertex of $S'$ is adjacent to at least one vertex of $S_{\leq 1} \cap S$. If the endpoints of an edge of $M$ in $S'$ are adjacent to vertices of the same component $C_i$, $1 \leq i \leq d'$, then there is an $S$-cycle in $F$ since every vertex of $C_i$ belongs to $S$. Thus the endpoints of every edge of $M$ are adjacent to different connected components of $F[S_{\leq 1} \cap S]$. Now obtain a bipartite graph by contracting every component $C_i$ into a single vertex and every edge of $M$ into a single vertex and keep only the adjacencies between the components and the edges of $M$. Let $(A, B)$ be the bipartition of the resulting bipartite graph such that $A$ contains the components of $F[S_{\leq 1} \cap S]$ and $B$ contains the edges of $M$. Since $|A| \leq |B|$ and every vertex of $B$ is adjacent to at least two vertices of $A$, there is a cycle in the bipartite graph. Then, it is not difficult to see that the cycle of the contracted vertices corresponds to an $S$-cycle in $F$. Therefore there is an $S$-cycle in an $S$-forest, leading to a contradiction.

**Case 2.** Let $2d - 1 \leq |S_{\leq 1} \cap S| \leq 2d$. Assume for contradiction that $|S'| > 2d - |S_{\leq 1} \cap S|$. This means that $S'$ contains at least one vertex. We pick a nonempty subset $W$ of $S'$ as follows. If $|S_{\leq 1} \cap S| = 2d - 1$ then $W$ consists of any two vertices of $S'$. If $|S_{\leq 1} \cap S| = 2d$ then $W$ consists of an arbitrary vertex of $S'$. In both cases, notice that $|S_{\leq 1} \cap S| + |W| > 2d$ by the fact $2d - 1 \leq |S_{\leq 1} \cap S|$. Then Observation 1 (1) implies that there is a cycle in $F[(S_{\leq 1} \cap S) \cup W]$. Since $W$ has at most two vertices, we conclude that the induced cycle of $F[(S_{\leq 1} \cap S) \cup W]$ has at least one vertex from $S$, hence it is an $S$-cycle in $F$. Therefore we reach a contradiction which implies that $|S'| \leq 2d - |S_{\leq 1} \cap S|$. ◀

Lemma 3 shows that we can compute all possible candidates for $S_{\leq 1}$ in polynomial time as follows.

- We first try, by brute force, all subsets $S'$ of $S$ having at most $2d$ vertices, according to Observation 1 (2).
- Then, for each such subset $S'$, we incorporate a set $X' \subseteq N(S')$ for which either $|X'| + |S'| \leq 4d - 2$, or $|X'| + |S'| \leq 2d$, according to Lemma 3.
- Given the described sets $S'$ and $X'$, we check if $G[S' \cup X']$ induces an $S$-forest and, if so, we include them into a list $L_1$ containing all candidates for $S_{\leq 1}$.

The correctness follows from Observation 1 and Lemma 3. Regarding the running time, notice that we create at most $n^{\mathcal{O}(d)}$ subsets for each of $S'$ and $X' \subseteq N(S')$. Thus, in time $n^{\mathcal{O}(d)}$ we can compute a list $L_1$ that contains all possible subsets of the vertices corresponding to $S_{\leq 1}$. Notice that such vertices are enough to build the part $F_{\leq 1}$.

Let $S_{\leq 1}$ be a set of $L_1$. We now focus on the graph $G' = G - (S_{\leq 1} \cup S)$ that contains the vertices that are at distance of at least two from $S_{\leq 1} \cap S$. Observe that for any $S$-forest $F$, the set of all vertices in $F$ which are at distance of at most one from the vertices of $S \cap V(F)$, are present as an element of $L_1$. Let $d'$ be the number of connected components of $G'$. It is clear that $d' \leq d$. In fact, if $S_{\leq 1} \cap S$ contains at least one vertex then $d' < d$, since the vertices of $G'$ are at distance of at least two from $S_{\leq 1} \cap S$. Moreover, observe that if $S_{\leq 1} \cap S = \varnothing$ then $G - S$ is a trivial solution, since we try all subsets of $S$, having at most $2d$ vertices, for the set $S_{\leq 1}$. From now on, we assume that $|S_{\leq 1} \cap S| \geq 1$ so that $d' < d$.

By brute force, we find all tuples $(A_1, \ldots, A_{d'})$ such that the following hold:

**(i)** $A_i \subseteq (S_{\leq 1} \setminus S)$, for every $1 \leq i \leq d'$, and

**(ii)** the graph $\widehat{G}$ with respect to $S_{\leq 1}$ and $(A_1, \ldots, A_{d'})$ is an $S$-forest.

Notice that by Lemma 2 (ii) it is sufficient to consider only such tuples. Since $A_i \subseteq S_{\leq 1}$, $d' < d$, and $|S_{\leq 1}| \leq 4d$, the number of tuples is $d^{\mathcal{O}(d)}$, so that we can obtain the desired set of tuples that satisfy both conditions in polynomial time.

In what follows, we consider the case for $d \leq 3$. By the previous arguments, we are given a set $S_{\leq 1} \subseteq N[S]$ and tuples of the form $A_1$ or $(A_1, A_2)$ which are subsets of $S_{\leq 1} \setminus S$. Our task is to compute a subset $V'$ of the vertices of $G'$ such that the vertices of $S_{\leq 1} \cup V'$ induce a maximum $S$-forest and the cut $(G[S_{\leq 1}], G[V'])$ satisfies $A_1$ or $(A_1, A_2)$, respectively. We distinguish the two cases with the following two lemmas.

▶ **Lemma 4.** *Let $X \subseteq N[S]$ and let $A_1$ be a subset of $X \setminus S$ such that both $F_{\leq 1} = G[X]$ and $\widehat{G}$ with respect to $X$ and $A_1$ are $S$-forests. There exists a polynomial-time algorithm that computes a maximum $S$-forest $F$ with an $S$-distance partition $(F_{\leq 1}, F_{>1})$ having a cut satisfying $A_1$.*

**Proof.** Since $F_{\leq 1}$ is a fixed $S$-forest of $F$, we need to determine the vertices of $V \setminus (X \cup S)$ that are included in $F_{>1}$. By the desired cut of $(F_{\leq 1}, F_{>1})$, we are restricted to the vertices of $V \setminus (X \cup S)$ that have neighbors in $F_{\leq 1}$ only to $A_1$. Those vertices can be described as follows:

$$B_1 = (V \setminus (X \cup S)) \setminus \{w \in V : N(w) \cap (X \setminus (S \cup A_1)) \neq \varnothing\}.$$

Notice that $B_1$ contains vertices that are at distance at least two from the $S$-vertices of $X \cap S$. Since the cut satisfies a single subset $A_1$, we have at most one connected component of $G[B_1]$ in $F_{>1}$. In order to choose the correct connected component of $G[B_1]$, we try to include each of them in $F_{>1}$ and select the one having the maximum total weight. Notice that adding any component of $G[B_1]$ into $F_{>1}$ cannot create any $S$-cycle, because $\widehat{G}$ with respect to $X$ and $A_1$ is an $S$-forest. Thus, by Lemma 2, we correctly compute a maximum $S$-forest with the desired properties. Clearly the set $B_1$ can be constructed in polynomial time. Since the number of connected components $G[B_1]$ is at most two, all steps can be executed in polynomial time. ◀

▶ **Lemma 5.** *Let $X \subseteq N[S]$ and let $A_1, A_2$ be subsets of $X \setminus S$ such that both $F_{\leq 1} = G[X]$ and $\widehat{G}$ with respect to $X$ and $(A_1, A_2)$ are $S$-forests. There exists a polynomial-time algorithm that computes a maximum $S$-forest $F$ with an $S$-distance partition $(F_{\leq 1}, F_{>1})$ having a cut satisfying $(A_1, A_2)$.*

**Proof.** Similar to the proof of Lemma 4, we first construct the sets $B_1, B_2$ that contain vertices of $V \setminus (X \cup S)$ and satisfy the cut obtained from $X$:

$$B_1 = (V \setminus (X \cup S)) \setminus \{w \in V : N(w) \cap (X \setminus (S \cup A_1)) \neq \varnothing\} \quad \text{and}$$
$$B_2 = (V \setminus (X \cup S)) \setminus \{w \in V : N(w) \cap (X \setminus (S \cup A_2)) \neq \varnothing\}.$$

As the desired cut of $(F_{\leq 1}, F_{>1})$ satisfies $(A_1, A_2)$, there are two connected components of $F_{>1}$ which are subsets of the two sets $B_1$ and $B_2$, respectively. Let $C_1$ and $C_2$ be the connected components of $F_{>1}$ such that $C_1 \subseteq B_1$ and $C_2 \subseteq B_2$. Now observe that there should be two non-adjacent vertices $w_1 \in B_1$ and $w_2 \in B_2$ that belong to $C_1$ and $C_2$, respectively. We iterate over all possible pairs of non-adjacent vertices $w_1 \in B_1 \cap C_1$ and $w_2 \in B_2 \cap C_2$ in $\mathcal{O}(n^2)$ time. Assuming a given choice for $w_1$ and $w_2$, observe the following:

- Since $w_1$ and $w_2$ are vertices of different connected components of $F_{>1}$, the components themselves are further restricted to be subsets of $B_1 \setminus N[w_2]$ and $B_2 \setminus N[w_1]$, respectively. That is, $C_1 \subseteq (B_1 \setminus N[w_2])$ and $C_2 \subseteq B_2 \setminus N[w_1]$.
- Since $F$ has at least one vertex of $S$, $w_1, w_2 \in V \setminus (X \cup S)$ are non-adjacent, and by the fact $d \leq 3$, we have that $B_1 \setminus N[w_2]$ and $B_2 \setminus N[w_1]$ induce cliques in $G$. Thus $B_1 \setminus N[w_2] \subseteq N[w_1]$ and $B_2 \setminus N[w_1] \subseteq N[w_2]$, respectively.

Then by the second statement it is not difficult to see that $B_1 \setminus N[w_2]$ and $B_2 \setminus N[w_1]$ are disjoint. Let $B_1' = (B_1 \setminus N[w_2]) \setminus \{w_1\}$ and $B_2' = (B_2 \setminus N[w_1]) \setminus \{w_2\}$. Now in order to find the maximum induced $S$-forest under the stated conditions and our assumption that $w_1$ and $w_2$ belong to the two connected components of $F_{>1}$, it suffices to find the maximum subset $C_1 \cup C_2$ of $B_1' \cup B_2'$ such that there are no edges between the vertices of $C_1 \cap B_1'$ and the vertices of $C_2 \cap B_2'$. This boils down to compute a minimum weighted vertex cover on the bipartite graph $G'$ obtained from $G[B_1' \cup B_2']$ and removing the edges inside $G[B_1']$ and $G[B_2']$. By maximum flow standard techniques, we compute a minimum weighted vertex cover $U$ on $G'$ in polynomial time [30]. Therefore, $G[B_1' \cup B_2'] - U$ contains the connected components $C_1 \setminus \{w_1\}$ and $C_1 \setminus \{w_2\}$, as required. ◀

Now we are equipped with our necessary tools in order to obtain our main result, namely a polynomial-time algorithm that solves Weighted Subset Feedback Vertex Set on graphs of independent set of size at most 3.

▶ **Theorem 6.** Weighted Subset Feedback Vertex Set *on graphs of independent set of size at most* 3 *can be solved on time* $n^{\mathcal{O}(1)}$.

**Proof.** Let us briefly explain such an algorithm for computing a maximum $S$-forest $F$ of a graph $G$ having independent set size at most three. Let $d = 3$. Initially we set $F^* = G - S$. Then, for every set $X \subseteq N[S]$ with $|X| \leq 4d$ such that $G[X]$ is an $S$-forest, we try by brute force all subsets $A_1$ and $(A_1, A_2)$ with $A_i \subseteq (X \setminus S)$ such that $\widehat{G}$ with respect to $X$ and $A_1$ or $(A_1, A_2)$ is an $S$-forest. For each of such subsets, we find a maximum $S$-forest $F$ with an $S$-distance partition $(G[X], F_{>1})$ having a cut satisfying $A_1$ or $(A_1, A_2)$, respectively, by applying the algorithms described in Lemma 4 and Lemma 5. At each step, we maintain the maximum weighted $S$-forest $F^*$ by comparing $F$ with $F^*$. Finally we provide the vertices $V \setminus V(F^*)$ as the set with the minimum total weight that are removed from $G$.

By Lemma 3, it is sufficient to consider the described subsets $X$. Since every induced subgraph of $G - X$ contains at most two connected components, Lemma 2 implies that all possible subsets $A_1$ or $(A_1, A_2)$ with the described properties are enough to consider. Thus, the correctness follows from Lemmata 3–5. Regarding the running time, notice that whether a graph contains an $S$-cycle can be tested in linear time. Thus we can construct all

described and valid subsets in $n^{\mathcal{O}(1)}$ time. Therefore the total running time of the algorithm is $n^{\mathcal{O}(1)}$, since each of the algorithms given in Lemma 4 and Lemma 5, respectively, requires polynomial time. ◀

Let us now show that extending Theorem 6 to graphs of larger independent sets is not possible. More precisely, with the following result we show that Weighted Subset Feedback Vertex Set is para-NP-complete parameterized by $\alpha(G)$.

▶ **Theorem 7.** Weighted Subset Feedback Vertex Set *is NP-complete on graphs of independent set of size at most* 4.

**Proof.** We will provide a polynomial reduction from the Vertex Cover problem on tripartite graphs which is NP-complete [17] and asks whether a tripartite graph $G$ contains a vertex cover of weight at most $k$. Let $G = (A, B, C, E)$ be a tripartite graph on $n$ vertices, where $(A, B, C)$ is the partition of $V(G)$. We construct a weighted graph $G'$ from $G$ in polynomial time as follows.

- We turn the three independent sets $A$, $B$ and $C$ into cliques by adding all necessary edges and we give all vertices unary weight.
- We add a vertex $r_A$ that is adjacent to all of the vertices of $A$ and we assign weight $n$ to $r_A$. In a completely symmetric way, we add vertices $r_B$ and $r_C$ with respect to the sets $B$ and $C$, respectively.
- We add a vertex $s$ that is adjacent to all three vertices $r_A, r_B, r_C$ having weight $n$.

This completes the construction of $G'$. Observe that all vertices of $V(G') \setminus \{s, r_A, r_B, r_C\}$ have weight equal to one. It is not difficult to verify that the constructed graph $G'$ is a graph having an independent set at most 4, since the vertex set of $G' - \{s\}$ can be partitioned into three cliques.

Next we claim that $G$ has a vertex cover $U$ of weight at most $k < n$ if and only if $G'$ with $S = \{s\}$ has a subset feedback vertex set of weight at most $k$. Assume a vertex cover $U$ of $G$. By definition, $U$ covers all edges of $G$, so that $G[(A \cup B \cup C) \setminus U]$ is an independent set. This means that $G'[(A \cup B \cup C) \setminus U]$ is a vertex-disjoint union of cliques. Since $s$ is non-adjacent to any vertex of $G$ and $G'[r_A, r_B, r_C]$ is an independent set, every cycle of $G' - U$ contains a vertex of $r_A, r_B$ and $r_C$ with at least two vertices from $A, B$ and $C$, respectively. Thus, $G' - U$ is a connected $S$-forest. Therefore, $U$ is a subset feedback vertex set of $(G', \{s\})$ of size at most $k$.

For the opposite direction, assume a subset feedback vertex set $F$ of $(G', \{s\})$. If $F$ is not a subset of $A \cup B \cup C$, then its sum of weights is greater or equal to $n$. Then $F$ is not a minimum subset feedback vertex set of $(G', \{s\})$, since $A \cup B \cup C$ minus a single vertex is trivially a subset feedback vertex set of $(G', \{s\})$ of total weight $n - 1$. Thus $F$ is indeed a subset of $A \cup B \cup C$. Assume that $F$ is not a vertex cover of $G$. By definition, there is an edge of $G$ that remains uncovered. Without loss of generality, assume that this edge has its endpoints on the vertices $x \in A$ and $y \in B$. Then $\langle s, r_A, x, y, r_B \rangle$ is an induced cycle of $G'$, which contradicts the fact that $F$ is a subset feedback vertex set of $(G', \{s\})$. Therefore $F$ is a vertex cover of $G$. ◀

We stress that Theorem 7 further implies that the NP-completeness result carries along to graphs of clique cover number at most four, since the constructed graph given in the proof can be partitioned into four disjoint cliques.

## 4 SFVS on Graphs of Bounded Independent Set

Here we show that despite the complexity dichotomy result for the Weighted Subset Feedback Vertex Set, whenever the weights of the vertices are equal Subset Feedback Vertex Set can be solved in polynomial time on graphs of bounded independent set number.

▶ **Theorem 8.** Subset Feedback Vertex Set *on graphs of independent set of size at most $d$ can be solved in time $n^{\mathcal{O}(d)}$.*

**Proof.** Let $G = (V, E)$ be a graph with $\alpha(G) \leq d$ and let $S \subseteq V$. Denote by $X \subseteq V$ a minimum subset feedback vertex set of $G$. Let $F = G - X$ be a maximum $S$-forest of $G$. By Observation 1 (2), the vertices of $S$ that belong to $F$ are at most $2d$. Thus for every optimum solution $X$, the set $S \setminus X$ has at most $2d$ vertices.

Now we claim that it is enough to try guessing all subsets $X'$ of $X$ for which $|X'| \leq 2d$. To see this, observe that if $X \setminus S$ has order more than $2d$, then $G - S$ has more vertices than $G - X$, leading to a contradiction to the optimality of $X$. Hence, $X \setminus S$ has at most $2d$ vertices. In order to find an optimal solution, it suffices to consider all such candidates $S'$ for $S \setminus X$ and $X'$ for $X \setminus S$. To check whether an induced subgraph of $G$ consists an $S$-forest takes $\mathcal{O}(n + m)$ time. Since the number of such sets $S'$ is at most $n^{2d}$ and the number of the considered sets $X'$ is at most $n^{2d}$, the total running time is bounded by $n^{\mathcal{O}(d)}$. Therefore in time $n^{\mathcal{O}(d)}$ we compute a minimum subset feedback vertex set showing the claimed result. ◀

Regarding the dependence of the exponent in the running time of the algorithm given in Theorem 8, note that we can hardly avoid this fact, since Feedback Vertex Set is W[1]-hard parameterized by the independent set number as explicitly given in [24]. At the same time such an observation follows from the W[1]-hardness result from the construction given in [22] with respect to the maximum induced matching width. In the full version of this extended abstract [32], we provide a different and simpler reduction from the Multicolored Independent Set problem [14, 33] which shows an interesting connection with graphs of bounded independent set size.

## 5 Extending to other Terminal Set Problems

Let us now consider further *terminal set* problems that are related to Subset Feedback Vertex Set. In these type of problems we are given a graph $G = (V, E)$, a terminal set $T \subseteq V$, and a nonnegative integer $k$ and the goal is to find a set $X \subseteq V$ with $|X| \leq k$ which intersects all "structures" (such as cycles or paths) passing through the vertices in $T$ [6]. In this setting, Subset Feedback Vertex Set is a particular terminal set problem when the objective structure is a cycle. We show that the ideas that we developed for Subset Feedback Vertex Set on graphs of bounded independent set size, can be extended to further terminal set problems when the objective structure is a path instead of a cycle.

The (unweighted) Node Multiway Cut problem is concerned with finding a set $X \subseteq V \setminus T$ of size at most $k$ such that any path between two different terminals intersects $X$. Notice that in this problem we are not allowed to remove any terminal. For graphs having bounded independent set size, we completely characterize the complexity of Node Multiway Cut. In particular, for $\alpha(G) = 3$ we can adopt the reduction given in Theorem 7.

▶ **Theorem 9.** *Let $G$ be a graph of independent set of size at most $d$. If $d \leq 2$ then* Node Multiway Cut *can be solved on time $n^{\mathcal{O}(1)}$. Otherwise,* Node Multiway Cut *is NP-complete on graphs of independent set of size at most* 3.

Due to the difficulty of Node Multiway Cut even for the unweighted version and with small independent set number, we consider a relaxed variation in which we are allowed to remove terminal vertices. The Node Multiway Cut with Deletable Terminals problem seeks for a solution $X$ with $X \subseteq V$ (instead of $X \subseteq V \setminus T$). Next we show that the (unweighted) Node Multiway Cut with Deletable Terminals problem can be solved in polynomial time on graphs of bounded independent set number, using an idea similar to the one given in Theorem 8.

▶ **Theorem 10.** Node Multiway Cut with Deletable Terminals *on graphs of independent set of size at most $d$ can be solved in time $n^{\mathcal{O}(d)}$.*

**Proof.** Let $(G, T, k)$ be an instance of Node Multiway Cut with Deletable Terminals where $G$ is a graph having independent set size at most $d$. Observe that every solution $X$ has size at most $|T|$. Assume first that $|T| \leq d$. Then we can enumerate all subsets having at most $|T|$ vertices in time $n^{\mathcal{O}(|T|)}$ and pick the smallest subset that separates all terminals. Thus in time $n^{\mathcal{O}(d)}$ we output a valid solution $X$, if it exists.

Next assume that $d < |T|$. We consider the graph $G[T]$. As an induced subgraph of $G$, $G[T]$ has independent set size at most $d$. Thus, $G[T]$ contains at least one edge. If both endpoints of an edge in $G[T]$ do not belong to solution $X$, then there is a path between terminal vertices. This means that there is a vertex cover $U$ of $G[T]$ such that $U \subseteq X$. To compute such a set $U$, we enumerate all independent sets $T' \subseteq T$ of size at most $d$ in time $|T|^{\mathcal{O}(d)}$ and construct $U = T \setminus T'$. For each constructed $U$, we consider the graph $G' = G - U$ with terminals $T'$. Since $T'$ is an independent set in $G'$, we know that $|T'| \leq d$. Thus in time $n^{\mathcal{O}(|T'|)}$ we can compute a set $X'$ of minimum size such that all terminals of $G' - X'$ are separated. Therefore, the total running time is bounded by $|T|^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(|T'|)}$ which is bounded by $n^{\mathcal{O}(d)}$, because $|T| \leq n$ and $|T'| \leq d$. ◀

Let us also stress that we can hardly avoid the dependence of the exponent in the running time given in Theorem 10. This comes from the fact that Node Multiway Cut with Deletable Terminals with $T = V(G)$ is equivalent to asking whether the graph contains an independent set of size at least $k$. That is, we have to solve the Independent Set which is known to be W[1]-hard parameterized by the size of the independent set [12].

Regarding the node-weighted variant of Node Multiway Cut with Deletable Terminals, we provide a dichotomy result with respect to $\alpha(G)$. In fact, for $\alpha(G) \leq 2$ we can invoke the algorithm for the Weighted Subset Feedback Vertex Set given in Theorem 6, by adding a new vertex with a large weight that is adjacent to all terminals. Moreover, due to its close connection to the Node Multiway Cut, for $\alpha(G) > 2$ we can assign appropriate weights to the terminals in a way that they become undeletable.

▶ **Theorem 11.** *Let $G$ be a graph of independent set of size at most $d$. If $d \leq 2$ then* Weighted Node Multiway Cut with Deletable Terminals *can be solved in time $n^{\mathcal{O}(1)}$. Otherwise,* Weighted Node Multiway Cut with Deletable Terminals *is NP-complete on graphs of independent set of size at most* 3.

## 6 Concluding Remarks

Despite the fact that the Weighted Subset Feedback Vertex Set is NP-complete on graphs with bounded independent set number, it is still interesting to settle the complexity of Subset Feedback Vertex Set on graphs of maximum induced matching width by extending the approach given in [23]. Moreover, Feedback Vertex Set is known to be

polynomially-time solvable on cocomparability graphs [28], and, more generally, on AT-free graphs [26]. To our knowledge, Subset Feedback Vertex Set has not been studied on such graphs, besides the existence of a fast exponential-time algorithm for the unweighted variant of the problem [6]. Furthermore, Theorem 9 shows that Node Multiway Cut remains NP-complete on graphs having maximum induced matching three. However, on graphs of bounded maximum induced matching the complexity of Node Multiway Cut with Deletable Terminals is still unknown.

## References


1. A. Brandstädt, V. B. Le, and J. Spinrad. *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, 1999.
2. B.-M. Bui-Xuan, O. Suchý, J. A. Telle, and M. Vatshelle. Feedback vertex set on graphs of low clique-width. *Eur. Journal of Combinatorics*, 34(3):666–679, 2013.
3. G. Calinescu. Multiway Cut. In *Encyclopedia of Algorithms*. Springer, 2008.
4. J. Chen, Y. Liu, and S. Lu. An improved parameterized algorithm for the minimum node multiway cut problem. *Algorithmica*, 55:1–13, 2009.
5. R. H. Chitnis, F. V. Fomin, D. Lokshtanov, P. Misra, M. S. Ramanujan, and S. Saurabh. Faster exact algorithms for some terminal set problems. In *Proceedings of IPEC 2013*, pages 150–162, 2013.
6. R. H. Chitnis, F. V. Fomin, D. Lokshtanov, P. Misra, M. S. Ramanujan, and S. Saurabh. Faster exact algorithms for some terminal set problems. *Journal of Computer and System Sciences*, 88:195–207, 2017.
7. D. G. Corneil and J. Fonlupt. The complexity of generalized clique covering. *Discrete Applied Mathematics*, 22(2):109–118, 1988.
8. M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. M. M. van Rooij, and J. O. Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *Proceedings of FOCS 2011*, pages 150–159, 2011.
9. M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. On multiway cut parameterized above lower bounds. *ACM Trans. Comput. Theory*, 5(1):3:1–3:11, 2013.
10. M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. Subset feedback vertex set is fixed-parameter tractable. *SIAM J. Discrete Math.*, 27(1):290–309, 2013.
11. R. Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate Texts in Mathematics*. Springer, 2012.
12. R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.
13. G. Even, J. Naor, and L. Zosin. An 8-approximation algorithm for the subset feedback vertex set problem. *SIAM J. Comput.*, 30(4):1231–1252, 2000.
14. M. R. Fellows, D. Hermelin, F. A. Rosamond, and S. Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410(1):53–61, 2009.
15. F. V. Fomin, S. Gaspers, D. Lokshtanov, and S. Saurabh. Exact algorithms via monotone local search. In *Proceedings of STOC 2016*, pages 764–775, 2016.
16. F. V. Fomin, P. Heggernes, D. Kratsch, C. Papadopoulos, and Y. Villanger. Enumerating minimal subset feedback vertex sets. *Algorithmica*, 69(1):216–231, 2014.
17. M. R. Garey and D. S. Johnson. *Computers and Intractability*. W. H. Freeman and Co., 1978.
18. N. Garg, V. V. Vazirani, and M. Yannakakis. Multiway cuts in node weighted graphs. *J. Algorithms*, 50(1):49–61, 2004.
19. P. A. Golovach, P. Heggernes, D. Kratsch, and R. Saei. Subset feedback vertex sets in chordal graphs. *J. Discrete Algorithms*, 26:7–15, 2014.

20 M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Annals of Discrete Mathematics 57, Elsevier, 2004.

21 E. C. Hols and S. Kratsch. A randomized polynomial kernel for subset feedback vertex set. *Theory Comput. Syst.*, 62:54–65, 2018.

22 L. Jaffke, O. Kwon, and J. A. Telle. A note on the complexity of feedback vertex set parameterized by mim-width. *CoRR*, abs/1711.05157, 2017.

23 L. Jaffke, O. Kwon, and J. A. Telle. A unified polynomial-time algorithm for feedback vertex set on graphs of bounded mim-width. In *Proceedings of STACS 2018*, pages 42:1–42:14, 2018.

24 B. Jansen, V. Raman, and M. Vatshelle. Parameter ecology for feedback vertex set. *Tsinghua Sci. and Technol.*, 19(4):387–409, 2014.

25 K. Kawarabayashi and Y. Kobayashi. Fixed-parameter tractability for the subset feedback set problem and the S-cycle packing problem. *J. Comb. Theory, Ser. B*, 102(4):1020–1034, 2012.

26 D. Kratsch, H. Müller, and I. Todinca. Feedback vertex set on AT-free graphs. *Discrete Applied Mathematics*, 156(10):1936–1947, 2008.

27 S. Kratsch and M. Wahlstrom. Representative sets and irrelevant vertices: new tools for kernelization. In *Proceedings of FOCS 2012*, pages 450–459, 2012.

28 Y. D. Liang and M.-S. Chang. Minimum feedback vertex sets in cocomparability graphs and convex bipartite graphs. *Acta Informatica*, 34(5):337–346, 1997.

29 D. Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351:399–406, 2006.

30 J. B. Orlin. Max flows in O(nm) time, or better. In *Proceedings of STOC 2013*, pages 765–774, 2013.

31 C. Papadopoulos and S. Tzimas. Polynomial-time algorithms for the subset feedback vertex set problem on interval graphs and permutation graphs. In *Proceedings of FCT 2017*, pages 381–394, 2017.

32 C. Papadopoulos and S. Tzimas. Subset feedback vertex set on graphs of bounded independent set size. *CoRR*, abs/1805.07141, 2018. `arXiv:1805.07141`.

33 K. Pietrzak. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *J. Comput. Syst. Sci.*, 67(4):757–771, 2003.

34 J. P. Spinrad. *Efficient Graph Representations*. American Mathematical Society, Fields Institute Monograph Series 19, 2003.

35 M. Vatshelle. *New Width Parameters of Graphs*. PhD thesis, University of Bergen, Norway, 2012.

36 M. Yannakakis. Node-deletion problems on bipartite graphs. *SIAM J. Comput.*, 10(2):310–327, 1981.

# Integer Programming in Parameterized Complexity: Three Miniatures

**Tomáš Gavenčiak**
Department of Applied Mathematics, Charles University, Prague, Czech Republic
gavento@kam.mff.cuni.cz

**Dušan Knop**[1]
Department of Informatics, University of Bergen, Bergen, Norway *and*
Department of Theoretical Computer Science, Faculty of Information Technology,
Czech Technical University in Prague, Prague, Czech Republic
dusan.knop@uib.no

**Martin Koutecký**[2]
Technion - Israel Institute of Technology, Haifa, Israel *and*
Computer Science Institute, Charles University, Prague, Czech Republic
koutecky@technion.ac.il

## Abstract

Powerful results from the theory of integer programming have recently led to substantial advances in parameterized complexity. However, our perception is that, except for Lenstra's algorithm for solving integer linear programming in fixed dimension, there is still little understanding in the parameterized complexity community of the strengths and limitations of the available tools. This is understandable: it is often difficult to infer exact runtimes or even the distinction between FPT and XP algorithms, and some knowledge is simply unwritten folklore in a different community. We wish to make a step in remedying this situation.

To that end, we first provide an easy to navigate quick reference guide of integer programming algorithms from the perspective of parameterized complexity. Then, we show their applications in three case studies, obtaining FPT algorithms with runtime $f(k)\,\mathrm{poly}(n)$. We focus on:

- *Modeling*: since the algorithmic results follow by applying existing algorithms to new models, we shift the focus from the complexity result to the modeling result, highlighting common patterns and tricks which are used.
- *Optimality program:* after giving an FPT algorithm, we are interested in reducing the dependence on the parameter; we show which algorithms and tricks are often useful for speed-ups.
- *Minding the* $\mathrm{poly}(n)$*:* reducing $f(k)$ often has the unintended consequence of increasing $\mathrm{poly}(n)$; so we highlight the common trade-offs and show how to get the best of both worlds.

Specifically, we consider graphs of bounded neighborhood diversity which are in a sense the simplest of dense graphs, and we show several FPT algorithms for Capacitated Dominating Set, Sum Coloring, and Max-$q$-Cut by modeling them as convex programs in fixed dimension, $n$-fold integer programs, bounded dual treewidth programs, and indefinite quadratic programs in fixed dimension.

**2012 ACM Subject Classification** Theory of computation → Parameterized complexity and exact algorithms, Theory of computation → Graph algorithms analysis

**Keywords and phrases** graph coloring, parameterized complexity, integer linear programming, integer convex programming

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.21

---

[1] Author supported by the project NFR MULTIVAL.
[2] Author supported by Technion postdoctoral fellowship.

© Tomáš Gavenčiak, Dušan Knop, and Martin Koutecký;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 21; pp. 21:1–21:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

**Related Version** A full version of the paper is available at `https://arxiv.org/abs/1711.02032`.

## 1 Introduction

Our focus is on modeling various problems as INTEGER PROGRAMMING (IP), and then obtaining FPT algorithms by applying known algorithms for IP. IP is the problem

$$\min\{f(\mathbf{x}) \mid \mathbf{x} \in S \cap \mathbb{Z}^n, S \subseteq \mathbb{R}^n \text{ is convex}\} \ . \tag{IP}$$

We give special attention to two restrictions of IP. First, when $S$ is a polyhedron, we get

$$\min\{f(\mathbf{x}) \mid A\mathbf{x} \leq \mathbf{b},\, \mathbf{x} \in \mathbb{Z}^n\}, \tag{LinIP}$$

where $A \in \mathbb{Z}^{m \times n}$ and $\mathbf{b} \in \mathbb{Z}^m$; we call this problem *linearly-constrained IP*, or LINIP. Further restricting $f$ to be a linear function gives INTEGER LINEAR PROGRAMMING (ILP):

$$\min\{\mathbf{w}\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b},\, \mathbf{x} \in \mathbb{Z}^n\}, \tag{ILP}$$

where $\mathbf{w} \in \mathbb{Z}^n$. The function $f\colon \mathbb{Z}^n \to \mathbb{Z}$ is called the *objective function*, $S$ is the *feasible set* (defined by *constraints* or various *oracles*), and $\mathbf{x}$ is a vector of *(decision) variables*. By $\langle\cdot\rangle$ we denote the binary encoding length of numbers, vectors and matrices.

In 1983 Lenstra showed that ILP is polynomial in fixed dimension and solvable in time $n^{\mathcal{O}(n)}\langle A, \mathbf{b}, \mathbf{w}\rangle$ (including later improvements [22, 36, 45]). Two decades later this algorithm's potential for applications in parameterized complexity was recognized, e.g. by Niedermeier [52]:

> *[...] It remains to investigate further examples besides* CLOSEST STRING *where the described ILP approach turns out to be applicable. More generally, it would be interesting to discover more connections between fixed-parameter algorithms and (integer) linear programming.*

This call has been answered in the following years, for example in the context of graph algorithms [19, 20, 24, 44], scheduling [30, 35, 38, 51] or computational social choice [8].

In the meantime, many other powerful algorithms for IP have been devised; however it seemed unclear exactly *how* could these tools be used, as Lokshtanov states in his PhD thesis [46], referring to FPT algorithms for convex IP in fixed dimension:

> *It would be interesting to see if these even more general results can be useful for showing problems fixed parameter tractable.*

Similarly, Downey and Fellows [14] highlight the FPT algorithm for so called $n$-fold IP:

> *Conceivably, [*MINIMUM LINEAR ARRANGEMENT*] might also be approached by the recent (and deep) FPT results of Hemmecke, Onn and Romanchuk [28] concerning nonlinear optimization.*

Interestingly, MINIMUM LINEAR ARRANGEMENT was shown to be FPT by yet another new algorithm for IP due to Lokshtanov [47].

In the last 3 years we have seen a surge of interest in, and an increased understanding of, these IP techniques beyond Lenstra's algorithm, allowing significant advances in fields such as parameterized scheduling [9, 30, 33, 38, 51], computational social choice [39, 40, 42], multichoice optimization [23], and stringology [39]. This has increased our understanding of the strengths and limitations of each tool as well as the modeling patterns and tricks which are typically applicable and used.

## 1.1 Our Results

We start by giving a quick overview of existing techniques in Section 2, which we hope to be an accessible reference guide for parameterized complexity researchers. Then, we resolve the parameterized complexity of three problems when parameterized by the neighborhood diversity of a graph (we defer the definitions to the relevant sections). However, since our complexity results follow by applying an appropriate algorithm for IP, we also highlight our modeling results. Moreover, in the spirit of the optimality program (introduced by Marx [49]), we are not content with obtaining *some* FPT algorithm, but we attempt to decrease the dependence on the parameter $k$ as much as possible. This sometimes has the unintended consequence of increasing the polynomial dependence on the graph size $|G|$. We note this and, by combining several ideas, get the "best of both worlds". Driving down the $\mathrm{poly}(|G|)$ factor is in the spirit of "minding the $\mathrm{poly}(n)$" of Lokshtanov et al. [48].

We denote by $|G|$ the number of vertices of the graph $G$ and by $k$ its neighborhood diversity; graphs of neighborhood diversity $k$ have a succinct representation (constructible in linear time) with $\mathcal{O}(k^2 \log |G|)$ bits and we assume to have such a representation on input.

▶ **Theorem 1.** Capacitated Dominating Set

**(a)** *Has a convex IP model in $\mathcal{O}(k^2)$ variables and can be solved in time and space $k^{\mathcal{O}(k^2)} \log |G|$.*

**(b)** *Has an ILP model in $\mathcal{O}(k^2)$ variables and $\mathcal{O}(|G|)$ constraints, and can be solved in time $k^{\mathcal{O}(k^2)} \mathrm{poly}(|G|)$ and space $\mathrm{poly}(k, |G|)$.*

**(c)** *Can be solved in time $k^{\mathcal{O}(k)} \mathrm{poly}(|G|)$ using model a and a proximity argument.*

**(d)** *Has a polynomial $OPT + k^2$ approximation algorithm by rounding a relaxation of a.*

▶ **Theorem 2.** Sum Coloring

**(a)** *Has an $n$-fold IP model in $\mathcal{O}(k|G|)$ variables and $\mathcal{O}(k^2|G|)$ constraints, and can be solved in time $k^{\mathcal{O}(k^3)}|G|^2 \log^2 |G|$.*

**(b)** *Has a* LinIP *model in $\mathcal{O}(2^k)$ variables and $k$ constraints with a non-separable convex objective, and can be solved in time $2^{2^{k^{\mathcal{O}(1)}}} \log |G|$.*

**(c)** *Has a* LinIP *model in $\mathcal{O}(2^k)$ variables and $\mathcal{O}(2^k)$ constraints whose constraint matrix has dual treewidth $k + 2$ and whose objective is separable convex, and can be solved in time $k^{\mathcal{O}(k^2)} \log |G|$.*

▶ **Theorem 3.** Max-$q$-Cut *has a* LinIP *model with an indefinite quadratic objective and can be solved in time $g(q, k) \log |G|$ for some computable function $g$.*

## 1.2 Related Work

Graphs of neighborhood diversity constitute an important stepping stone in the design of algorithms for dense graphs, because they are in a sense the simplest of dense graphs [2, 3, 6, 20, 24, 25, 50]. Studying the complexity of Capacitated Dominating Set on graphs of bounded neighborhood diversity is especially interesting because it was shown to be W[1]-hard parameterized by treewidth by Dom et al. [13]. Sum Coloring was shown to be FPT parameterized by treewidth [32]; its complexity parameterized by clique-width is open as far as we know. Max-$q$-Cut is FPT parameterized by $q$ and treewidth (by reduction to CSP), but W[1]-hard parameterized by clique-width [21].

## 1.3 Preliminaries

For positive integers $m, n$ with $m \leq n$ we set $[m, n] = \{m, \ldots, n\}$ and $[n] = [1, n]$. We write vectors in boldface (e.g., $\mathbf{x}, \mathbf{y}$) and their entries in normal font (e.g., the $i$-th entry of $\mathbf{x}$ is $x_i$). For a graph $G$ we denote by $V(G)$ its set of vertices, by $E(G)$ the set of its edges, and by $N_G(v) = \{u \in V(G) \mid uv \in E(G)\}$ the (open) neighborhood of a vertex $v \in V(G)$. For a matrix $A$ we define

- the *primal graph* $G_P(A)$, which has a vertex for each column and two vertices are connected if there exists a row such that both columns are non-zero, and,
- the *dual graph* $G_D(A) = G_P(A^\intercal)$, which is the above with rows and columns swapped.

We call the treedepth and treewidth of $G_P(A)$ the *primal treedepth* $\mathrm{td}_P(A)$ and *primal treewidth* $\mathrm{tw}_P(A)$, and analogously for the *dual treedepth* $\mathrm{td}_D(A)$ and *dual treewidth* $\mathrm{tw}_D(A)$.

We define a partial order $\sqsubseteq$ on $\mathbb{R}^n$ as follows: for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ we write $\mathbf{x} \sqsubseteq \mathbf{y}$ and say that $\mathbf{x}$ is *conformal* to $\mathbf{y}$ if $x_i y_i \geq 0$ (that is, $\mathbf{x}$ and $\mathbf{y}$ lie in the same orthant) and $|x_i| \leq |y_i|$ for all $i \in [n]$. It is well known that every subset of $\mathbb{Z}^n$ has finitely many $\sqsubseteq$-minimal elements.

▶ **Definition 4** (Graver basis)**.** The *Graver basis* of $A \in \mathbb{Z}^{m \times n}$ is the finite set $\mathcal{G}(A) \subset \mathbb{Z}^n$ of $\sqsubseteq$-minimal elements in $\{\mathbf{x} \in \mathbb{Z}^n \mid A\mathbf{x} = 0,\ \mathbf{x} \neq \mathbf{0}\}$.

**Neighborhood Diversity.** Two vertices $u, v$ are called *twins* if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. The *twin equivalence* is the relation on vertices of a graph where two vertices are equivalent if and only if they are twins.

▶ **Definition 5** (Lampis [44])**.** The *neighborhood diversity* of a graph $G$, denoted by $\mathrm{nd}(G)$, is the minimum number $k$ of classes (called *types*) of the twin equivalence of $G$.

We denote by $V_i$ the classes of twin equivalence on $G$ for $i \in [k]$. A graph $G$ with $\mathrm{nd}(G) = k$ can be described in a compressed way using only $\mathcal{O}\big(\log |G| \cdot k^2\big)$ space by its type graph, which is computable in linear time [44]:

▶ **Definition 6.** The *type graph* $T(G)$ of a graph $G$ is a graph on $k = \mathrm{nd}(G)$ vertices $[k]$, where each $i$ is assigned weight $|V_i|$, and where $i, j$ is an edge or a loop in $T(G)$ if and only if two distinct vertices of $V_i$ and $V_j$ are adjacent.

**Modeling.** Loosely speaking, by *modeling* an optimization problem $\Pi$ as a different problem $\Lambda$ we mean encoding the features of $\Pi$ by the features of $\Lambda$, such that the optima of $\Lambda$ encode *at least some* optima of $\Pi$. Modeling differs from reduction by highlighting which features of $\Pi$ are captured by which features of $\Lambda$.

In particular, when modeling $\Pi$ as an integer program, the same feature of $\Pi$ can often be encoded in several ways by the variables, constraints or the objective. For example, an objective of $\Pi$ may be encoded as a convex objective of the IP, or as a linear objective which is lower bounded by a convex constraint; similarly a constraint of $\Pi$ may be modeled as a linear constraint of IP or as minimizing a penalty objective function expressing how much is the constraint violated. Such choices greatly influence which algorithms are applicable to solve the resulting model. Specifically, in our models we focus on the parameters #variables (dimension), #constraints, the largest coefficient in the constraints $\|A\|_\infty$ (abusing the notation slightly when the constraints are not linear), the largest right hand side $\|\mathbf{b}\|_\infty$, the largest *domain* $\|\mathbf{u} - \mathbf{l}\|_\infty$, and the largest coefficient of the objective function $\|\mathbf{w}\|_\infty$ (linear objectives), $\|Q\|_\infty$ (quadratic objectives) or $f_{\max} = \max_{\mathbf{x}: \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}} |f(\mathbf{x})|$ (in general), and noting other relevant features.

**Solution structure.** We concur with Downey and Fellows that FPT and structure are essentially one [14]. Here, it typically means restricting our attention to certain structured solutions and showing that nevertheless such structured solutions contain optima of the problem at hand. We always discuss these structural properties before formulating a model.

## 2 Integer Programming Toolbox

We give a list of the most relevant algorithms solving IP, highlighting their fastest known runtimes (marked $\top$), typical use cases and strengths ($+$), limitations ($-$), and a list of references to the algorithms ($\heartsuit$) and their most illustrative applications ($\triangleright$), both in chronological order.

### 2.1 Small Dimension

The following tools generally rely on results from discrete geometry.

**ILP in small dimension.** Problem (ILP) with small $n$.

$\top$ $n^{2.5n}\langle A, \mathbf{b}, \mathbf{w}\rangle$ [36, 22]

$+$ Can use large coefficients, which allows encoding logical connectives using Big-$M$ coefficients [5]. Runs in polynomial space. Most people familiar with ILP.

$-$ Small dimension can be an obstacle in modeling polynomially many "types" of objects [7, Challenge #2]. Models often use exponentially many variables in the parameter, leading to double-exponential runtimes (applies to all small dimension techniques below). Encoding a convex objective or constraint requires many constraints (cf. Model 9). Big-$M$ coefficients are impractical.

$\heartsuit$ [45, 36, 22]

$\triangleright$ [52, 19, 35, 20, 18]

**Convex IP in small dimension.** Problem (IP) with $f$ a convex function; $S$ can be represented by polynomial inequalities, a first-order oracle, a separation oracle, or as a semialgebraic set.

$\top$ $n^{\frac{4}{3}n}\langle B\rangle$, where $S$ is contained in a ball of radius $B$ [12].

$+$ Strictly stronger than ILP. Representing constraints implicitely by an oracle allows better dependence on instance size (cf. Model 8).

$-$ Exponential space. Algorithms usually impractical. Proving convexity can be difficult.

$\heartsuit$ [26, Theorem 6.7.10] (weak separation oracle), [37] (semialgebraic sets), [27, 31] (polynomials), [11] randomized / [12] deterministic (strong separation oracle), [53] reduction to Mixed ILP subproblems (first-order oracle).

$\triangleright$ [30, 8, 51, 38, 41], Model 8

**Indefinite quadratic IP in small dimension.** Problem (LinIP) with $f(\mathbf{x}) = \mathbf{x}^\intercal Q\mathbf{x}$ indefinite (non-convex) quadratic.

$\top$ $g(n, \|A\|_\infty, \|Q\|_\infty)\langle \mathbf{b}\rangle$ [55]

$+$ Currently the only tractable indefinite objective.

$-$ Limiting parameterization.

$\heartsuit$ [47, 55]

$\triangleright$ [47], Model 10

**Parametric ILP in small dimension.** Given a $Q = \{\mathbf{b} \in \mathbb{R}^m \mid B\mathbf{b} \leq \mathbf{d}\}$, decide

$$\forall \mathbf{b} \in Q \cap \mathbb{Z}^m \, \exists \mathbf{x} \in \mathbb{Z}^n : \, A\mathbf{x} \leq \mathbf{b} \ .$$

⊤ $g(n, m)\,\mathrm{poly}(\|A, B, \mathbf{d}\|_\infty)$ [16]
+ Models one quantifier alternation. Useful in expressing game-like constraints (e.g., "∀ moves ∃ a counter-move"). Allows unary big-$M$ coefficients to model logic [42, Theorem 4.5].
− Input has to be given in unary (vs. e.g. Lenstra's algorithm).
♡ [16, Theorem 4.2], [10, Corollary 1]
▷ [10, 42]

## 2.2 Variable Dimension

In this section it will be more natural to consider the following *standard form* of (LinIP)

$$\min\{f(\mathbf{x}) \mid A\mathbf{x} = \mathbf{b},\, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},\, \mathbf{x} \in \mathbb{Z}^n\}, \tag{SLinIP}$$

where $\mathbf{b} \in \mathbb{Z}^m$ and $\mathbf{l}, \mathbf{u} \in \mathbb{Z}^n$. Let $L = \langle f_{\max}, \mathbf{b}, \mathbf{l}, \mathbf{u}\rangle$. In contrast with the previous section, the following algorithms typically rely on algebraic arguments and dynamic programming.

**ILP with few rows.** Problem (SLinIP) with small $m$ and a linear objective $\mathbf{wx}$ for $\mathbf{w} \in \mathbb{Z}^n$.
⊤ $\mathcal{O}((m\|A\|_\infty)^{2m})\langle \mathbf{b}\rangle$ if $\mathbf{l} \equiv \mathbf{0}$ and $\mathbf{u} \equiv +\infty$, and $n \cdot (m\|A\|_\infty)^{\mathcal{O}(m^2)}\langle \mathbf{b}, \mathbf{l}, \mathbf{u}\rangle$ in general [34]
+ Useful for configuration IPs with small coefficients, leading to exponential speed-ups. Best runtime in the case without upper bounds. Linear dependence on $n$.
− Limited modeling power. Requires small coefficients.
♡ [54, 17, 34]
▷ [34]

$$A_{\mathrm{nfold}} = \begin{pmatrix} A_1 & A_1 & \cdots & A_1 \\ A_2 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_2 \end{pmatrix} \qquad A_{\mathrm{stoch}} = \begin{pmatrix} B_1 & B_2 & 0 & \cdots & 0 \\ B_1 & 0 & B_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ B_1 & 0 & 0 & \cdots & B_2 \end{pmatrix}$$

**n-fold IP, tree-fold IP, and dual treedepth.** *$n$-fold IP* is problem (SLinIP) in dimension $nt$, with $A = A_{\mathrm{nfold}}$ for some two blocks $A_1 \in \mathbb{Z}^{r\times t}$ and $A_2 \in \mathbb{Z}^{s\times t}$, $\mathbf{l}, \mathbf{u} \in \mathbb{Z}^{nt}$, $\mathbf{b} \in \mathbb{Z}^{r+ns}$, and with $f$ a separable convex function, i.e., $f(\mathbf{x}) = \sum_{i=1}^n \sum_{j=1}^t f_j^i(x_j^i)$ with each $f_j^i : \mathbb{Z} \to \mathbb{Z}$ convex. *Tree-fold IP* is a generalization of $n$-fold IP where the block $A_2$ is itself replaced by an $n$-fold matrix, and so on, recursively, $\tau$ times. Tree-fold IP has bounded $\mathrm{td}_D(A)$.

⊤ $(\|A\|_\infty rs)^{\mathcal{O}(r^2 s + rs^2)}(nt)^2 \log(nt)\langle L\rangle$ $n$-fold IP [1, 15]; $(\|A\|_\infty + 1)^{2^{\mathrm{td}_D(A)}}(nt)^2 \log(nt)\langle L\rangle$ for (SLinIP) [43].
+ Variable dimension useful in modeling many "types" of objects [40, 42]. Useful for obtaining exponential speed-ups (not only configuration IPs). Seemingly rigid format is in fact not problematic (blocks can be different provided coefficients and dimensions are small).
− Requires small coefficients.
♡ [28, 39, 9, 15, 1, 43]
▷ [38, 40, 39, 9, 33], Model 11

**2-stage and multi-stage stochastic IP, and primal treedepth.** 2-stage stochastic IP is problem (SLinIP) with $A = A_{\text{stoch}}$ and $f$ a separable convex function; multi-stage stochastic IP is problem (SLinIP) with a multi-stage stochastic matrix, which is the transpose of a tree-fold matrix; multi-stage stochastic IP is in turn generalized by IP with small primal treedepth $\text{td}_P(A)$.

⊤ $g(\text{td}_P(A), \|A\|_\infty)n^2 \log n\langle L\rangle$, $g$ computable [43]

\+ Similar to Parametric ILP in fixed dimension, but quantification $\forall \mathbf{b} \in Q \cap \mathbb{Z}^n$ is now over a polynomial sized but possibly non-convex set of explicitely given right hand sides.

− Not clear which problems are captured. Requires small coefficients. Parameter dependence $g$ is possibly non-elementary; no upper bounds on $g$ are known, only computability.

♡ [29, 4, 43]

▷ N/A

**Small treewidth and Graver norms.** Let $g_\infty(A) = \max_{\mathbf{g}\in\mathcal{G}(A)} \|\mathbf{g}\|_\infty$ and $g_1(A) = \max_{\mathbf{g}\in\mathcal{G}(A)} \|\mathbf{g}\|_1$ be maximum norms of elements of $\mathcal{G}(A)$.

⊤ $\min\{g_\infty(A)^{\mathcal{O}(\text{tw}_P(A))}, g_1(A)^{\mathcal{O}(\text{tw}_D(A))}\}n^2 \log n\langle L\rangle$ [43]

\+ Captures IPs beyond the classes defined above (cf. Section 5.3).

− Bounding $g_1(A)$ and $g_\infty(A)$ is often hard or impossible.

♡ [43]

▷ Model 14

## 3 Convex Constraints: Capacitated Dominating Set

| Capacitated Dominating Set | |
|---|---|
| **Input:** | A graph $G = (V, E)$ and a capacity function $c\colon V \to \mathbb{N}$. |
| **Task:** | Find a smallest possible set $D \subseteq V$ and a mapping $\delta\colon V \setminus D \to D$ such that for each $v \in D$, $\lvert\delta^{-1}(v)\rvert \le c(v)$. |

**Solution Structure.** Let $<_c$ be a linear extension of ordering of $V$ by vertex capacities, i.e., $u <_c v$ if $c(u) \le c(v)$. For $i \in T(G)$ and $\ell \in [|V_i|]$ let $V_i[1:\ell]$ be the set of the first $\ell$ vertices of $V_i$ in the ordering $<_c$ and let $f_i(\ell) = \sum_{v\in V_i[2:\ell]} c(v)$; for $\ell > |V_i|$ let $f_i(\ell) = f_i(|V_i|)$. Let $D$ be a solution and $D_i = D \cap V_i$. We call the functions $f_i$ the *domination capacity functions*. Intuitively, $f_i(\ell)$ is the maximum number of vertices dominated by $V_i[1:\ell]$. Observe that since $f_i(\ell)$ is a partial sum of a non-increasing sequence of numbers, it is a piece-wise linear concave function. We say that $D$ is *capacity-ordered* if, for each $i \in T(G)$, $D_i = V_i[1:|D_i|]$. The following observation allows us to restrict our attention to such solutions; the proof goes by a simple exchange argument.

▶ **Lemma 7** (⋆). *There is a capacity-ordered optimal solution.*

Observe that a capacity-ordered solution is fully determined by the sizes $|D_i|$ and $<_c$ rather than the actual sets $D_i$, which allows modeling CDS in small dimension.

---

▶ **Model 8** (Capacitated Dominating Set as convex IP in fixed dimension).
**Variables & notation:**

- $x_i = |D_i|$
- $y_{ij} = |\delta^{-1}(D_i) \cap D_j|$
- $f_i(x_i) =$ maximum #vertices dominated by $D_i$ if $|D_i| = x_i$

**Objective & Constraints:**

$$\min \sum_{i \in T(G)} x_i \qquad \min |D| = \sum_{i \in T(G)} |D_i| \qquad \text{(cds:cds-obj)}$$

$$\sum_{j \in N_{T(G)}(i)} y_{ij} \leq f_i(x_i) \qquad \forall i \in T(G) \qquad \text{respect capacities} \qquad \text{(cds:cap)}$$

$$\sum_{i \in N_{T(G)}(j)} y_{ij} \geq |V_j| - x_j \qquad \forall j \in T(G) \qquad \text{every } v \in V_j \setminus D_j \text{ dominated} \qquad \text{(cds:dom)}$$

$$0 \leq x_i \leq |V_i| \qquad \forall i \in T(G) \qquad \text{(cds:bounds)}$$

**Parameters & Notes:**

| #vars | #constraints | $\|A\|_\infty$ | $\|\mathbf{b}\|_\infty$ | $\|\mathbf{l},\mathbf{u}\|_\infty$ | $\|\mathbf{w}\|_\infty$ |
|---|---|---|---|---|---|
| $\mathcal{O}(k^2)$ | $\mathcal{O}(k)$ | 1 | $\|G\|$ | $\|G\|$ | 1 |

- constraint (cds:cap) is convex, since it bounds the area *under* a concave function, and is piece-wise linear.

---

Then, applying for example Dadush's algorithm [11] to Model 8 yields Theorem 1a. We can trade the non-linearity of the previous model for an increase in the number of constraints and the largest coefficient. That, combined with Lenstra's algorithm, yields Theorem 1b, where we get a larger dependence on $|G|$, but require only $\text{poly}(k, |G|)$ space.

---

▶ **Model 9** (Capacitated Dominating Set as ILP in fixed dimension).
Exactly as Model 8 but replace constraints (cds:cap) with the following equivalent set of $|G|$ linear constraints:

$$\sum_{ij \in E(T(G))} y_{ij} \leq f_i(\ell - 1) + c(v_\ell)(x_i - \ell + 1) \qquad \forall i \in T(G)\, \forall \ell \in [|V_i|] \qquad \text{(cds:cap-lin)}$$

The parameters then become:

| #vars | #constraints | $\|A\|_\infty$ | $\|\mathbf{b}\|_\infty$ | $\|\mathbf{l},\mathbf{u}\|_\infty$ | $\|\mathbf{w}\|_\infty$ |
|---|---|---|---|---|---|
| $\mathcal{O}(k^2)$ | $\mathcal{O}(k + \|G\|)$ | $\|G\|$ | $\|G\|$ | $\|G\|$ | 1 |

---

**[Additive approximation] Proof of Theorem 1d.** Let $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{k+k^2}$ be an optimal solution to the *continuous relaxation* of Model 8, i.e., we relax the requirement that $(\mathbf{x}, \mathbf{y})$ are integral; note that such $(\mathbf{x}, \mathbf{y})$ can be computed in polynomial time using the ellipsoid method [26], or by applying a polynomial LP algorithm to Model 9. We would like to round $(\mathbf{x}, \mathbf{y})$ up to an integral $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ to obtain a feasible integer solution which would be an approximation of an integer optimum. Ideally, we would take $\hat{\mathbf{y}} = \lceil \mathbf{y} \rceil$ and compute $\hat{\mathbf{x}}$ accordingly, i.e., set $\hat{x}_i$ to be smallest possible such that $\sum_{j \in N_{T(G)}(i)} \hat{y}_{ij} \geq f_i(\hat{x}_i)$; note that $\hat{x}_i \leq x_i + k$, since we add at most $k$ neighbors (to be dominated) in neighborhood of $V_i$. However, this might result in a non-feasible solution if, for some $i$, $\hat{x}_i > |V_i|$. In such a case, we solve the relaxation again with an additional constraint $x_i = |V_i|$ and try rounding again, repeating this aforementioned fixing procedure if rounding fails, and so on. After at most $k$ repetitions this rounding results in a feasible integer solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$, in which case we have $\|\hat{\mathbf{x}} - \mathbf{x}\|_1 \leq k^2$ and thus the solution represented by $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ has value at most $OPT + k^2$; the relaxation must eventually become feasible as setting $x_i = |V_i|$ for all $i \in T(G)$ yields a feasible solution. ◀

**[Speed trade-offs] Proof of Theorem 1c.** Notice that on our way to proving Theorem 1d we have shown that Model 8 has *integrality gap* at most $k^2$, i.e., the value of the continuous optimum is at most $k^2$ less than the value of the integer optimum. This implies that an integer optimum $(\mathbf{x}^*, \mathbf{y}^*)$ satisfies, for each $i \in [k]$, $\max\{0, \lfloor x_i - k^2 \rfloor\} \leq x_i^* \leq \min\{|V_i|, x_i + \lceil k^2 \rceil\}$.

We can exploit this to improve Theorem 1a in terms of the parameter dependence at the cost of the dependence on $|G|$. Let us assume that we have a way to test, for a given integer vector $\hat{\mathbf{x}}$, whether it models a capacity-ordered solution, that is, whether there exists a capacitated dominating set with $D_i = V_i[1 : \hat{x}_i]$ for each $i$. Then we can simply go over all possible $(2k^2 + 2)^k$ choices of $\hat{\mathbf{x}}$ and choose the best. So we are left with the task of, given a vector $\hat{\mathbf{x}}$, deciding if it models a capacity-ordered solution.

But this is easy. Let $<_c$ be the assumed order and define $D$ as above. Now, we construct an auxiliary bipartite matching problem, where we put $c(v)$ copies of each vertex from $D$ on one side of the graph, and all vertices of $V \setminus D$ on the other side, and connect a copy of $v \in D$ to $u \in V \setminus D$ if $uv \in E(G)$. Then, $D$ is a capacitated dominating set if and only if all vertices in $V \setminus D$ can be matched. The algorithm is then simply to compute the continuous optimum $\mathbf{x}$, and go over all integer vectors $\hat{\mathbf{x}}$ with $\|\mathbf{x} - \hat{\mathbf{x}}\|_1 \leq k^2$, verifying whether they model a solution and choosing the smallest (best) one. ◀

## 4 Indefinite Quadratics: Max $q$-Cut

| Max-$q$-Cut | |
|---|---|
| **Input:** | A graph $G = (V, E)$. |
| **Task:** | A partition $W_1 \dot\cup \cdots \dot\cup W_q = V$ maximizing the number of edges between distinct $W_\alpha$ and $W_\beta$, i.e., $|\{uv \in E(G) \mid u \in W_\alpha, v \in W_\beta, \alpha \neq \beta\}|$. |

**Solution structure.** As before, it is enough to describe *how many* vertices from type $i \in T(G)$ belong to $W_\alpha$ for $\alpha \in [q]$, and their specific choice does not matter; this gives us a small dimensional encoding of the solutions.

---

▶ **Model 10** (Max-$q$-Cut as LinIP with indefinite quadratic objective).

**Variables & Notation:**

- $x_{i\alpha} = |V_i \cap W_\alpha|$
- $x_{i\alpha} \cdot x_{j\beta} = \#$edges between $V_i \cap W_\alpha$ and $V_j \cap W_\beta$ if $ij \in E(T(G))$.

**Objective & Constraints:**

$$\min \sum_{\substack{\alpha,\beta \in [q]: \\ \alpha \neq \beta}} \sum_{ij \in E(T(G))} x_{i\alpha} \cdot x_{j\beta} \qquad \text{min \#edges across partites} \qquad \text{(mc:obj)}$$

$$\sum_{\alpha \in [q]} x_{i\alpha} = |V_i| \qquad \forall i \in T(G) \quad (V_i \cap W_\alpha)_{\alpha \in [q]} \text{ partitions } V_i \qquad \text{(mc:part)}$$

**Parameters & Notes:**

| #vars | #constraints | $\|A\|_\infty$ | $\|\mathbf{b}\|_\infty$ | $\|\mathbf{l}, \mathbf{u}\|_\infty$ | $\|Q\|_\infty$ |
|---|---|---|---|---|---|
| $kq$ | $k$ | 1 | $\|G\|$ | $\|G\|$ | 1 |

- objective (mc:obj) is indefinite quadratic.

---

Applying Lokshtanov's [47] or Zemmer's [55] algorithm to Model 10 yields Theorem 3. Note that since we do not know anything about the objective except that it is quadratic, we have to make sure that $\|Q\|_\infty$ and $\|A\|_\infty$ are small.

# 5 Convex Objective: Sum Coloring

| Sum Coloring | |
|---|---|
| **Input:** | A graph $G = (V, E)$. |
| **Task:** | A proper coloring $c\colon V \to \mathbb{N}$ minimizing $\sum_{v \in V} c(v)$. |

In the following we first give a single-exponential algorithm for Sum Coloring with a polynomial dependence on $|G|$, then a double-exponential algorithm with a logarithmic dependence on $|G|$, and finally show how to combine the two ideas together to obtain a single-exponential algorithm with a logarithmic dependence on $|G|$.

## 5.1 Sum Coloring via $n$-fold IP

**Structure of Solution.** The following observation was made by Lampis [44] for the Coloring problem, and it holds also for the Sum Coloring problem: every color $C \subseteq V(G)$ intersects each clique type in at most one vertex, and each independent type in either none or all of its vertices. The first follows simply by the fact that it is a clique; the second by the fact that if both colors $\alpha, \beta$ with $\alpha < \beta$ are used for an independent type, then recoloring all vertices of color $\beta$ to be of color $\alpha$ remains a valid coloring and decreases its cost. We call a coloring with this structure an *essential coloring*.

---

▶ **Model 11** (Sum Coloring as $n$-fold IP).

**Variables & Notation:**

- $x_i^\alpha = 1$ if color $\alpha$ intersects $V_i$
- $\alpha \cdot x_i^\alpha =$ cost of color $\alpha$ at a clique type $i$
- $\alpha |V_i| \cdot x_i^\alpha =$ cost of color $\alpha$ at an independent type $V_i$
- $S_{\text{nfold}}(\mathbf{x}) = \sum_{\alpha=1}^{|G|} \left( (\sum_{\text{clique } i \in T(G)} \alpha x_i^\alpha) + (\sum_{\text{indep. } i \in T(G)} \alpha |V_i| x_i^\alpha) \right) =$ total cost of $\mathbf{x}$

**Objective & Constraints:**

$$\min S_{\text{nfold}}(\mathbf{x}) \tag{sc:nf:obj}$$

$$\sum_{\alpha=1}^{|G|} x_i^\alpha = |V_i| \qquad \forall i \in T(G), V_i \text{ is clique} \qquad V_i \text{ is colored} \tag{sc:nf:cliques}$$

$$\sum_{\alpha=1}^{|G|} x_i^\alpha = 1 \qquad \forall i \in T(G), V_i \text{ is independent} \qquad V_i \text{ is colored} \tag{sc:nf:indeps}$$

$$x_i^\alpha + x_j^\alpha \leq 1 \qquad \forall \alpha \in [|G|]\ \forall ij \in E(T(G)) \qquad \mathbf{x}^\alpha \text{ is independent set} \tag{sc:nf:xi-indep}$$

**Parameters & Notes:**

- | #vars | #constraints | $\|A\|_\infty$ | $\|\mathbf{b}\|_\infty$ | $\|\mathbf{l}, \mathbf{u}\|_\infty$ | $\|\mathbf{w}\|_\infty$ | $r$ | $s$ | $t$ |
  |---|---|---|---|---|---|---|---|---|
  | $k\|G\|$ | $k + k^2\|G\|$ | 1 | $\|G\|$ | 1 | $\|G\|$ | $k$ | $k^2$ | $k$ |
- Constraints have an $n$-fold format: (sc:nf:cliques) and (sc:nf:indeps) form the $(A_1 \cdots A_1)$ block and (sc:nf:xi-indep) form the $A_2$ blocks; see parameters $r, s, t$ above.

---

Applying the algorithm of Altmanová et al. [1] to Model 11 yields Theorem 2a. Model 11 is a typical use case of $n$-fold IP: we have a vector of multiplicities $\mathbf{b}$ (modeling $(|V_1|, \ldots, |V_k|)$) and we optimize over its decompositions into independent sets of $T(G)$. A clever objective function models the objective of Sum Coloring.

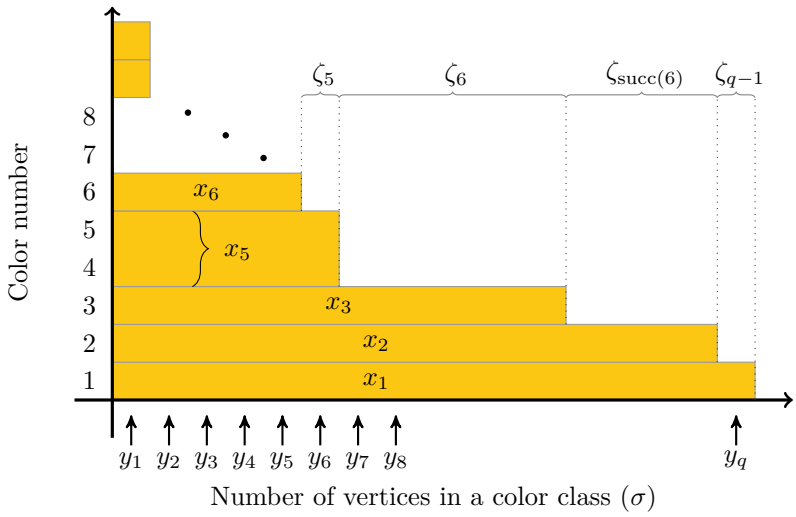



**Figure 1** An illustration of the cost decomposition to the individual classes. Note that $i$-th row (color $i$) has cost $i$ per vertex.

## 5.2 Sum Coloring via Convex Minimization in Fixed Dimension

**Structure of Solution.** The previous observations also allow us to encode a solution in a different way. Let $\mathcal{I} = \{I_1, \ldots, I_K\}$ be the set of all independent sets of $T(G)$; note that $K < 2^k$. Then we can encode an essential coloring of $G$ by a vector of multiplicities $\mathbf{x} = (x_{I_1}, \ldots, x_{I_K})$ of elements of $\mathcal{I}$ such that there are $x_{I_j}$ colors which color exactly the types contained in $I_j$. The difficulty with Sum Coloring lies in the formulation of its objective function. Observe that given an $I \in \mathcal{I}$, the number of vertices every color class of this type will contain is independent of the actual multiplicity $x_I$. Define the *size of a color class* $\sigma\colon \mathcal{I} \to \mathbb{N}$ as $\sigma(I) = \sum_{\text{clique } i \in I} 1 + \sum_{\text{indep. } i \in I} |V_i|$.

▶ **Lemma 12** ($\star$). *Let $G = (V, E)$ be a graph and let $c\colon V \to \mathbb{N}$ be a proper coloring of $G$ minimizing $\sum_{v \in V} c(v)$. Let $\mu(p)$ denote the quantity $|\{v \in V \mid c(v) = p\}|$. Then $\mu(p) \geq \mu(q)$ for every $p \leq q$.*

Our goal now is to show that the objective function can be expressed as a convex function in terms of the variables $\mathbf{x}$. We will get help from auxiliary variables $y_1, \ldots, y_{|G|}$ which are a linear projection of variables $\mathbf{x}$; note that we do not actually introduce these variables into the model and only use them for the sake of proving convexity. Namely, $y_j$ indicates how many color classes contain at least $j$ vertices: $y_j = \sum_{\sigma(I) \geq j} x_I$. Then, the objective function can be expressed as $S_{\text{convex}}(\mathbf{x}) = \sum_{i=1}^{p} |i\sigma(I_i)| = \sum_{j=1}^{|G|} \binom{y_j}{2}$, where $i = 1, \ldots, p$ is the order of the color classes given by Lemma 12, every class of type $I$ is present $x_I$ times, where we enumerate only those $I$ with $x_I \geq 1$. The equivalence of the two is straightforward to check.

Finally, $S_{\text{convex}}$ is convex with respect to $\mathbf{x}$ because,

**1)** all $x_I$ are linear (thus affine) functions,

**2)** $y_i = \sum_{I : \sigma(I) \geq i} x_I$ is a sum of affine functions, thus affine,

**3)** $y_i(y_i - 1)/2$ is convex: it is a basic fact that $h(x) = g(f(x))$ is convex if $f$ is affine and $g$ is convex. Here $f = y_i$ is affine by the previous point and $g = f(f-1)/2$ is convex.

**4)** $S_{\text{convex}}$ is the sum of $y_i(y_i - 1)/2$, which are convex by the previous point.

▶ **Model 13** (Sum Coloring as LinIP in fixed dimension with convex objective).
**Variables & Notation:**
- $x_I$ = #of color class $I$
- $y_i$ = #of color classes $I$ with $\sigma(I) \leq i$
- $\binom{y_i}{2}$ cost of column $y_i$ (Figure 1)
- $S_{\text{convex}} = \sum_{i=1}^{|G|} \binom{y_i}{2}$ = cost of all columns

**Objective & Constraints:**

$$\min S_{\text{convex}}(\mathbf{x}) \qquad \text{(sc:convex:obj)}$$

$$\sum_{I_j : i \in I_j} x_{I_j} = |V_i| \qquad \forall\text{clique } i \in T(G) \qquad \text{clique } V_i \text{ gets } |V_i| \text{ colors} \qquad \text{(sc:convex:cliques)}$$

$$\sum_{I_j : i \in I_j} x_{I_j} = 1 \qquad \forall\text{indep. } i \in T(G) \qquad \text{indep. } V_i \text{ gets 1 color} \qquad \text{(sc:convex:indeps)}$$

**Parameters & Notes:**
- 

| #vars | #constraints | $\|A\|_\infty$ | $\|\mathbf{b}\|_\infty$ | $\|\mathbf{l},\mathbf{u}\|_\infty$ | $f_{\max}$ |
|---|---|---|---|---|---|
| $2^k$ | $k$ | 1 | $\|G\|$ | $\|G\|$ | $\|G\|^2$ |

- Objective $S_{\text{convex}}$ is non-separable convex, and can be computed in time $2^k \log |G|$ by noticing that there are at most $2^k$ different $y_i$'s (see below).

Applying the algorithm of Dadush [11] to Model 13 yields Theorem 2b. Notice that we could not apply Lokshtanov's algorithm because the objective has large coefficients. Also notice that we do not need separability of $S_{\text{convex}}$ or any structure of $A$.

## 5.3 Sum Coloring and Graver Bases

Consider Model 13. The fact that the number of rows and the largest coefficient $\|A\|_\infty$ is small, and that we can formulate $S_{\text{convex}}$ as a separable convex objective in terms of the $y_i$ variables gives us some hope that Graver basis techniques would be applicable.

Since $|\mathcal{I}| \leq 2^k$, we can replace the $y_i$'s by a smaller set of variables $z_i$ for a set of "critical sizes" $\Gamma = \{i \in [|G|] \mid \exists I \in \mathcal{I} : \sigma(I) = i\}$. For each $i \in \Gamma$ let $\text{succ}(i) = \min\{j \in \Gamma \mid j > i\}$ (and let $\text{succ}(\max \Gamma) = \max \Gamma$), define $z_i = \sum_{I \in \mathcal{I} : \sigma(I) \geq i} x_I$, and let $\zeta_i = (\text{succ}(i) - i)$ be the size difference between a color class of size $i$ and the smallest larger color class. Then,

$$S_{\text{convex}}(\mathbf{x}) = \sum_{i=1}^{|G|} \binom{y_i}{2} = \sum_{i \in \Gamma} \zeta_i \binom{z_i}{2} = S_{\text{sepconvex}}(\mathbf{z}) \ .$$

Now we want to construct a system of inequalities of bounded dual treewidth $\text{tw}_D(A)$; however, adding the $z_i$ variables as we have defined them amounts to adding many inequalities containing the $z_1$ variable, thus increasing the dual treewidth to $k + 2^k$. To avoid this, let us define $z_i$ equivalently as $z_i = z_{\text{succ}(i)} + \sum_{\substack{I \in \mathcal{I}: \\ \text{succ}(i) > \sigma(I) \geq i}} x_I = z_{\text{succ}(i)} + \sum_{\substack{I \in \mathcal{I}: \\ \sigma(I) = i}} x_I$.

▶ **Model 14** (Sum Coloring as LinIP with small $\text{tw}_D(A)$ and small $g_1(A)$).
**Variables & Notation:**
- $x_I$ = #of color class $I$
- $z_i$ = #of color classes $I$ with $\sigma(I) \geq i$
- $\zeta_i$ = size difference between $I \in \mathcal{I}$ with $\sigma(I) = i$ and closest larger $J \in \mathcal{I}$
- $\zeta_i \binom{z_i}{2}$ cost of all columns between $y_i$ and $y_{\text{succ}(i)}$ (Figure 1)

- $\Gamma$ = set of critical sizes
- $S_{\text{sepconvex}}(\mathbf{z}) = \sum_{i\in\Gamma} \zeta_i \binom{z_i}{2}$ = total cost

**Objective & Constraints:** constraints (sc:convex:cliques) and (sc:convex:indeps), and:

$$\min S_{\text{sepconvex}}(\mathbf{z}) \tag{sc:graver:obj}$$

$$z_i = z_{\text{succ}(i)} + \sum_{I\in\mathcal{I}:\sigma(I)=i} x_I \qquad \forall i \in \Gamma \tag{sc:graver:sep}$$

**Parameters & Notes:**

| #vars | #constraints | $\|A\|_\infty$ | $\|\mathbf{b}\|_\infty$ | $\|\mathbf{l},\mathbf{u}\|_\infty$ | $f_{\max}$ | $g_1(A)$ | $\text{tw}_D(A)$ |
|---|---|---|---|---|---|---|---|
| $\mathcal{O}(2^k)$ | $\mathcal{O}(2^k)$ | 1 | $\|G\|$ | $\|G\|$ | $\|G\|^2$ | $\mathcal{O}(k^k)$ | $k+2$ |

- Bounds on $g_1(A)$ and $\text{tw}_D(A)$ by Lemmas 16 and 15, respectively.
- Objective $S_{\text{sepconvex}}$ is separable convex. ◀

---

Applying the algorithm of Koutecký et al. [43] to Model 14 yields Theorem 2c.

Let us denote the matrix encoding the constraints (sc:convex:cliques) and (sc:convex:indeps) as $F \in \mathbb{Z}^{k\times 2\cdot 2^k}$ (notice that we also add the empty columns for the $z_i$ variables), and the matrix encoding the constraints (sc:graver:sep) by $L \in \mathbb{Z}^{2^k\times 2\cdot 2^k}$; thus $A = \binom{F}{L}$.

▶ **Lemma 15** ($\star$). *In Model 14 it holds that* $\text{tw}_D(A) \le k+1$.

**Proof Idea.** $G_D(F)$ is a $k$-clique $K_k$, and $G_D(L)$ is a $2^k$-path $P_{2^k}$. Thus, $G_D(A)$ are these two graphs connected by all possible edges, and we construct a path decomposition, whose consecutive nodes contain $G_D(F)$ and consecutive vertices of $G_D(L)$. ◀

▶ **Lemma 16** ($\star$). *In Model 14 it holds that* $g_1(A) \le k^{\mathcal{O}(k)}$.

**Proof Idea.** We first simplify the structure of $L$ by deleting duplicitous columns, and then explicitely construct a decomposition of any $\mathbf{h}$ s.t. $L\mathbf{h} = \mathbf{0}$ into conformal vectors $\mathbf{g}$ of small $\ell_1$-norm. Combining with known bounds on matrices with few rows ($F$) and stacked matrices ($A$) yields the bound. ◀

## References


1. Kateřina Altmanová, Dušan Knop, and Martin Koutecký. Evaluating and Tuning n-fold Integer Programming. In Gianlorenzo D'Angelo, editor, *17th International Symposium on Experimental Algorithms, SEA 2018, June 27-29, 2018, L'Aquila, Italy*, volume 103 of *LIPIcs*, pages 10:1–10:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.SEA.2018.10.
2. Sancrey Rodrigues Alves, Konrad Kazimierz Dabrowski, Luérbio Faria, Sulamita Klein, Ignasi Sau, and Uéverton dos Santos Souza. On the (Parameterized) Complexity of Recognizing Well-Covered (r, l)-graphs. In *10th International Conference on Combinatorial Optimization and Applications (COCOA'16)*, volume 10043 of *Lecture Notes in Computer Science*, pages 423–437, 2016. doi:10.1007/978-3-319-48749-6.
3. NR Aravind, Subrahmanyam Kalyanasundaram, Anjeneya Swami Kare, and Juho Lauri. Algorithms and hardness results for happy coloring problems. *arXiv preprint arXiv:1705.08282*, 2017.
4. Matthias Aschenbrenner and Raymond Hemmecke. Finiteness theorems in stochastic integer programming. *Foundations of Computational Mathematics*, 7(2):183–227, 2007.
5. Johannes Bisschop. *AIMMS Optimization Modeling*. Paragon Decision Technology BV, 2006.

6 Édouard Bonnet and Florian Sikora. The Graph Motif problem parameterized by the structure of the input graph. *Discrete Applied Mathematics*, 2016.

7 Robert Bredereck, Jiehua Chen, Piotr Faliszewski, Jiong Guo, Rolf Niedermeier, and Gerhard J Woeginger. Parameterized algorithmics for computational social choice: Nine research challenges. *Tsinghua Science and Technology*, 19(4):358–373, 2014.

8 Robert Bredereck, Piotr Faliszewski, Rolf Niedermeier, Piotr Skowron, and Nimrod Talmon. Elections with Few Candidates: Prices, Weights, and Covering Problems. In *Proc. ADT 2015*, volume 9346 of *Lecture Notes Comput. Sci.*, pages 414–431, 2015.

9 Lin Chen and Daniel Marx. Covering a tree with rooted subtrees–parameterized and approximation algorithms. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 2801–2820. SIAM, 2018.

10 Jason Crampton, Gregory Gutin, Martin Koutecký, and Rémi Watrigant. Parameterized resiliency problems via integer linear programming. In *Proc. CIAC 2017*, volume 10236 of *Lecture Notes Comput. Sci.*, pages 164–176, 2017.

11 Daniel Dadush, Chris Peikert, and Santosh Vempala. Enumerative Lattice Algorithms in any Norm Via M-ellipsoid Coverings. In Rafail Ostrovsky, editor, *FOCS*, page 580–589. IEEE, 2011. doi:10.1109/FOCS.2011.31.

12 Daniel Dadush and Santosh S Vempala. Near-optimal deterministic algorithms for volume computation via M-ellipsoids. *Proceedings of the National Academy of Sciences*, 110(48):19237–19245, 2013.

13 Michael Dom, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. Capacitated domination and covering: A parameterized perspective. In *International Workshop on Parameterized and Exact Computation*, pages 78–90. Springer, 2008.

14 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

15 Friedrich Eisenbrand, Christoph Hunkenschröder, and Kim-Manuel Klein. Faster Algorithms for Integer Programs with Block Structure. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 49:1–49:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.49.

16 Friedrich Eisenbrand and Gennady Shmonin. Parametric integer programming in fixed dimension. *Math. Oper. Res.*, 33(4), 2008.

17 Friedrich Eisenbrand and Robert Weismantel. Proximity results and faster algorithms for Integer Programming using the Steinitz Lemma. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 808–816. SIAM, 2018.

18 Piotr Faliszewski, Rica Gonen, Martin Koutecký, and Nimrod Talmon. Opinion diffusion and campaigning on society graphs. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 219–225. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi:10.24963/ijcai.2018/30.

19 Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Frances A. Rosamond, and Saket Saurabh. Graph Layout Problems Parameterized by Vertex Cover. In Seok-Hee Hong, Hiroshi Nagamochi, and Takuro Fukunaga, editors, *ISAAC*, volume 5369 of *Lecture Notes in Computer Science*, page 294–305. Springer, 2008. doi:10.1007/978-3-540-92182-0.

20 Jiří Fiala, Tomáš Gavenčiak, Dušan Knop, Martin Kouteckỳ, and Jan Kratochvíl. Parameterized complexity of distance labeling and uniform channel assignment problems. *Discrete Applied Mathematics*, 2017.

21 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Almost Optimal Lower Bounds for Problems Parameterized by Clique-Width. *SIAM J. Comput.*, 43(5):1541–1563, 2014. doi:10.1137/130910932.

22 András Frank and Eva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987. doi:10.1007/BF02579200.

23 Jakub Gajarský, Petr Hliněný, Martin Koutecký, and Shmuel Onn. Parameterized shifted combinatorial optimization. In *International Computing and Combinatorics Conference*, pages 224–236. Springer, 2017.

24 Robert Ganian. Using neighborhood diversity to solve hard problems. *arXiv preprint arXiv:1201.3091*, 2012.

25 Luisa Gargano and Adele A. Rescigno. Complexity of conflict-free colorings of graphs. *Theoretical Computer Science*, 566(??):39–49, February 2015. URL: http://www.sciencedirect.com/science/article/pii/S0304397514009463.

26 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

27 Sebastian Heinz. Complexity of integer quasiconvex polynomial optimization. *J. Complexity*, 21(4):543–556, 2005. doi:10.1016/j.jco.2005.04.004.

28 Raymond Hemmecke, Shmuel Onn, and Lyubov Romanchuk. n-Fold integer programming in cubic time. *Math. Program*, 137(1-2):325–341, 2013. doi:10.1007/s10107-011-0490-y.

29 Raymond Hemmecke and Rüdiger Schultz. Decomposition methods for two-stage stochastic integer programs. In *Online optimization of large scale systems*, pages 601–622. Springer, 2001.

30 Danny Hermelin, Judith-Madeleine Kubitza, Dvir Shabtay, Nimrod Talmon, and Gerhard Woeginger. Scheduling two competing agents when one agent has significantly fewer jobs. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 43. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

31 Robert Hildebrand and Matthias Köppe. A new Lenstra-type algorithm for quasiconvex polynomial integer minimization with complexity $2^{O(n\log n)}$. *Discrete Optimization*, 10(1):69–84, 2013. doi:10.1016/j.disopt.2012.11.003.

32 Klaus Jansen. Complexity results for the optimum cost chromatic partition problem. Manuscript, 1997.

33 Klaus Jansen, Kim-Manuel Klein, Marten Maack, and Malin Rau. Empowering the Configuration-IP – New PTAS Results for Scheduling with Setups Times. *arXiv preprint arXiv:1801.06460*, 2018.

34 Klaus Jansen and Lars Rohwedder. On Integer Programming and Convolution. *arXiv preprint arXiv:1803.04744*, 2018.

35 Klaus Jansen and Roberto Solis-Oba. An OPT+ 1 Algorithm for the Cutting Stock Problem with Constant Number of Object Lengths. In *IPCO*, pages 438–449. Springer, 2010.

36 Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, August 1987.

37 Leonid Khachiyan and Lorant Porkolab. Integer Optimization on Convex Semialgebraic Sets. *Discrete & Computational Geometry*, 23(2):207–224, 2000. doi:10.1007/PL00009496.

38 Dušan Knop and Martin Koutecký. Scheduling meets $n$-fold Integer Programming. *Journal of Scheduling*, November 2017. doi:10.1007/s10951-017-0550-0.

39 Dušan Knop, Martin Koutecký, and Matthias Mnich. Combinatorial n-fold Integer Programming and Applications. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms (ESA 2017)*, volume 87 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 54:1–54:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ESA.2017.54.

40 Dušan Knop, Martin Koutecký, and Matthias Mnich. Voting and Bribing in Single-Exponential Time. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on*

*Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 46:1–46:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

41 Dušan Knop, Tomáš Masařík, and Tomáš Toufar. Parameterized Complexity of Fair Vertex Evaluation Problems. *CoRR*, abs/1803.06878, 2018. `arXiv:1803.06878`.

42 Dušan Knop, Martin Koutecký, and Matthias Mnich. A Unifying Framework for Manipulation Problems. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pages 256–264, 2018. URL: `http://dl.acm.org/citation.cfm?id=3237427`.

43 Martin Koutecký, Asaf Levin, and Shmuel Onn. A Parameterized Strongly Polynomial Algorithm for Block Structured Integer Programs. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 85:1–85:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. `doi:10.4230/LIPIcs.ICALP.2018.85`.

44 Michael Lampis. Algorithmic Meta-theorems for Restrictions of Treewidth. *Algorithmica*, 64(1):19–37, 2012.

45 Hendrik W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

46 Daniel Lokshtanov. *New Methods in Parameterized Algorithms and Complexity*. PhD thesis, University of Bergen, 2009.

47 Daniel Lokshtanov. Parameterized integer quadratic programming: Variables and coefficients. *arXiv preprint arXiv:1511.00310*, 2015.

48 Daniel Lokshtanov, MS Ramanujan, and Saket Saurabh. A linear time parameterized algorithm for directed feedback vertex set. *arXiv preprint arXiv:1609.04347*, 2016.

49 Dániel Marx. What's next? Future directions in parameterized complexity. In *The Multivariate Algorithmic Revolution and Beyond*, pages 469–496. Springer, 2012.

50 Tomáš Masařík and Tomáš Toufar. Parameterized Complexity of Fair Deletion Problems. In *International Conference on Theory and Applications of Models of Computation (TAMC'16)*, pages 628–642, 2017.

51 Matthias Mnich and Andreas Wiese. Scheduling and fixed-parameter tractability. *Math. Program.*, 154(1-2, Ser. B):533–562, 2015.

52 Rolf Niedermeier. Ubiquitous Parameterization - Invitation to Fixed-Parameter Algorithms. In Jirí Fiala, Václav Koubek, and Jan Kratochvíl, editors, *MFCS*, volume 3153 of *Lecture Notes in Computer Science*, page 84–103. Springer, 2004. `doi:10.1007/978-3-540-28629-5_4`.

53 Timm Oertel, Christian Wagner, and Robert Weismantel. Integer convex minimization by mixed integer linear optimization. *Oper. Res. Lett*, 42(6-7):424–428, 2014. `doi:10.1016/j.orl.2014.07.005`.

54 Christos H Papadimitriou. On the complexity of integer programming. *Journal of the ACM (JACM)*, 28(4):765–768, 1981.

55 Kevin Zemmer. *Integer Polynomial Optimization in Fixed Dimension*. PhD thesis, ETH Zurich, 2017.

# Solving Target Set Selection with Bounded Thresholds Faster than $2^n$

**Ivan Bliznets**
St. Petersburg Department of Steklov Institute of Mathematics of the Russian Academy of Sciences, St. Petersburg, Russia
iabliznets@gmail.com
https://orcid.org/0000-0003-2291-2556

**Danil Sagunov**
St. Petersburg Department of Steklov Institute of Mathematics of the Russian Academy of Sciences, St. Petersburg, Russia
danilka.pro@gmail.com
https://orcid.org/0000-0003-3327-9768

## Abstract

In this paper we consider the Target Set Selection problem. The problem naturally arises in many fields like economy, sociology, medicine. In the Target Set Selection problem one is given a graph $G$ with a function $\mathrm{thr} : V(G) \to \mathbb{N} \cup \{0\}$ and integers $k, \ell$. The goal of the problem is to activate at most $k$ vertices initially so that at the end of the activation process there is at least $\ell$ activated vertices. The activation process occurs in the following way: (i) once activated, a vertex stays activated forever; (ii) vertex $v$ becomes activated if at least $\mathrm{thr}(v)$ of its neighbours are activated. The problem and its different special cases were extensively studied from approximation and parameterized points of view. For example, parameterizations by the following parameters were studied: treewidth, feedback vertex set, diameter, size of target set, vertex cover, cluster editing number and others.

Despite the extensive study of the problem it is still unknown whether the problem can be solved in $\mathcal{O}^*\left((2-\epsilon)^n\right)$ time for some $\epsilon > 0$. We partially answer this question by presenting several faster-than-trivial algorithms that work in cases of constant thresholds, constant dual thresholds or when the threshold value of each vertex is bounded by one-third of its degree. Also, we show that the problem parameterized by $\ell$ is W[1]-hard even when all thresholds are constant.

**2012 ACM Subject Classification** Theory of computation → Graph algorithms analysis, Theory of computation → Parameterized complexity and exact algorithms, Theory of computation → Branch-and-bound

**Keywords and phrases** exact exponential algorithms, target set, vertex thresholds, social influence, irreversible conversions of graphs, bootstrap percolation

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.22

**Related Version** Full version permanently available at `https://arxiv.org/abs/1807.10789`.

**Funding** This research was supported by the Russian Science Foundation (project 16-11-10123)

## 1 Introduction

In this paper we consider the Target Set Selection problem. In the problem one is given a graph $G$ with a function $\mathrm{thr} : V(G) \to \mathbb{N} \cup \{0\}$ (a *threshold function*), and two integers $k, \ell$. The question of the problem is to find a vertex subset $S \subseteq V(G)$ (a *target set*) such that $|S| \leq k$ and if we initially activate $S$ then eventually at least $\ell$ vertices of $G$ become activated.

© Ivan Bliznets and Danil Sagunov;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 22; pp. 22:1–22:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The activation process is defined by the following two rules: (i) if a vertex becomes activated it stays activated forever; (ii) vertex $v$ becomes activated if either it was activated initially or at some moment there is at least $\text{thr}(v)$ activated vertices in the set of its neighbours $N(v)$. Often in the literature by Target Set Selection people refer to the special case of Target Set Selection where $\ell = |V(G)|$, i.e. where we need to activate all vertices of the graph. We refer to this special case as Perfect Target Set Selection.

Target Set Selection problem naturally arises in such areas as economy, sociology, medicine. Let us give an example of a scenario [24, 6] under which Target Set Selection may arise in the marketing area. Often people start using some product when they find out that some number of their friends are already using it. Keeping this in mind, it is reasonable to start the following advertisement campaign of a product: give out the product for free to some people; these people start using the product, and then some friends of these people start using the product, then some friends of these friends and so on. For a given limited budget for the campaign we would like to give out the product in a way that eventually we get the most users of the product. Or we may be given the desired number of users of the product and we would like to find out what initial budget is sufficient. It is easy to see that this situation is finely modelled by the Target Set Selection problem.

The fact that Target Set Selection naturally arises in many different fields leads to a situation that the problem and its different special cases were studied under different names: Irreversible $k$-Conversion Set [10, 16], $P_3$-Hull Number [3], $r$-Neighbour Bootstrap Percolation [4], $(k, \ell)$-Influence [5], monotone dynamic monopolies [27], a generalization of Perfect Target Set Selection on the case of oriented graphs is known as Chain Reaction Closure and $t$-Threshold Starting Set [1]. In [10], Centeno et al. showed that Perfect Target Set Selection is NP-hard even when all threshold values are equal to two.

There is an extensive list of results on Target Set Selection from parameterized and approximation point of view. Many different parameterizations were studied in the literature such as size of the target set, treewidth, feedback vertex set, diameter, vertex cover, cluster editing number and others (for more details, see Table 1). Most of these studies consider the Perfect Target Set Selection problem, i.e. the case where $\ell = |V(G)|$. However, FPT membership results for parameters treewidth [6] and cliquewidth [23] were given for the general case of Target Set Selection. From approximation point of view, it is known that the minimization version (minimize the number of vertices in a target set for a fixed $\ell$) of the problem is very hard and cannot be approximated within $\mathcal{O}(2^{\log^{1-\epsilon} n})$ factor for any $\epsilon > 0$, unless $\text{NP} \subseteq \text{DTIME}(n^{polylog(n)})$. This inappoximability result holds even for graphs of constant degree with all thresholds being at most two [11]. Also, the maximization version of the problem (maximize the number of activated vertices for a fixed $k$) is NP-hard to approximate within a factor of $n^{1-\epsilon}$ for any $\epsilon > 0$ [24].

Taking into account many intractability results for the problem, it is natural to ask whether we can beat a trivial brute-force algorithm for this problem or its important subcase Perfect Target Set Selection. In other words, can we construct an algorithm with running time $\mathcal{O}^*\left((2-\epsilon)^n\right)$ for some $\epsilon > 0$. Surprisingly, the answer to this question is still unknown. Note that the questions whether we can beat brute-force naturally arise in computer science and have significant theoretic importance. Probably, the most important such question is SETH hypothesis which informally can be stated as:

▶ **Hypothesis 1** (SETH). *There is no algorithm for SAT with running time $\mathcal{O}^*\left((2-\epsilon)^n\right)$ for any $\epsilon > 0$.*

Another example of such question is the following hypothesis:

**Table 1** Some known results on different parameterizations of Perfect Target Set Selection. In the Thresholds column we indicate restrictions on the threshold function under which the results were obtained. Here $t$ denotes the maximum threshold value.

| Parameter | Thresholds | Result | Reference |
|---|---|---|---|
| Bandwidth $b$ | general | $\mathcal{O}^*\left(b^{\mathcal{O}(b \log b)}\right)$ | Chopin et al. [12] |
| Clique Cover Number $c$ | general | NP-hard for $c = 2$ | Chopin et al. [12] |
| Cliquewidth cw | constant | $\mathcal{O}^*\left((\mathrm{cw} \cdot t)^{\mathcal{O}(\mathrm{cw} \cdot t)}\right)$ | Hartmann [23] |
| Cluster Editing Number $\zeta$ | general | $\mathcal{O}^*\left(16^{\zeta}\right)$ | Nichterlein et al. [26] |
| Diameter $d$ | general | NP-hard for $d = 2$ | Nichterlein et al. [26] |
| Feedback Edge Set Number $f$ | general | $\mathcal{O}^*\left(4^{f}\right)$ | Nichterlein et al. [26] |
| Feedback Vertex Set Number | general | W[1]-hard | Ben-Zwi et al. [6] |
| Neighborhood Diversity nd | majority | $\mathcal{O}^*\left(\mathrm{nd}^{\mathcal{O}(\mathrm{nd})}\right)$ | Dvořák et al. [17] |
| | general | W[1]-hard | Dvořák et al. [17] |
| Target Set Size $k$ | constant | W[P]-complete | Abrahamson et al. [1], Bazgan et al. [5] |
| Treewidth $w$ | constant | $\mathcal{O}^*\left(t^{\mathcal{O}(w \log w)}\right)$ | Ben-Zwi et al. [6] |
| | majority | W[1]-hard | Chopin et al. [12] |
| Vertex Cover Number $\tau$ | general | $\mathcal{O}^*\left(2^{(2^{\tau}+1)\cdot\tau}\right)$ | Nichterlein et al. [26] |

▶ **Hypothesis 2.** *[29] For every hereditary graph class $\Pi$ that can be recognized in polynomial time, the* Maximum Induced $\Pi$-Subgraph *problem can be solved in $\mathcal{O}^*\left((2-\epsilon)^n\right)$ time for some $\epsilon > 0$.*

There is a significant number of papers [9, 28, 19, 18, 14, 30, 22, 13, 15, 7, 20] with the main motivation to present an algorithm faster than the trivial one.

As in the stated hypotheses and mentioned papers, our goal is to come up with an algorithm that works faster than brute-force. We partially answer this question by presenting several $\mathcal{O}^*\left((2-\epsilon)^n\right)$ running time algorithms for Target Set Selection when thresholds, i.e. the values of $\mathrm{thr}(v)$, are bounded by some fixed constant and in case when the values of $\mathrm{thr}(v) - \deg(v)$, so-called *dual thresholds*, are bounded by some fixed constant for every $v \in V(G)$. We think that this result may be interesting mainly because of the following two reasons. Firstly, the result is established for a well-studied problem with many applications and hence can reveal some important combinatorial or algorithmic structure of the problem. Secondly, maybe by resolving the asked question we could make progress in resolving hypotheses 1, 2.

**Our results.** In this paper, we establish the following algorithmic results.

Perfect Target Set Selection can be solved in

- $\mathcal{O}^*\left(1.90345^n\right)$ if for every $v \in V(G)$ we have $\mathrm{thr}(v) \leq 2$;
- $\mathcal{O}^*\left(1.98577^n\right)$ if for every $v \in V(G)$ we have $\mathrm{thr}(v) \leq 3$;
- $\mathcal{O}^*\left((2-\epsilon_d)^n\right)$ randomized time if for every $v \in V(G)$ we have $\mathrm{thr}(v) \geq \deg(v) - d$.

Target Set Selection can be solved in

- $\mathcal{O}^*\left(1.99001^n\right)$ if for every $v \in V(G)$ we have $\mathrm{thr}(v) \leq \lceil \frac{\deg(v)}{3} \rceil$;
- $\mathcal{O}^*\left((2-\epsilon_t)^n\right)$ if for every $v \in V(G)$ we have $\mathrm{thr}(v) \leq t$.

We also prove the following lower bound.

Target Set Selection parameterized by $\ell$ is W[1]-hard even if

- $\mathrm{thr}(v) = 2$ for every $v \in V(G)$.

# 2 Preliminaries

## 2.1 Notation and problem definition

We use standard graph notation. We consider only simple graphs, i.e. undirected graphs without loops and multiple edges. By $V(G)$ we denote the set of vertices of $G$ and by $E(G)$ we denote the set of its edges. We let $n = |V(G)|$. $N(v)$ denotes the set of neighbours of vertex $v \in V(G)$, and $N[v] = N(v) \cup \{v\}$. $\Delta(G) = \max_{v \in V(G)} \deg(v)$ denotes the maximum degree of $G$. By $G[F]$ we denote the subgraph of $G$ induced by a set $F$ of its vertices. Define by $\deg_F(v)$ the degree of $v$ in the subgraph $G[F]$.

By $X_1 \sqcup X_2 \sqcup \ldots \sqcup X_m$ we denote the disjoint union of sets $X_1, X_2, \ldots, X_m$, i.e. $X_1 \sqcup X_2 \sqcup \ldots \sqcup X_m = X_1 \cup X_2 \cup \ldots \cup X_m$ with the additional restriction that $X_i \cap X_j = \emptyset$ for any distinct $i, j$.

For a graph $G$, threshold function thr and $X \subseteq V(G)$ we put $\mathcal{S}_0(X) = X$ and for every $i > 0$ we define $\mathcal{S}_i(X) = \mathcal{S}_{i-1}(X) \cup \{v \in V(G) : |N(v) \cap \mathcal{S}_{i-1}(X)| \geq \mathrm{thr}(v)\}$. We say that *$v$ becomes activated in the $i^{th}$ round*, if $v \in \mathcal{S}_i(X) \setminus \mathcal{S}_{i-1}(X)$, i.e. $v$ is not activated in the $(i-1)^{\text{th}}$ round and is activated in the $i^{\text{th}}$ round. By *activation process* yielded by $X$ we mean the sequence $\mathcal{S}_0(X), \mathcal{S}_1(X), \ldots, \mathcal{S}_i(X), \ldots, \mathcal{S}_n(X)$. Note that $\mathcal{S}_n(X) = \mathcal{S}_{n+1}(X)$ as $\mathcal{S}_i(X) \subseteq \mathcal{S}_{i+1}(X)$ and $n$ rounds is always enough for the activation process to converge. By $\mathcal{S}(X)$ we denote the set of vertices that eventually become activated, and we say that $X$ *activates* $\mathcal{S}(X)$ *in* $(G, \mathrm{thr})$. Thus, $\mathcal{S}(X) = \mathcal{S}_n(X)$.

We recall the definition of Target Set Selection.

Target Set Selection
**Input:** A graph $G$ with thresholds $\mathrm{thr} : V(G) \to \mathbb{N} \cup \{0\}$, integers $k$, $\ell$.
**Question:** Is there a set $X \subseteq V(G)$ such that $|X| \leq k$ and $|\mathcal{S}(X)| \geq \ell$?

We call a solution $X$ of Target Set Selection a *target set of* $(G, \mathrm{thr})$.

By Perfect Target Set Selection we understand a special case of Target Set Selection with $\ell = n$. We call $X$ a *perfect target set of* $(G, \mathrm{thr})$, if it activates all vertices of $G$, i.e. $\mathcal{S}(X) = V(G)$.

Most of the algorithms described in this paper are recursive algorithms that use branching technique. Such algorithms are described by *reduction rules*, that are used to simplify a problem instance, and *branching rules*, that are used to solve an instance by recursively solving smaller instances. If a branching rule branches an instance of size $n$ into $r$ instances of size $n - t_1, n - t_2, \ldots, n - t_r$, we call $(t_1, t_2, \ldots, t_r)$ a *branching vector* of this branching rule. By a *branching factor* of a branching rule we understand a constant $c$ that is a solution of a linear reccurence corresponding to some branching vector of this rule; such constants are used to bound the running time of an algorithm following the rule with $c^n$. Note that a branching rule may have multiple corresponding branching vectors and multiple corresponding branching factors. By the *worst branching factor* of a branching rule (or multiple branching rules, if they are applied within the same algorithm) we understand the largest among its branching factors. We refer to [21] for a more detailed explanation of these aspects.

In our work we also use the following folklore result.

▶ **Lemma 1.** *For any positive integer $n$ and any $\alpha$ such that $0 < \alpha \leq \frac{1}{2}$, we have $\sum_{i=0}^{\lfloor \alpha n \rfloor} \binom{n}{i} \leq 2^{H(\alpha)n}$, where $H(\alpha) = -\alpha \log_2(\alpha) - (1-\alpha)\log_2(1-\alpha)$.*

**Algorithm:** `minimal_pvcs`$(G, F, A, Z)$

**Input:** Graph $G$ with $\Delta(G) < t$, vertex subsets $F, A, Z$ such that $F \sqcup A \sqcup Z = V(G)$.
**Output:** All minimal partial vertex covers $S$ of $G$ such that $S \cap (A \sqcup Z) = A$.

```
if ∃v : N[v] ⊆ F then
    foreach R ⊊ N[v] do
        minimal_pvcs(G, F \ N[v], A ⊔ R, Z ⊔ (N[v] \ R))
else
    foreach R ⊆ F do
        if A ⊔ R is a minimal partial vertex cover of G then
            output A ⊔ R
```

**Figure 1** Algorithm enumerating all minimal partial vertex covers of a graph.

## 2.2 Minimal partial vertex covers

▶ **Definition 2.** Let $G$ be a graph. We call a subset $S \subseteq V(G)$ of its vertices a *$T$-partial vertex cover of $G$* for some $T \subseteq E(G)$, if the set of edges covered by vertices in $S$ is exactly $T$, i.e. $T = \{uv : \{u, v\} \cap S \neq \emptyset, uv \in E(G)\}$.

We call a $T$-partial vertex cover $S$ of $G$ a *minimal partial vertex cover of $G$* if there is no $T$-partial vertex cover $S'$ of $G$ with $S' \subsetneq S$. Equivalently, there is no vertex $v \in S$ so that $S \setminus \{v\}$ is a $T$-partial vertex cover of $G$.

The following theorem bounds the number of minimal partial vertex covers in graphs of bounded degree. We note that somewhat similar results were proven by Björklund et al. [8].

▶ **Theorem 3.** *For any positive integer $t$, there is a constant $\omega_t < 1$ and an algorithm that, given an $n$-vertex graph $G$ with $\Delta(G) < t$ as input, outputs all minimal partial vertex covers of $G$ in $\mathcal{O}^*\left(2^{\omega_t n}\right)$ time.*

**Proof.** We present a recursive branching algorithm that lists all minimal partial vertex covers of $G$. Pseudocode of the algorithm is presented in Figure 1. As input, the algorithm takes three sets $F, A, Z$ such that $F \sqcup A \sqcup Z = V(G)$. The purpose of the algorithm is to enumerate all minimal partial vertex covers that contain $A$ as a subset and do not intersect with $Z$. So the algorithm outputs all minimal partial vertex covers $S$ of $G$ satisfying $S \cap (A \sqcup Z) = A$. It easy to see that then `minimal_pvcs`$(G, V(G), \emptyset, \emptyset)$ enumerates all minimal partial vertex covers of $G$.

The algorithm uses only the following branching rule. If there is a vertex $v \in F$ such that $N(v) \subseteq F$ then consider $2^{|N[v]|} - 1$ branches. In each branch, take some $R \subsetneq N[v]$ and run `minimal_pvcs`$(G, F \setminus N[v], A \sqcup R, Z \sqcup (N[v] \setminus R))$. In other words, we branch on which vertices in $N[v]$ belong to minimal partial vertex cover and which do not. Note that if $S$ is a minimal partial vertex cover then it cannot contain $N[v]$, since otherwise $S \setminus \{v\}$ is its proper subset and covers the same edges. Hence, above branching consider all possible cases. Since $\Delta(G) < t$, the worst branching factor is $(2^t - 1)^{\frac{1}{t}}$.

If the branching rule cannot be applied then we apply brute-force on all possible variants of the intersection of the minimal partial vertex cover $S$ and the set $F$. So we consider all $2^{|F|}$ variants of $S \cap F$, and filter out variants that do not correspond to a minimal partial vertex cover. Minimality of a partial vertex cover can be checked in polynomial time, so filtering out adds only a polynomial factor.

Note that we run brute-force only if every vertex in $F$ has at least one neighbour in $A \sqcup Z$, in other words, $A \sqcup Z$ is a dominating set of $G$. Since $\Delta(G) < t$, any dominating set of $G$ consists of at least $\frac{n}{t}$ vertices. Hence, $|F| \leq \frac{(t-1)n}{t}$. This leads to the following upper bound on the running time of the algorithm:

$$\left(\left(2^t - 1\right)^{\frac{1}{t}}\right)^{\frac{n}{t}} \cdot 2^{\frac{(t-1)n}{t}} \cdot n^{\mathcal{O}(1)}.$$

Hence, we can put $\omega_t = \frac{1}{t^2} \log\left(2^t - 1\right) + \frac{t-1}{t} < 1$. ◀

# 3 Algorithms for bounded thresholds

## 3.1 Algorithm for thresholds bounded by fixed constant

In this subsection we prove the following theorem.

▶ **Theorem 4.** *Let $t$ be a fixed constant. For* Target Set Selection *with all thresholds bounded by $t$ there is a $\mathcal{O}^*\left((2-\epsilon_t)^n\right)$-time algorithm, where $\epsilon_t$ is a positive constant that depends only on $t$.*

Our algorithm consists of three main stages. In the first stage we apply some simple reduction and branching rules. If the instance becomes small enough we then apply brute-force and solve the problem. Otherwise, we move to the second stage of the algorithm. In the second stage we perform branching rules that help us describe the activation process. After that we move to the third stage in which we run special dynamic program that finally solves the problem for each branch. Let us start the description of the algorithm.

### 3.1.1 Stage I

In the first stage our algorithm applies some branching rules. In each branch we maintain the following partition of $V(G)$ into three parts $A, Z, F$. These parts have the following meaning: $A$ is the set of vertices that are known to be in our target set, $Z$ — the set of vertices that are known to be not in the target set, $F$ — the set of all other vertices (i.e. vertices about that we do not know any information so far). At the beginning, we have $A = Z = \emptyset$ and $F = V(G)$.

We start the first stage with exhaustive application of reduction rule 1 and branching rule 1.

▶ **Reduction rule 1.** *If there is any vertex $v \in \mathcal{S}(A)$, but $v \notin A \sqcup Z$, then assign $v$ to $Z$.*

Reduction rule 1 is correct as there is no need to put a vertex in a target set if it will become activated eventually by the influence of its neighbours.

▶ **Branching rule 1.** *If there is a vertex $v \in F$ such that $\deg_F(v) \geq \mathrm{thr}(v)$ then arbitrarily choose a subset $T \subseteq N(v) \cap F$ such that $|T| = \mathrm{thr}(v)$ and branch on the following branches:*

1. *For each subset of vertices $S \subseteq T \cup \{v\}$ of size less than $\mathrm{thr}(v)$ consider a branch in which we put $S$ into $A$ and we put other vertices $T \cup \{v\} \setminus S$ into $Z$;*
2. *Additionally consider the branch in which we assign all vertices from $T$ to $A$ and $v$ is assigned to $Z$.*

It is enough to consider only above-mentioned branches. All other possible branches assign at least $\mathrm{thr}(v)$ vertices from $T \cup \{v\}$ to $A$, and we always can replace such branch with the branch assigning $T$ to $A$, since it leads to the activation of all vertices in $T \cup \{v\}$

and adds at most the same number of vertices into a target set. Branching rule 1 considers $2^{\mathrm{thr}(v)+1} - \mathrm{thr}(v) - 1$ options for $\mathrm{thr}(v) + 1$ vertices, thus it gives the biggest branching factor of $(2^{t+1} - t - 1)^{\frac{1}{t+1}}$ (here and below $t = \max_{v \in V(G)} \mathrm{thr}(v)$).

▶ **Branching rule 2.** *If $|F| \leq \gamma n$, where $\gamma$ is a constant to be chosen later, then simply apply brute-force on how vertices in $F$ should be assigned to $A$ and $Z$.*

If branching rule 2 is applied in all branches then the running time of the whole algorithm is at most $2^{\gamma n}(2^{t+1} - t - 1)^{\frac{(1-\gamma)n}{t+1}}$ and we do not need to use stages II and III, as the problem is already solved in this case.

### 3.1.2 Stage II

After exhaustive application of reduction rule 1 and branching rules 1 and 2, in each branch we either know the answer or we have the following properties:

1. $\Delta(G[F]) < t$;
2. $|F| > \gamma n$;
3. $\mathcal{S}(A) \subseteq A \sqcup Z$.

Now, in order to solve the problem it is left to identify the vertices of a target set that belong to $F$. It is too expensive to consider all $2^{|F|}$ subsets of $F$ as $F$ is too big. Instead of this direct approach (brute-force on all subsets of $F$) we consider several subbranches. In each such branch we almost completely describe the activation process of the graph. For each branch, knowing this information about the activation process, we find an appropriate target set by solving a special dynamic program in stage III.

Let $X$ be an answer (a target set). $X$ can be expressed as $X = A \sqcup B$ where $B \subseteq F$. At the beginning of the activation process only vertices in $\mathcal{S}_0(X) = X = A \sqcup B$ are activated, after the first round vertices in $\mathcal{S}_1(A \sqcup B)$ are activated, and so on. It is clear that $\mathcal{S}(A \sqcup B) = \mathcal{S}_n(A \sqcup B)$. Unfortunately, we cannot compute the sequence of $\mathcal{S}_i(A \sqcup B)$ as we do not know $B$. Instead we compute the sequence $P_0, P_1, \ldots, P_n = P$ such that $P_i \setminus B = \mathcal{S}_i(X) \setminus B$ and $P_i \subseteq P_{i+1}$ for any $i$.

First of all, using Theorem 3 we list all minimal partial vertex covers of the graph $G[F]$. For each minimal partial vertex cover $C$ we create a branch that indicates that $C \subseteq B$ and, moreover, $C$ covers exactly the same edges in $G[F]$ as $B$ does. In other words, any edge in $G[F]$ has at least one endpoint in $B$ if and only if it has at least one endpoint in $C$. Note that such $C$ exists for any $B$. One can obtain $C$ by removing vertices from $B$ one by one while it covers the same edges as $B$. When no vertex can be removed, then, by definition, the remaining vertices form a minimal partial vertex cover.

Put $P_0 = A \sqcup C$. It is correct since $\mathcal{S}_0(X) \setminus B = A = P_0 \setminus B$. We now show how to find $P_{i+1}$ having $P_i$. Recall that to do such transition from $\mathcal{S}_i(X)$ to $\mathcal{S}_{i+1}(X)$ it is enough to find vertices with the number of neighbours in $\mathcal{S}_i(X)$ being at least the threshold value of that vertex. As for $P_i$ and $P_{i+1}$, it is sufficient to check that the number of activated neighbours has reached the threshold only for vertices that are not in $B$. Thus any transition from $P_i$ to $P_{i+1}$ can be done by using a procedure that, given $P_i$ and any vertex $v \notin P_i$, checks whether $v$ becomes activated in the $(i+1)^{\text{th}}$ round or not, under the assumption that $v \notin B$.

Given $P_i$ it is not always possible to find a unique $P_{i+1}$ as we do not know $B$. That is why in such cases we create several subbranches that indicate potential values of $P_{i+1}$.

Let us now show how to, for each vertex $v \notin P_i$, figure out whether $v$ is in $P_{i+1}$ (see pseudocode in Figure 2). Since we know $P_i$ and $P_i \subseteq P_{i+1}$, we assume that $v \notin P_i$.

If $|N(v) \cap P_i| \geq \mathrm{thr}(v)$ then we simply include $v$ in $P_{i+1}$. We claim that this check is enough for $v \in F$.

▶ **Claim 1.** *If $v \in F \setminus B$, then $v$ becomes activated in the $i^{th}$ round if and only if $|N(v) \cap P_i| \geq \mathrm{thr}(v)$.*

**Proof.** We show that by proving that $\mathcal{S}_i(X) \cap N(v) = P_i \cap N(v)$ for every $v \in F \setminus B$. Note that $\mathcal{S}_i(X) \setminus B = P_i \setminus B$ by definition of $P_i$. So it is enough to prove that $\mathcal{S}_i(X) \cap N(v) \cap B = P_i \cap N(v) \cap B$, which is equivalent to $N(v) \cap B = P_i \cap N(v) \cap B$, as $B \subseteq \mathcal{S}_i(X)$. Since $v \notin B$, then any $uv \in E(G[F])$ is covered by $B$ if and only if $u \in B$. $C$ covers the same edges in $G[F]$ as $B$ does, and also $v \notin C$, hence $C \cap N(v) = B \cap N(v)$. Thus, since $C \subseteq P_0 \subseteq P_i$, we get $P_i \cap B \cap N(v) = P_i \cap C \cap N(v) = C \cap N(v) = B \cap N(v)$. ◀

If $v \in B$, the decision for $v$ does not matter. Thus if $v \in F$ and $|N(v) \cap P_i| < \mathrm{thr}(v)$, we may simply not include $v$ in $P_{i+1}$.

If $v \in Z$, at this point, we cannot compute the number of activated neighbours of $v$ exactly as we do not know what neighbours of $v$ are in $B$. Note that we do not need the exact number of such neighbours if we know that this value is at least $\mathrm{thr}(v)$. Thus we branch into $\mathrm{thr}(v) + 1$ subbranches corresponding to the value of $\min\{|N(v) \cap B|, \mathrm{thr}(v)\}$, from now on we denote this value as $dg(v)$.

On the other hand, we know all activated neighbours of $v$ that are in $V(G) \setminus F$ since $\mathcal{S}_i(X) \cap (V(G) \setminus F) = P_i \cap (V(G) \setminus F)$, as $B \subseteq F$. Let this number be $m = |N(v) \cap (P_i \setminus F)|$. So the number of activated neighbours of $v$ is at least $m + dg(v)$. Also there may be some activated neighbours of $v$ in $N(v) \cap P_i \cap F$. However, we cannot simply add $|N(v) \cap P_i \cap F|$ to $m + dg(v)$ since vertices in $P_i \cap B$ will be computed twice. So we are actually interested in the value of $|(N(v) \cap P_i \cap F) \setminus B|$. That is why for vertices from $N(v) \cap P_i \cap F$ we simply branch whether they are in $B$ or not. After that we compare $m + dg(v) + |(N(v) \cap P_i \cap F) \setminus B|$ with $\mathrm{thr}(v)$ and figure out whether $v$ becomes activated in the current round or not.

Note that once we branch on the value of $\min\{|N(v) \cap B|, \mathrm{thr}(v)\}$, or on whether $v \in B$ or not for some $v$, we will not branch on the same value or make a decision for the same vertex again as it makes no sense. Once fixed, the decision should not change along the whole branch and all of its subbranches, otherwise the information about $B$ would just become inconsistent.

Let us now bound the number of branches created. There are three types of branchings in the second stage:

1. Branching on the value of the minimal partial vertex cover $C$. By Theorem 3, there is at most $\mathcal{O}^*\left(2^{\omega_t |F|}\right)$ such branches.
2. Branching on the value of $dg(v) = \min\{|N(v) \cap B|, \mathrm{thr}(v)\}$ with $v \in Z$. There is at most $(t+1)^{|Z|}$ such possibilities since $t \geq \min\{|N(v) \cap B|, \mathrm{thr}(v)\} \geq 0$.
3. Branching on whether vertex $u$ is in $B$ or not. We perform this branching only for vertices in the set $N(v) \cap P_i \cap F$ with $v \in Z$ only when its size is strictly smaller than $\mathrm{thr}(v) \leq t$. Hence we perform a branching of this type on at most $(t-1)|Z|$ vertices.

Hence, the total number of the branches created in stage II is at most

$$2^{\omega_t |F|} \cdot (t+1)^{|Z|} \cdot 2^{(t-1)|Z|} \cdot n^{\mathcal{O}(1)}.$$

### 3.1.3 Stage III

Now, for each branch our goal is to find the smallest set $X$ which activates at least $\ell$ vertices and agrees with all information obtained during branching in a particular branch. That is,

- $A \subseteq X, Z \cap X = \emptyset$ (branchings made in stage I);
- $C \subseteq X$ (branching of the first type in stage II);

**Algorithm:** `is_activated`$(G, \mathrm{thr}, A, Z, F, P_i, v)$

**Input:** $G, \mathrm{thr}, A, Z, F$ as usual, $P_i$ such that $P_i \setminus B = \mathcal{S}_i(A \sqcup B) \setminus B$ for some $B$, and a vertex $v \notin P_i$.

**Output:** True, if $v \notin B$ and $v \in \mathcal{S}_{i+1}(A \sqcup B)$;
False, if $v \notin B$ and $v \notin \mathcal{S}_{i+1}(A \sqcup B)$;
any answer, otherwise.

```
if |N(v) ∩ P_i| ≥ thr(v) then
    return True
else if v ∈ F then
    return False
m ⟵ |N(v) ∩ (P_i \ F)|
branch on the value of dg(v) = min {|N(v) ∩ B|, thr(v)}
m ⟵ m + dg(v)
foreach u ∈ P_i ∩ N(v) ∩ F do
    branch on whether u ∈ B
    if u ∉ B then
        m ⟵ m + 1
return m ≥ thr(v)
```

**Figure 2** Procedure determining whether a vertex becomes activated in the current round.

- information about $\min\{|N(v) \cap B|, \mathrm{thr}(v)\}$ (second type branchings in stage II);
- additional information whether certain vertices belong to $X$ or not (third type branchings in stage II).

From now on we assume that we are considering some particular branching leaf. Let $A'$ be the set of vertices that are known to be in $X$ for a given branch and $Z'$ be the set of vertices known to be not in $X$ (note that $A \subseteq A'$ and $Z \subseteq Z'$). Let $Z = \{v_1, v_2, \ldots, v_z\}$ and $F' = V(G) \setminus A' \setminus Z' = \{u_1, u_2, \ldots, u_{f'}\}$. So actually it is left to find $B' \subseteq F'$ (in these new terms, $B = (A' \setminus A) \sqcup B'$) such that $|A' \sqcup B'| \leq k$, $|P \cup A' \cup B'| \geq \ell$ and for each $i \in \{1, 2, \ldots, z\}$ the value $\min\{\mathrm{thr}(v_i), |N(v_i) \cap B|\}$ equals $dg(v_i)$. This is true since the information obtained during branching completely determines the value of $P$.

In order to solve the obtained problem we employ dynamic programming. We create a table $TS$ of size $f' \times \ell \times (t+1)^z$. For all $B'_1$ such that $|(B'_1 \cup P) \cap \{u_1, u_2, \ldots, u_i\}| = p$ and $\min\{\mathrm{thr}(v_j), |N(v_j) \cap ((A' \setminus A) \sqcup B'_1)|\} = d_j$, in the field $TS(i, p, d_1, d_2, \ldots, d_z)$ we store any set $B'_2$ of minimum size such that $A' \sqcup B'_1 \sqcup B'_2$ is a potential solution, i.e. $|\mathcal{S}(A' \sqcup B'_1 \sqcup B'_2)| = |(P \cup B'_1 \cup B'_2)| = |P \cap (V(G) \setminus F')| + p + |B'_2| \geq \ell$ and for every $j$ we have $\min\{\mathrm{thr}(v_j), |N(v_j) \cap ((A' \setminus A) \sqcup B'_1 \sqcup B'_2)|\} = \min\{\mathrm{thr}(v_j), |N(v_j) \cap B'_2| + d_j\} = dg(v_j)$. Note that the choice of $B'_2$ depends only on values $i, p, d_1, d_2, \ldots, d_z$, but not on the value of $B'_1$ directly. In other words, $TS(i, p, d_1, d_2, \ldots, d_z)$ stores one of optimal ways of how the remaining $f' - i$ vertices in $F'$ should be chosen into $B'$ if the first $i$ vertices in $F'$ was chosen correspondingly to the values of $p$ and $d_j$.

Note that for some fields in the $TS$ table there may be no appropriate value of $B'_2$ (there is no appropriate solution). In such cases, we put the corresponding element to be equal to $V(G)$. It is a legitimate operation since we are solving a minimization problem. Note that

the desired value of $B'$ will be stored as

$$TS(0, 0, \min\{|N(v_1) \cap (A' \setminus A)|, \mathrm{thr}(v_1)\}, \ldots, \min\{|N(v_z) \cap (A' \setminus A)|, \mathrm{thr}(v_z)\}).$$

We assign $TS(f', p, dg(v_1), dg(v_2), \ldots dg(v_z)) = \emptyset$ for every $p$ such that $p + |P \cap (V(G) \setminus F')| \geq \ell$. We do this since values $p, dg(v_1), dg(v_2), \ldots dg(v_z)$ indicate that $A' \sqcup B'_1$ is already a solution. In all other fields of type $TS(f', \cdot, \cdots, \cdot)$ we put the value of $V(G)$. We now show how to evaluate values $TS(i, p, d_1, d_2, \ldots, d_z)$ for any $i \geq 0$ smaller than $f'$. We can evaluate any $TS(i, \cdot, \cdot, \ldots, \cdot)$ in polynomial time if we have all values $TS(i+1, \cdot, \cdot, \ldots, \cdot)$ evaluated. For each $j \in \{1, 2, \ldots, z\}$, let $d_j^{i+1} = \min\{\mathrm{thr}(v_j), d_j + |N(v_j) \cap \{u_{i+1}\}|\}$. In order to compute $TS(i, p, d_1, d_2, \ldots, d_z)$, we need to decide whether $u_{i+1}$ is in a target set or not. If $u_{i+1}$ is taken into $B'$ then $d_j$ becomes equal to $d_j^{i+1}$ for each $j$, if it is not, none of $d_j$ should change. Hence, $TS(i, p, \langle d_j \rangle) = \min\left[TS(i+1, p+1, \langle d_j^{i+1} \rangle) \cup \{u_{i+1}\},\ TS(i+1, p + |P \cap \{u_{i+1}\}|, \langle d_j \rangle)\right]$.

Since $0 \leq d_j \leq dg(v_j)$ for any $j$, the $TS$ table has $\mathcal{O}^*\left((t+1)^{|Z|}\right)$ fields. Each field of the table is evaluated in polynomial time. So the desired $B'$ is found (hence, the solution is found) in $\mathcal{O}^*\left((t+1)^{|Z|}\right)$ time for any branch fixed in stage II. Stages II and III together run in $2^{\omega_t |F|} \cdot (t+1)^{|Z|} \cdot 2^{(t-1)|Z|} \cdot (t+1)^{|Z|} \cdot n^{\mathcal{O}(1)}$ time for any fixed subbranch of stage I.

Actually, the $(t+1)^{2|Z|}$ multiplier in the upper bound can be improved. Recall that it corresponds to the number of possible variants of $dg(v_j)$ and the number of possible variants of $d_j$. However, note that $d_j \leq dg(v_j)$. So after each of $dg(v_j)$ is fixed in stage II, for $d_j$ there is only $dg(v_j) + 1$ options in stage III. Hence, each of the pairs $(d_j, dg(v_j))$ can be presented only in $\binom{t+2}{2}$ variants. This gives an improvement of the $(t+1)^{2|Z|}$ multiplier to a $\binom{t+2}{2}^{|Z|}$ multiplier. So, the upper bound on the running time in stages II and III becomes $\mathcal{O}^*\left(2^{\omega_t |F|} \cdot \binom{t+2}{2}^{|Z|} \cdot 2^{(t-1)|Z|}\right)$.

We rewrite this upper bound in terms of $n$ and $|F|$. Since $|Z| \leq n - |F|$, the upper bound is

$$2^{\omega_t |F|} \cdot \binom{t+2}{2}^{n-|F|} \cdot 2^{(t-1)(n-|F|)} \cdot n^{\mathcal{O}(1)}.$$

Now we are ready to choose $\gamma$. We set the value of $\gamma$ so that computation in each branch created at the end of stage I takes at most $\mathcal{O}^*(2^{\gamma n})$ time. Note that the upper bound on the running time required for stages II and III increases while the value of $|F|$ decreases. So we can find $\gamma$ as the solution of equation $2^{\gamma n} = 2^{\omega_t \gamma n} \cdot \binom{t+2}{2}^{(1-\gamma)n} \cdot 2^{(t-1)(1-\gamma)n}$. Hence, $\gamma = \frac{(t-1)+\log_2\binom{t+2}{2}}{(t-\omega_t)+\log_2\binom{t+2}{2}} < 1$, as $\omega_t < 1$. So the overall running time is

$$2^{\gamma n}(2^{t+1} - t - 1)^{\frac{(1-\gamma)n}{t+1}} \cdot n^{\mathcal{O}(1)},$$

which is $\mathcal{O}^*\left((2 - \epsilon_t)^n\right)$ for some $\epsilon_t > 0$ since $\gamma < 1$.

## 3.2 Two algorithms for constant thresholds in the perfect case

Here, we present two algorithms for special cases of Perfect Target Set Selection with thresholds being at most two or three. These algorithms use the idea that cannot be used in the general case of Target Set Selection, so the running times of these algorithms are significantly faster than the running time of the algorithm from the previous subsection. We provide their full descriptions in the full version of our paper.

▶ **Theorem 5.** Perfect Target Set Selection *with thresholds being at most two can be solved in* $\mathcal{O}^*(1.90345^n)$ *time.*

▶ **Theorem 6.** Perfect Target Set Selection *with thresholds being at most three can be solved in* $\mathcal{O}^*(1.98577^n)$ *time.*

### 3.3 Algorithm for thresholds bounded by one-third of degrees

Here, we prove the following.

▶ **Theorem 7.** *Let $G$ be a connected graph with at least three vertices. Assume that* $\mathrm{thr}(v) \leq \lceil \frac{\deg(v)}{3} \rceil$ *for every $v \in V(G)$. Then there is a perfect target set of $(G, \mathrm{thr})$ of size at most* $0.45|V(G)|$.

▶ **Corollary 8.** Target Set Selection *with thresholds bounded by one-third of degree rounded up can be solved in* $\mathcal{O}^*(1.99001^n)$ *time.*

In our proofs we use a combinatorial model proposed by Ackerman et al. in [2]. We provide these proofs in the full version of our paper.

## 4 Algorithm for bounded dual thresholds

Let $(G, \mathrm{thr})$ be a graph with thresholds. By *dual threshold* of vertex $v \in V(G)$ we understand the value $\overline{\mathrm{thr}}(v) = \deg(v) - \mathrm{thr}(v)$. In terms of dual thresholds, $v$ becomes activated if it has at most $\overline{\mathrm{thr}}(v)$ not activated neighbours. For bounded dual thresholds we prove the following theorem.

▶ **Theorem 9.** *For any non-negative integer $d$,* Perfect Target Set Selection *with dual thresholds bounded by $d$ can be solved in* $\mathcal{O}^*((2-\epsilon_d)^n)$ *randomized time for some* $\epsilon_d > 0$.

This result follows from the result of Pilipczuk and Pilipczuk in [28], where they presented an algorithm for the Maximum $d$-Degenerate Induced Subgraph problem with the same running time. One may find the detailed proof in the full version of our paper.

## 5 Lower bounds

### 5.1 ETH lower bound

First of all, we show a $2^{o(n+m)}$ lower bound for Perfect Target Set Selection, where $m$ denotes the number of edges in the input graph. We have not found any source that claims this result. Thus, for completeness, we state it here. The result follows from the reduction given by Centeno et al. in [10]. They showed a linear reduction from a special case of 3-SAT, where each variable appears at most three times, to Perfect Target Set Selection where thresholds are equal to two and maximum degree of the graph is constant. Note that in their work they refer to the problem as $\mathrm{IRR}_2$-Conversion Set.

▶ **Theorem 10.** Perfect Target Set Selection *cannot be solved in* $2^{o(n+m)}$ *time unless ETH fails, even when thresolds are equal to two and maximum degree of the graph is constant.*

One can find the detailed proof in the full version of our paper.

### 5.2 Parameterization by $\ell$

We now look at Target Set Selection from parameterized point of view. In [5], Bazgan et al. proved that Target Set Selection $\ell$ is W[1]-hard with respect to parameter $\ell$, when all dual thresholds are equal to 0. This result also follows from the proof of W[1]-hardness

of Cutting $\ell$ Vertices given by Marx in [25], with a somewhat different construction. Inspired by his proof, we show that this result holds even when all thresholds are constant.

▶ **Theorem 11.** Target Set Selection *parameterized by $\ell$ is W[1]-hard even when all thresholds are equal to two.*

**Proof.** Let $(G, k)$ be an instance of the Clique problem. In order to provide the reduction, we construct a graph $G'$ in which each vertex corresponds to a vertex or an edge of graph $G$ i.e. $V(G') = V(G) \sqcup E(G)$. We add edges in $G'$ between vertices corresponding to $v \in V(G)$ and $e \in E(G)$ if and only if $v$ and $e$ are incident in $G$.

We will refer to the vertex in $G'$ corresponding to an edge $e \in E(G)$ as $v_e \in V(G')$. If a vertex from $G'$ corresponds to a vertex $u \in G$ we refer to it as $v_u$. Slightly abusing notation we will refer to the set of vertices in $G'$ corresponding to the vertices $V(G)$ as $V$ and to the set of vertices corresponding to the edges $E(G)$ as $E$, $V \sqcup E = V(G')$. Consider now an instance of Target Set Selection for $G'$, with the same $k$, $\ell = k + \binom{k}{2}$ and all thresholds equal to $t = 2$.

If $G$ has a clique of size $k$, then selecting corresponding vertices as a target set of $G'$ leads to activation of the vertices corresponding to the edges of the clique. Hence, $k + \binom{k}{2}$ vertices will be activated in total.

Let us now prove that if $G'$ has a target set of size at most $k$ activating at least $\ell = k + \binom{k}{2}$ vertices, then $G$ has a clique on $k$ vertices. Let $S$ be such target set of $G'$. Denote by $k_v = |S \cap V|$ the number of vertices in $S$ corresponding to the vertices of $G$ and by $k_e = |S \cap E|$ the number of vertices in $S$ corresponding to the edges of $G$, $k_v + k_e \leq k$.

Now, we show how to convert any target set $S$ of size at most $k$ activating at least $k + \binom{k}{2}$ vertices into a target set $S'$ such that $|S'| \leq k$, $S' \subseteq V$ and $S'$ activates at least $k + \binom{k}{2}$ vertices.

Observe that if there is an edge $u_1u_2 = e \in E(G)$ such that $v_e \in S$ and $v_{u_1} \in S$ then $S' = S \setminus \{v_e\} \cup \{v_{u_2}\}$ also activates at least $k + \binom{k}{2}$ vertices and the size of $S'$ is at most $k$. Thus we can assume that if $v_{u_1u_2} \in S$, then $v_{u_1}, v_{u_2} \notin S$.

Observe that any initially not activated vertex in $E$ becomes activated only if all two of its neighbours are activated. It means that any such vertex does not influence the activation process in future. Hence, since $G'$ is bipartite, the activation process always finishes within two rounds, and no vertex in $V$ becomes activated in the second round.

Let $V_1$ be the set of vertices of $V$ that become activated by $S$ in the first round, i.e. $V_1 = \mathcal{S}_1(S) \setminus \mathcal{S}_0(S) \cap V$. Note that these vertices are activated directly by $k_e$ vertices in $S \cap E$. Let $S_{E,i}$ be the set of vertices in $S \cap E$ that have exactly $i$ endpoints in $V_1$. Denote by $k_{e,i}$ the size of $S_{E,i}$. Then we have $k_{e,0} + k_{e,1} + k_{e,2} = k_e$. Note that if there is a vertex in $S \cap E$ with no endpoints in $V_1$ then one can replace it with any neighbour and size of $S$ will not change and it will activate at least the same number of vertices in $G'$. Thus we can assume that $k_{e,0} = 0$.

We show that $|V_1| \leq \frac{k_{e,1}}{2} + k_{e,2}$. Indeed, in order to be activated, any vertex from $V_1$ requires at least two vertices from $E$ to be in the target set. Each vertex from $S_{E,i}$ contributes to exactly $i$ vertices from $V_1$, and the total number of contributions is $k_{e,1} + 2k_{e,2}$. This number should be at least $2|V_1|$. Hence, $|V_1| \leq \frac{k_{e,1}}{2} + k_{e,2}$.

Consider $S' = S \setminus E \cup V_1$ i.e. we replace all $k_e$ vertices from $E$ with all vertices from $V_1$. Note that $|S'| \leq |S| - \frac{k_{e,1}}{2}$. Vertices from $S_{E,2}$ become activated in the first round since all of them have two endpoints in $S'$. Thus $S'$ is now a target set of size not greater than $k - \frac{k_{e,1}}{2}$ activating at least $\ell - k_{e,1}$ vertices in $G'$.

Note that any vertex from $S_{E,1}$ can be activated by adding one more vertex to $S'$. Consider set $H = N(S_{E,1}) \setminus V_1$. If $|H| \leq \frac{k_{e,1}}{2}$ then consider $S_1 = H \cup S'$. $S_1$ compared to $S'$

will additionally activate all vertices in $S_{E,1}$. Note that $S_1$ is a target set $S$ of size at most $k$ activating at least $\ell$ vertices.

If $|H| > \frac{k_{e,1}}{2}$ then construct $S_1$ from $S'$ by simply adding $\frac{k_{e,1}}{2}$ arbitrary vertices from $H$. Each of these vertices will additionally activate at least one vertex corresponding to edge, thus $S_1$ is a target set of size at most $k$ activating at least $\ell$ vertices.

We have shown how to transform any target set $S$ activating at least $k + \binom{k}{2}$ vertices in $G'$ into a target set $S_1$ such that $S_1 \subseteq V$ and $S_1$ activates at least the same number of vertices in $G'$. As we have shown earlier, no vertex in $E \setminus S_1$ influence the activation process after becoming activated. Then, since $S_1 \cap E = \emptyset$, $S_1$ activates only vertices in $E$ in the first round and the process finishes. Hence, if the instance for $G'$ has a solution, then $G$ has a clique of size $k$. ◀

## References


1. Karl A. Abrahamson, Rodney G. Downey, and Michael R. Fellows. Fixed-parameter tractability and completeness IV: On completeness for W[P] and PSPACE analogues. *Annals of Pure and Applied Logic*, 73(3):235–276, June 1995. doi:10.1016/0168-0072(94)00034-z.
2. Eyal Ackerman, Oren Ben-Zwi, and Guy Wolfovitz. Combinatorial model and bounds for target set selection. *Theoretical Computer Science*, 411(44-46):4017–4022, October 2010. doi:10.1016/j.tcs.2010.08.021.
3. R.T. Araújo, R.M. Sampaio, and J.L. Szwarcfiter. The convexity of induced paths of order three. *Electronic Notes in Discrete Mathematics*, 44:109–114, November 2013. doi:10.1016/j.endm.2013.10.017.
4. József Balogh, Béla Bollobás, and Robert Morris. Bootstrap percolation in high dimensions. *Combinatorics, Probability and Computing*, 19(5-6):643–692, 2010.
5. Cristina Bazgan, Morgan Chopin, André Nichterlein, and Florian Sikora. Parameterized approximability of maximizing the spread of influence in networks. *Journal of Discrete Algorithms*, 27:54–65, July 2014. doi:10.1016/j.jda.2014.05.001.
6. Oren Ben-Zwi, Danny Hermelin, Daniel Lokshtanov, and Ilan Newman. Treewidth governs the complexity of target set selection. *Discrete Optimization*, 8(1):87–96, February 2011. doi:10.1016/j.disopt.2010.09.007.
7. Daniel Binkele-Raible, Ljiljana Brankovic, Marek Cygan, Henning Fernau, Joachim Kneis, Dieter Kratsch, Alexander Langer, Mathieu Liedloff, Marcin Pilipczuk, Peter Rossmanith, and Jakub Onufry Wojtaszczyk. Breaking the $2^n$-barrier for Irredundance: Two lines of attack. *Journal of Discrete Algorithms*, 9(3):214–230, September 2011. doi:10.1016/j.jda.2011.03.002.
8. Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. The traveling salesman problem in bounded degree graphs. *ACM Transactions on Algorithms*, 8(2):1–13, April 2012. doi:10.1145/2151171.2151181.
9. Ivan Bliznets, Fedor V. Fomin, Michał Pilipczuk, and Yngve Villanger. Largest Chordal and Interval Subgraphs Faster Than $2^n$. In *Lecture Notes in Computer Science*, pages 193–204. Springer Berlin Heidelberg, 2013. doi:10.1007/978-3-642-40450-4_17.
10. Carmen C. Centeno, Mitre C. Dourado, Lucia Draque Penso, Dieter Rautenbach, and Jayme L. Szwarcfiter. Irreversible conversion of graphs. *Theoretical Computer Science*, 412(29):3693–3700, July 2011. doi:10.1016/j.tcs.2011.03.029.
11. Ning Chen. On the Approximability of Influence in Social Networks. *SIAM Journal on Discrete Mathematics*, 23(3):1400–1415, January 2009. doi:10.1137/08073617x.
12. Morgan Chopin, André Nichterlein, Rolf Niedermeier, and Mathias Weller. Constant Thresholds Can Make Target Set Selection Tractable. *Theory of Computing Systems*, 55(1):61–83, September 2013. doi:10.1007/s00224-013-9499-3.

13 Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. Scheduling Partially Ordered Jobs Faster than $2^n$. *Algorithmica*, 68(3):692–714, October 2012. doi:10.1007/s00453-012-9694-7.

14 Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. Solving the 2-Disjoint Connected Subgraphs Problem Faster than $2^n$. *Algorithmica*, 70(2):195–207, May 2013. doi:10.1007/s00453-013-9796-x.

15 Marek Cygan, Marcin Pilipczuk, and Jakub Onufry Wojtaszczyk. Capacitated Domination Faster Than $O(2^n)$. In *Lecture Notes in Computer Science*, pages 74–80. Springer Berlin Heidelberg, 2010. doi:10.1007/978-3-642-13731-0_8.

16 Paul A. Dreyer Jr. and Fred S. Roberts. Irreversible $k$-threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion. *Discrete Applied Mathematics*, 157(7):1615–1627, 2009.

17 Pavel Dvořák, Dušan Knop, and Tomáš Toufar. Target Set Selection in Dense Graph Classes. *arXiv preprint arXiv:1610.07530*, 2016.

18 Fedor V. Fomin, Serge Gaspers, Artem V. Pyatkin, and Igor Razgon. On the Minimum Feedback Vertex Set Problem: Exact and Enumeration Algorithms. *Algorithmica*, 52(2):293–307, December 2007. doi:10.1007/s00453-007-9152-0.

19 Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. Solving Connected Dominating Set Faster than $2^n$. *Algorithmica*, 52(2):153–166, December 2007. doi:10.1007/s00453-007-9145-z.

20 Fedor V. Fomin, Pinar Heggernes, Dieter Kratsch, Charis Papadopoulos, and Yngve Villanger. Enumerating Minimal Subset Feedback Vertex Sets. *Algorithmica*, 69(1):216–231, December 2012. doi:10.1007/s00453-012-9731-6.

21 Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.

22 Fedor V. Fomin, Ioan Todinca, and Yngve Villanger. Exact Algorithm for the Maximum Induced Planar Subgraph Problem. In *Algorithms – ESA 2011*, pages 287–298. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-23719-5_25.

23 Tim A. Hartmann. Target Set Selection Parameterized by Clique-Width and Maximum Threshold. In *SOFSEM 2018: Theory and Practice of Computer Science*, pages 137–149. Springer International Publishing, December 2017. doi:10.1007/978-3-319-73117-9_10.

24 David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '03*. ACM Press, 2003. doi:10.1145/956750.956769.

25 Dániel Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394–406, February 2006. doi:10.1016/j.tcs.2005.10.007.

26 André Nichterlein, Rolf Niedermeier, Johannes Uhlmann, and Mathias Weller. On tractable cases of Target Set Selection. *Social Network Analysis and Mining*, 3(2):233–256, May 2012. doi:10.1007/s13278-012-0067-7.

27 D. Peleg. Size bounds for dynamic monopolies. *Discrete Applied Mathematics*, 86(2-3):263–273, September 1998. doi:10.1016/s0166-218x(98)00043-2.

28 Marcin Pilipczuk and Michał Pilipczuk. Finding a Maximum Induced Degenerate Subgraph Faster Than $2^n$. In *Parameterized and Exact Computation*, pages 3–12. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-33293-7_3.

29 Michał Pilipczuk. Exact Algorithms for Induced Subgraph Problems. In *Encyclopedia of Algorithms*, pages 1–5. Springer US, 2015. doi:10.1007/978-3-642-27848-8_520-1.

30 Igor Razgon. Computing Minimum Directed Feedback Vertex Set in $O^*(1.9977^n)$. In *Theoretical Computer Science*, pages 70–81. World Scientific, 2007.

# Resolving Conflicts for Lower-Bounded Clustering

**Katrin Casel**
Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
Universität Trier, Fachbereich IV - Informatikwissenschaften, Germany
casel@informatik.uni-trier.de

## Abstract

This paper considers the effect of non-metric distances for lower-bounded clustering, i.e., the problem of computing a partition for a given set of objects with pairwise distance, such that each set has a certain minimum cardinality (as required for anonymisation or balanced facility location problems). We discuss lower-bounded clustering with the objective to minimise the maximum radius or diameter of the clusters. For these problems there exists a 2-approximation but only if the pairwise distance on the objects satisfies the triangle inequality, without this property no polynomial-time constant factor approximation is possible, unless $\mathsf{P} = \mathsf{NP}$. We try to resolve or at least soften this effect of non-metric distances by devising particular strategies to deal with violations of the triangle inequality (*conflicts*). With parameterised algorithmics, we find that if the number of such conflicts is not too large, constant factor approximations can still be computed efficiently.

In particular, we introduce parameterised approximations with respect to not just the number of conflicts but also for the vertex cover number of the *conflict graph* (graph induced by conflicts). Interestingly, we salvage the approximation ratio of 2 for diameter while for radius it is only possible to show a ratio of 3. For the parameter vertex cover number of the conflict graph this worsening in ratio is shown to be unavoidable, unless $\mathsf{FPT} = \mathsf{W}[2]$. We further discuss improvements for diameter by choosing the (induced) $\mathcal{P}_3$-cover number of the conflict graph as parameter and complement these by showing that, unless $\mathsf{FPT} = \mathsf{W}[1]$, there exists no constant factor parameterised approximation with respect to the parameter split vertex deletion set.

**2012 ACM Subject Classification** Theory of computation → Approximation algorithms analysis, Theory of computation → Parameterized complexity and exact algorithms, Theory of computation → Facility location and clustering

**Keywords and phrases** clustering, triangle inequality, parameterised approximation

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.23

**Acknowledgements** I would like to thank Lorik Dumani for his invaluable help with implementing and testing the algorithms in this paper. This work was supported by the DFG, grant FE 560/6-1.

## 1 Introduction

For most clustering problems, the quality of a solution is usually assessed with respect to a given pairwise distance on the input objects. Approximate solutions for such tasks often rely on this distance to be a metric. But what happens if this property does not hold? Many clustering problems are much more difficult to solve or even approximate if the pairwise distance violates the triangle inequality. The problem Uncapacitated Facility Location, for example, can be approximated with ratio 1.488 if restricted to metric instances, see [20]. For general, possibly non-metric, distances, it is only possible to compute a $\log(n)$-approximation; see [18] for one of many algorithms with this performance. The relation to the Set Cover problem does not just provide the basis for this positive approximation result,

© Katrin Casel;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 23; pp. 23:1–23:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

but also transfers non-approximability. In particular, is known that that $\log(n)$ is the best approximation ratio for Set Cover by [10], assuming $\mathsf{P} \neq \mathsf{NP}$, and this hardness transfers to Uncapacitated Facility Location by a very simple approximation-preserving reduction identifying sets with facilities and the universe with the set of customers.

Such helpful consequences of a restriction to triangle inequality have lead to many approaches which assume that this property holds. Another example of this kind is given in [13], where the properties that come with a restriction to distances which satisfy the triangle inequality are used to speed up the famous heuristic algorithm *k-means*, named after the clustering problem it is designed to approximate efficiently.

For many applications, the requirement that the associated distance is a metric seems to be pretty natural and is not really considered a restriction. For lower-bounded clustering, the problem of computing a partition for a given set of objects with pairwise distance such that each set has a certain minimum cardinality, we also made this assumption in order to enable approximation algorithms with a provable performance ratio, see [1]. In particular, with the objective to minimise the maximum radius or diameter of the clusters, it turned out that, unless $\mathsf{P} = \mathsf{NP}$, there exists no polynomial-time constant factor approximation if the pairwise distance on the objects violates the triangle inequality while a ratio of 2 can be achieved without such violations. With an attempt to use this problem to model a clustering which can be used for recommender systems, we however found that the pairwise distance does not in general satisfy the triangle inequality. The so-called *Pearson distance*, which is usually used for recommendations, does not have this useful property and, as also observed in [23], practical instances actually show this non-metric behaviour. Such unfortunate situations seem to be unavoidable when it comes to human preferences which raises the question of how the resulting negative effects can be avoided, or at least controlled.

One option that comes to mind concerning non-metric instances in general is editing, i.e., a pre-processing step which tries to transform a given instance, with preferably few changes, such that triangle inequality holds and known algorithms for such well-behaved instances can then be applied. This idea however has several drawbacks. Changes to a given instance always come at the price of distortion; altering distances or even deleting objects results in perturbation of the original input. This effect hence raises the task to find alterations which bring as little change to the original instance as possible. Such editing problems are then usually already difficult to solve themselves. In our specific case of lower-bounded clustering, the task to find a minimum number of vertices such that their removal from a given instance deletes all violations of the triangle inequality is closely related to the 3-hitting set problem. But much more troublesome than the complexity of computing such minimal alterations for these types of problems is the danger of accidentally worsening the optimum value with vertex deletion. Observe that by requiring a lower bound on the cardinality of the clusters, the optimum value is not necessarily monotone, in the sense that a larger set of input objects might enable a better solution.

Here, we therefore seek a different approach which employs extra treatment for violations of the triangle inequality within the approximations designed for metric instances. The basic idea is to investigate the consequences of violations and devise strategies to deal with those within moderate exponential time depending on, roughly speaking, how much the given pairwise distance differs from a metric. More precisely, we will, for a pairwise distance $d$, look at the set of pairs $\{u, v\}$ which directly violate the triangle inequality, i.e., there exists another object $x$ such that $d(u, v) > d(u, x) + d(v, x)$. We call such pairs *conflicts*. If the set of conflicts for a given instance is empty, the associated distance obviously satisfies the triangle inequality, which makes the cardinality of the set of conflicts a reasonable measure

for our purposes. Our strategy then is to alter the algorithms for metric instances in such a way that they also yield constant-factor approximations for non-metric instances while only spending exponential effort with respect to the conflicts. Formally, this gives a parameterised approximation with structural parameterisation by the number of conflicts.

This kind of parameterisation by conflicts to improve approximabilty can be seen as a generalisation of the *distance from triviality* approach introduced in [17]. The idea there is to define some *distance* which specifies how much a given instance differs from some structural property which makes it easy to solve, and use this measure as parameter. The term *triviality* there already refers to the broader case of polynomial time solvable instances, not just trivial inputs as one might think, and in our case we go one step further and see the number of conflicts as the distance to an instance which can be approximated efficiently.

We discuss conflicts for the problems of minimising the maximum radius or diameter for lower-bounded clustering. We first develop parameterised approximations with respect to conflicts and then improve those to only require exponential time with respect to the vertex cover number of the *conflict graph* $G_c$ (the graph induced by conflicts interpreted as edges). For diameter, we then consider even smaller parameters given by the size of an (induced) $\mathcal{P}_3$-cover of $G_c$, but conversely show that the even smaller size of a split vertex deletion set for $G_c$ is not a suitable parameter. Curiously, we find that while the ratio of 2 remains for diameter, it is only possible to prove a ratio of 3 for radius. For the parameter vertex cover of $G_c$, we prove that this worsening is unavoidable under the assumption that $\mathsf{FPT} \neq \mathsf{W}[2]$.

## 2 Preliminaries

We mostly use standard notation and refer to textbooks such as [6] for graph theory, [3] for approximation algorithms and [12] for parameterised complexity terminology.

When estimating running times we use $\mathcal{O}^*$-notation which, compared to $\mathcal{O}$-notation, also suppresses factors which are polynomial in the input size. We use $B_n$ to denote the $n$th *Bell number* which is the number of partitions of a set of size $n$.

The algorithms discussed here combine parameterisation and approximation and fall into the category of *fpt-approximation algorithms with parameter* $\kappa$, as discussed for example in [22], i.e., approximation algorithms with provable ratio and running time in $\mathcal{O}(g(\kappa) \cdot f(n))$, for computable function $g$ and polynomial $f$, so $\mathcal{O}^*(g(\kappa))$ in $\mathcal{O}^*$-notation. For lower bounds in this context, there does not seem to exist a unified notation, so for these kinds of results, we will not use hardness notions but link the existence of certain parameterised approximations to the (unlikely) equivalence of certain complexity classes.

### 2.1 Problem Definition

As formal model for lower-bounded clustering we use the abstract problem $(\|\cdot\|, f)$-$k$-CLUSTER from [1]. An instance of this problem is an undirected graph $G = (V, E)$ with edge-weights $w_E \colon E \to \mathbb{R}_+$ and a lower bound $k \in \mathbb{N}$. A feasible solution is any partition $P_1, \ldots, P_s$ of $V$ such that $|P_i| \geq k$ for all $i \in \{1, \ldots, s\}$, we refer to such a partition as *$k$-cluster*.

For two vertices $u, v \in V$ with $u \neq v$ we define $d(u, v) := w_E(\{u, v\})$ if $\{u, v\} \in E$, and if $\{u, v\} \notin E$, the distance $d(u, v)$ is defined by the shortest path from $u$ to $v$ in $G$. For simplicity we always extend $d$ to a function on the whole set $V \times V$ by defining $d(v, v) = 0$ for all $v \in V$. This definition of the *induced distance* $d$ derived from weights only known for a set $E$ of given edges models missing information about pairwise distances.

For the objectives to minimise maximum radius or diameter we formally define $\mathrm{rad}(P) := \min\{\max\{d(x, y) \colon y \in P\} \colon x \in P\}$ and $\mathrm{diam}(P) := \max\{\max\{d(x, y) \colon y \in P\} \colon x \in P\}$. The

**Figure 1** In the graph on the left, $\{c, d\}$ has a weight larger than the shortest path from $c$ to $d$ but is not in $C$. Lowering the weights on $\{a, d\}$ and $\{b, c\}$ to 2 to remove these conflicts turns $\{c, d\}$ into a conflict. The graph on the right has a 2-cluster of maximum radius and diameter 1. Removing $a$ to delete conflicts results in a graph for which only a trivial 2-cluster of radius $\Delta$ is possible.

resulting problems $(\|\cdot\|_\infty, \text{rad})$-$k$-CLUSTER and $(\|\cdot\|_\infty, \text{diam})$-$k$-CLUSTER ask for a $k$-cluster $P_1, \ldots, P_s$ minimising $\max\{\text{rad}(P_i) \colon 1 \leq i \leq s\}$ and $\max\{\text{diam}(P_i) \colon 1 \leq i \leq s\}$, respectively. A vertex $x \in V$ with $\max\{d(x, y) \colon y \in P\} = \text{rad}(P)$ is called *central* for $P$.

## 2.2 Conflicts

A function $d \colon V \times V \to \mathbb{R}^+$ satisfies the *triangle inequality* if $d(u, v) \leq d(u, w) + d(w, v)$ for all $u, v, w \in V$. We will call an instance of $(\|\cdot\|, f)$-$k$-CLUSTER *metric* if the induced distance satisfies the triangle inequality although this does not necessarily make $d$ a metric in the classical definition of this word, as we allow the existence of $u \neq v$ with $d(u, v) = 0$ (violation of the so-called *identity of indiscernibles* property of metrics); recall that by the formal definition, $d$ derived from edge-weights is just non-negative, symmetric and reflexive.

Observe that our definition allows non-metric instances, see Figure 1. We define the set of *conflicts* for an instance $(G, k)$ with $G = (V, E)$ and induced distance $d$ as the collection $C$ of vertex pairs $\{u, v\}$ such that the triangle inequality is violated for $u$ and $v$, formally:

$$C = \{\{u, v\} \in V \times V \colon \exists\, x \in V \colon d(u, v) > d(u, x) + d(v, x)\}\,.$$

One curious property is that the set of conflicts is not necessarily the whole set of edges with a weight larger than the cheapest path in the graph (for a counterexample see Figure 1), it however is always a subset of $E$. Considering the option of weight reduction to achieve triangle inequality, $C$ might be smaller than the set of edge-weights which have to be adjusted in order to arrive at a graph without conflicts. Figure 1 also gives a small example which illustrates why vertex removal can be a dangerous editing step for problems with non-monotone objective function such as lower bounded clustering. Observe how the optimum value may increase arbitrarily from the original graph to a graph created by deleting vertices to avoid conflicts. This effect is another reason to favour parameterisation over editing.

Actually, we will mostly consider parameterisation by the cardinality of the set $P$ of *conflict vertices*, which simply are the vertices involved in a conflict, formally defined by:

$$P = \bigcup_{\{u,v\} \in C} \{u, v\}\,.$$

In the following we will use $c$ and $p$ for the parameters number of conflicts and number of conflict vertices, respectively. Parameterisation by $p$ yields the same general tractability as parameterisation by $p$ as the parameters are related by the inequalities $p \leq 2c \leq (p(p-1))$. For the concrete running times, it is however still relevant to distinguish between $p$ and $c$ as the bounds given by this inequality are sharp.

We further refer to the graph $G_c = (P, C)$ as *conflict graph*; observe that $G_c$ is always a subgraph of the input graph. The structure of the conflict graph reflects the entanglement

of conflicts in a given instance. When designing algorithms which devise specific strategies to resolve conflicts, it is not surprising that the relations between these can be exploited for improvement. The structure of the conflict graph hence yields further possibilities for a parameterisation which measures the distance to a metric instance.

## 3 Parameterisation by Conflict Vertices

Without restriction to metric instances, there exists no constant factor approximation in polynomial time for $(\|\cdot\|_\infty, \text{rad})$- or $(\|\cdot\|_\infty, \text{diam})$-$k$-CLUSTER, unless $\mathsf{P} = \mathsf{NP}$ (see Proposition 6 from [1]). The 2-approximation presented for these problems in [1], Theorem 9 hence highly relies on the assumption that the input instance is metric. In short, on input $((V, E), w_E, k)$ with induced distance $d$, the approximation algorithm with the performance ratio of 2 for both $(\|\cdot\|_\infty, \text{rad})$- and $(\|\cdot\|_\infty, \text{diam})$-$k$-CLUSTER simply performs a binary search for the smallest value $D$ for which the following 2-step greedy procedure is successful:

**step (1)** While $V$ is not empty, pick some $c \in V$, build the set $P(c) := \{w \in V : d(c, w) \leq D\}$ and set $V = V \setminus P(c)$.

**step (2)** Let $P(c_1), \ldots, P(c_s)$ be the partition built in step (1). Try to create from this a $k$-cluster $P'(c_1), \ldots, P'(c_s)$ such that $c_i \in P'(c_i)$ and $d(v, c_i) \leq D$ for all $v \in P'(c_i)$, $i \in \{1, \ldots, s\}$. (Such a partition can be efficiently computed with the help of a network flow formulation over vertices $V \cup \{s, t\}$. With arcs of capacity 1 from $s$ to every $v \in V$, arcs of capacity $k$ from $c_i$ to $t$ and arcs of capacity 1 from $v \in V$ to $c_i$ if $d(v, c_i) \leq D$ for each $i \in \{1, \ldots, s\}$. A flow of value $sk$ in this network interpreted as moving a vertex $v \in P(c_i)$ into the set $P(c_j)$ if and only if the flow uses the arc from $v$ to $c_j$, gives a polynomial procedure to create a $k$-cluster.)

The ratio of 2 for $(\|\cdot\|_\infty, \text{rad})$- and $(\|\cdot\|_\infty, \text{diam})$-$k$-CLUSTER can be shown by proving that the greedy procedure is successful if $D$ is fixed to be twice the optimum radius or the optimum diameter, respectively, which is due to the following two properties for this choice of $D$:

**(i)** By the choice of the vertices $c_i$ in step (1), it follows that $d(c_i, c_j) > D$ for all $i \neq j$, which means that $c_i$ and $c_j$ belong to different sets in an optimum solution. So, for each $i$ there exist enough vertices at distance at most $D$ from $c_i$ to be put into $P'(c_i)$ in step (2).

**(ii)** If the greedy procedure is successful, the construction immediately yields $d(v, c_i) \leq D$ for each $v \in P'(c_i)$, so the resulting $k$-cluster $P'(c_1), \ldots, P'(c_s)$ has maximum radius $D$ and maximum diameter $2D$.

The above properties however do not hold in case the input instance is not metric. More precisely, for each objective function (radius, diameter), one of them is no longer true. For radius, property (i) fails, as without triangle inequality, vertices at distance more than twice the optimum can still be contained in the same cluster in an optimum solution. For diameter, property (ii) fails, as a radius of $D$ does no longer guarantee a diameter of $2D$.

We will now try to salvage these properties for non-metric instances by adding exponential effort with respect to conflicts to the above approximation procedure. Given that different properties are lost for the two objective functions, it is not surprising that this approach yields two different parameterised approximations, the basic algorithmic idea of fixing a maximum radius $D$, building a preliminary clustering with clusters of radius $D$ and then balancing the cardinalities with a network however always remains and we will only sketch the crucial points which have to be adjusted in each case.

At first, we observe that starting from the approximation algorithm for metric instances, it is not too hard to see that a constant number of conflicts does not yield too much trouble. More precisely, we simply guess a suitable central vertex for each conflict vertex and fix the

resulting partition of $P$ and centres for step (1). This optimal guessing for the problematic vertices resolves the problems for both objective functions, as the fixed centres are optimal by choice and the greedy algorithm is only responsible for partitioning remaining vertices in $V \setminus P$ for which the triangle inequality holds. This kind of guessing yields:

▶ **Theorem 1.** *A 2-approximation for* $(\|\cdot\|_\infty, \mathrm{rad})$*- and* $(\|\cdot\|_\infty, \mathrm{diam})$*-$k$-CLUSTER can be computed with a running time in* $\mathcal{O}^*(n^p)$.

This result raises the question whether an improvement to a more efficient running time is possible, i.e., some constant factor parameterised approximation. To this end, we want to mention that an improvement of the approximation ratio of 2 is unlikely, as this is already shown to imply $\mathsf{P} = \mathsf{NP}$ for metric instances in [1], Corollaries 1 and 3.

For diameter, we have to only be careful with property (ii) which means that in the above guessing, we did not really require the knowledge of centres but only the partition they imply on the vertices in $P$. In fact, it can be shown that knowing this partition is enough, and simply enforcing it in the metric approximation algorithm still yields a 2-approximation for $(\|\cdot\|_\infty, \mathrm{diam})$-$k$-CLUSTER. The cost of trying all partitions of $P$ can be estimated with the Bell number (which can be bounded by $B_n < \left(\frac{0.792n}{\log(n+1)}\right)^n$, see [5]) and yields:

▶ **Theorem 2.** *A 2-approximation for* $(\|\cdot\|_\infty, \mathrm{diam})$*-$k$-CLUSTER can be computed with a running time in* $\mathcal{O}^*(B_p)$.

For radius, the fixed centres are important, but by compromising a little on the approximation ratio it is still possible to design a parameterised approximation with parameter $p$. This algorithm only requires guessing which vertices in $P$ are central in an optimal solution and accordingly forcing step (1) to build clusters around those first; observe that this knowledge salvages property (i). Picking a suitable cluster for the vertices in $P$ which are not chosen to be central however blows up the approximation ratio to 3 (an effect which is explained later in connection to the lower bounds), which yields:

▶ **Theorem 3.** *A 3-approximation for* $(\|\cdot\|_\infty, \mathrm{rad})$*-$k$-CLUSTER can be computed with a running time in* $\mathcal{O}^*(2^p)$.

## 4 Structural Parameters of the Conflict Graph

One possibility to speed up the parameterised approximation algorithms presented so far, is to choose a smaller parameter. In this section, we want to focus on strategies to only spend exponential time for vertices in a subset of $P$. More precisely, as advertised in the section-title, we will consider parameterisation by structural parameters of the conflict graph.

### 4.1 Vertex Cover

Looking closer at the problems caused by the conflicts in $C$, it is not necessary to consider all vertices in $P$ but it appears to be sufficient to pick a subset which covers all conflicts. Formally, this idea translates into parameterisation by a vertex cover for the conflict graph $G_c = (P, C)$. In the following, we will use $p_c$ to denote the size of a minimum vertex cover for $G_c$ and discuss parameterised approximation with respect to this parameter.

Again, a first easy observation is that, at least for $(\|\cdot\|_\infty, \mathrm{rad})$-$k$-CLUSTER, instances for which $p_c$ is a constant can be approximated efficiently. In fact, we can simply switch from the set $P$ to a minimum vertex cover for $G_c$ in the procedure discussed for Theorem 1, as property (i) can only be violated if the algorithm can choose two vertices of a conflict as centres. The additionally required computing of a constant size vertex cover for $G_c$ is not an expensive task, so this idea immediately yields:

▶ **Theorem 4.** *A 2-approximation for* $(\|\cdot\|_\infty, \mathrm{rad})$*-*$k$*-*CLUSTER *can be computed with a running time in* $\mathcal{O}^*(n^{p_c})$*.*

The parameterised approximation for $(\|\cdot\|_\infty, \mathrm{diam})$-$k$-CLUSTER from Section 3 requires little algorithmic adjustment to switch from parameter $p$ to parameter $p_c$. Proving correctness of the given procedure, i.e., guaranteeing a performance ratio, is however more complicated.

Consider only guessing a partition of a vertex cover $\mathcal{V}$ of the conflict graph and fixing this partition for step (1). The only further change required to make sure that property (ii) remains true, is to adjust the algorithm when building clusters including the guessed sets $V_1, \ldots, V_t$ in the partition of $\mathcal{V}$. Now, both when building the preliminary clusters including some $V_i$ in step (1) and moving vertices in step (2) into such a cluster, we not just require a distance of at most $D$ to one chosen centre but to all vertices in $V_i$. Then all distances in a set of the resulting $k$-cluster involving vertices in the cover $\mathcal{V}$ are bounded by $D$. As $\mathcal{V}$ is a vertex cover of $G_c$, distances not involving a vertex in $\mathcal{V}$ are not a conflict which means triangle inequality can be used for all remaining cases to prove that property (ii) still holds.

This approach requires computing a vertex cover for the conflict graph, a problem which can be solved by [9] with a running time in $\mathcal{O}^*(1.2738^{p_c})$. As this single-exponential effort is only performed once in the beginning and dominated by the Bell number, we can conclude:

▶ **Theorem 5.** *A 2-approximation for* $(\|\cdot\|_\infty, \mathrm{diam})$*-*$k$*-*CLUSTER *can be computed with a running time in* $\mathcal{O}^*(B_{p_c})$*.*

For $(\|\cdot\|_\infty, \mathrm{rad})$-$k$-CLUSTER, it is not obvious how to reduce $p$ in Theorem 3 to $p_c$. While knowing which vertices in a vertex cover are central is sufficient to avoid picking centres from the same cluster in step (1), the problem is finding a suitable cluster for the vertices from the vertex cover which are not central. In particular, this is a problem when two vertices from the vertex cover which are not central are wrongfully put in the same cluster; more precisely, if two vertices $u, v \in \mathcal{V}$ are not central in any optimal solution but belong to two different clusters, with centres $c_u, c_v \notin \mathcal{V}$, while these two correct centres are put into $P(c)$ with $\{u, c\}, \{v, c\} \in C$ in step (1) of our algorithm. In such a case there is no general clean way to identify how to split up $P(c)$ into sets of cardinality at least $k$ and such that $u$ and $v$ can be assigned at a radius which can in any way be bounded by the optimum.

With a more involved algorithm which additionally guesses a partition, like for the diameter measure, it is possible to find an approximation for $(\|\cdot\|_\infty, \mathrm{rad})$-$k$-CLUSTER parameterised by $p_c$. After guessing a partition and a set of centres for the vertex cover, this approach further requires computing a suitable centre for each set in the partition before running steps (1) and (2), which are altered to respect the fixed partition and central vertices. This just means finding for each set $V_i$ in the partition of the vertex cover $\mathcal{V}$ for which no vertex in $V_i$ is fixed as center, a vertex $v \in V \setminus \mathcal{V}$ such that $d(v, w) \leq \frac{D}{2}$ for all $w \in V_i$ (where $\frac{D}{2}$ relates to the optimum radius). As two sets $V_i$ and $V_j$ might compete over such candidates for centres, we use maximum matching to enable finding, in case our fixed guesses are correct, a centre for each set $V_i$. These adjustments are sufficient to salvage property (i). Although now there is no longer the problem of finding a suitable cluster for vertices which are not allowed to be central, the performance ratio is still only 3, as for this procedure the worst-case now comes from choosing a wrong center $c' \in V \setminus \mathcal{V}$ for some $V_i$: If $P \subseteq V$ is the cluster containing $V_i$ in an optimum solution and $c$ is its correct center, it follows that the distance of $v \in P$ to the wrong center $c'$ can only be bounded by $d(v, c') \leq d(v, c) + d(c, c') \leq \frac{D}{2} + d(c, v_i) + d(c', v_i) \leq 3\frac{D}{2}$ (using some $v_i \in V_i$ for this equation). The asymptotic running time of this approach is dominated by guessing the partition and centre-choice for a minimum vertex cover of $G_c$, so:

▶ **Theorem 6.** *A 3-approximation for* $(\|\cdot\|_\infty, \mathrm{rad})$-$k$-CLUSTER *can be computed with a running time in* $\mathcal{O}^*(2^{p_c} \cdot B_{p_c})$.

## 4.2 $\mathcal{P}_3$-Covers

With more changes to the algorithms discussed for $(\|\cdot\|_\infty, \mathrm{diam})$-$k$-CLUSTER, it is possible to further reduce the size of the subset of $P$ which requires the expensive guessing of the partition. When building the first partition in step (1), it is always possible to correctly assign conflict vertices to a set, by branching on the conflicts to decide which vertex has to be excluded. This way it is possible to find a correct choice of central vertices. The network used to model the reassignment of vertices according to the fixed centres can be altered to prevent two conflict vertices to move into the same cluster, by routing their flow through an additional network-vertex with a capacity of only 1 to move into a cluster. If the conflicts are isolated, an additional network-vertex for each conflict can be used to correctly model all conflict-free reassignments.

We can of course not assume that conflicts are pairwise disjoint, but we can fix the partition of a subset of conflict vertices, as we did for the vertex cover of the conflict graph, and use the above ideas for the remaining vertices which induce a graph with isolated conflicts. The set of vertices which have to be removed in order to arrive at a graph with isolated edges, or equivalently a graph which does not contain any path of length 2 usually denoted $\mathcal{P}_3$, is smaller than the vertex cover of the conflict graph (unless the instance is initially metric). Formally, such a set is called a $\mathcal{P}_3$-*cover*. In the following, we use $p_{3c}$ to denote the cardinality of a smallest $\mathcal{P}_3$-cover for $G_c$ and with the parameterised algorithm from [24], such a set can be computed with a running time in $\mathcal{O}^*(1.7485^{p_{3c}})$. Using the more expensive strategy of guessing the correct partition only for a minimum $\mathcal{P}_3$-cover of the conflict graph computed with the algorithm in [24], branching on the remaining isolated conflicts for the pre-clustering and modifying the network to avoid conflicts as described above gives the following result; observe that the number of remaining conflicts is bounded by $\frac{p}{2}$:

▶ **Theorem 7.** *A 2-approximation for* $(\|\cdot\|_\infty, \mathrm{diam})$-$k$-CLUSTER *can be computed with a running time in* $\mathcal{O}^*(\sqrt{2}^p \cdot B_{p_{3c}})$.

As already mentioned, branching on conflicts in step (1) works for any set of conflicts, not just for the restriction to isolated ones. The reassignment restriction for conflict vertices modelled with the capacities in the network however requires a situation where, in case of conflict, at most one vertex can be moved into a cluster. Capacities on arcs from some additional network-vertices which handle conflicts, can not model a scenario where out of three vertices $u, v, w$, a cluster is restricted to either only contain $u$ or any subset of $\{v, w\}$; this situation occurs if $u$ is in conflict with $v$ and $w$ but $\{u, w\}$ is not a conflict. This structure means that the vertices $u, v, w$ induce a $\mathcal{P}_3$ in the conflict graph. If the conflict graph, or any graph for that matter, does not contain an induced $\mathcal{P}_3$, its connected components are cliques. For this structure, the network can be adjusted to correctly model conflict-free vertex-reassignments. The problem to find, for a given graph, a smallest vertex set whose removal yields a $\mathcal{P}_3$-free graph is called INDUCED $\mathcal{P}_3$-COVER or CLUSTER VERTEX DELETION.

It is possible to, in a sense, generalise the algorithm for Theorem 7 to consider a cluster vertex deletion set instead of a $\mathcal{P}_3$-cover to reduce the cost for guessing the partition. We denote the corresponding parameter, size of a cluster vertex deletion set for the conflict graph, by $p_{3d}$. While the relation $p_{3d} \leq p_{3c}$ obviously makes this generalisation an improvement, we have to pay for this in the branching for the pre-clustering, as the remaining conflicts are no longer bounded by $\frac{1}{2}|P|$. A minimum cluster vertex deletion set for $G_c$ can be computed in

**Table 1** Summary of the running time of the parameterised 2-approximation for $(\|\cdot\|_\infty, \text{diam})$-$k$-CLUSTER, where $p_c$, $p_{3c}$ and $p_{3d}$ denote the size of a minimum vertex, $\mathcal{P}_3$ and induced $\mathcal{P}_3$-cover for the conflict graph, respectively.

| $p_c$ | $p_{3c}$ | $p_{3d}$ |
|---|---|---|
| $\mathcal{O}^*(B_{p_c})$ (Theorem 5) | $\mathcal{O}^*(\sqrt{2}^p B_{p_{3c}})$ (Theorem 7 ) | $\mathcal{O}^*(2^c B_{p_{3d}})$ (Theorem 8 ) |

time $\mathcal{O}^*(1.9102^{p_{3d}})$ with the help of the parameterised algorithm in [7], so again a negligible effort compared to checking all partitions, and this idea hence yields the following result:

▶ **Theorem 8.** *A 2-approximation for* $(\|\cdot\|_\infty, \text{diam})$-$k$-CLUSTER *can be computed with a running time in* $\mathcal{O}^*(2^c \cdot B_{p_{3d}})$.

The results to improve the parameterised approximation from Theorem 5 are presented here in a way which suggest a stepwise improvement of the running time. In principle, reducing the number of vertices which require partitioning appears to be the best option. But the reductions of this set used for Theorems 7 and 8 require additional branching costs on conflict vertices and conflicts, respectively. Depending on the structure of the conflict graph, any one of the three algorithms can have the best worst-case running time. An overview of the parameterised 2-approximations for $(\|\cdot\|_\infty, \text{diam})$-$k$-CLUSTER with respect to the structural parameters of the conflict graph discussed in this section is given in Table 1.

## 5 Lower Bounds

In this section, we investigate the limitations of parameterised approximation for lower bounded clustering with structural parameters of the conflict graph. Especially the increase from ratio 2 for metric instances to ratio 3 for non-metric instances for $(\|\cdot\|_\infty, \text{rad})$-$k$-CLUSTER appears strange. We will however see that ratio 3 is, under certain complexity theoretic assumptions, optimal for parameter $p_c$ and that the approach we use to design parameterised approximations is generally limited to the performance ratio of 3 for radius. Further, we will discuss the limits of choosing structural parameters for $(\|\cdot\|_\infty, \text{diam})$-$k$-CLUSTER. More precisely, we will see that while the previous section gave approaches to move from $p$ to $p_c$, $p_{3c}$ and $p_{3d}$, a next step towards a parameterisation by a split vertex deletion set does not seem to allow for a constant factor parameterised approximation.

For the negative results of this section, we use a kind of reduction which links the existence of a parameterised approximation with certain ratio to a parameterised algorithm for a problem which is believed not to be in FPT. Such an algorithm can be seen as *fpt gap-reduction* as introduced in [4]. As the problem to reduce from for $(\|\cdot\|_\infty, \text{rad})$-$k$-CLUSTER, we use MULTICOLOURED DOMINATING SET, which asks for input graph $G = (V, E)$ with vertex partition $V = V_1, \ldots, V_\ell$ for the existence of a subset $\mathcal{D} \subseteq V$ such that $N[\mathcal{D}] = V$ ($\mathcal{D}$ is a dominating set for $G$) and $|\mathcal{D} \cap V_i| = 1$ for all $i \in \{1, \ldots, \ell\}$. The colour-coding technique from [2] shows that the W[2]-hardness of classical MINIMUM DOMINATING SET, which is shown in [11], transfers to this restricted version we called *multicoloured* in reference to MULTICOLOURED CLIQUE and the corresponding reduction technique introduced in [15].

We will in the following sketch a reduction from MULTICOLOURED DOMINATING SET to $(\|\cdot\|_\infty, \text{rad})$-$k$-CLUSTER which shows that a parameterised approximation with parameter $p_c$ for the clustering problem could be used to show fixed parameter tractability of the W[2]-hard domination problem. This kind of reduction yields a lower bound for parameterised approximation of $(\|\cdot\|_\infty, \text{rad})$-$k$-CLUSTER under the assumption FPT $\neq$ W[2].

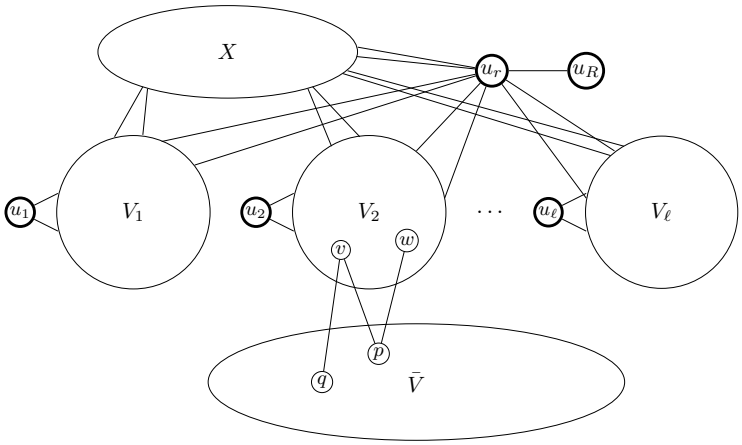



**Figure 2** Illustration of the reduction used for Theorem 9, vertices in the vertex cover for the conflict graph drawn with thick border.

For an instance $G = (V, E)$ with $V = V_1, \ldots, V_\ell$ and $|V| = n$ of Multicoloured Dominating Set, we construct an instance $((V', E'), w_{E'}, n + 2)$ of $(\|\cdot\|_\infty, \mathrm{rad})$-$k$-cluster with vertex set $V'$ containing $V$, a copy of $V$, denoted $\bar{V} = \{\bar{v} \colon v \in V\}$, $\ell + 2$ vertices (which will become the vertex cover of the conflict graph) denoted $u_1, \ldots, u_\ell$ and $u_r, u_R$ and an additional set $X$ of $(\ell - 1)n + \ell$ vertices. These vertices are connected with the following edges of weight 1 (see also the illustrated in Figure 9):

- $\{v, \bar{w}\}, \{\bar{w}, v\} \in E'$ iff $\{v, w\} \in E$ (these model the structure of the original graph),
- $\{u_i, v\} \in E'$ for all $v \in V_i$, $i \in \{1, \ldots, \ell\}$ (these force to pick one center from each $V_i$),
- $\{v, x\} \in E'$ for all $x \in X$, $v \in V \cup \{u_r\}$ (enables balancing cardinalities with the set $X$),
- $\{u_r, v\} \in E'$ for all $v \in V$ and $\{u_r, u_R\} \in E'$ (forces $u_r$ to be central and allows to assign $v \in V$ not picked for the dominating set at radius 1 to the corresponding cluster).

Further $E'$ contains all other edges involving the vertices $u_1, \ldots, u_\ell$ and $u_r, u_R$ with weight 3 which clearly makes this set of $\ell + 2$ vertices a vertex cover of the conflict graph for the resulting $(\|\cdot\|_\infty, \mathrm{rad})$-$k$-cluster instance $((V', E'), w_{E'}, n + 2)$.

Assuming the existence of a parameterised approximation with parameter $p_c$ and ratio better than 3 for $(\|\cdot\|_\infty, \mathrm{rad})$-$k$-cluster, this algorithm would be able to build clusters with maximum radius less than 3 for $((V', E'), w_{E'}, n+2)$ if and only if there exists a multicoloured dominating set for $G = (V, E)$; more precisely, the centres of the clusters containing the vertices $u_1, \ldots, u_\ell$ in a $k$-cluster of maximum radius less than 3 for $((V', E'), w_{E'}, n + 2)$ correspond to a multicoloured dominating set for $G = (V, E)$. To understand why this is true, observe that the lower bound $n + 2$ only allows to build at most $\ell + 1$ clusters, while a maximum radius of less than 3 requires at least one cluster with $u_r$ and one cluster with a vertex from $V_i \cup \{u_i\}$ for each $i \in \{1, \ldots, \ell\}$ as centre. A vertex $\bar{V}$ can then only be at distance less than 3 from a centre if the corresponding vertex in the original graph $G$ is adjacent to this centre. As this parameterised approximation has a running-time in $\mathcal{O}^*(g(p_c))$ for some computable function $g$, and $p_c$ is bounded by $\ell + 2$, the given polynomial construction of $((V', E'), w_{E'}, n + 2)$ combined with this assumed parameterised approximation could be used to solve Multicoloured Dominating Set in $\mathcal{O}^*(g(\ell))$, which formally yields:

▶ **Theorem 9.** *There exists no $(3 - \varepsilon)$-approximation for $(\|\cdot\|_\infty, \mathrm{rad})$-$k$-cluster with a running time in $\mathcal{O}^*(g(p_c))$ for any $\varepsilon > 0$ and computable function $g$, unless* $\mathsf{FPT} = \mathsf{W}[2]$.

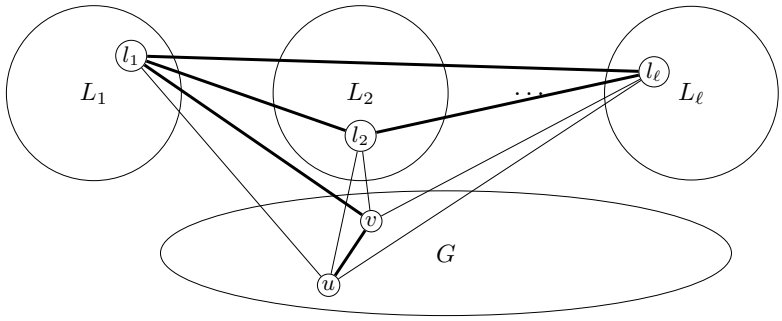



**Figure 3** Illustration of the reduction used for Theorem 10, bold lines represent conflict edges with weight $r+1$, all other edges in the complete graph have weight 1. Vertices $u, v \in V$ are such that $\{u, v\} \in E$, $1 \in L(u)$, $1 \notin L(v)$, $2 \in L(u) \cap L(v)$ and $\ell \notin L(u) \cup L(v)$.

The reduction used to prove Theorem 9 also illustrates why our parameterised approximation for $(\|\cdot\|_\infty, \text{rad})$-$k$-CLUSTER with parameter $p$ does not have a performance ratio better than 3. The situation illustrated in Figure 2 is also a case where our algorithmic strategy fails; if the algorithm chooses $w$ instead of $v$ as central vertex in step (1) (observe that these two are both not in the set $P$), then $q$ is one of the vertices in $P \setminus P'$ (in fact $P' = \emptyset$ is the only correct choice) which has to be put into a suitable cluster without choosing it as centre, which places it at the worst possible distance (3 times the optimum) from the central vertex.

For diameter, we want to investigate the limits of choosing smaller sets of vertices which require partitioning in our parameterised approximations. A graph $G = (V, E)$ is called a *split graph*, if its vertex set can be partitioned into two disjoint sets $A$ and $B$ such that $A$ is an independent set in $G$ and $G[B]$ is the complete graph on vertex set $B$. Especially for the application to ratings to build recommendation systems, it appears that the conflict graph almost has the structure of a split graph: with a small set of users which give unusual ratings and are hence in conflict among each other (set $B$) and with a larger set of more average users (set $A$). This observation raises the question whether it is helpful to turn the conflict graph into a split graph, as this transformation appears to require very little change.

Formally, a *split vertex deletion set* of a graph $G = (V, E)$ is a subset $V'$ of $V$ such that $G[V \setminus V']$ is a split graph. Looking at the previous strategies to lower the parameter from vertex cover to $\mathcal{P}_3$-cover to cluster vertex deletion, the size of a minimum split vertex deletion set appears to be a promising next smaller parameter-choice. Unfortunately, it seems that this parameterisation can not be used for $(\|\cdot\|_\infty, \text{diam})$-$k$-CLUSTER as the following result will show. We will use a similar kind of fpt gap-reduction between $(\|\cdot\|_\infty, \text{diam})$-$k$-CLUSTER parameterised by split deletion set from the problem LIST COLOURING, which for a given graph $G = (V, E)$, colours $\{1, \ldots, \ell\}$ and a set of possible colours for each vertex $v \in V$, given by a list $L(v) \subseteq \{1, \ldots, \ell\}$ for each $v \in V$ asks if there exists a colouring $f \colon V \to \{1, \ldots, r\}$ such that $f(v) \in L(v)$ for all $v \in V$ and $f(v) \neq f(w)$ for all $\{v, w\} \in E$. LIST COLOURING with parameter $\tau(G)$ (vertex cover number of $G$) is W[1]-hard, see [14, 16].

Our reduction from LIST COLOURING to $(\|\cdot\|_\infty, \text{diam})$-$k$-CLUSTER constructs a clustering instance $((V', E'), w_{E'}, n+2)$ as sketched in Figure 3, where $G$ is the input graph for LIST COLOURING and each $L_i$, is a set of $n+2$ new vertices. Observe that a vertex cover for $G$ is a split vertex deletion set of the conflict graph for this instance. It can be shown that a "yes"-instance for LIST COLOURING corresponds to a $k$-cluster of maximum diameter 1 while a "no"-instance yields a diameter of at least $r+1$.

This would enable solving List Colouring in $\mathcal{O}^*(g(\tau))$ if there existed a parameterised approximation with ratio $r$ and cluster vertex deletion set as parameter for $(\|\cdot\|_\infty, \text{diam})$-$k$-cluster, hence:

▶ **Theorem 10.** *There exists no constant factor approximation for* $(\|\cdot\|_\infty, \text{diam})$-$k$-cluster *with a running time in* $\mathcal{O}^*(f(p_s))$*, for any computable function* $f$*, unless* $\mathsf{FPT} = \mathsf{W}[1]$.

## 6 Conclusions

As the exponential time hypothesis implies $\mathsf{FPT} \neq \mathsf{W}[1]$ by [8], the negative results in this paper especially hold assuming the *exponential time hypothesis* [19]. Both reductions used to show these create instances of $(\|\cdot\|, f)$-$k$-cluster with large values for $k$. In most applications however, $k$ is a fixed, not too large integer, which raises the question whether an additional parameterisation by $k$ (additional to the number of conflicts) would help overcome the negative results. For the greedy strategies used for the parameterised approximations in this paper, it is not clear how $k$ could be included in a useful way. An improvement with parameterisation by both conflicts and $k$ probably means using a very different algorithmic approach. The gap between our positive results and the presented lower bounds further suggests room for improvement. Stronger lower bounds seem to require new techniques for reductions which consider both parameterisation and approximation.

We would like to mention that a relaxation of the cardinality constraint, i.e., asking for an approximate solution in the sense that this partition is allowed to contain clusters with only $\alpha k$ vertices, for some factor $0 < \alpha \leq 1$, does not help with the problems caused by conflicts. In fact, the inapproximability for general non-metric instances from [1] holds for fixed values of $k$ with $k \geq 3$, which means that this kind of additional cardinality relaxation either yields a polynomial time solvable problem, in case $\alpha k \leq 2$, or a problem with the same approximation hardness.

One other aspect we did not consider here is the optimality of the asymptotic running times of our positive results. Techniques for such more fine-grained considerations need a more careful analysis. A concrete question in this regard is whether it is possible to improve the parameterised approximations for $(\|\cdot\|_\infty, \text{diam})$-$k$-cluster to only require single-exponential time. In this regard it would be very interesting to see if slightly superexponential lower bounds as shown in [21] can be proven for a 2-approximation of $(\|\cdot\|_\infty, \text{diam})$-$k$-cluster with parameter $p$. For improvement on the running time on the other hand, it might be interesting to analyse the algorithms with a closer look at the enumerations of the partitions of $P$. We always estimated this with the Bell number although we only consider partitions with specific properties (those which are possible in a clustering of maximum diameter $D$) which in a sense relate to colourings of the conflict graph. It might be possible to enumerate the relevant partitions of $P$ more efficiently with the help of colouring strategies.

Aside from the problems discussed here, there are many other related clustering-type problems which exhibit similar difficulties with violations of the triangle inequality. The parameterisation by conflicts and related parameters might provide a useful way to approach these problems as well.

## References

1 F. N. Abu-Khzam, C. Bazgan, K. Casel, and H. Fernau. Clustering with Lower-Bounded Sizes - A General Graph-Theoretic Framework. *Algorithmica*, 80(9):2517–2550, 2018.

2 N. Alon, R. Yuster, and U. Zwick. Color Coding. In M.-Y. Kao, editor, *Encyclopedia of Algorithms*. Springer, 2008.

3 G. Ausiello. *Complexity and approximation: combinatorial optimization problems and their approximability properties*. Springer, 1999.

4 C. Bazgan, M. Chopin, A. Nichterlein, and F. Sikora. Parameterized Inapproximability of Target Set Selection and Generalizations. *Computability*, 3(2):135–145, 2014.

5 D. Berend and T. Tassa. Improved bounds on Bell numbers and on moments of sums of random variables. *Probability and Mathematical Statistics*, 30(2):185–205, 2010.

6 B. Bollobás. *Modern Graph Theory*, volume 184 of *Graduate texts in mathematics*. Springer, 1998.

7 A. Boral, M. Cygan, T. Kociumaka, and M. Pilipczuk. A Fast Branching Algorithm for Cluster Vertex Deletion. *Theory Comput. Syst.*, 58(2):357–376, 2016.

8 J. Chen, X. Huang, I. A. Kanj, and G. Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006. doi:10.1016/j.jcss.2006.04.007.

9 J. Chen, I. A. Kanj, and G. Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40–42):3736–3756, 2010. doi:10.1016/j.tcs.2010.06.026.

10 I. Dinur and D. Steurer. Analytical approach to parallel repetition. In D. B. Shmoys, editor, *Symposium on Theory of Computing, STOC*, pages 624–633. ACM, 2014.

11 R. G. Downey and M. Fellows. Fixed parameter tractability and completeness. *Congressus Numerantium*, 87:161–187, 1992.

12 R. G. Downey and M. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

13 C. Elkan. Using the Triangle Inequality to Accelerate k-Means. In T. Fawcett and N. Mishra, editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 147–153. AAAI Press, 2003.

14 M. Fellows, F. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh, S. Szeider, and C. Thomassen. On the complexity of some colorful problems parameterized by treewidth. *Information and Computation*, 209(2):143–153, 2011.

15 M. Fellows, D. Hermelin, F. A. Rosamond, and S. Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410(1):53–61, 2009.

16 J. Fiala, P. Golovach, and J. Kratochvíl. Parameterized complexity of coloring problems: Treewidth versus vertex cover. *Theor. Comput. Sci.*, 412(23):2513–2523, 2011. doi:10.1016/j.tcs.2010.10.043.

17 J. Guo, F. Hüffner, and R. Niedermeier. A Structural View on Parameterizing Problems: Distance from Triviality. In R. G. Downey, M. Fellows, and F. K. H. A. Dehne, editors, *Parameterized and Exact Computation, First International Workshop, IWPEC 2004, Bergen, Norway, September 14-17, 2004, Proceedings*, volume 3162 of *LNCS*. Springer, 2004.

18 D. S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22(1):148–162, 1982.

19 R. Impagliazzo and R. Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

20 S. Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013.

21 D. Lokshtanov, D. Marx, and S. Saurabh. Slightly Superexponential Parameterized Problems. *SIAM J. Comput.*, 47(3):675–702, 2018.

22 D. Marx. Parameterized complexity and approximation algorithms. *The Computer Journal*, 51(1):60–78, 2008.

23 J. B. Schafer, D Frankowski, J. L. Herlocker, and S. Sen. Collaborative Filtering Recommender Systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web, Methods and Strategies of Web Personalization*, volume 4321 of *LNCS*, pages 291–324. Springer, 2007.

24 M. Xiao and S. Kou. Kernelization and Parameterized Algorithms for 3-Path Vertex Cover. In *Theory and Applications of Models of Computation - 14th Annual Conference, TAMC 2017, Bern, Switzerland, April 20-22, 2017, Proceedings*, pages 654–668, 2017.

# Counting Induced Subgraphs: A Topological Approach to #W[1]-hardness

**Marc Roth**
Saarland University and Cluster of Excellence (MMCI), Saarbrücken, Germany
mroth@mmci.uni-saarland.de

**Johannes Schmitt**[1]
ETH Zürich, Zürich, Switzerland
johannes.schmitt@math.ethz.ch

## Abstract

We investigate the problem #IndSub($\Phi$) of counting all induced subgraphs of size $k$ in a graph $G$ that satisfy a given property $\Phi$. This continues the work of Jerrum and Meeks who proved the problem to be #W[1]-hard for some families of properties which include, among others, (dis)connectedness [JCSS 15] and even- or oddness of the number of edges [Combinatorica 17]. Using the recent framework of graph motif parameters due to Curticapean, Dell and Marx [STOC 17], we discover that for monotone properties $\Phi$, the problem #IndSub($\Phi$) is hard for #W[1] if the reduced Euler characteristic of the associated simplicial (graph) complex of $\Phi$ is non-zero. This observation links #IndSub($\Phi$) to Karp's famous Evasiveness Conjecture, as every graph complex with non-vanishing reduced Euler characteristic is known to be evasive. Applying tools from the "topological approach to evasiveness" which was introduced in the seminal paper of Khan, Saks and Sturtevant [FOCS 83], we prove that #IndSub($\Phi$) is #W[1]-hard for every monotone property $\Phi$ that does not hold on the Hamilton cycle as well as for some monotone properties that hold on the Hamilton cycle such as being triangle-free or not $k$-edge-connected for $k > 2$. Moreover, we show that for those properties #IndSub($\Phi$) can not be solved in time $f(k) \cdot n^{o(k)}$ for any computable function $f$ unless the Exponential Time Hypothesis (ETH) fails. In the final part of the paper, we investigate non-monotone properties and prove that #IndSub($\Phi$) is #W[1]-hard if $\Phi$ is any non-trivial modularity constraint on the number of edges with respect to some prime $q$ or if $\Phi$ enforces the presence of a fixed isolated subgraph.

**2012 ACM Subject Classification** Theory of computation → Parameterized complexity and exact algorithms, Theory of computation → Problems, reductions and completeness, Mathematics of computing → Combinatorics, Mathematics of computing → Graph theory, General and reference → General literature

**Keywords and phrases** counting complexity, Euler characteristic, homomorphisms, parameterized complexity, simplicial complexes

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.24

**Related Version** `https://arxiv.org/abs/1807.01920`

**Acknowledgements** We are very grateful to Cornelius Brand, Radu Curticapean, Holger Dell and Johannes Lengler for helpful advice and fruitful discussions. Furthermore we wish to point out that Sloane's OEIS [1] was very useful in the course of this work.

[1] The second author was supported by the grants SNF-200020162928 and ERC-2017-AdG-786580-MACI.

© Marc Roth and Johannes Schmitt;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 24; pp. 24:1–24:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

In their work about the parameterized complexity of counting problems [11] Flum and Grohe introduced the parameterized analogue of the theory of computational counting as layed out by Valiant in his seminal paper about the complexity of computing the permanent [28]. Since then parameterized counting has evolved into a well-studied subfield of parameterized complexity theory. In particular, there has been remarkable progress in the classification of problems that require to count small structures in large graphs. It turned out that many families of such counting problems allow so-called dichotomy results, that is, every problem in the family is either fixed-parameter tractable or hard for the class #W[1] — the counting equivalent of W[1]. One result of that kind is the dichotomy for counting homomorphisms [10, 13]. Here one is given a graph $H$ from a class of graphs $\mathcal{H}$ and an arbitrary graph $G$ and the task is to compute the number of homomorphisms from $H$ to $G$. When parameterized by $|H|$ this problem is fixed-parameter tractable if there exists a constant upper bound on the treewidth of graphs in $\mathcal{H}$ and #W[1]-hard otherwise. Similar results have been shown for the problems of counting subgraph embeddings [9], induced subgraphs [7] and locally injective homomorphisms [26]. As results like Ladner's theorem (see e.g. [19, 2]) rule out such dichotomies in the general case one might ask why all of the above problems indeed do allow such complexity classifications. The answer to that question was given very recently by Curticapean, Dell and Marx [8] who proved that, in some sense, all of those problems are the same. To this end, they defined the problem of computing linear combinations of homomorphisms which they called *graph motif parameters*. Here one is given a graph $G$ and a function $a$ of finite support that maps graphs to rational numbers and the task is to compute

$$\sum_H a(H) \cdot \#\mathsf{Hom}(H, G)\,, \tag{1}$$

where the sum is over all (unlabeled) simple graphs and $\#\mathsf{Hom}(H, G)$ denotes the number of homomorphisms from $H$ to $G$. A result of Lovász (see e.g. Chapt. 5 in [20]) implies that the number of subgraph embeddings $\#\mathsf{Emb}(H, G)$ as well as the number of induced subgraphs $\#\mathsf{IndSub}(H, G)$ can be expressed as a linear combination of homomorphisms. In case of embeddings the result states that

$$\#\mathsf{Emb}(H, G) = \sum_{\rho \geq \emptyset} \mu(\emptyset, \rho) \cdot \#\mathsf{Hom}(H/\rho, G)\,, \tag{2}$$

where the sum is over the partition lattice of the vertices of $H$, $\mu$ is the Möbius function over that lattice and $H/\rho$ is obtained from $H$ by identifying vertices along $\rho$. Now, intuitively, the main result of Curticapean, Dell and Marx states that computing a linear combination of homomorphisms is precisely as hard as computing the hardest term in the linear combination. Together with the dichotomy for counting homomorphisms this implies that every problem expressible as a linear combination of homomorphisms is either fixed-parameter tractable or #W[1]-hard. The purpose of this work is a thorough investigation of the problem of counting induced subgraphs through the lense of the framework of graph motif parameters. Chen, Thurley and Weyer [7] proved that the problem $\#\mathsf{IndSub}(\mathcal{H})$ of, given a graph $H \in \mathcal{H}$ and an arbitrary graph $G$, computing $\#\mathsf{IndSub}(H, G)$, is fixed-parameter tractable when parameterized by $|H|$ if and only if $\mathcal{H}$ is finite and #W[1]-hard otherwise. While this result resolves the parameterized complexity of problems such as computing the number of induced

cycles of length $k$,[2] it is not applicable to problems such as computing the number of connected induced subgraphs of size $k$. For this reason, Jerrum and Meeks [15, 14, 22, 16] introduced and studied the following problem: Let $\Phi$ be a (computable) graph property, then the problem $\#\mathsf{IndSub}(\Phi)$ asks, given a graph $G$ and a natural number $k$, to count all induced subgraphs of size $k$ in $G$ that satisfy $\Phi$.[3] In other words, the goal is to compute $\sum_{H\in\Phi_k} \#\mathsf{IndSub}(H,G)$, where $\Phi_k$ is the set of all (unlabeled) graphs with $k$ vertices that satisfy $\Phi$. The generality of $\#\mathsf{IndSub}(\Phi)$ allows to count almost arbitrary substructures in graphs, subsuming lots of parameterized counting problems that have been studied before, and hence the problem deserves a thorough complexity analysis with respect to the property $\Phi$. Jerrum and Meeks proved it to be #W[1]-hard for the property of connectivity [15], for the property of having an even (or odd) number of edges [16] as well as for some other properties (see Section 1.2). As noted in [8], the theory of graph motif parameters immediately implies that for every property $\Phi$, the problem $\#\mathsf{IndSub}(\Phi)$ is either fixed-parameter tractable or #W[1]-hard. However, for a concrete $\Phi$ it might be highly non-trivial to prove for which graphs $H$ the term $\#\mathsf{Hom}(H,G)$ is contained with a non-zero coefficient in the equivalent expression as linear combination of homomorphisms. Unfortunately, this is precisely what needs to be done to find out whether $\#\mathsf{IndSub}(\Phi)$ is fixed-parameter tractable or #W[1]-hard. In our investigation we will focus on the coefficient of $\#\mathsf{Hom}(K_k,G)$, where $K_k$ is the complete graph on k vertices. We will see that for monotone properties, non-zeroness of this coefficient is sufficient for the property to be evasive.

## 1.1 Results and techniques

The framework of graph motif parameters [8] implies that for every property $\Phi$ and natural number $k$, there exists a function $a$ from graphs to rationals such that for all graphs $G$ it holds that $\sum_{H\in\Phi_k} \#\mathsf{IndSub}(H,G) = \sum_H a(H)\cdot\#\mathsf{Hom}(H,G)$. Our most important observation is concerned with the coefficient of the complete graph.

▶ **Theorem 1** (Intuitive version). *Let $\Phi$, $k$ and $a$ be as above. Then it holds that $a(K_k)=0$ if and only if $\sum_{A\in\mathsf{E}_k^\Phi}(-1)^{\#A}=0$, where $\mathsf{E}_k^\Phi$ is the set of all edge-subsets $A$ of the labeled complete graph with $k$ vertices such that $\Phi$ holds for the graph induced by $A$.*

We will provide an introduction to the theory of graph motif parameters as well as the proof of Theorem 1 in Section 3. In Section 4 we turn towards monotone properties, i.e., properties that are closed under the removal of edges, and observe that in this case the term $\sum_{A\in\mathsf{E}_k^\Phi}(-1)^{\#A}$ is equal to the reduced Euler characteristic $\hat{\chi}$ of the simplicial graph complex $\Delta(\Phi_k)$ of $\Phi_k$. Recall that a simplicial complex is a set of sets that is closed under taking non-empty subsets and a simplicial graph complex is a simplicial complex whose elements are subsets of the edges of the labeled complete graph. We will make this formal in Section 2. As computing the number of cliques of size $k$ is #W[1]-complete [11] and computing a linear combination of homomorphisms is precisely as hard as computing its hardest term [8], the complexity of $\#\mathsf{IndSub}(\Phi)$ is resolved whenever $\Phi$ is monotone and the reduced Euler characteristic of $\Delta(\Phi_k)$ is known to be non-zero for infinitely many $k$.

[2] This problem can be equivalently expressed as $\#\mathsf{IndSub}(\mathcal{C})$, where $C$ is the class of all cycles.

[3] Strictly speaking, $\#\mathsf{IndSub}(\Phi)$ is the unlabeled version of $p$-#INDUCED SUBGRAPH WITH PROPERTY($\Phi$), both of which have been introduced in [15]. However, as Jerrum and Meeks point out, those problems are equivalent for graph properties that are invariant under relabeling of vertices (see Section 1.3.1 in [15]), which is true for all properties we are concerned with in this work.

▶ **Corollary 2.** *Let $\Phi$ be a monotone graph property such that $\hat{\chi}(\Delta(\Phi_k)) \neq 0$ for infinitely many $k$. Then the problem $\#\mathsf{IndSub}(\Phi)$ is $\#\mathrm{W}[1]$-hard.*

We will also obtain a matching lower bound under the Exponential Time Hypothesis (ETH) if the set of all $k$ for which $\hat{\chi}(\Delta(\Phi_k)) \neq 0$ satisfies a certain density condition (see Section 2).

The (reduced) Euler characteristic is well-understood for many graph complexes. For example, Chapt. 10.5 in the book of Jonsson [17] provides a large list of graph properties, each of whose reduced Euler characteristics are non-zero infinitely often. For those properties Corollary 2 is hence applicable. The study of the (reduced) Euler characteristic is, among others, motivated by Karp's famous evasiveness conjecture, stating that every non-trivial monotone graph property is evasive. A property $\Phi_k$ on graphs with $k$ vertices is evasive if every decision-tree algorithm that branches on the presence or absence of edges of a given graph $G$ needs to perform $\binom{k}{2}$ branches in the worst case to correctly decide whether $\Phi_k$ holds on $G$. We refer the reader to Miller's survey [23] for a detailed introduction. While the conjecture is still unresolved, there has been a major breakthrough due to Khan, Saks and Sturtevant [18] who proved the conjecture to be true whenever $k$ is a prime power. Their paper "A Topological Approach to Evasiveness" was, as the name suggests, the first one to use topological tools such as fixed-point complexes under group operations to prove evasiveness of a given graph complex. One of their results reads as follows:

▶ **Theorem 3** ([18][4])**.** *Let $\Phi_k$ be a non-trivial monotone graph property. If $\hat{\chi}(\Delta(\Phi_k)) \neq 0$ then $\Phi_k$ is evasive.*

Unfortunately, the converse of this theorem does not hold. A counterexample is given in Chapt. 10.6 in Jonsson's book [17]. Nevertheless it turns out that some tools of the topological approach to evasiveness suit as well for a topological approach to #W[1]-hardness of $\#\mathsf{IndSub}(\Phi)$. The most important ingredient in our proofs is a theorem that goes back to Smith [27] (see also [24] and Chapt. 3 in [3]), intuitively stating that, given a simplicial complex $\Delta$ and a $p$-power group $\Gamma$ for some prime $p$ that operates on the ground set of $\Delta$ in a way that leaves the complex stable, it holds that $\hat{\chi}(\Delta) \equiv \hat{\chi}(\Delta^\Gamma) \mod p$, where $\Delta^\Gamma$ is the fixed-point complex of $\Delta$ with respect to $\Gamma$. Again, this will be made formal in Section 2. Applying this theorem to a rather simple group, we will be able to prove our main result which reads as follows:

▶ **Theorem 4.** *Let $\Phi$ be a non-trivial monotone graph property. Then $\#\mathsf{IndSub}(\Phi)$ is $\#\mathrm{W}[1]$-hard and, assuming ETH, can not be solved in time $f(k) \cdot n^{o(k)}$ for any computable function $f$ if at least one of the following conditions is true*

1. *$\Phi$ is false for odd cycles.*
2. *$\Phi$ is true for odd anti-holes.*
3. *There exists $c \in \mathbb{N}$ such that for all $H$ it holds that $\Phi(H) = 1$ if and only if $H$ is not $c$-edge-connected.*
4. *There exists a graph $F$ such that for all $H$ it holds that $\Phi(H) = 1$ if and only if there is no homomorphism from $F$ to $H$.*

We remark that Rivest and Vuillemin [25] implicitly proved that the reduced Euler characteristic of a graph complex does not vanish if the first condition is true. Furthermore

[4] In fact, Khan, Saks and Sturtevant show that any non-evasive complex is collapsible. However, every collapsible complex has a reduced Euler characteristic of zero (see e.g. [21]). Hence the contraposition implies the theorem as stated.

we note that (non-)triviality of a monotone property needs to be defined with some care to exclude properties that depend only on the number of vertices of a graph. Details are given in Section 4. Examples of properties that satisfy the first condition are the ones of being bipartite, cycle-free, disconnected and non-hamiltonian. One example for the second condition is the property of having a chromatic number smaller or equal than half of the size of the graph (rounded up) and the third condition includes the properties of exclusion of a fixed complete graph as a subgraph.

In Section 5 we turn to non-monotone properties and illustrate that Theorem 1 itself is a useful criterion when it comes to establishing #W[1]-hardness of $\#\mathsf{IndSub}(\Phi)$. In particular, we will prove hardness whenever $\Phi$ is a non-trivial modularity constraint on the number of edges with respect to some prime or if $\Phi$ enforces the presence of a fixed isolated subgraph.

## 1.2 Related work

Jerrum and Meeks introduced and studied the problem $\#\mathsf{IndSub}(\Phi)$ for the following properties. In [15] they prove the problem to be #W[1]-hard if $\Phi$ is the property of being connected, which immediately follows from Theorem 4 as $\#\mathsf{IndSub}(\Phi)$ and $\#\mathsf{IndSub}(\neg\Phi)$ are equivalent[5] and the property of being disconnected is monotone and false for every cycle. In [16] hardness is established for the property of having an even (or odd) number of edges, which follows from Theorem 1 as every term in the sum $\sum_{A\in\mathsf{E}_k^\Phi}(-1)^{\#A}$ will have the same sign. In [14] Jerrum and Meeks prove the problem to be #W[1]-hard whenever the edge-density of graphs in $\Phi_k$ grows asymptotically slower than $k^2$ and in [22] Meeks shows that whenever $\Phi$ is co-monotone, i.e., $\neg\Phi$ is monotone, and the set of (edge-)minimal elements of $\Phi$ has unbounded treewidth, the problem is hard as well. Those latter results are independent from ours in the sense that ours do not imply theirs and vice versa. One example of a property whose hardness does not follow from the results of Jerrum and Meeks is bipartiteness: The edge-densities of both, the properties of being bipartite and not bipartite grow asymptotically as fast as $k^2$ and the edge-minimal non-bipartite graphs are odd cycles, hence having treewidth 2. However hardness for the property of being bipartite follows from the first condition of Theorem 4 as odd cycles are not bipartite. Moreover, we point out that Meeks' reduction in [22] uses the Excluded Grid Theorem and hence does not imply a tight lower bound under ETH. Finally we remark that due to space constraints some proofs are only sketched or omitted and we refer the interested reader to the related version of this paper.

# 2 Preliminaries

First we will introduce some basic notions. Given a finite set $S$, we write $\#S$ for the cardinality of $S$. We say that a set $\mathcal{K}$ of natural numbers is *dense* if there exists a constant $c > 0$ such that for all but finitely many $k \in \mathbb{N}$ there exists $k' \in \mathcal{K}$ such that $k \leq k' \leq c \cdot k$. Given a function $a$ from a (not necessarily finite) set $S$ to rational numbers, the *support* of $a$ is the set of elements $s \in S$ such that $a(s) \neq 0$. We write $\mathsf{supp}(a)$ for the support of $a$. Given a natural number $k$, we write $[k]$ for the set $\{0, \ldots, k-1\}$. Given a finite group $\Gamma$ of order $p^s$ for some prime $p$ and natural number $s$, we say that $\Gamma$ is a *$p$-power group*.

[5] We just need to substract one from $\binom{n}{k}$ to get the other.

### Graph theory

In this work all graphs are considered to be undirected, simple and to not contain self-loops. Given a graph $G$ we write $V(G)$ for the vertices and $E(G)$ for the edges of $G$. We denote the complete graph on $\ell$ vertices as $K_\ell$. A *labeled* graph is a graph $G$ with a bijective labeling $\ell : V(G) \to [\#V(G)]$ of the vertices and we will sloppily identify vertices with their labels. A *subgraph* of $G$ is a graph obtained from $G$ by deleting vertices (including incident edges) and/or edges. Given a subset $S \subseteq V(G)$, the *induced subgraph* $G[S]$ is the graph with vertices $S$ and edges $E(G) \cap S^2$. A *homomorphism* from a graph $H$ to a graph $G$ is a function $\varphi : V(H) \to V(G)$ that is edge-preserving, i.e. for every edge $\{u, v\} \in E(H)$ it holds that $\{\varphi(u), \varphi(v)\} \in E(G)$. We write $\mathsf{Hom}(H, G)$ for the set of all homomorphisms from $H$ to $G$. A homomorphism $\varphi$ is called an *embedding* if $\varphi$ is injective. We write $\mathsf{Emb}(H, G)$ for the set of all embeddings from $H$ to $G$. An *isomorphism* from a graph $H$ to a graph $G$ is a bijective homomorphism. We say that $H$ and $G$ are *isomorphic*, denoted by $H \cong G$, if such an isomorphism exists and we denote $\mathcal{G}$ as the set of all (isomorphism types of) graphs. An *automorphism* of a graph $H$ is an isomorphism from $H$ to $H$. We write $\mathsf{Aut}(H)$ for the set of all automorphisms of $H$. An embedding $\varphi$ from $H$ to $G$ is called a *strong embedding* if for all vertices $u, v \in V(H)$ it holds that $\{u, v\} \in E(H) \Leftrightarrow \{\varphi(u), \varphi(v)\} \in E(G)$. We write $\mathsf{StrEmb}(H, G)$ for the set of all strong embeddings from $H$ to $G$. Given graphs $H$ and $G$, we write $\mathsf{Sub}(H, G)$ for the set of all subgraphs of $G$ that are isomorphic to $H$ and $\mathsf{IndSub}(H, G)$ for the set of all induced subgraphs of $G$ that are isomorphic to $H$.

▶ **Fact 5.** *For all graphs $H$ and $G$ it holds that $\#\mathsf{Emb}(H, G) = \#\mathsf{Sub}(H, G) \cdot \#\mathsf{Aut}(H)$ and that $\#\mathsf{StrEmb}(H, G) = \#\mathsf{IndSub}(H, G) \cdot \#\mathsf{Aut}(H)$.*

A *graph property* $\Phi$ is a function from graphs to $\{0, 1\}$ such that $\Phi(G) = \Phi(G')$ whenever $G$ and $G'$ are isomorphic. We say that $\Phi$ holds on $G$ if $\Phi(G) = 1$ and we are not going not distinguish between $\Phi$ and the set of graphs for which $\Phi$ holds as this will be clear from the context. We write $\Phi_k$ for the set of all (isomorphism types of) graphs with $k$ vertices on which $\Phi$ holds. For technical reasons we define $\mathsf{E}_k^\Phi$ to be the set of all edge-subsets $A$ of the labeled complete graph with $k$ vertices such that $\Phi$ holds on the graph with the same vertices and edges $A$. A graph property is called *monotone* if it is closed under the removal of edges, that is, if $G'$ is obtained from $G$ by removing edges and $\Phi$ holds for $G$, then $\Phi$ holds for $G'$ as well. A property is called *co-monotone* if its complement is monotone[6].

### Transformation groups and simplical (graph) complexes

Let $\Omega$ be a finite set. A *simplicial complex* over the ground set $\Omega$ is a set $\Delta$ of non-empty subsets of $\Omega$ such that whenever a set $A$ is contained in $\Delta$ and $A'$ is a non-empty subset of $A$, then $A'$ is contained in $\Delta$ as well. An element $A$ of $\Delta$ is called a *simplex* and the *dimension* of $A$, denoted as $\mathsf{dim}(A)$, is defined to be $\#A - 1$. The *Euler characteristic* $\chi$ of a simplical complex $\Delta$ is defined to be $\chi(\Delta) := \sum_{i \geq 0} (-1)^i \cdot \#\{A \in \Delta \mid \mathsf{dim}(A) = i\}$ and the *reduced Euler characteristic* of $\Delta$ is defined to be $\hat{\chi}(\Delta) := 1 - \chi(\Delta)$.

▶ **Fact 6.** $\hat{\chi}(\Delta) = \sum_{i \geq 0} (-1)^i \cdot \#\{A \in \Delta \cup \{\emptyset\} \mid \#A = i\}$.

Given a simplicial complex $\Delta$ and a finite group $\Gamma$ that operates on the ground set $\Omega$ of $\Delta$, we say that $\Delta$ is a $\Gamma$-*simplicial complex* if the induced action of $\Gamma$ on subsets of $\Omega$

[6] We remark that in some literature, e.g. [22, 25], the notions of monotonicity and co-monotonicity are reversed.

preserves $\Delta$. More precisely, if $A \in \Delta$ and $g \in \Gamma$ then the set $g \triangleright A := \{g \triangleright a \mid a \in A\}$ is contained in $\Delta$ as well. If this is the case we can define the *fixed-point complex* $\Delta^\Gamma$ as follows. Let $\mathcal{O}_1, \ldots, \mathcal{O}_k$ be the orbits of $\Omega$ with respect to $\Gamma$. Then

$$\Delta^\Gamma := \left\{ S \subseteq \{1, \ldots, k\} \;\middle|\; S \neq \emptyset \wedge \bigcup_{i \in S} \mathcal{O}_i \in \Delta \right\}$$

The following theorem, which is due to Smith [27] (see also [24] and Chapt. 3 in [3]), will be of crucial importance in Section 4.

▶ **Theorem 7.** *Let $\Gamma$ a group of order $p^s$ for some prime $p$ and natural number $s$ and let $\Delta$ be a $\Gamma$-simplicial complex. Then $\chi(\Delta) \equiv \chi(\Delta^\Gamma) \mod p$ and hence $\hat{\chi}(\Delta) \equiv \hat{\chi}(\Delta^\Gamma) \mod p$.*

Now let $\Phi$ be a monotone graph property. Then $\mathsf{E}_k^\Phi \setminus \{\emptyset\}$ is a simplicial complex, called the *graph complex* of $\Phi_k$. The ground set is the set of all edges of the complete labeled graph $K_k$ on $k$ vertices and we emphasize $\mathsf{E}_k^\Phi \setminus \{\emptyset\}$ being a simplicial complex for monotone properties by denoting it as $\Delta(\Phi_k)$. If $\Gamma$ is any permutation group on the set $[k]$ then $\Gamma$ induces a group operation on the ground set of $\Phi_k$, i.e. the edges of the labeled complete graph of size $k$, by relabeling the vertices according to the group element. In particular, $\Delta(\Phi_k)$ is a $\Gamma$-simplicial complex as $\Phi_k$ is invariant under relabeling of vertices. We write $\Delta^\Gamma(\Phi_k)$ for the fixed-point complex $\Delta(\Phi_k)^\Gamma$ under this operation.

## Parameterized (counting) complexity

We will follow the definitions of Chapt. 14 of the textbook of Flum and Grohe [12]. A *parameterized counting problem* is a function $F : \{0,1\}^* \to \mathbb{N}$ together with a computable parameterization $\kappa : \{0,1\}^* \to \mathbb{N}$. $(F, \kappa)$ is called *fixed-parameter tractable* (FPT) if there exists a deterministic algorithm $\mathbb{A}$ and a computable function $f$ such that $\mathbb{A}$ computes $F$ in time $f(\kappa(x)) \cdot |x|^{O(1)}$ for any input $x$. Given two parameterized counting problems $(F, \kappa)$ and $(F', \kappa')$, a *parameterized Turing reduction* from $(F, \kappa)$ to $(F', \kappa')$ is an FPT algorithm w.r.t. $\kappa$ that has oracle access to $F'$ and that on input $x$ computes $F(x)$ with the additional restriction that there exists a computable function $g$ such that for any oracle query $y$ it holds that $\kappa'(y) \leq g(\kappa(x))$. We write $(F, \kappa) \leq_P^T (F', \kappa')$.

The parameterized counting problem #Clique aks, given a graph $G$ and a natural number $k$, to compute the number of complete subgraphs of size $k$ in $G$ and the problem is parameterized by $k$. The class #W[1] contains all problems $(F, \kappa)$ such that $(F, \kappa) \leq_P^T$ #Clique holds. Given a recursively enumerable class of graphs $\mathcal{H}$ the problems #Hom($\mathcal{H}$), #Emb($\mathcal{H}$), #Sub($\mathcal{H}$), #StrEmb($\mathcal{H}$) and #IndSub($\mathcal{H}$) ask, given a graph $H \in \mathcal{H}$ and an arbitrary (unlabeled) graph $G$, to compute #Hom$(H, G)$, #Emb$(H, G)$, #Sub$(H, G)$, #StrEmb$(H, G)$ and #IndSub$(H, G)$, respectively. All problems are parameterized by $|H|$. As stated in the introduction, there are dichotomy results for each of the aforementioned problems [10, 13, 9, 7]. We emphasize on the following, which is crucial for the framework of graph motif parameters.

▶ **Theorem 8** ([10, 13])**.** *The problem #Hom($\mathcal{H}$) is fixed-parameter tractable if there exists $b \in \mathbb{N}$ such that the treewidth*[7] *of every graph in $\mathcal{H}$ is bounded by $b$. Otherwise, the problem is #W[1]-hard.*

[7] We remark that the graph parameter of treewidth is not used explicitely in this work. Hence we refer the reader e.g. to Chapt. 11 in [12].

In this work we deal with a generalization of $\#\mathsf{IndSub}(\mathcal{H})$. Let $\Phi$ be a computable graph property. The problem $\#\mathsf{IndSub}(\Phi)$ asks, given a graph $G$ and a number $k \in \mathbb{N}$ to compute $\sum_{H \in \Phi_k} \#\mathsf{IndSub}(H, G)$. The parameter is $k$.

## 3 Graph motif parameters

In [8] Curticapean, Dell and Marx generalized the problem $\#\mathsf{Hom}(\mathcal{H})$ to linear combinations, called *graph motif parameters*. To this end, let $\mathcal{A}$ be a recursively enumerable set of functions $a : \mathcal{G} \to \mathbb{Q}$ such that $\mathsf{supp}(a)$ is finite. Then the problem $\#\mathsf{Hom}(\mathcal{A})$ asks, given $a \in \mathcal{A}$ and a graph $G$, to compute $\sum_{H \in \mathcal{G}} a(H) \cdot \#\mathsf{Hom}(H, G)$. The parameter is the description length of $a$, denoted by $|a|$. Their main result states that computing a linear combination of homomorphisms is as hard as computing all terms with non-zero coefficients:

▶ **Theorem 9** ([8]). *There exists a deterministic algorithm that, on input a function $a : \mathcal{G} \to \mathbb{Q}$ with finite support, a graph $F \in \mathsf{supp}(a)$ and a graph $G$ and given oracle access to the function $G \mapsto \sum_{H \in \mathcal{G}} a(H) \cdot \#\mathsf{Hom}(H, G)$, computes $\#\mathsf{Hom}(F, G)$ in time $g(|a|) \cdot \#V(G)^{O(1)}$ and additionally satisfies that the number of vertices of every graph $G'$ for which the oracle is queried is of size bounded by $\max_{H \in \mathsf{supp}(a)} \#V(H) \cdot \#V(G)$.*

Using this result, Curticapean, Dell and Marx proved that the problem $\#\mathsf{Hom}(\mathcal{A})$ is fixed-parameter tractable if there is a constant upper bound on the treewidth of all graphs that occur in the support of a function $a \in \mathcal{A}$, and #W[1]-hard otherwise. After that they showed that all of the problems $\#\mathsf{Emb}(\mathcal{H})$, $\#\mathsf{StrEmb}(\mathcal{H})$, ... are expressible as linear combinations of homomorphisms, immediately implying the existence of dichotomy results for those problems. However, establishing a concrete criterion for fixed-parameter tractability requires to find out which graphs are contained in the support of a function $a$ when the problem is translated to a linear combination of homomorphisms, and this can be highly non-trivial.

In what follows, we will establish a concrete criterion for properties $\Phi$ such that the coefficient of $K_k$ is non-zero when the function $G \mapsto \sum_{H \in \Phi_k} \#\mathsf{IndSub}(H, G)$ is translated to a linear combination of homomorphisms. This is motivated by the fact that, in this case, Theorem 9 allows us to compute the number $\#\mathsf{Hom}(K_k, G)$ which is equal to $k!$ times the number of cliques of size $k$ in $G$. As $\#\mathsf{Clique}$ can not be solved in time $f(k) \cdot \#V(G)^{o(k)}$ for any computable function $f$ under the Exponential Time Hypothesis [5, 6], we will not only obtain #W[1]-hardness but also a tight lower bound under the lense of fine-grained complexity theory.

▶ **Theorem 10** (Theorem 1 restated). *Let $\Phi$ be a graph property, let $k$ be a non-zero natural number and let $a : \mathcal{G} \to \mathbb{Q}$ be the function such that for all graphs $G$ the following is true*

$$\sum_{H \in \Phi_k} \#\mathsf{IndSub}(H, G) = \sum_{H} a(H) \cdot \#\mathsf{Hom}(H, G)\,.$$

*Then $|k! \cdot a(K_k)| = |\sum_{A \in \mathsf{E}_k^{\Phi}} (-1)^{\#A}|$.*

**Proof.** Using the principle of inclusion-exclusion we can express the number of strong embeddings in terms of the number of embeddings (see e.g. Chapt. 5.2.3 in [20]):

$$\#\mathsf{StrEmb}(H, G) = \sum_{\substack{H' \supseteq H \\ V(H) = V(H')}} (-1)^{\#E(H') - \#E(H)} \cdot \#\mathsf{Emb}(H', G)\,, \tag{3}$$

where $H'$ ranges over all graphs obtained from $H$ by adding edges. Next we collect terms in (3) that correspond to isomorphic graphs. To this end we let $\#\{H' \supseteq H\}$ denote the number of possibilities to add edges to $H$ such that the resulting graph is isomorphic to $H'$. Note that $\#\{K_k \supseteq H\} = 1$ if $H$ has $k$ vertices. We obtain

$$\#\mathsf{StrEmb}(H, G) = \sum_{H' \in \mathcal{G}} (-1)^{\#E(H') - \#E(H)} \cdot \#\{H' \supseteq H\} \cdot \#\mathsf{Emb}(H', G)\,. \tag{4}$$

Next we translate the number of embeddings to a linear combination of homomorphisms. This can be done using Möbius inversion[8] (see [8] or Chapt. 5.2.3 in [20]):

$$\#\mathsf{Emb}(H', G) = \sum_{\rho \geq \emptyset} \mu(\emptyset, \rho) \cdot \#\mathsf{Hom}(H'/\rho, G)\,, \tag{5}$$

where the sum and the Möbius function $\mu$ are over the partition lattice of $V(H')$ and $H'/\rho$ is obtained from $H'$ by contracting every pair of vertices that is contained in the same block in $\rho$. We observe that the coefficient of $\#\mathsf{Hom}(K_k, G)$ in the above sum is $\mu(\emptyset, \emptyset) = 1$ if $H'$ is isomorphic to $K_k$ and zero otherwise as every vertex contraction of a graph with $k$ vertices that is not the complete graph can not result in the complete graph with $k$ vertices. Hence the coefficient of $\#\mathsf{Hom}(K_k, G)$ in Equation (4) is precisely $(-1)^{\#E(K_k) - \#E(H)}$. Next we use Fact 5 and obtain that

$$\sum_{H \in \Phi_k} \#\mathsf{IndSub}(H, G) = \sum_{H \in \Phi_k} \#\mathsf{StrEmb}(H, G) \cdot \#\mathsf{Aut}(H)^{-1}\,. \tag{6}$$

It follows that the coefficient $a(K_k)$ of $\#\mathsf{Hom}(K_k, G)$ in Equation (6) satisfies

$$a(K_k) = \sum_{H \in \Phi_k} (-1)^{\#E(K_k) - \#E(H)} \cdot \#\mathsf{Aut}(H)^{-1}\,. \tag{7}$$

We now multiply this equation by $k!$, which we interpret as the number $\#\mathsf{Sym}_k$ of elements of the symmetric group of the $k$ vertices. Taking also the absolute value on both sides allows us to drop the constant factor $(-1)^{\#E(K_k)}$ and we obtain

$$|k! \cdot a(K_k)| = \left| \sum_{H \in \Phi_k} (-1)^{\#E(H)} \cdot \frac{\#\mathsf{Sym}_k}{\#\mathsf{Aut}(H)} \right|\,. \tag{8}$$

For any graph $H$ in the above sum choose a set $A_0$ of edges of the labeled complete graph $K_k$ on $k$ vertices such that the corresponding subgraph $G(A_0)$ is isomorphic to $H$. The group $\mathsf{Sym}_k$ acts on the vertices and thus on the edges of $K_k$ and by the definition of a graph automorphism, the stabilizer of the set $A_0$ has exactly $\#\mathsf{Aut}(H)$ elements. On the other hand the orbit of $A_0$ under $\mathsf{Sym}_k$ is the collection of all sets $A$ such that $G(A) \cong H$. By the Orbit Stabilizer theorem we have $\frac{\#\mathsf{Sym}_k}{\#\mathsf{Aut}(H)} = \#\{A \subseteq E(K_k) \mid G(A) \cong H\}$. Inserting in equation (8) we obtain $|k! \cdot a(K_k)| = |\sum_{H \in \Phi_k} \sum_{\substack{A \subseteq E(K_k) \\ G(A) \cong H}} (-1)^{\#E(H)}| = |\sum_{A \in \mathsf{E}_k^{\Phi}} (-1)^{\#A}|$. ◀

Theorem 10 implies the following sufficient criterion for hardness of $\#\mathsf{IndSub}(\Phi)$ which we will use in the remainder of the paper.

[8] We omit the formal introduction to Möbius inversion as we will only need that $\mu(\emptyset, \emptyset) = 1$. We refer the interested reader to [20], where the concept is introduced and Equation (5) is proved.

▶ **Corollary 11.** *Let $\Phi$ be a graph property and let $\mathcal{K} = \{k \in \mathbb{N} \mid \sum_{A \in \mathsf{E}_k^\Phi} (-1)^{\#A} \neq 0\}$. If $\mathcal{K}$ is infinite, then $\#\mathsf{IndSub}(\Phi)$ is $\#\mathrm{W}[1]$-hard. If additionally $\mathcal{K}$ is dense, $\#\mathsf{IndSub}(\Phi)$ can not be solved in time $f(k) \cdot \#V(G)^{o(k)}$ for any computable function $f$, unless ETH fails.*

Due to space constraints the proof is deferred to the related version of the paper. However, let us give the rough idea: By Theorem 9 and Theorem 10 there exists a (tight) reduction from the problem $\#\mathsf{Clique}(\mathcal{K})$ of counting cliques of size $k$ where $k \in \mathcal{K}$ is promised.

As $\mathcal{K}$ is infinite we can apply a standard technique using colors and the inclusion-exclusion principle to reduce $\#\mathsf{Clique}$ to $\#\mathsf{Clique}(\mathcal{K})$, the former of which is $\#\mathrm{W}[1]$-complete and, assuming ETH, cannot be solved in time $f(k) \cdot n^{o(k)}$ for any computable function $f$ [5, 6]. This immediately yields $\#\mathrm{W}[1]$-hardness of $\#\mathsf{Clique}(\mathcal{K})$ and hence $\#\mathsf{IndSub}(\Phi)$. If additionally $\mathcal{K}$ is dense, the reduction is tight and the lower bound under ETH transfers as well.

## 4 Monotone properties

Recall that monotone graph properties are closed under the removal of edges. In what follows we assume every monotone graph property to hold on the independent set, i.e., the graph containing no edges, because otherwise the property would be trivially false. For technical reasons we say that a property is *non-trivial* if it is false on $K_k$ for all but finitely many $k \in \mathbb{N}$.[9] Due to space constraints we might omit or only sketch proofs and defer the details to the related version of the paper. We start by refining Theorem 10 for monotone properties.

▶ **Lemma 12.** *Let $\Phi$ be a monotone graph property and let $k$ be a non-zero natural number. Then it holds that $\sum_{A \in \mathsf{E}_k^\Phi} (-1)^{\#A} = \hat{\chi}(\Delta(\Phi_k))$.*

▶ **Corollary 13** (Corollary 2 restated)**.** *Let $\Phi$ be a monotone graph property and let*

$$\mathcal{K} = \{k \in \mathbb{N} \mid \hat{\chi}(\Delta(\Phi_k)) \neq 0\}.$$

*If $\mathcal{K}$ is infinite, then $\#\mathsf{IndSub}(\Phi)$ is $\#\mathrm{W}[1]$-hard. If additionally $\mathcal{K}$ is dense, $\#\mathsf{IndSub}(\Phi)$ can not be solved in time $f(k) \cdot \#V(G)^{o(k)}$ for any computable function $f$, unless ETH fails.*

**Proof.** Follows immediately from Lemma 12 and Corollary 11. ◀

The above criterion yields hardness of $\#\mathsf{IndSub}(\Phi)$ for every monotone graph property $\Phi$ whose graph complex is well-understood with respect to the (reduced) Euler characteristic. The thesis of Jonsson (see Chapt. 10.5 in [17]) provides a large list of graph complexes including e.g. disconnectivity, colorability and coverability, only to name a few, whose reduced Euler characteristics are non-zero infinitely often and to which Corollary 13 is hence applicable. We would also like to point out the work of Chakrabarti, Khot and Shi [4] who proved the reduced Euler characteristic of a large family of graph complexes to be odd. Their result will be used to prove the fourth condition of Theorem 4 and reads as follows — we state it in terms of homomorphisms.

▶ **Lemma 14** ([4])**.** *Let $F$ be a graph and let $\Phi^{[F]}$ be the graph property that holds true on a graph $G$ if and only if $\mathsf{Hom}(F, G) = \emptyset$, i.e., there is no homomorphism from $F$ to $G$. Furthermore let $T_F := \min\{2^{2^t} - 1 \mid 2^{2^t} \geq \#V(F)\}$ and let $k \in \mathbb{N}$ such that $k \equiv 1 \mod T_F$. Then it holds that $\chi(\Phi_k^{[F]}) \equiv 0 \mod 2$ and hence $\hat{\chi}(\Phi_k^{[F]}) \equiv 1 \mod 2$.*

[9] This is required to exlude properties like $\Phi(G) = 0 \Leftrightarrow \#V(G) \equiv 1 \mod 2$ which indeed is monotone as it is closed under the removal of edges.

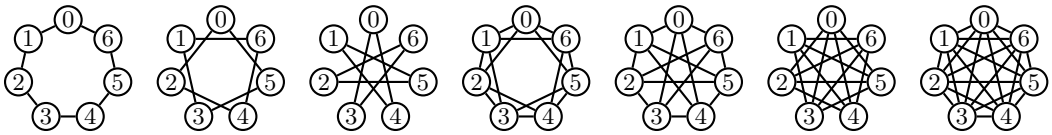



**Figure 1** Non-empty unions of orbits on the operation of $\mathbb{Z}_7$ on the edge set of the labeled graph with 7 vertices. If $\Phi$ is trivially true then $\Delta^{\mathbb{Z}_7}(\Phi_7)$ contains all of the above subsets of orbits. If $\Phi$ holds only for bipartite graphs then none of the above subsets is contained in $\Delta^{\mathbb{Z}_7}(\Phi_7)$. If $\Phi$ is planarity then $\Delta^{\mathbb{Z}_7}(\Phi_7) = \{\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3\}$. More exotically, if $\Phi$ is the property of not being 5-edge-connected then $\Delta^{\mathbb{Z}_7}(\Phi_7)$ contains every subset of orbits except for $\mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{O}_3$.

Unfortunately, as the proof of the above lemma shows, it is often quite tedious to argue about the (reduced) Euler characteristic of the graph complex induced by a more complicated property $\Phi$ and hence proving hardness of $\#\mathsf{IndSub}(\Phi)$. In the remainder of this section we will therefore demonstrate that Corollary 13 together with Theorem 7 yields a fruitful topological approach to prove #W[1]-hardness and conditional lower bounds for $\#\mathsf{IndSub}(\Phi)$, given that $\Phi$ is a monotone graph property. We outline the approach in the following lemma.

▶ **Lemma 15.** *Let $\Phi$ be a monotone graph property, let $\mathcal{K}$ be an infinite subset of $\mathbb{N}$ and let $\Gamma = \{\Gamma_k \mid k \in \mathcal{K}\}$ be a set of permutation groups such that for every $k \in \mathcal{K}$ the group $\Gamma_k$ is a $p_k$-power group for some prime $p_k$. If $\hat{\chi}(\Delta^{\Gamma}(\Phi_k)) \not\equiv 0 \mod p_k$ holds for every $k \in \mathcal{K}$ then $\#\mathsf{IndSub}(\Phi)$ is #W[1]-hard. If additionally $\mathcal{K}$ is dense, $\#\mathsf{IndSub}(\Phi)$ can not be solved in time $f(k) \cdot \#V(G)^{o(k)}$ for any computable function $f$, unless ETH fails.*

**Proof.** Follows immediately from Corollary 13 and Theorem 7. ◀

Intuitively, Lemma 15 states that instead of analyzing $\hat{\chi}(\Delta(\Phi_k))$ which might be tedious, it suffices to prove that the reduced Euler characteristic of the fixed-point complex of $\Delta(\Phi_k)$ with respect to a $p$-power group is not 0 modulo $p$. For our purposes it will suffice to use the groups $\mathbb{Z}_p$ for prime numbers $p$, explained as follows. Recall that the ground set of $\Delta(\Phi_p)$ is the set of all edges of the labeled complete graph on $p$ vertices. Now $b \in \mathbb{Z}_p$ is interpreted as a relabeling $x \mapsto x + b$ of the vertices[10], which induces an operation on the edges by mapping the edge $\{x, y\}$ to the edge $\{x + b, y + b\}$. We remark that this group was also used in an intermediate step in [18]. It can easily be verified that this mapping is a group operation. Furthermore $\Delta(\Phi_p)$ is a $\mathbb{Z}_p$-simplicial complex with respect to this operation as $\Phi$ is invariant under relabeling of vertices. Hence the fixed-point complex $\Delta^{\mathbb{Z}_p}(\Phi_p)$ is defined. Furthermore observe that every orbit of the group operation is an Hamilton cycle. We illustrate $\Delta^{\mathbb{Z}_7}(\Phi_7)$ for some properties $\Phi$ in Figure 1.

Note that, given a prime $p > 2$, the ground set of $\Delta^{\mathbb{Z}_p}(\Phi_p)$ consists of exactly $\frac{1}{2}(p-1)$ elements. In particular those are the Hamiltonian cycles $H_1 = (0, 1, 2, \dots)$, $H_2 = (0, 2, 4, \dots)$, $H_3 = (0, 3, 6, \dots)$, $\dots$, $H_{\frac{1}{2}(p-1)} = (0, \frac{1}{2}(p-1), p-1, \dots)$. Equivalently, $H_i$ is the orbit of the (labeled) edge $\{0, i\}$ under the operation of $\mathbb{Z}_p$ for $i \in \{1, \dots, \frac{1}{2}(p-1)\}$ and it can easily be verified that those are all orbits of the group operation. In what follows, given a non-empty set $P \subseteq \{1, \dots, \frac{1}{2}(p-1)\}$, we write $H_P$ for the graph with vertices (labeled with) $\{0, \dots, p-1\}$ and edges $\bigcup_{i \in P} H_i$.

▶ **Fact 16.** *Let $P$ be non-empty subset of $\{1, \dots, \frac{1}{2}(p-1)\}$. Then $P \in \Delta^{\mathbb{Z}_p}(\Phi_p) \Leftrightarrow H_P \in \Phi_p$.*

[10] Here $+$ is addition modulo $p$.

Now we have everything we need to prove our main result. We start with monotone properties that are false on odd cycles or true on odd antiholes.

▶ **Lemma 17.** *Let $\Phi$ be a non-trivial monotone graph property. If $\Phi$ does not hold on odd cycles or if $\Phi$ holds on odd anti-holes then there exists a constant $N \in \mathbb{N}$ such that $\hat{\chi}(\Delta^{\mathbb{Z}_p}(\Phi_p)) \not\equiv 0 \mod p$ for every prime $p > N$.*

**Proof sketch.** If $\Phi$ does not hold on odd cycles then $\Delta^{\mathbb{Z}_p}(\Phi_p) = \emptyset$ and hence we have that $\hat{\chi}(\Delta^{\mathbb{Z}_p}(\Phi_p)) = 1 - \chi(\Delta^{\mathbb{Z}_p}(\Phi_p)) = 1 - 0 = 1 \not\equiv 0 \mod p$. As $\Phi$ is non-trivial there exists $N \in \mathbb{N}$ such that $\Phi(K_k) = 0$ for all $k > N$. Now if $\Phi$ holds on odd anti-holes then $\Delta^{\mathbb{Z}_p}(\Phi_p) = \{P \mid \emptyset \subsetneq P \subsetneq \{1, \dots, \frac{1}{2}(p-1)\}\}$ for all $p > N$ since $\Phi$ is monotone and $H_P$ is an anti-hole if and only if $\#P = \frac{p-1}{2} - 1$. Furthermore, $\Phi$ does not hold on $H_{\{1,\dots,\frac{1}{2}(p-1)\}} \cong K_p$ as $p > N$. Now it can be easily verified that $\hat{\chi}(\Delta^{\mathbb{Z}_p}(\Phi_p)) = (-1)^{\frac{1}{2}(p-1)+1} \not\equiv 0 \mod p$. ◀

We continue with one more exotic property which illustrates the utility of the topological approach by exploiting the simple structure of $\Delta^{\mathbb{Z}_p}(\Phi_p)$.

▶ **Lemma 18.** *Let $c \in \mathbb{N}$ be an arbitrary constant and let $\Phi$ be the graph property of being not $(c+1)$-edge-connected. Then $\hat{\chi}(\Delta^{\mathbb{Z}_p}(\Phi_p)) \not\equiv 0 \mod p$ for every prime $p > c + 3$.*

**Proof sketch.** We rely on the observation that the graph $H_P$ is $(c+1)$-edge-connected if and only if $\#P > \lfloor \frac{c}{2} \rfloor$. Hence $\Delta^{\mathbb{Z}_p}(\Phi_p) = \{P \subseteq \{1, \dots, \frac{1}{2}(p-1) \mid P \neq \emptyset \wedge \#P \leq \lfloor \frac{c}{2} \rfloor\}$. Now it can be easily verified that $\hat{\chi}(\Delta^{\mathbb{Z}_p}(\Phi_p)) = (-1)^{\lfloor \frac{c}{2} \rfloor} \cdot \binom{\frac{1}{2}(p-1)-1}{\lfloor \frac{c}{2} \rfloor} \not\equiv 0 \mod p$. ◀

Finally, Theorem 4 follows from Lemma 15, 14, 17 and 18. Details are given in the related version of the paper.

## 5 Non-monotone properties

In this section, we present #W[1]-hardness results for two non-monotone properties. For the first one, we generalize [16] where hardness was established for counting induced subgraphs with an even (or odd) number of edges. It turns out that any non-trivial modularity constraint with respect to a prime induces #W[1]-hardness. To this end let $q$ be a prime and $\mathcal{Q}$ be a subset of $\{0, \dots, q-1\}$. Then the property $\mathsf{Mod}[q, \mathcal{Q}]$ holds on a graph $H$ if and only if $(\#E(H) \bmod q) \in \mathcal{Q}$. For the second property, let $F$ be a connected graph. Then the property $\mathsf{Iso}[F]$ holds on a graph $H$ if and only if $H$ contains an isolated subgraph that is isomorphic to $F$. Due to space constraints the proof of the following theorem is deferred to the related version of the paper.

▶ **Theorem 19.** *For all primes $q$, non-trivial subsets $\mathcal{Q}$ of $\{0, \dots, q-1\}$ and connected graphs $F$, the problems $\#\mathsf{IndSub}(\mathsf{Mod}[q, \mathcal{Q}])$ and $\#\mathsf{IndSub}(\mathsf{Iso}[F])$ are #W[1]-hard and can not be solved in time $f(k) \cdot n^{o(k)}$ for any computable function $f$, unless ETH fails.*

## 6 Conclusion and future work

We used the framework of graph motif parameters to provide a sufficient criterion for #W[1]-hardness of $\#\mathsf{IndSub}(\Phi)$. For monotone properties $\Phi$ this amounts to the reduced Euler characteristic of the associated graph complex to be non-zero infinitely often. In particular, our results provide a fine-grained reduction from the problem of counting cliques of size $k$ to counting induced subgraphs of size $k$ with property $\Phi$ whenever $\Phi$ is monotone and $\hat{\chi}(\Delta(\Phi_k)) \neq 0$. Using a topological approach, we established hardness for a large class of

non-trivial monotone graph properties. The obvious next question, whose answer would settle the parameterized complexity of $\#\mathsf{IndSub}(\Phi)$ for monotone properties completely, is whether for every non-trivial monotone property $\Phi$ the set of $k$ such that $\hat{\chi}(\Delta(\Phi_k)) \neq 0$ is infinite.

## References


1. OEIS Foundation Inc. (2018). The On-Line Encyclopedia of Integer Sequences. URL: http://oeis.org.
2. Karl R. Abrahamson, Rodney G. Downey, and Michael R. Fellows. Fixed-Parameter Intractability II (Extended Abstract). In *STACS 93, 10th Annual Symposium on Theoretical Aspects of Computer Science, Würzburg, Germany, February 25-27, 1993, Proceedings*, pages 374–385, 1993. doi:10.1007/3-540-56503-5_38.
3. Glen E Bredon. *Introduction to compact transformation groups*, volume 46. Academic press, 1972.
4. Amit Chakrabarti, Subhash Khot, and Yaoyun Shi. Evasiveness of Subgraph Containment and Related Properties. *SIAM J. Comput.*, 31(3):866–875, 2001. doi:10.1137/S0097539700382005.
5. Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David W. Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized NP-hard problems. *Inf. Comput.*, 201(2):216–231, 2005. doi:10.1016/j.ic.2005.05.001.
6. Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006. doi:10.1016/j.jcss.2006.04.007.
7. Yijia Chen, Marc Thurley, and Mark Weyer. Understanding the Complexity of Induced Subgraph Isomorphisms. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, pages 587–596, 2008. doi:10.1007/978-3-540-70575-8_48.
8. Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In *Proceedings of the 49th ACM Symposium on Theory of Computing, STOC*, pages 210–223, 2017. doi:10.1145/3055399.3055502.
9. Radu Curticapean and Dániel Marx. Complexity of Counting Subgraphs: Only the Boundedness of the Vertex-Cover Number Counts. In *Proceedings of the 55th Annual Symposium on Foundations of Computer Science, FOCS*, pages 130–139, 2014. doi:10.1109/FOCS.2014.22.
10. Víctor Dalmau and Peter Jonsson. The complexity of counting homomorphisms seen from the other side. *Theoretical Computer Science*, 329(1):315–323, 2004.
11. Jörg Flum and Martin Grohe. The Parameterized Complexity of Counting Problems. *SIAM J. Comput.*, 33(4):892–922, 2004. doi:10.1137/S0097539703427203.
12. Jörg Flum and Martin Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
13. Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM (JACM)*, 54(1):1, 2007.
14. Mark Jerrum and Kitty Meeks. Some Hard Families of Parameterized Counting Problems. *TOCT*, 7(3):11:1–11:18, 2015. doi:10.1145/2786017.
15. Mark Jerrum and Kitty Meeks. The parameterised complexity of counting connected subgraphs and graph motifs. *J. Comput. Syst. Sci.*, 81(4):702–716, 2015. doi:10.1016/j.jcss.2014.11.015.

16 Mark Jerrum and Kitty Meeks. The parameterised complexity of counting even and odd induced subgraphs. *Combinatorica*, 37(5):965–990, 2017. doi:10.1007/s00493-016-3338-5.

17 Jakob Jonsson. *Simplicial complexes of graphs*, volume 1928 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 2008. doi:10.1007/978-3-540-75859-4.

18 Jeff Kahn, Michael E. Saks, and Dean Sturtevant. A Topological Approach to Evasiveness. In *24th Annual Symposium on Foundations of Computer Science, Tucson, Arizona, USA, 7-9 November 1983*, pages 31–33, 1983. doi:10.1109/SFCS.1983.4.

19 Richard E. Ladner. On the Structure of Polynomial Time Reducibility. *J. ACM*, 22(1):155–171, 1975. doi:10.1145/321864.321877.

20 László Lovász. *Large networks and graph limits*, volume 60. American Mathematical Society Providence, 2012.

21 Frank H. Lutz. Some Results Related to the Evasiveness Conjecture. *J. Comb. Theory, Ser. B*, 81(1):110–124, 2001. doi:10.1006/jctb.2000.2000.

22 Kitty Meeks. The challenges of unbounded treewidth in parameterised subgraph counting problems. *Discrete Applied Mathematics*, 198:170–194, 2016. doi:10.1016/j.dam.2015.06.019.

23 Carl A. Miller. Evasiveness of Graph Properties and Topological Fixed-Point Theorems. *Foundations and Trends in Theoretical Computer Science*, 7(4):337–415, 2013. doi:10.1561/0400000055.

24 Robert Oliver. Fixed-point sets of group actions on finite acyclic complexes. *Commentarii Mathematici Helvetici*, 50(1):155–177, 1975.

25 Ronald L. Rivest and Jean Vuillemin. On Recognizing Graph Properties from Adjacency Matrices. *Theor. Comput. Sci.*, 3(3):371–384, 1976. doi:10.1016/0304-3975(76)90053-0.

26 Marc Roth. Counting Restricted Homomorphisms via Möbius Inversion over Matroid Lattices. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, pages 63:1–63:14, 2017. doi:10.4230/LIPIcs.ESA.2017.63.

27 PA Smith. Fixed-point theorems for periodic transformations. *American Journal of Mathematics*, 63(1):1–8, 1941.

28 Leslie G. Valiant. The Complexity of Computing the Permanent. *Theor. Comput. Sci.*, 8:189–201, 1979. doi:10.1016/0304-3975(79)90044-6.

# A Strongly-Uniform Slicewise Polynomial-Time Algorithm for the Embedded Planar Diameter Improvement Problem

**Daniel Lokshtanov**[1]

Department of Informatics - University of Bergen, Bergen, Norway
daniel.lokshtanov@uib.no

**Mateus de Oliveira Oliveira**[2]

Department of Informatics - University of Bergen, Bergen, Norway
mateus.oliveira@uib.no

**Saket Saurabh**[3]

Department of Informatics - University of Bergen, Bergen, Norway
saket.saurabh@uib.no

## Abstract

In the EMBEDDED PLANAR DIAMETER IMPROVEMENT problem (EPDI) we are given a graph $G$ embedded in the plane and a positive integer $d$. The goal is to determine whether one can add edges to the planar embedding of $G$ in such a way that planarity is preserved and in such a way that the resulting graph has diameter at most $d$. Using non-constructive techniques derived from Robertson and Seymour's graph minor theory, together with the effectivization by self-reduction technique introduced by Fellows and Langston, one can show that EPDI can be solved in time $f(d) \cdot |V(G)|^{O(1)}$ for some function $f(d)$. The caveat is that this algorithm is *not* strongly uniform in the sense that the function $f(d)$ is not known to be computable. On the other hand, even the problem of determining whether EPDI can be solved in time $f_1(d) \cdot |V(G)|^{f_2(d)}$ for *computable* functions $f_1$ and $f_2$ has been open for more than two decades [Cohen at. al. Journal of Computer and System Sciences, 2017]. In this work we settle this later problem by showing that EPDI can be solved in time $f(d) \cdot |V(G)|^{O(d)}$ for some computable function $f$. Our techniques can also be used to show that the EMBEDDED $k$-OUTERPLANAR DIAMETER IMPROVEMENT problem ($k$-EOPDI), a variant of EPDI where the resulting graph is required to be $k$-outerplanar instead of planar, can be solved in time $f(d) \cdot |V(G)|^{O(k)}$ for some computable function $f$. This shows that for each fixed $k$, the problem $k$-EOPDI is strongly uniformly fixed parameter tractable with respect to the diameter parameter $d$.

**2012 ACM Subject Classification** Theory of computation → Parameterized complexity and exact algorithms

**Keywords and phrases** Embedded Planar Diameter Improvement, Constructive Algorithms, Nooses

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.25

[1] Daniel Lokshtanov acknowledges support from the ERC grant agreement no. 715744.
[2] Mateus de Oliveira Oliveira acknowledges support from the Bergen Research Foundation.
[3] Saket Saurabh acknowledges support from the ERC grant agreement no. 306992.

© Daniel Lokshtanov, Mateus de O. Oliveira, and Saket Saurabh;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 25; pp. 25:1–25:13
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

# 1 Introduction

In this work, a *plane graph* is a pair $G^g = (G, g)$ consisting of a planar graph $G$ together with a planar embedding $g$ of $G$ [4]. In the EMBEDDED PLANAR DIAMETER IMPROVEMENT problem (EPDI), we are given a plane graph $G^g$ and a positive integer $d$, and the goal is to determine whether $G^g$ has a plane supergraph $H^h$ of diameter at most $d$. The set of YES instances of EPDI is closed under minors. In other words, if $G^g$ has a plane supergraph of diameter at most $d$, then any minor of $G^g$ also has such a supergraph. Therefore, using non-constructive arguments from Robertson and Seymour's graph minor theory [14, 15] in conjunction with the fact that planar graphs of constant diameter have constant treewidth, one can show that for each fixed $d$, there exists an algorithm $\mathfrak{A}_d$ which determines in linear time whether a given plane graph $G^g$ has a plane supergraph of diameter at most $d$. The caveat is that the non-constructive techniques mentioned above provide us with no clue about what the algorithm $\mathfrak{A}_d$ actually is. This problem can be partially remedied using a technique called effectivization by self-reduction introduced by Fellows and Langston [10, 8]. Using this technique one can show that for some function $f : \mathbb{N} \to \mathbb{N}$, there exists a single algorithm $\mathfrak{A}$ which takes a plane graph $G^g$ and a positive integer $d$ as input, and determines in time $f(d) \cdot |V(G)|^{O(1)}$ whether $G^g$ has a plane supergraph of diameter at most $d$. Nevertheless, the function $f : \mathbb{N} \to \mathbb{N}$ bounding the influence of the parameter $d$ in the running time of the algorithm mentioned above is not known to be computable. The problem of determining whether EPDI admits an algorithm running in time $f(d) \cdot |V(G)|^{O(1)}$ for some *computable* function $f$ is a notorious and long-standing open problem in parameterized complexity theory [8, 9, 5]. Interestingly even the problem of determining whether EPDI can be solved in time $f_1(d) \cdot |V(G)|^{f_2(d)}$ for *computable* functions $f_1$ and $f_2$ has remained open until now [4]. In this work we settle this latter problem by showing that EPDI can be solved in time $2^{d^{O(d)}} \cdot |V(G)|^{O(d)}$. The problem of determining whether EPDI can be solved in time $f(d) \cdot |V(G)|^{O(1)}$ for some *computable* function $f : \mathbb{N} \to \mathbb{N}$ remains widely open.

▶ **Theorem 1.** *There is an algorithm $\mathfrak{A}$ that takes as input, a positive integer $d$, and a plane graph $G^g$, and determines in time $2^{d^{O(d)}} \cdot |V(G)|^{O(d)}$ whether $G^g$ has a plane supergraph $H^h$ of diameter at most $d$.*

A graph is 1-outerplanar if it can be embedded in the plane in such a way that every vertex lies in the outer face of the embedding. A graph is $k$-outerplanar if it can be embedded in the plane in such a way that after deleting all vertices in the outer face, the remaining graph is $(k-1)$-outerplanar. In [4] Cohen et al. have considered the $k$-OUTERPLANAR DIAMETER IMPROVEMENT problem ($k$-OPDI), a variant of the PDI problem in which the target supergraph is required to be $k$-outerplanar instead of planar. In particular, they have shown that the 1-OPDI problem can be solved in polynomial time. The complexity of the $k$-OPDI problem with respect to explicit algorithms was left as an open problem for $k \geq 2$. By adapting our algorithm for the EPDI problem we are able to show that when the input graph is given together with an embedding that must be preserved, then the resulting problem, the $k$-EOPDI problem, can be solved in time $2^{d^{O(d)}} \cdot |V(G)|^{O(k)}$ for each fixed $k$. In other words, this problem is strongly uniformly fixed parameter tractable with respect to the diameter parameter for each fixed value of outerplanarity.

[4] See Section 2 for formal definitions.

▶ **Theorem 2.** *There is an algorithm $\mathfrak{A}$ that takes as input, positive integers $d, k$, and a plane graph $G^g$, and determines in time $2^{d^{O(d)}} \cdot |V(G)|^{O(k)}$ whether $G^g$ has a $k$-outerplanar plane supergraph $H^h$ of diameter at most $d$.*

**Related Work.** In the *planar diameter improvement problem* (PDI), the input consists of a planar graph $G$ and a positive integer $d$ and the goal is to determine whether one can add edges to $G$ in such a way that the resulting graph is planar and has diameter at most $d$. Note that the difference between EPDI and PDI is that in the former we are given an embedding that must be preserved, while in the latter no embedding is given at the input. Recently, using automata theoretic techniques, the second author was able to provide strongly uniform FPT and XP algorithms for many graph completion problems where the parameter to be improved is definable in counting monadic second order logic [6]. In particular, when specialized to the PDI problem, the techniques in [6] yield a strongly uniform algorithm that solves PDI in time $f(d) \cdot 2^{O(\Delta \cdot d)} \cdot |V(G)|^{O(d)}$, where $f$ is a computable function and $\Delta$ is the maximum degree of the input graph $G$. Nevertheless, the problem of determining whether PDI admits a strongly uniform algorithm running in time $f_1(d) \cdot |V(G)|^{f_2(d)}$ for computable functions $f_1$ and $f_2$ is still open if no bound is imposed on the degree of the input graph. We note that currently it is not known either whether PDI is reducible to EPDI or whether EPDI is reducible to PDI in XP time. Therefore it is not clear if our algorithm for EPDI can be used to provide a strongly uniform XP algorithm for PDI. It is worth noting that no hardness results for either PDI or EPDI are known. Indeed, determining whether either of these problems is NP-hard is also a long-standing open problem.

While the techniques employed in [6] to tackle the PDI problem on graphs of bounded degree are automata theoretic, the techniques employed in the present work to tackle the EPDI problem on general graphs are based on dynamic programming. In particular, our main algorithm carefully exploits the view of separators in plane graphs as *nooses* - simple closed curves in the plane that touch the graph only in the vertices (see e.g. [2]). The terminology *noose* for such curves comes from the graph minors project of Robertson and Seymour [13]. Our algorithm processes nooses in a way reminiscent of the dynamic programming algorithm of Bouchitte and Todinca over potential maximal cliques [3]. Although this method has found numerous applications in the field of graph algorithms [7, 12, 11], this work is the first which apply these techniques in the context of completion problems.

For each fixed $d$, let $\mathcal{G}_d$ be the *subgraph-closure*[5] of the class of planar graphs of diameter at most $d$. Then clearly a graph $G$ is a yes instance of PDI if and only if $G \in \mathcal{G}_d$. When considering the task of constructing strongly uniform algorithms for PDI, two general approaches come to mind. The first follows by observing that graphs in $\mathcal{G}_d$ have treewidth at most $O(d)$, and that for each fixed $d \in \mathbb{N}$, $\mathcal{G}_d$ is MSO definable. Therefore, one could try to devise an algorithm $\mathfrak{A}$ that takes an integer $d$ as input and constructs an MSO formula $\varphi_d$ defining $\mathcal{G}_d$. With such a formula $\varphi_d$ in hands, one could apply Courcelle's model checking theorem to determine whether a given graph $G$ is a yes instance of PDI. The existence of such an algorithm $\mathfrak{A}$ is however an open problem. We note that one can easily define by induction on $d$ an MSO sentence $\phi_d$ which is true on a graph $G$ if and only if $G$ is planar and has diameter at most $d$. Nevertheless, it is not clear how to use $\phi_d$ to construct $\varphi_d$. It is worth noting that there is no algorithm that takes an MSO sentence $\varphi$ as input and constructs a sentence $\varphi'$ defining the subgraph closure of the models of $\varphi$. For instance, let

[5] Note that the class of planar graphs of diameter at most $d$ is closed under contractions but *not* under subgraphs, since removing edges may increase the diameter.

$\mathcal{L} = \{L_n\}_{n\in\mathbb{N}}$ be the family of ladder graphs, where $L_n$ is the ladder with $n$ steps. It is easy to see that $\mathcal{L}$ is MSO definable and every graph in $\mathcal{L}$ has treewidth at most 2. Nevertheless, the subgraph closure of $\mathcal{L}$ does not have finite index. Therefore, such subgraph closure is *not* MSO definable, since Courcelle's theorem implies that MSO-definable families of graphs of bounded treewidth have finite index.

The second approach is based on the fact that for each fixed $d \in \mathbb{N}$, the set of graphs $\mathcal{G}_d$ is minor closed. In particular, this implies that the class $\mathcal{G}_d$ can be characterized by a finite set $\mathcal{M}_d$ of forbidden minors. Therefore, one could try to devise an effective algorithm that takes an integer $d \in \mathbb{N}$ as input and gives as output the list of all forbidden minors in $\mathcal{M}_d$. By using the fact that minor-freeness can be tested in time FPT in the size of the minors, such an algorithm $\mathfrak{A}$ would solve PDI in time FPT in $d$. We observe however that the problem of listing the elements of $\mathcal{M}_d$ may be much more difficult than the problem of solving PDI in time FPT in $d$. It is worth noting that Adler, Kreutzer and Grohe have shown that if a minor-free graph property $\mathcal{P}$ is MSO definable and has constant treewidth, then one can effectively enumerate the set of forbidden minors for $\mathcal{P}$ [1]. Nevertheless, the problem with this approach is that, as discussed in the previous paragraph, it is not clear how to construct an MSO sentence $\varphi_d$ defining $\mathcal{G}_d$.

## 2 Preliminaries

**Graphs.** For each $n \in \mathbb{N}$ we let $[n] = \{1, ..., n\}$. For each finite set $S$ we let $\mathcal{P}(S, 2) = \{\{u, v\} \subseteq S \mid u \neq v\}$ be the set of unordered pairs of elements from $S$. In this work, a graph is a pair $G = (V(G), E(G))$ where $V(G)$ is a finite set of vertices and $E(G) \subseteq \mathcal{P}([n], 2)$ is a set of undirected edges. A path of length $m$ in $G$ is a sequence $p = v_0v_1...v_m$ of distinct vertices where for each $i \in \{0, ..., m-1\}$, $\{v_i, v_{i+1}\} \in E(G)$. We say that $v_0$ and $v_m$ are the endpoints of $p$. The distance $dist(u, v)$ between vertices $u$ and $v$ is defined as the length of the shortest path with endpoints $u$ and $v$. The diameter of $G$, is defined as $d(G) = \max_{u,v} dist(u, v)$.

**Embeddings.** A *simple arc* in $\mathbb{R}^2$ is a subset $\alpha \subseteq \mathbb{R}^2$ that is a homeomorphic image of the closed real interval $[0, 1]$. We let $endpts(\alpha)$ be the endpoints of $\alpha$, and $Int(\alpha) = \alpha \backslash endpts(\alpha)$ be the interior of $\alpha$. We let $\mathcal{A}$ be the set of simple arcs in $\mathbb{R}^2$. A planar embedding of $G$ is a map $g : V(G) \cup E(G) \to \mathbb{R}^2 \cup \mathcal{A}$ that assigns a point $g(v) \in \mathbb{R}^2$ to each vertex $v \in V(G)$ and a simple arc $g(\{u, v\}) \in \mathcal{A}$ with each edge $\{u, v\}$ in such a way that the following conditions are satisfied.

1. For each $u, v \in V(G)$, $g(u) \neq g(v)$.
2. For each $\{u, v\} \in E(G)$, $\{g(u), g(v)\}$ are the endpoints of the simple arc $g(\{u, v\})$.
3. For each $\{u, v\} \in E(G)$, and each $w \in V(G)$ such that $w \neq u$ and $w \neq v$, $g(w) \notin g(\{u, v\})$.
4. For each $\{u, v\}, \{u', v'\} \in E(G)$, $Int(g(\{u, v\})) \cap Int(g(\{u', v'\})) = \emptyset$.

Intuitively, a planar embedding of a graph $G$ is a drawing of $G$ on the plane where each vertex $v$ is represented by a point and each edge $e$ is represented by a non self-intersecting curve that connects the points corresponding to the endpoints of $e$, and no crossings are allowed. A *plane graph* is a pair $G^g = (G, g)$ where $G$ is a graph and $g$ is a planar embedding of $G$. For technical reasons, in this work we assume that the origin $(0, 0) \in \mathbb{R}^2$ is distinct from $g(v)$ for each $v \in V(G)$ and does not belong to $g(e)$ for each edge $e \in E(G)$.

**Plane Completion.** Let $G$ and $H$ be graphs. We say that $G$ is a *subgraph* of $H$ if $V(G) \subseteq V(H)$ and $E(G) \subseteq E(H)$. If $G^g$ and $H^h$ are plane graphs, then we say that $G^g$ is a *plane*

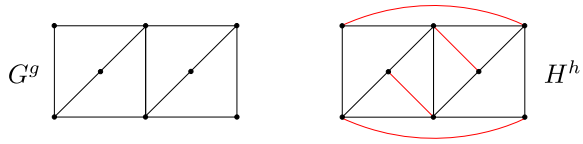



**Figure 1** Left: a plane graph $G$ of diameter 3. Right: a plane completion of $G$ of diameter 2.

*subgraph* of $H^h$ if $G$ is a subgraph of $H$, $g|_{V(G)} = h|_{V(G)}$ and $g_{E(G)} = h|_{E(G)}$ [6]. Alternatively, we say that $H^h$ is a *plane completion* of $G^g$ (Figure 1). We say that such a completion $H^h$ is triangulated if each face of $H^h$ has three vertices.

**Combinatorial face.** Let $G^g$ be a plane graph. We say that a point $p \in \mathbb{R}^2$ is *independent* of $G^g$ if $p \neq g(v)$ for every $v \in V(G)$ and $p \notin g(e)$ for every $e \in E(G)$. We say that an independent point $p$ *reaches* a point $p' \in \mathbb{R}^2$ if there is a simple arc $\alpha$ with endpoints $p$ and $p'$ that does not contain any vertex in $g(V)\backslash\{p, p'\}$ and does not intersect the interior of any arc in $g(E)$. If $p$ is an independent point then we let $\mathcal{F}(G^g, p)$ be the subgraph of $G$ whose vertex set is $V(\mathcal{F}(G^g, p)) = \{v \in V(G) \mid p \text{ reaches } g(v)\}$ and whose edge set is $E(\mathcal{F}(G^g, p)) = \{e \in E(G) \mid p \text{ reaches each point in } g(e)\}$.

We let $b(p)$ be a boolean value that is 0 if the origin $(0, 0)$ is reachable from $p$, and 1 otherwise. We say that a pair $F = (X, b)$, where $X$ is a subgraph of $G$ and $b \in \{0, 1\}$, is a *combinatorial face* if there exists a $p \in \mathbb{R}^2$ such that $X = \mathcal{F}(G^g, p)$ and $b = b(p)$. For instance, if $G^g$ is a plane graph where $G$ is a tree, then the unique combinatorial face of $G^g$ is the pair $F = (G, 1)$. On the other hand, if $G$ is a cycle, then $G^g$ has two faces: $F_1 = (G, 0)$ and $F_2 = (G, 1)$. We write $F(G^g)$ to denote the set of all faces of $G^g$. We note that $F(G^g)$ has $O(|V(G)|)$ faces. If $F = (X, b)$ is a combinatorial face, then we define $V(F) = V(X)$ and $E(F) = E(X)$.

We say that two embedded versions $G^g$ and $G^{g'}$ of a graph $G$ are equivalent if $F(G^g) = F(G^{g'})$. We write $G^g \equiv G^{g'}$ to denote that $G^g$ and $G^{g'}$ are equivalent.

## 3 Nooses

▶ **Definition 3.** Let $G^g$ be a plane graph. A $G^g$*-noose* is a subset $\eta \subseteq \mathbb{R}^2$ homeomorphic to the unit circle $S_1$ such that $\eta \cap Int(g(uv)) = \emptyset$ for every edge $uv \in E(G)$.

We note that if $\eta$ is a $G^g$-noose, the intersection $\eta \cap g(V(G))$ may be non-empty. We let $V(\eta) = \{v \in V(G) \mid g(v) \in \eta\}$ be the set of vertices of $G$ whose image lies in the noose $\eta$. The size of $\eta$ is defined as $|\eta| = |V(\eta)|$.

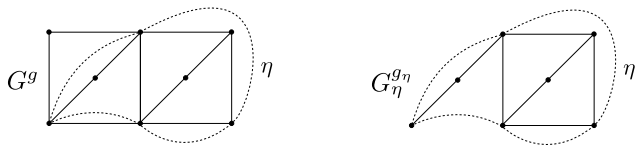



**Figure 2** Left: a plane graph $G$ and one of its nooses $\eta$. Right: the graph $G_\eta^{g_\eta}$ that lies in the interior of $\eta$.

[6] Where $g|_{V(G)}$ and $g|_{E(G)}$ denote the restrictions of $g$ to $V(G)$ and to $E(G)$ respectively.

**Combinatorial cycle.** Let $\Sigma$ be a finite set. We let $\Sigma^k$ be the set of sequences of length $k$ over $\Sigma$. If $a_0a_1...a_{k-1}$ is a sequence in $\Sigma^k$, then we let

$$[a_0a_1...a_{k-1}] = \{a_{j_0}a_{j_1}...a_{j_{k-1}} \mid \exists l \in \{0,...,k-1\} \forall i \in \{0,...,k-1\},\ j_i = i + l \ mod\ k\}.$$

be the set of all cyclic shifts of the string $a_0a_1...a_{k-1}$. We say that the set $[a_0a_1...a_{k-1}]$ is a *combinatorial cycle* over $\Sigma$.

**Noose Type.** Let $G^g$ be a plane graph and $\Sigma(G) = V(G) \cup F(G)$. A *$G^g$-noose type* is a cycle $\tau = [v_0F_0v_1F_1...v_{r-1}F_{r-1}]$ over $\Sigma(G)$ where $\{v_0,...,v_{r-1}\} \subseteq V(G)$, $\{F_0,...,F_{r-1}\} \subseteq F(G^g)$, and $\{v_i, v_{i+1 \bmod r}\} \subseteq V(F_i)$ for each $i \in \{0,1,...,r-1\}$. We say that a $G^g$ noose $\eta$ is compatible with $\tau$ if there exist simple arcs $\ell_0,...,\ell_{r-1}$ satisfying the following properties.

1. $\eta = \bigcup_{i\in\{0,...,r-1\}} \ell_i$.
2. For each $i \in \{0,...,r-1\}$, $endpts(\ell_i) = \{g(v_i), g(v_{i+1 \bmod r})\}$.
3. For each $i \in \{0,...,r-1\}$, $\ell_i \cap \ell_{i+1 \bmod r} = v_{i+1 \bmod r}$.

We say that two $G^g$-nooses $\eta_1$ and $\eta_2$ are equivalent if there exists a $G^g$-noose type $\tau$ such that both $\eta_1$ and $\eta_2$ are compatible with $\tau$. We note that uncountably many $G^g$-nooses may be compatible with a given $G^g$-noose type $\tau$.

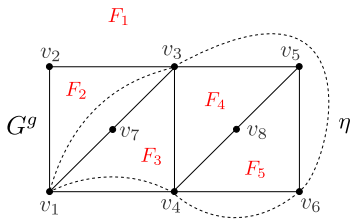



**Figure 3** The type of the noose type $\eta$ is the cycle $[v_1F_2v_3F_1v_6F_1v_4F_3]$. Note that the segment of $\eta$ between $v_1$ and $v_3$ lies in the area delimited by the face $F_2$, the segment between $v_3$ and $v_6$ lies in the area delimited by face $F_1$ and so on.

Let $\gamma$ be a subset of $\mathbb{R}^2$ homeomorphic to the unit circle $S_1$. We say that a point $p \in \mathbb{R}^2$ belongs to the closed interior of $\gamma$ if $\alpha \cap \gamma \neq \emptyset$ for every simple arc $\alpha$ with $endpts(\alpha) = \{(0,0), p\}$. We let $\hat{\gamma}$ be the set of points in the closed interior of $\gamma$. If $G^g = (G, g)$ is a plane graph and $\eta$ is a $G^g$-noose, then we let $G^{g_\eta}_\eta = (G_\eta, g_\eta)$ be the plane graph where

1. $V(G_\eta) = \{v \in V(G) \mid g(v) \in \hat{\eta}\}$,
2. $E(G_\eta) = \{uv \in E(G) \mid g(uv) \subset \hat{\eta}\}$,
3. $g_\eta : V(G_\eta) \cup E(G_\eta) \to \mathbb{R}_2 \cup \mathcal{A}$ with $g_\eta|_{V(G_\eta)} = g|_{V(G_\eta)}$ and $g_\eta|_{E(G_\eta)} = g|_{E(G_\eta)}$.

Intuitively, the graph $G^{g_\eta}_\eta$ is the plane subgraph of $G^g$ that lies in the closed interior of $\eta$.

▶ **Observation 1.** *Let $G^g$ be a plane graph and $\eta_1$ and $\eta_2$ be equivalent $G^g$-nooses. Then $G^{g_{\eta_1}}_{\eta_1} = G^{g_{\eta_2}}_{\eta_2}$.*

▶ **Definition 4.** Let $G^g$ be a plane graph and $\tau$ be a $G^g$-noose type. We say that a plane completion $H^h$ of $G^g_\tau$ is $\tau$-respecting if there is a $G^g$-noose $\eta$ of type $\tau$ which is also a $H^h$-noose.

We note that in Definition 4, although the $G^g$-noose type of $\eta$ is $\tau$, the $H^h$-noose type of $\eta$ is not necessarily $\tau$ since $H^h$ may have more faces than $G^g$.

## 4 Representative Sets

Let $G^g$ be a plane graph and $\eta$ be a $G^g$-noose. We say that a plane completion $H^h$ of $G^g$ is *$\eta$-respecting* if $\eta$ is also an $H^h$-noose. We say that a plane graph $X^x$ is a $(G^g, \eta)$-completion if the following conditions are satisfied.

1. $\eta$ is a $X^x$-noose.
2. $X^x = X^{x_\eta}_\eta$.
3. $X^x$ is a plane completion of $G^{g_\eta}_\eta$.

Intuitively, a $(G^h, \eta)$-completion is a plane completion $X^x$ of $G^{h_\eta}_\eta$ where all vertices and edges are drawn inside $\eta$.

▶ **Proposition 1.** *If $H^h$ is an $\eta$-respecting plane completion of $G^g$, then $H^{g_\eta}_\eta$ is a $(G^g, \eta)$-completion.*

If $G^g$ is a plane graph and $H^h$ is a plane subgraph of $G^g$ then we let $G^g - H^h$ be the plane graph $Y^y$ where $V(Y) = V(G)$, $E(Y) = E(G) \backslash E(H)$ and $y = g|_{V(Y)}$. In other words, $G^g - H^h$ is the graph obtained by deleting from $G$ the edges which are shared with $H$. On the other hand, let $G^g$ and $H^h$ be plane graphs such that $V(H) \subseteq V(G)$, $Int(h(e)) \cap g(e') = \emptyset$ for every $e \in E(H) \backslash E(G)$ and every $e' \in E(G)$, and such that $h(e) = g(e)$ for every $e \in E(G) \cap E(H)$. We let $G^g + H^h$ be the plane graph $Y^y$ with vertex set $V(Y) = V(G)$, edge set $E(Y) = E(G) \cup E(H)$, and embedding $y$ such that $y|_{V(G)} = g|_{V(G)}$, $y|_{E(G)} = g|_{E(G)}$ and $y|_{E(H)\backslash E(G)} = h|_{E(H)\backslash E(G)}$. In other words, $G^g + H^h$ is the graph obtained by adding all edges in $E(H) \backslash E(G)$ to $G$ and by drawing these edges in the plane according to $h$.

We say that $H^h$ is a $\eta$-respecting diameter-$d$ plane completion of $G^g$ if $H^h$ is a $\eta$-respecting plane completion of $G^g$ of diameter at most $d$.

▶ **Definition 5.** Let $G^g$ be a plane graph and $\eta$ be a $G^g$-noose. Let $A$ and $B$ be sets of $(G^g, \eta)$-completions. We say that $A$ represents $B$ if for every diameter-$d$ $\eta$-respecting plane completion $H^h$ of $G^g$, such that $H^{h_\eta}_\eta \in B$, there exists some $X^x \in A$ such that $(H^h - H^{h_\eta}_\eta) + X^x$ is also a diameter-$d$ plane completion of $G^g$.

Let $G^g$ be a plane graph and $\eta$ be a $G^g$-noose. We let $V(\eta) = \{v \in V(G) \mid g(v) \in \eta\}$ be the set of vertices of $G$ whose image under $g$ lies on $\eta$, and $\hat{V}(\eta)$ be the set of vertices of $G$ that lie in the closed interior of $\eta$.

Let $X^x$ be a $(G^g, \eta)$-completion. The truncated distance between any two vertices $v, v'$ of $\hat{V}(\eta)$, denoted by $d(X^x, v, v')$ is defined as the distance between $v$ and $v'$ in $X^x$ if this distance is at most $d$, and $\infty$ otherwise. We let $D(X^x) = [d(X^x, v, v')]_{v,v' \in V(\eta)}$, be the matrix of truncated distances between any two vertices in $V(\eta)$. For any vertex $v$ in $\hat{V}(\eta)$, we let $D(X^x, v) = [d(X^x, v, v')]_{v' \in V(\eta)}$ be the vector of distances between $v$ and the vertices whose image lie in the noose $\eta$. We say that two vertices $u$ and $u'$ in $\hat{V}(\eta)$ are unresolved if their distance in $X^x$ is greater than $d$. For each pair of unresolved vertices we let $D(X^x, u, u') = (D(X^x, u), D(X^x, u'))$ be the pair of distance vectors from $u$ to $V(\eta)$ and from $u'$ to $V(\eta)$ respectively. We let

$$\mathbb{D}(X^x) = \{D(X^x, u, u') \mid (u, u') \text{ is an unresolved pair}\}.$$

The signature of $X^x$ is defined as follows.

$$\mathcal{S}(X^x) = (D(X^x), \mathbb{D}(X^x)). \tag{1}$$

▶ **Proposition 2.** *Let $|V(\eta)| \leq 8d$. Then there exist at most $2^{d^{O(d)}}$ distinct signatures.*

▶ **Lemma 6.** *Let $H^h$ be a plane completion of $G^g$ of diameter at most $d$ and let $X^x$ be a $(G^g, \eta)$-completion such that $\mathcal{S}(H^{h_\eta}_\eta, \eta) = \mathcal{S}(X^x, \eta)$. Then if $H^g$ has diameter at most $d$, $(H^h - H^{h_\eta}_\eta) + X^x$ has diameter at most $d$.*

▶ **Lemma 7.** *Let $G^g$ be a plane graph, $\eta$ be a $G^g$-noose and $B$ be a set of $(G^g, \tau)$-completions. Then one can construct in time $2^{d^{O(d)}} \cdot |B|$ a set $Trunc(B)$ of $(G^g, h)$-completions such that $|Trunc(B)| \leq d^{O(d)}$ and $Trunc(B)$ represents $B$.*

**Proof.** Let $B$ be a set of $(G^g, \eta)$ completions. For each signature $S$, let $B(S)$ be the set of all graphs $H^h$ in $B$ with $\mathcal{S}(H^h, \eta) = S$. Finally, let $Trunc(B)$ be the set obtained by selecting a unique graph $H^h_S$ from set $B(S)$ whenever $B(S)$ is non-empty. Then by Lemma 6, $Trunc(B)$ represents $B$. ◀

## 5 Algorithm for Embedded Planar Diameter Improvement

In this section we will devise an algorithm that takes a plane graph $G^g$ and a positive integer $d$ as input and determines in time $|V(G)|^{O(d)}$ whether $G^g$ has a diameter-$d$ plane completion $H^h$. In the reminder of this section we assume that the input plane graph $G^g$ and the input positive integer $d$ are fixed.

▶ **Definition 8.** Let $\eta$ be a $G^g$-noose . We define the following set.

$$\mathcal{F}_\eta = \{H^{h_\eta}_\eta \mid H^h \text{ is a diameter-}d \text{ plane completion of } G^g\}.$$

Intuitively, $\mathcal{F}_\eta$ is the set of all plane completions of the graph $G^{g_\eta}_\eta$ that can be extended to some diameter-$d$ plane completion of the graph $G^g$. For each $G^g$-noose $\eta$ we will define a family $\tilde{\mathcal{F}}_\eta \subseteq \mathcal{F}_\eta$ of $(G^g, \eta)$-completions such that $\tilde{\mathcal{F}}_\eta$ represents $\mathcal{F}_\eta$ and $|\tilde{\mathcal{F}}_\eta| \leq 2^{d^{O(d)}}$.

We will define a partial order on nooses as follows. First, for each noose $\eta$, we consider the following triple.

$$\phi(\eta) = [|V(G^{g_\eta}_\eta)|, |E(G^{g_\eta}_\eta)|, -|\eta|].$$

We set $\eta < \eta'$ if and only if $\phi(\eta) < \phi(\eta')$. In other words, a noose $\eta$ is smaller than a noose $\eta'$ if the closed interior of $\eta$ has less vertices than the closed interior of $\eta'$, or if these interiors have the same number of vertices, but the first has less edges, or if both interiors have the same number of edges and vertices and the first noose has more vertices than the latter nose. Note that there is an inversion in the third coordinate, since the larger the noose-size, the lesser is the order.

We will compute $\tilde{\mathcal{F}}_\eta$ under the assumption that $\tilde{\mathcal{F}}_{\eta'}$ has been computed for every $\eta'$ such that $\phi(\eta') < \phi(\eta)$. There are three cases to be considered. In all cases the size of the involved nooses will be at most $8d$. In the first case, assuming that the size of $\eta$ is at most $8d$, we will show how to compute $\tilde{\mathcal{F}}_\eta$ under the assumption that $\tilde{\mathcal{F}}_{\eta'}$ has been computed for every noose $\eta'$ whose closed interior has fewer edges than the one of $\eta$. The second case concerns nooses of size strictly less than $8d$. In this case, we will show how to compute $\tilde{\mathcal{F}}_\eta$ assuming that we have computed $\tilde{\mathcal{F}}_{\eta'}$ for every noose $\eta'$ of size $|\eta| + 1$ whose closed interior is identical to the one of $\eta$. Note that due to the negative sign in the definition of $\phi(\eta)$, if two nooses $\eta$ and $\eta'$ have identical closed interior and $|\eta'| > |\eta|$, then $\phi(\eta') < \phi(\eta)$. Finally, the most important case is the third, in which nooses have size exactly $8d$. In this case we show how to compute $\tilde{\mathcal{F}}_\eta$ assuming we have computed $\tilde{\mathcal{F}}_{\eta'}$ for every noose of size $8d$ whose closed interior is strictly contained in the closed interior of $\eta$. We note that we only need to consider one noose $\eta_\tau$ for each noose-type $\tau$. Therefore, the number of nooses to be considered will be upper bounded by $n^{O(d)}$.

**Case One.** Let $G^g$ be a plane graph and $\eta$ be a $G^g$-noose with $|\eta| < 8d$. We say that an edge $uv \in E(G)$ is *parallel* to $\eta$ if there exists a simple arc $\ell \in \mathcal{A}$ such that the following conditions are satisfied.

1. $\ell \subseteq \eta$.
2. $endpts(\ell) = \{g(u), g(v)\}$.
3. Let $\alpha = \ell \cup g(uv)$ and let $\hat{\alpha}$ be the closed interior of $\alpha$.
   a. $\hat{\alpha} \cap g(V(G)) = \{g(u), g(v)\}$.
   b. $\hat{\alpha} \cap g(u'v') \subseteq \{g(u), g(v)\}$ for each $u'v' \in E(G)$.

In other words, $uv$ is parallel to $\eta$ if there is a simple arc $\ell \subset \eta$ with endpoints $\{g(u), g(v)\}$ such that the closed interior of the curve $\ell \cup g(uv)$ only intersects the drawing of $G$ in the points of $g(u, v)$ (Figure 4).

We say that an edge $uv \in E(G)$ is *internally* (resp. *externally*) parallel to $\eta$ if $uv$ is parallel to $\eta$ and $g(uv) \subset \hat{\eta}$ (resp. $g(uv) \cap \hat{\eta} = \{g(u), g(v)\}$). We note that if an edge $uv$ is parallel to $\eta$, then $uv$ is either internally or externally parallel to $\eta$. Let $G^g$ be a plane graph and $uv \in E(G)$. We let $G^g - uv$ be the plane graph which is obtained by deleting the edge $uv$ from $E(G)$ and by restricting the mapping $g$ to the remaining edges.

▶ **Proposition 3.** *Let $\eta$ be a $G^g$-noose, and let $uv$ be an edge in $E(G)$ such that $uv$ is internally parallel to $\eta$. Then there exists a $G^g$-noose $\eta'$ such that $V(\eta) = V(\eta')$, $uv$ is externally parallel to $\eta'$ and $G^{g_{\eta'}}_{\eta'} = G^{g_\eta}_{\eta} - uv$. (See Figure 4).*

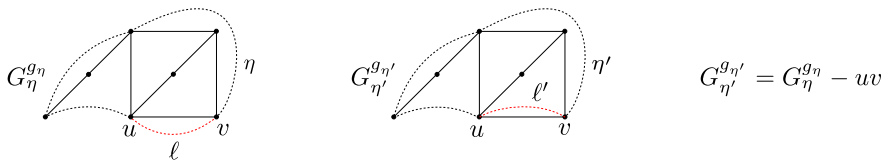



**Figure 4** The edge $uv$ is internally parallel to $\eta$ and externally parallel to $\eta'$. Note that the graph $G^{g_{\eta'}}_{\eta'}$ is equal to the graph $G^{g_\eta}_{\eta}$ minus the edge $uv$. Intuitively, the noose $\eta'$ is obtained from $\eta$ by deleting the arc $\ell$ and by gluing the arc $\ell'$ in its place.

▶ **Lemma 9.** *Let $uv$ be an edge in $E(G)$, and let $\eta$ be a $G^g$-noose such that $uv$ is internally parallel to $\eta$. Let $\eta'$ be a $G^g$-noose such that $V(\eta) = V(\eta')$, $G^{g_{\eta'}}_{\eta'} = G^{g_\eta}_{\eta} - uv$ and $uv$ is externally parallel to $\eta'$. Note that such a noose $\eta'$ exists by Proposition 3. Suppose that $X^x \in \mathcal{F}_\eta$. Then there exists $Y^y \in \mathcal{F}_{\eta'}$ such that $Y^y = X^x - uv$.*

Let $Y^y$ be a $(G^g, \eta)$-completion and $uv$ be an edge in $E(G) \backslash E(Y)$ such that $u, v \in V(\eta)$. We let $Y^y + uv$ be the plane graph $X^x$ where $V(X) = V(Y)$, $E(X) = E(Y)$, $x|_{V(G) \cup E(G)} = y$ and $x(uv) = g(uv)$.

▶ **Lemma 10.** *Let $uv$ be an edge in $E(G)$, and let $\eta$ and $\eta'$ be $G^g$-nooses such that $V(\eta) = V(\eta')$ and $G^g_\eta = G^g_{\eta'} + uv$. Assume that $\tilde{\mathcal{F}}_{\eta'}$ represents $\mathcal{F}_{\eta'}$. Then $\tilde{\mathcal{F}}_\eta = \tilde{\mathcal{F}}_{\eta'} + uv$ represents $\mathcal{F}_\eta$.*

**Case Two.** Let $k < 8d$ and let $\eta$ be a $G^g$-noose of type $\tau = [v_0 F_0 v_1 F_1 ... v_{k-1} F_{k-1}]$. A trivial extension of $\eta$ is a noose $G^g$-noose $\eta'$ of type

$$\tau = [v_0 F_0 v_1 F_1 ... v_j F_j u F_j v_{j+1} ... v_{k-1} F_{k-1}]$$

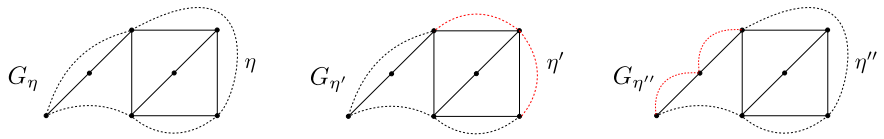



**Figure 5** The nooses $\eta'$ and $\eta''$ are trivial extensions of $\eta$. The red dashed region indicates where a new vertex was added. Note that $G^g_\eta = G^g_{\eta'} = G^g_{\eta''}$.

for some $j \in \{0, ..., k-1\}$ and some $u$ which belongs to face $F_j$ and to the closed interior of $\eta$. Note that if $\eta'$ is a trivial extension of $\eta$, then $\eta$ and $\eta'$ have identical closed interiors. In Figure 5 we depict two trivial extensions $\eta'$ and $\eta''$ of a noose $\eta$.

We say that a noose $\eta$ is extensible if $|V(\eta)| < 8d$ and if $\eta$ has at least one trivial extension. We let $\mathrm{Ext}(\eta)$ be the set of (equivalence classes of) trivial extensions of $\eta$. It is immediate that for each extensible noose $\eta$, $\mathcal{F}_\eta = \bigcup_{\eta' \in \mathrm{Ext}(\eta)} \mathcal{F}_{\eta'}$. Based on this equality, for each extensible noose $\eta$, we define the set $\hat{\mathcal{F}}_\eta = \bigcup_{\eta' \in \mathrm{Ext}(\eta)} \tilde{\mathcal{F}}_{\eta'}$.

▶ **Lemma 11.** *Let $\eta$ be an extensible noose. Suppose that $\tilde{\mathcal{F}}_{\eta'}$ represents $\mathcal{F}_{\eta'}$ for every $\eta' \in \mathrm{Ext}(\eta)$. Then $\hat{\mathcal{F}}_\eta$ represents $\mathcal{F}_\eta$.*

Note that the number of extensions of a noose $\eta$ may be linear in $|V(G)|$. Therefore, the number of plane graphs in $\hat{\mathcal{F}}_\eta$ may be linear in $|V(G)|$. To decrease the size of this family, we apply the *Trunc* operator introduced in Section 4.

▶ **Lemma 12.** *Let $\eta$ be an extensible noose and let $\tilde{\mathcal{F}}_\eta = Trunc(\hat{\mathcal{F}}_\eta)$. Then $\tilde{\mathcal{F}}_\eta$ represents $\mathcal{F}_\eta$.*

**Case Three.** In this section we will deal with the case in which $|\eta| = 8d$. Let $G^g$ be a plane graph. A face-vertex sequence is a sequence $X = F_1 v_1 F_2 ... v_{r-1} F_r$ where $r \geq 1$, $\{v_1, ..., v_{r-1}\} \subseteq V(G)$, $\{F_1, ..., F_r\} \subseteq F(G)$, and for each $i \in \{1, ..., r-2\}$, $v_i$ and $v_{i+1}$ are in $F_i$. We note that any noose type $\tau = [v_0 F_0 v_1 ... v_m F_m]$ where $m \geq 1$ can be written as $\tau = [v_0 X v_r Y]$ where $X = F_1 v_1 F_2 ... v_{r-1} F_{r-1}$ and $Y = F_r v_{r+1} F_{r+1} ... v_m F_m$ are face-vertex sequences.

The *reverse* of a face-vertex sequence $X = F_1 v_1 F_2 ... v_{r-1} F_r$ is the face-vertex sequence $X^R = F_r v_{r-1} ... F_2 v_1 F_1$. Let $\tau_1$ and $\tau_2$ be noose types. We say that $\tau_1$ and $\tau_2$ are *summable* if there is a unique maximal[7] face-vertex sequence $X = X(\tau_1, \tau_2)$ with the property that there exist vertices $v, v'$ and face-vertex sequences $Y$ and $Z$ such that $\tau_1 = [vYv'X]$ and $\tau_2 = [v'ZvX^R]$. If this is the case, the sum of $\tau_1$ with $\tau_2$ is defined as $\tau_1 \oplus \tau_2 = [vYv'Z]$. We let $V(X(\tau_1, \tau_2))$ denote the vertices which lie in the face vertex sequence $X(\tau_1, \tau_2)$.

We note that a noose $\eta$ has type $\tau_1 \oplus \tau_2$ if and only if there exist vertices $v, v'$ and simple arcs $\ell_1, \ell'_1, \ell_2, \ell'_2$ with endpoints $\{v, v'\}$ satisfying the following properties.

1. $\ell_1 \cap \ell'_1 = \ell_2 \cap \ell'_2 = \ell_1 \cap \ell_2 = \{v, v'\}$.
2. $g(V(X(\tau_1, \tau_2))) \subseteq \ell'_1 \cap \ell'_2$.
3. $\eta = \ell_1 \cup \ell_2$.
4. $\eta_1 = \ell_1 \cup \ell'_1$ is a noose of type $\tau_1$.
5. $\eta_2 = \ell_2 \cup \ell'_2$ is a noose of type $\tau_2$.

Intuitively, $\eta$ is obtained from $\eta_1$ and $\eta_2$ by the following process. First, we delete the interior of $\ell'_1$ from $\eta_1$ to obtain the segment $\ell_1$, and we delete the interior of $\ell'_2$ from $\eta_2$ to

[7] Maximal with respect to length.

obtain the segment $\ell_2$. Subsequently, we glue $\ell_1$ with $\ell_2$ along their common endpoints in order to obtain the noose $\eta$. We note that if $\eta = \eta_1 \oplus \eta_2$ then $G_\eta^{g_\eta} = G_{\eta_1}^{g_{\eta_1}} \cup G_{\eta_2}^{g_{\eta_1}}$.

Let $\eta$ be a $G^g$-noose with $|\eta| = 8d$ and let $H^h$ be a triangulated $\eta$-respecting diameter-$d$ plane completion of $G^g$. Let $V_1 \dot\cup V_2 \dot\cup V_3 \dot\cup V_4$ be a partition of $V(\eta)$ where for each $i \in \{1,2,3,4\}$, $V_i$ has $d$ consecutive vertices. Let $\hat V_1 \dot\cup \hat V_2 \dot\cup \hat V_3 \dot\cup \hat V_4$ be a partition of the vertex set of $H_\eta^{h_\eta}$ where for each $v \in V(H_\eta^{h_\eta})$,

$$v \in \hat V_i \;\Leftrightarrow\; \left[ dist(v, V_i) < \min_{j<i} dist(v, V_j) \;\wedge\; dist(v, V_i) \leq \min_{j>i} dist(v, V_j) \right].$$

Intuitively, the set of vertices incident with the noose $\eta$ is partitioned into four consecutive sections of size $d$: north ($V_1$), east ($V_2$), south ($V_3$) and west ($V_4$). Subsequently, each vertex $v \in V(G_\eta^g)$ is classified as being north ($\hat V_1$), east ($\hat V_2$), south ($\hat V_3$) or west ($\hat V_4$) according to whether $v$ is closer to a northern, eastern, southern or western vertex from the noose. In the case in which a vertex $v$ is as close to $V_i$ as it is to $V_j$, for some $i \neq j$, ties are broken by considering that north is smaller than east, which is smaller than south, which is smaller than west. We note that the way in which we have decided to break ties is completely arbitrary.

▶ **Lemma 13.** *There is an edge $uv \in E(H_\eta^{h_\eta})$ such that either $u \in \hat V_1$ and $v \in \hat V_3$, or $u \in \hat V_2$ and $v \in \hat V_4$.*

▶ **Lemma 14.** *At least one of the following statements must be satisfied.*

1. *There is a path of length at most $2d+1$ between a vertex in $V_1$ to a vertex in $V_3$.*
2. *There is a path of length at most $2d+1$ between a vertex in $v_2$ and a vertex in $v_4$.*

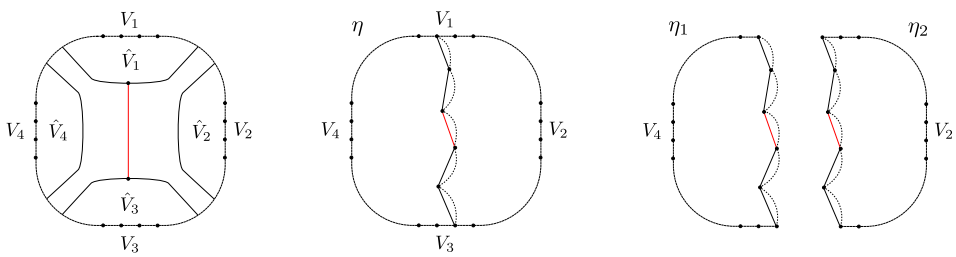



**Figure 6** Left: A noose $\eta$ with $8d$ vertices, and the sets $V_i$ and $\hat V_i$. Either there is an edge from a maximal element of $\hat V_1$ to a maximal element of $\hat V_3$, or an edge between a maximal element of $\hat V_2$ and a maximal element of $\hat V_4$. Middle: A path from $V_1$ to $V_3$ is depicted. Right: the path from $V_1$ to $V_3$ can be used to show that $\eta = \eta_1 \oplus \eta_2$ for nooses $\eta_1$ and $\eta_2$ where can be split into nooses $\eta_1 \oplus \eta_2$.

▶ **Lemma 15.** *Let $H^h$ be a $\eta$-respecting diameter-$d$ plane triangulated completion of $G^g$. Then there exist $G^g$-nooses $\eta_1$ and $\eta_2$ satisfying the following properties.*

1. *$|V(\eta_1)| \leq 8d$ and $|V(\eta_2)| \leq 8d$.*
2. *$\eta = \eta_1 \oplus \eta_2$.*
3. *$H^h$ is both $\eta_1$-respecting and $\eta_2$-respecting.*
4. *$H_\eta^{h_\eta} = H_{\eta_1}^{h_{\eta_1}} \cup H_{\eta_2}^{h_{\eta_2}}$.*

Let $\eta$, $\eta_1$ and $\eta_2$ be $G^g$ nooses such that $\eta = \eta_1 \oplus \eta_2$. Then we define the following set: $\mathcal{F}_{\eta_1} \oplus \mathcal{F}_{\eta_2} = \{X^x \cup Y^y \mid X^x \in \mathcal{F}_{\eta_1},\ Y^y \in \mathcal{F}_{\eta_2}\}$. Then, Lemma 15 implies that whenever $|\eta| = 8d$, $\mathcal{F}_\eta = \bigcup_{\eta=\eta_1\oplus\eta_2} \mathcal{F}_{\eta_1} \oplus \mathcal{F}_{\eta_2}$.

▶ **Lemma 16.** *Let $\eta$ be a $G^g$-noose with $|\eta| = 8d$, and assume that for every two $G^g$-nooses $\eta_1$ and $\eta_2$ such that $\eta = \eta_1 \oplus \eta_2$ we have that $\tilde{\mathcal{F}}_{\eta_1}$ represents $\mathcal{F}_{\eta_1}$ and $\tilde{\mathcal{F}}_{\eta_2}$ represents $\mathcal{F}_{\eta_2}$. Then $\mathcal{F}_\eta$ is represented by the following set.*

$$\tilde{\mathcal{F}}_\eta = \mathit{Trunc}\left(\bigcup_{\eta=\eta_1\oplus\eta_2} \tilde{F}_{\eta_1} \oplus \tilde{F}_{\eta_2}\right) = \mathit{Trunc}\left(\bigcup_{\eta=\eta_1\oplus\eta_2} \{X^x \cup Y^y \mid X^x \in \tilde{\mathcal{F}}_{\eta_1},\ Y^y \in \tilde{\mathcal{F}}_{\eta_2}\}\right) \quad (2)$$

**Algorithm.** Now we summarize our algorithm for determining whether a given plane graph $G^g$ admits a plane completion of diameter at most $d$. We assume that the graph has at least $d$ vertices, since otherwise the graph has trivially a completion of diameter at most $d$. As a first step we enumerate all combinatorial faces of $G^g$, constructing in this way the set $F(G^g)$. We note that there exists at most $O(|V(G)|)$ such faces, and the set $F(G^g)$ can clearly be constructed in time $|V(G)|^{O(1)}$. As a second step, we enumerate all noose types containing at most $8d$ vertices. In other words, we enumerate all cycles $\tau = [v_0F_1v_1...v_{r-1}F_r]$ where $r-1 \leq 8d$, and verify if $\tau$ is a valid noose type. Since there are at most $O(|V(G)|)$ combinatorial faces, there are at most $|V(G)|^{O(d)}$ possible noose types with at most $8d$ vertices. Let $\mathcal{T}$ be the set of all such noose types. Now, for each noose type $\tau \in \mathcal{T}$, we select an arbitrary noose $\eta_\tau$ of type $\tau$. We say that $\eta_\tau$ is a representative for $\tau$. Let $\mathcal{N} = \{\eta_\tau\}_{\tau\in\mathcal{T}}$ be the set of all such nooses.

For each noose $\eta \in \mathcal{N}$, we will compute a set $\tilde{\mathcal{F}}_\eta$ containing at most $2^{d^{O(d)}}$ triangulated plane completions of the graph $G_\eta^{g_\eta}$. In particular the set $\tilde{\mathcal{F}}_\eta$ represents the set $\mathcal{F}_\eta$. The sets $\tilde{\mathcal{F}}_\eta$ are computed by dynamic programming.

In the base case, let $\mathcal{N}^0 \subseteq \mathcal{N}$ be the minimal elements of $\mathcal{N}$ with respect to the noose ordering defined in the beginning of Section 5. For each such minimal noose $\eta^0$, we have that $|V(\eta^0)| = 3$, and the graph $G_\eta^{g_\eta}$ has no edges. Therefore the set $\tilde{\mathcal{F}}_\eta$ can be constructed in constant time in this case.

Now assume that we are dealing with a non-minimal noose $\eta \in \mathcal{N}$ and that $\tilde{\mathcal{F}}_{\eta'}$ has been constructed for every $\eta' < \eta$. We will show how to construct $\tilde{\mathcal{F}}_\eta$. There are three cases to be considered.

1. If there is some edge $uv \in E(G)$ which is externally parallel to $\eta$, then we consider a noose $\eta'$ such that $G_{\eta'}^{g_{\eta'}} = G_\eta^{g_\eta} - uv$. Then we set $\tilde{\mathcal{F}}_\eta = \tilde{\mathcal{F}}_{\eta'} + uv$.
2. If $\eta$ is trivially extensible, then we let $\tilde{\mathcal{F}}_\eta = \mathit{Trunc}\left(\bigcup_{\eta'\in\mathrm{Ext}(\eta)} \tilde{\mathcal{F}}(\eta')\right)$.
3. If $|\eta| = 8d$, then we let $\tilde{\mathcal{F}}_\eta = \mathit{Trunc}\left(\bigcup_{\eta=\eta_1\oplus\eta_2} \tilde{\mathcal{F}}_{\eta_1} \oplus \tilde{\mathcal{F}}_{\eta_2}\right)$

Now let $\eta$ be a maximal noose in $\mathcal{N}$. Then we have that $|\eta| = 3$ and that the graph $G_\eta^{g_\eta} = G^g$. Therefore, we have that $G^g$ admits a diameter-$d$ plane completion if and only if the set $\tilde{\mathcal{F}}_\eta$ is non-empty. Additionally, if this is the case, then any graph in $\tilde{\mathcal{F}}_\eta$ is a diameter-$d$ plane completion of $G^g$.

Note that in any of the three cases above the time necessary to construct the set $\tilde{\mathcal{F}}_\eta$ from previously computed $\tilde{\mathcal{F}}_{\eta'}$ is at most $2^{d^{O(d)}} \cdot |V(G)|^{O(d)}$, since there are at most $|V(G)|^{O(d)}$ nooses in $\mathcal{N}$ and each $\tilde{\mathcal{F}}_{\eta'}$ has at most $2^{d^{O(d)}}$ elements. This implies that the computation of $\tilde{\mathcal{F}}_\eta$ for every $\eta \in \mathcal{N}$ also takes time at most $2^{d^{O(d)}} \cdot |V(G)|^{O(d)}$, as stated in Theorem 1.

***k*-Outerplanar Plane Diameter Improvement.** The algorithm for solving the embedded $k$-outerplanar diameter improvement problem is almost identical to the one to solve the embedded planar diameter improvement problem. The only difference is that, since the input graph is $k$-outerplanar, we only need to consider nooses of size at most $8k$, instead of $8d$ as in the previous algorithm.

In particular, the proof of Lemma 14 may be adapted in such a way that if we split the set $V(\eta)$ into sets $V_1, V_2, V_3, V_4$ as done previously, then there is either a path of length at most $2k$ between some vertex in $V_1$ and some vertex in $V_3$, or a path of length at most $2k$ between some vertex in $V_3$ and some vertex in $V_4$. As a consequence, the value $8d$ in Lemma 15 may be replaced with $8k$ when considering $k$-outerplanar graphs. With these adaptations our algorithm for the embedded $k$-outerplanar diameter improvement problem is guaranteed to run in time $2^{d^{O(d)}} \cdot |V(G)|^{O(k)}$ as stated in Theorem 2.

## References


1. Isolde Adler, Martin Grohe, and Stephan Kreutzer. Computing excluded minors. In *Proc. of SODA 2008*, pages 641–650. SIAM, 2008.
2. Vincent Bouchitté, Frédéric Mazoit, and Ioan Todinca. Chordal embeddings of planar graphs. *Discrete Mathematics*, 273(1):85–102, 2003.
3. Vincent Bouchitté and Ioan Todinca. Treewidth and minimum fill-in: Grouping the minimal separators. *SIAM Journal on Computing*, 31(1):212–232, 2001.
4. Nathann Cohen, Daniel Gonçalves, Eun Jung Kim, Christophe Paul, Ignasi Sau, Dimitrios M Thilikos, and Mathias Weller. A polynomial-time algorithm for outerplanar diameter improvement. *Journal of Computer and System Sciences*, 2017.
5. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
6. Mateus de Oliveira Oliveira. On Supergraphs satisfying CMSO properties. In *Proc. of the 26th Conference on Computer Science Logic (CSL 2017). To appear.*, volume 82 of *LIPIcs*, 2017.
7. Frederic Dorn, Eelko Penninkx, Hans L Bodlaender, and Fedor V Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica*, 58(3):790–810, 2010.
8. Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.
9. Michael R. Fellows and Rodney G. Downey. Parameterized Computational Feasibility. *Feasible Mathematics II*, 13:219–244, 1995.
10. Michael R Fellows and Michael A Langston. On search decision and the efficiency of polynomial-time algorithms. In *Proc. of STOC 1989*, pages 501–512. ACM, 1989.
11. Fedor V Fomin and Dimitrios M Thilikos. A simple and fast approach for solving problems on planar graphs. In *STACS*, volume 2996, pages 56–67. Springer, 2004.
12. Fedor V Fomin, Ioan Todinca, and Yngve Villanger. Large induced subgraphs via triangulations and CMSO. *SIAM Journal on Computing*, 44(1):54–87, 2015.
13. Neil Robertson and Paul D Seymour. Graph minors. XI. Circuits on a surface. *Journal of Combinatorial Theory, Series B*, 60(1):72–106, 1994.
14. Neil Robertson and Paul D Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of combinatorial theory, Series B*, 63(1):65–110, 1995.
15. Neil Robertson and Paul D Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004.

# The PACE 2018 Parameterized Algorithms and Computational Experiments Challenge: The Third Iteration

**Édouard Bonnet**
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France
edouard.bonnet@dauphine.fr

**Florian Sikora**
Université Paris-Dauphine, PSL University, CNRS, LAMSADE, 75016, Paris, France
florian.sikora@dauphine.fr

**Abstract**

The Program Committee of the Third Parameterized Algorithms and Computational Experiments challenge (PACE 2018) reports on the third iteration of the PACE challenge. This year, all three tracks were dedicated to solve the Steiner Tree problem, in which, given an edge-weighted graph and a subset of its vertices called *terminals*, one has to find a minimum-weight subgraph which spans all the terminals. In Track A, the number of terminals was limited. In Track B, a tree-decomposition of the graph was provided in the input, and the treewidth was limited. Finally, Track C welcomed heuristics. Over 80 participants on 40 teams from 16 countries submitted their implementations to the competition.

**2012 ACM Subject Classification** Theory of computation → Fixed parameter tractability

**Keywords and phrases** Steiner tree problem, contest, implementation challenge, FPT

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.26

**Funding** Partially supported by the LABEX MILYON (ANR-10- LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR) and project "ESIGMA" (ANR-17-CE23-0010).

**Acknowledgements** The PACE challenge was supported by Networks [35], an NWO Gravitation project of the University of Amsterdam, Eindhoven University of Technology, Leiden University, and the Center for Mathematics and Computer Science (CWI), by `data-experts.de`, and by *Freunde der Saarbrücker Informatik* (`FdSI`). The prize money (3750 €) and travel grants (1000 € for Daniel Rehfeldt who presented SCIP-Jack during the award ceremony) were given through the generosity of Networks [35] and `data-experts.de`. We are grateful to Szymon Wasik and Jan Badura for the fruitful collaboration and for hosting the competition at `optil.io`.

## 1 Introduction

The Parameterized Algorithms and Computational Experiments Challenge (PACE) was conceived in Fall 2015 to deepen the relationship between parameterized algorithms and practice. It aims to:

1. Bridge the divide between the theory of algorithm design and analysis, and the practice of algorithm engineering.
2. Inspire new theoretical developments.
3. Investigate in how far theoretical algorithms from parameterized complexity and related fields are competitive in practice.

© Édouard Bonnet and Florian Sikora;
licensed under Creative Commons License CC-BY
13th International Symposium on Parameterized and Exact Computation (IPEC 2018).
Editors: Christophe Paul and Michał Pilipczuk; Article No. 26; pp. 26:1–26:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

4. Produce universally accessible libraries of implementations and repositories of benchmark instances.
5. Encourage the dissemination of these findings in scientific papers.

The first iteration of PACE was held at IPEC 2016 [13]. Since then, PACE has been mentioned as an inspiration in many papers [1, 3, 21, 22, 25, 31, 33, 41, 42]. Hisao Tamaki's companion theory paper to his winning implementations of PACE 2017 [14] won the best paper award at ESA 2017 [40]. Very recent works acknowledging PACE or inspired by the challenge include the following papers [5, 7, 32, 20, 4, 17]. At this year's ESA, it was shown that parallelized dynamic programming (DP) on tree decompositions is a competitive approach to SAT solving [23]. According to the paper, this was made possible by the excellent programs developed for the treewidth tracks of PACE 2016 and 2017. At the same conference, Bannach and Brendt provided an efficient and simple interface to DP on tree decompositions [2].

In this article, we report on the third iteration of PACE. The PACE 2018 challenge was announced on November 14, 2017. The final version of the submissions was due on May 12, 2018. We informed the participants of the result on May 14, and announced them to the public on August 22nd, during the award ceremony at the International Symposium on Parameterized and Exact Computation (IPEC 2018) in Helsinki.

## 2 The Steiner Tree Problem: Theory and Practice

There are many different variants of the Steiner Tree problem. They all involve the task to interconnect objects called *terminals* by using minimum-length *wires*. Here, we focus on the main variant in graphs. For a subset of edges $F$ of an undirected graph, let $V(F)$ denote the set of vertices that are endpoints of those edges. A formal definition of the Steiner Tree problem can be stated as follows.

**Input:** An undirected graph $G$ with a non-negative weight function on its edges $w : E(G) \to \mathbb{R}^+$, and a set of *terminals* $T \subseteq V(G)$.

**Output:** A subset of edges $F \subseteq E(G)$ minimizing $\sum_{e \in F} w(e)$ such that: $T \subseteq V(F)$ holds and $(V(F), F)$ is a connected graph.

We will interchangeably regard $F$ and $(V(F), F)$ as the solution. In words, given an edge-weighted graph and a subset of its vertices (the *terminals*) the task is to find a minimum-weight subgraph that spans all terminals. As the weights are non-negative, any edge of a cycle in a feasible solution can be removed to obtain a solution that is still feasible but has a smaller or equal total weight. Hence, there is always a tree among the optimum solutions, which explains the second word, *Tree*, in the problem name. The first word, *Steiner*, refers to the 19th-century geometer Jakob Steiner.

We chose Steiner Tree to be the problem of PACE 2018 since it has a wide spectrum of real-life applications, including the design of VLSI, optical or wireless communication systems, and transportation networks [30]. Furthermore, as a result of the 11th DIMACS implementation challenge, there are established benchmarks for Steiner Tree and its variants. Finally, as we will see in the next section, this problem has interesting FPT algorithms.

### 2.1 Parameterized algorithms

In what follows, we denote the number of edges of an input graph $G$ by $n$, its number of edges by $m$, its number of terminals by $t$, and its treewidth by $w$. *Steiner vertices* are the non-terminal vertices touched by a solution $F \subseteq E(G)$, and we denote their number by $s$,

that is, we have $s = |V(F) \setminus T|$. As we will see, there are FPT algorithms for Steiner Tree with $t$ as a parameter and with $w$ as a parameter, that is, algorithms running in time $f(t)n^{O(1)}$ and $g(w)n^{O(1)}$ for some computable functions $f$ and $g$.

### The Dreyfus-Wagner algorithm

There is a classical algorithm from the 70s designed by Dreyfus and Wagner [16] (henceforth DW) with running time $O(3^t n + 2^t n^2 + n(n \log n + m))$. This was later improved to $O(3^t n + 2^t(n \log n + m))$ by Erickson, Monma, and Veinott [19] (henceforth EMV). It works by dynamic programming on pairs of disjoint subsets of the set of terminals $T$. To keep it simple, in a solution $(V(F), F)$, there is a vertex with degree at least 3 whose removal would disconnect two disjoint sets of terminals $T_1, T_2 \subset T$. One can store a lightest way of connecting $T_1$ and $T_2$ for all the pairs $T_1, T_2$ in a bottom-up manner. The number of pairs of disjoint sets in a universe of $t$ elements is $3^t$, which explains part of the running time of DM and EMV.

One can wonder if the exponential basis of 3 can be decreased. There has been quite a bit of work done in that direction, leading to a series of improvements going from 3 to 2. However, no submission to this year's PACE challenge have made use of these improvements, for good reason: They tend to come with a nasty blow-up in the polynomial factors. For instance, there are $O(2.5^t n^{15})$ and $O(2.1^t n^{58})$ time algorithms [24]. If the weights are bounded by some constant (and in the variant where the vertices, not the edges, are weighted), there is an algorithm with running time roughly $O(2^t n^2 + nm)$, based on the so-called fast subset convolution [6].

### Treewidth-based algorithms

Given a tree-decomposition $(\mathcal{T}, \{V_u\}_{u \in V(\mathcal{T})})$ of width $w$ of the input graph $G$ (with $\mathcal{T}$, a rooted tree and $\{V_u\}$, a family of subsets of $V(G)$ indexed by the nodes of $\mathcal{T}$), one can solve Steiner Tree in time $O^*(w^w)$ by bottom-up dynamic programming on the nodes of $\mathcal{T}$. Indeed, once the subtree of $\mathcal{T}$ at $u \in V(\mathcal{T})$ is processed, the information to remember, besides the invested budget, is the subset of vertices of $V_u$ touched by a partial solution and how these vertices are connected in the partial solution. This information can be represented by a partition of a subset of $V_u$. The number of such objects is bounded by $O(w^w)$ since $|V_u| \leqslant w + 1$, hence the size of the DP table and the running time.

For some time it was an open question if Steiner Tree and similar problems with a connectivity constraint can be solved in time $2^{O(w)} n^{O(1)}$ instead of $2^{O(w \log w)} n^{O(1)}$. This was finally resolved in 2011 with the Cut&Count technique of Cygan et al. [12]. In a nutshell, the Cut&Count technique starts by addressing the parity version of the problem: how many sets of $\ell$ connected edges span $T$, modulo 2? For this, one counts the number $p$ of pairs $(D, \mathcal{C})$ where $D$ is a set of at most $\ell$ edges spanning $T$ (and $D$ is not required to be connected) and $\mathcal{C}$ is a cut such that every connected component of $D$ is fully contained in one side of $\mathcal{C}$. If we restrict a designated terminal to always be on the *left* side of $\mathcal{C}$, the number of possible $\mathcal{C}$ for a given $D$ is $2^{\mathrm{cc}(D)-1}$, where $\mathrm{cc}(D)$ is the number of connected components of $D$. This number is 1 if $D$ is connected and even otherwise. So, the parity of $p$ is the answer to our question. Since computing $p$ does not exhibit a (*global*) connectivity constraint, it can be done in time $2^{O(w)} n^{O(1)}$ by standard dynamic programming. Finally, one can distinguish the case *zero solutions* from the case *non-zero even number of solutions* by using the Isolation Lemma. This lemma implies that by giving uniformly random integer weights between 1 and $100\ell m$ to edges, there will be, with high probability, exactly one solution of minimum total

weight (provided, of course, that there was a solution to begin with). For this total weight, we will therefore find an odd number of solutions, and may confidently report it.

One might then ask for a deterministic $2^{O(w)}n^{O(1)}$ time algorithm, which could handle weighted instances. The rank-based approach of Bodlaender et al. [8] does exactly that. The main idea is that when performing the standard $O^*(w^w)$ DP, one does not need to keep *every* partition of a bag. Instead, one can prune the partitions in the following way. If $\mathcal{S}$ is the set of partitions of a bag achievable by partial solutions of a certain cost, then one can safely restrict attention to a subset $\mathcal{S}' \subseteq \mathcal{S}$, such that: if there is a $\Pi_1 \in \mathcal{S}$ compatible with a partition $\Pi^*$ generated by a partial solution in the rest of the graph, then there should be at least one $\Pi_2 \in \mathcal{S}'$ also compatible with $\Pi^*$. By *compatible*, we mean that the two partitions would fully connect the common subset of the bag touched by the partial solutions. If one represents compatibility by a square matrix in $\mathbb{F}_2$ whose rows and columns are indexed by the partitions (with a 1 in the entries corresponding to compatible partitions, and a 0 otherwise), then it can be observed that any single row/column of a linearly dependent family of rows/columns can be safely removed. Indeed, for every index of the deleted vector containing a 1, there has to be another vector of the family also containing a 1 at this position. Then it is proved that the rank of the matrix with all the partitions is $2^{O(w)}$, and that Gaussian elimination can be done in the same time without having to explicitly store or compute all $O^*(w^w)$ permutations. This three-act story is told with the exact balance of details and conciseness in the book on parameterized complexity by Cygan et al. [11].

### Known theoretical obstacles

The Steiner Tree problem parameterized by the number $s$ of Steiner vertices is $W[2]$-hard [34]. Furthermore it is unlikely that there is an algorithm running in time $O(n^{s-\varepsilon})$, for any $\varepsilon > 0$. Such an algorithm would indeed give, by a simple reduction, an algorithm in $n^{k-\varepsilon}$ for $k$-Dominating Set, contradicting the Strong Exponential Time Hypothesis [39]. Theory also says that one cannot expect very efficient preprocessing: there is no polynomial kernel in $s + t$ (the total number of vertices in $V(F)$) unless the polynomial-time hierarchy collapes [15]. Although, in practice, there are dozens of listed reductions rules which can sometimes completely solve large real-world instances.

### Approximations

We hastily touch the approximability status of Steiner Tree as it might be relevant to heuristics. A minimum spanning tree in the subgraph induced by the terminals of the metric closure yields an approximation ratio of 2 [27]. There is a 1.39-approximation based on a scheme called *iterative randomized rounding* of an LP-formulation [9], and a combinatorial 1.55-approximation [38]. However, Steiner Tree is APX-hard: there is no 1.01-approximation unless P=NP [10].

## 2.2 DIMACS challenge

In 2014, the 11th (and latest) DIMACS implementation challenge was also dedicated to solving Steiner Tree problems, and extensive libraries of instances existed prior to our challenge. As we will detail later, we mainly selected our instances from the Steinlib sets. Furthermore, there were already very efficient programs solving Steiner Tree and its variants. One successful program during the DIMACS competition, SCIP-Jack, also participated in our challenge. SCIP-Jack is a branch-and-cut algorithm based on the free MIP-solver SCIP. As such, it provides a natural baseline and comparison for the FPT approaches.

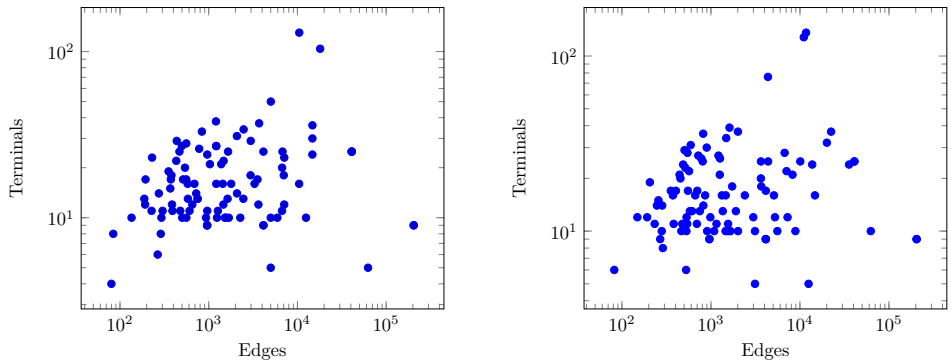



**Figure 1** The distribution (#edges,#terminals) of the public (left) and private (right) instances of Track A in log-log plot.

## 3 Selection of the instances and rules

Before we selected the instances, we first defined the precise goals of each track. In light of the efficient parameterized algorithms described in the previous paragraphs and the objectives of PACE, it will not come as a surprise that we decided that Track A would have instances where the number $t$ of terminals is relatively small, while in Track B the treewidth $w$ of the graphs would be relatively small.

Generating artificial instances for which a careful optimization of the Dreyfus-Wagner algorithm (for Track A) or of the rank-based approach for dynamic programming on tree decompositions (for Track B) would get a decisive advantage compared to a generic solver (ILP, SAT) was not our goal. So, we selected our instances among established benchmarks (mainly Steinlib, PUC, GAP, Vienna), which were also used four years ago in the DIMACS challenge on Steiner Tree.

### 3.1 Track A

For this track, we picked quite a lot of instances with a VLSI application. They are grid graphs with rectangular holes and Manhattan distance weights (in the ALUE, ALUT, DIW, DMXA, GAP, MSM, TAQ, and LIN sets). We used randomly generated sparse instances meant to be resistant to preprocessing (in the E, I160, I640, PUC sets). We also included industrial instances stemming from wire routing problems (in the WRP3 and WRP4 sets), and real geographical networks in the form of complete graphs with Euclidean distances (X set).

Among these instance sets, we selected 200 instances: 100 public (given to the contestants) and 100 private (used for the evaluation), with 4 to 136 terminals. The median number of terminals is 16 and its mean is 19.4, while 1480 is the mean number of vertices, and 8515, the mean number of edges (see Figure 1). On more than 20 of those instances, top implementations of the DIMACS challenge, such as Staynerd, take tens of minutes. We sorted the instances by increasing $(t, m)$ in the lexicographic order.

The main rule for Track A and B was that the algorithm (deterministic or randomized) should be exact. A sub-optimal output (in the public or private instances) would therefore mean disqualification. The final score is the number of private instances solved on the `optil.io` platform with 30 minutes per instance.

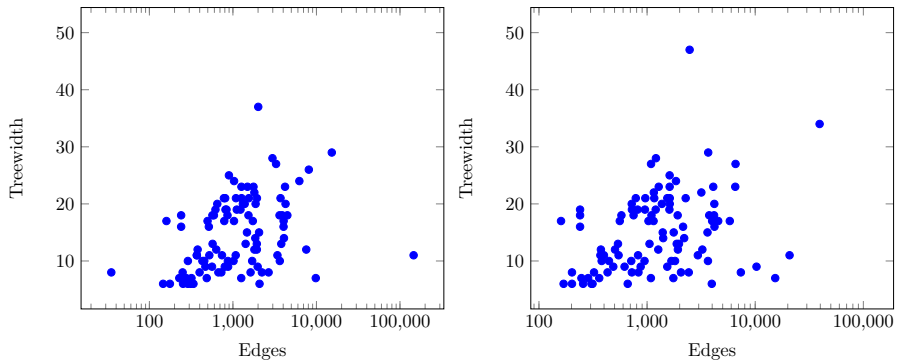



**Figure 2** The distribution (#edges,treewidth) of the public (left) and private (right) instances of Track B in lin-log plot.

## 3.2 Track B

For this track, we used a lot of rectilinear instances with a low treewidth but a high number of terminals (in the ES100FST, ES500FST, ES1000FST sets, where the number between ES and FST corresponds to the number of terminals). We also used many instances from WRP3 and WRP4 with an intermediate number of terminals (≈50). A tree decomposition of almost always minimum width was given with the input. We computed these decompositions using the winning implementations of the treewidth track of PACE 2017 by Strasser and by Tamaki and his team [14].

On the 200 instances that we picked, the average number of vertices is 1490, the average number of edges is 2847, the average number of terminals is 606, and its median is 100. Furthermore the width of the given tree decompositions range from 6 to 47 with a median of 19.5 (see Figure 2). We sorted the instances by increasing $(w, m)$. The rules were the same as in Track A.

## 3.3 Track C

Finally in Track C, dedicated to heuristics, we chose large and difficult instances with many terminals and high treewidth. To this end, we used the hardest instances of Steinlib. We also added a lot of instances from the Vienna set. These instances were generated from real-world telecommunication networks by Ivana Ljubic's group at the University of Vienna. A majority of the selected instances cannot be solved within one hour by the state-of-the-art program. In several cases, the actual optimum is unknown.

The average number of vertices is 27K, the average number of edges is 48K, and the average number of terminals is 1114, with a median at 360.5 (see Figure 3). Finally the best treewidth upper bound (obtained with the heuristics of previous PACE) on these instances is almost always above 40. We sorted the instances by increasing $(t, m)$.

In track C, the final score to maximize is the sum over all the private instances of the ratio opt/sol where opt is the optimum (if known) or the best known upper bound (otherwise) and sol is the value of the proposed output. Outputting a non-feasible solution would not disqualify the submission, but would give a score of 0 on that particular instance.

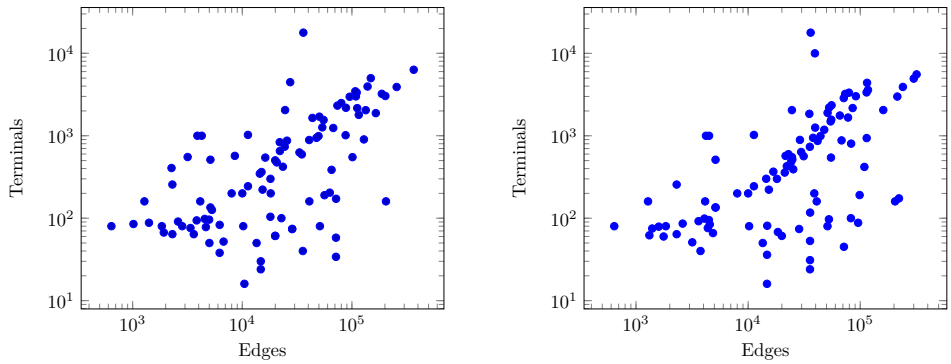



**Figure 3** The distribution (#edges,#terminals) of the public (left) and private (right) instances of Track C in log-log plot.

**Table 1** Participation per country (based on the initial registration form; more teams and participants uploaded their code on `optil.io` afterwards).

| Country | Teams | Participants |
|---|---|---|
| Austria | 2 | 4 |
| Brazil | 1 | 3 |
| Canada | 1 | 1 |
| Czechia | 2 | 4 |
| Denmark | 1 | 1 |
| England | 1 | 1 |
| Finland | 1 | 1 |
| France | 4 | 7 |
| Germany | 4 | 5 |
| India | 6 | 12 |
| Japan | 4 | 8 |
| Mexico | 1 | 4 |
| Netherlands | 2 | 6 |
| Norway | 2 | 4 |
| Poland | 2 | 11 |
| Romania | 1 | 3 |
| Total | 35 | 75 |

## 4 Participation and results

This year, we had over 40 teams and 80 participants coming from 16 countries and four continents: including Austria, Brazil, Canada, Czechia, Denmark, England, Finland, France, Germany, India, Japan, Mexico, the Netherlands, Norway, Poland, and Romania.

The number of teams and participants both doubled compared to PACE 2017. To be precise, the above numbers correspond to teams and participants who sent a final implementation to at least one track. If we also count people who uploaded some code on the `optil.io` platform but dropped out of the competition, the number of teams exceeds 50 and the number of participants exceeds 100. If we count the teams with their multiplicity (that is, the number of tracks in which they participated), this number exceeds 75.

## 4.1 Track A

Yoichi Iwata and Takuto Shigemura won this track by solving 95 private instances. Their algorithm builds upon the dynamic programming of Erickson-Monma-Veinott (EMV). A technical lemma permitted them to prune a lot of entries in the DP table. This happens to also solve instances with small treewidth even when the number of terminals is relatively large.

Krzysztof Maziarz and Adam Polak got the second place by preprocessing and improving Dreyfus-Wagner (DW) with the heuristic ideas presented by Hougardy et al. [29]. The latter consists of running a shortest path algorithm (Dijkstra, A*, etc.) in the partial solutions in order to prune the search space.

Koch and Rehfeldt took the third place with their program SCIP-Jack which has three main components. First, known and new reduction methods simplify the instance. Secondly, a set of heuristics provides upper and lower bounds. Finally, a reduction (of many variants) to the directed Steiner tree problem is performed and the core of the algorithm is a branch-and-cut approach using the MIP-solver SCIP (developed by a superset of this team).

At the fourth place, albeit still very close to the first place, solving 92 instances, Andre Schidler, Johannes Fichte, and Markus Hecher used reduction rules presented by Rehfeldt [37], including the so-called *dual ascent* and the improved DW by Hougardy et al. [29].

All the other teams implemented some refinements of DW or EMV.

- **1st place, 450 €:** Yoichi Iwata and Takuto Shigemura (team wata&sigma from Japanese National Institute of Informatics and University of Tokyo) solved 95 out of 100 instances
  `github.com/wata-orz/steiner_tree`
- **2nd place, 350 €:** Krzysztof Maziarz and Adam Polak (team Jagiellonian from Jagiellonian University) solved 94 out of 100 instances
  `https://bitbucket.org/krismaz/pace2018`
- **3rd place, 300 €:** Thorsten Koch and Daniel Rehfeldt (team reko from Zuse Institute Berlin and TU Berlin) solved 93 out of 100 instances
  `github.com/dRehfeldt/scipjack/`
- **4th place, 225 €:** Andre Schidler, Johannes Fichte, and Markus Hecher (team TUW from TU Vienna) solved 92 out of 100 instances
  `github.com/ASchidler/pace17/`
- **5th place:** Krzysztof Kiljan, Dominik Klemba, Marcin Mucha, Wojciech Nadara, Marcin Pilipczuk, Mateusz Radecki, and Michał Ziobro (team UWarsaw from University of Warsaw and Jagiellonian University) solved 67 out of 100 instances
  `https://bitbucket.org/marcin_pilipczuk/pace2018-steiner-tree`
- **6th place:** Suhas Thejaswi (team Noname from Aalto University) solved 66 out of 100 instances
  `github.com/suhastheju/pace-2018-exact`
- **6th place:** Peter Mitura and Ondřej Suchý (team FIT CTU in Prague from the Czech Technical University) solved 66 out of 100 instances
  `github.com/PMitura/pace2018`
- **6th place:** Johannes Varga (team johannes from TU Vienna) solved 66 out of 100 instances
  `github.com/josshy/st-tree`
- **9th place:** Saket Saurabh, P. S. Srinivasan, and Prafullkumar Tale (team SSPSPT from the Institute Of Mathematical Sciences, HBNI, Chennai and the International Institute of Information Technology, Bangalore) solved 48 out of 100 instances
  `github.com/pptale/PACE18`

- **10th place:** Sharat Ibrahimpur (team the65thbit from University of Waterloo) solved 69 out of 100 instances but was incorrect on 1 instance
  `github.com/sharat1105/PACE2018`
- **11th place:** S. Vaishali and Rathna Subramanian (team PCCoders from PSG College of Technology, Coimbatore) solved 14 out of 100 instances but was incorrect on several instances
  `github.com/ammuv/PACE-2018-`
- **12th place:** R. Vijayaragunathan, N. S. Narayanaswamy, and Rajesh Pandian M. (team Resilience from TCS Lab, Indian Institute of Technology, Madras) solved 9 instances but was incorrect on several instances
  `github.com/mrprajesh/pace2018`

One can see two well-defined clusters (90-95 and 65-70). The implementation at 10th place would have led the second cluster, if not for a small bug that showed only on one private instance.

## 4.2 Track B

In this track with low treewidth, the developers of SCIP-Jack, Thorsten Koch and Daniel Rehfeldt, took the first place by solving 92 of the 100 instances in the private set. They do not directly exploit the tree-decomposition given in input, but it is likely that the low treewidth indirectly translates into higher efficiency of their preprocessing, heuristics, and/or branch-and-cut. Otherwise, it is hard to explain how the same program, discarding the tree-decomposition, won Track B by solving 15 more instances than the winner of Track A (where they finished third).

At the second place, Yoichi Iwata and Takuto Shigemura implemented the standard $O^*(w^w)$ dynamic programming on the tree-decomposition. When the treewidth becomes too large, they switch to their improved EMV algorithm. They observe that their algorithm for Track A performs well when the treewidth is low since the existence of small separators prunes a lot of entries in the DP table.

Completing the podium, Tom van der Zanden added to the rank-based approach a handful of nice observations, crediting some of them to Luuk van der Graaff's master thesis [28]. This results in the only program of the competition solving all the instances with treewidth at most 15. In particular, this implementation is the only one to solve instance 39 of the private set, which has 5K+ vertices, 20K+ edges, 2.4K+ terminals, and treewidth 11.

- **1st place, 450 €:** Thorsten Koch and Daniel Rehfeldt (team reko from Zuse Institute Berlin and TU Berlin) solved 92 out of 100 instances
  `github.com/dRehfeldt/scipjack/`
- **2nd place, 350 €:** Yoichi Iwata and Takuto Shigemura (team wata&sigma from the Japanese National Institute of Informatics and the University of Tokyo) solved 77 out of 100 instances
  `github.com/wata-orz/steiner_tree`
- **3rd place, 300 €:** Tom van der Zanden (team Tom from Utrecht University) solved 58 out of 100 instances
  `github.com/TomvdZanden/SteinerTreeTW`
- **4th place:** Peter Mitura and Ondřej Suchý (team FIT CTU in Prague from the Czech Technical University) solved 52 out of 100 instances
  `github.com/PMitura/pace2018`

**Table 2** For each team: *score* is the number of private instances solved, *treewidth approach*, the treewidth-based algorithm used, if any, *first unsolved*, the number of the first unsolved private instance (we recall that the instances of Track B were sorted lexicographically by increasing $(w, e)$) and the corresponding value of the treewidth, *switching to DW-like*, if the treewidth approach is substituted to a terminal-based algorithm (DW = Dreyfus-Wagner algorithm, EMV = Erickson-Monma-Veinott algorithm), and *criterion to switch* what is the test to switch to such an algorithm.

| Team | score | treewidth approach | first unsolved | switching to DW-like | criterion to switch |
|---|---|---|---|---|---|
| reko | **92** | no | 26, $w = 9$ | no | – |
| wata_sigma | 77 | $w^w$ DP | 26, $w = 9$ | EMV | $w > 10$ or ($w > 8$ and $t < 300$) |
| Tom | 58 | $2^{O(w)}$ rank-based | **54**, $w = 16$ | EMV | $3^t < 5^w$ |
| FIT CTU in Prague | 52 | $2^{O(w)}$ rank-based | 39, $w = 11$ | no | – |
| yasu | 52 | $2^{O(w)}$ rank-based | 13, $w = 7$ | DW | $w > 15$ |
| fujiyoshi | 49 | $2^{O(w)}$ rank-based | 39, $w = 11$ | no | – |
| UWarsaw | 33 | $2^{O(w)}$ rank-based | 32, $w = 10$ | no | – |
| lapo | 33 | $2^{O(w)}$ rank-based | 13, $w = 7$ | no | – |

- **4th place:** Yasuaki Kobayashi (team Yasu from Kyoto University) solved 52 out of 100 instances
  `https://bitbucket.org/yasu0207/steiner_tree`
- **6th place:** Akio Fujiyoshi (team CBGfinder from Ibaraki University) solved 49 out of 100 instances
  `github.com/akio-fujiyoshi/CBGfinder_for_steiner_tree_problem`
- **7th place:** Krzysztof Kiljan, Dominik Klemba, Marcin Mucha, Wojciech Nadara, Marcin Pilipczuk, Mateusz Radecki, and Michał Ziobro (team UWarsaw from University of Warsaw and Jagiellonian University) solved 33 out of 100 instances
  `https://bitbucket.org/marcin_pilipczuk/pace2018-steiner-tree`
- **7th place:** Dilson Guimarães, Guilherme Gomes, João Gonçalves, and Vinícius dos Santos (team lapo from LAPO-UFMG) solved 33 out of 100 instances
  `github.com/dilsonguim/pace2018`

The rest of the teams implemented the rank-based approach of Bodlaender et al. [8] as the main or secondary component. At the shared fourth place, Peter Mitura and Ondřej Suchý (FIT CTU in Prague)'s implementation is the fastest of the competition on average, while Yasuaki Kobayashi's program also switches to a terminal-based approach when the treewidth becomes too large. Table 2 summarizes the approach used by all the teams.

Since the instances were edge-weighted, no submission tried to use the Cut&Count technique. It would be interesting to compare the speed of the rank-based and the Cut&Count techniques on unweighted instances of relatively low treewidth. Another direction would be to make Cut&Count work on edge-weighted instances, in practice and/or in theory.

## 4.3 Track C

This track attracted the most participants. Regarding the instance sizes, it should be observed that a naive implementation (without preprocessing) of the 2-approximation algorithm does not finish within the time limit. The contributions use either meta-heuristics (an evolutionary algorithm for the winning team, iterated local-search with noising for the third team, simulated annealing, etc.), or start from some solution (a spanning tree, an arbitrary feasible solution, or a 2-approximation) and then improve it with local search. Radek Hušek and teammates (Team Tarken) used a parameterized approximation approach [18].

- **1st place, 450 €:** Emmanuel Romero Ruiz, Emmanuel Antonio Cuevas, Irwin Enrique Villalobos López, and Carlos Segura González (CIMAT Team from the Center for Research in Mathematics, Guanajuato) got an average ratio of 99.93/100
  `github.com/HeathcliffAC/SteinerTreeProblem`
- **2nd place, 350 €:** Thorsten Koch and Daniel Rehfeldt (team reko from Zuse Institute Berlin and TU Berlin) got an average ratio of 99.85/100
  `github.com/dRehfeldt/scipjack/`
- **3rd place, 300 €:** Martin Josef Geiger (team MJG from HSU Hamburg) [26] got an average ratio of 99.80/100
  `https://data.mendeley.com/datasets/yf9vpkgwdr/1`
- **4th place, 225 €:** Radek Hušek, Tomáš Toufar, Dušan Knop, Tomáš Masařík, and Eduard Eiben (team CUiB from Charles University, Prague and University of Bergen) got an average ratio of 99.72/100
  `github.com/goderik01/PACE2018`
- **5th place:** Emmanuel Arrighi and Mateus de Oliveira Oliveira (team Gardeners from ENS Cachan and University of Bergen) got an average ratio of 98.93/100
  `github.com/SteinerGardeners/TrackC-Version1`
- **6th place:** Krzysztof Kiljan, Dominik Klemba, Marcin Mucha, Wojciech Nadara, Marcin Pilipczuk, Mateusz Radecki, and Michał Ziobro (team UWarsaw from University of Warsaw and Jagiellonian University) got an average ratio of 98.27/100
  `https://bitbucket.org/marcin_pilipczuk/pace2018-steiner-tree`
- **7th place:** Stéphane Grandcolas (team SGLS from LIS Marseille) got an average ratio of 97.54/100
  `http://www.dil.univ-mrs.fr/~gcolas/sgls.c`
- **8th place:** Max Hort, Marciano Geijselaers, Joshua Scheidt, Pit Schneider, and Tahmina Begum (team JMMPT from the Department of Data Science and Knowledge Engineering, Maastricht University) got an average ratio of 97.15/100
  `github.com/maxhort/Pacechallenge-TrackC/`
- **9th place:** Dimitri Watel and Marc-Antoine Weisser (team DoubleDoubleU from SAMOVAR-ENSIIE and LRI-Centrale-Supelec) got an average ratio of 96.92/100
  `github.com/mouton5000/PACE2018/`
- **10th place:** R. Vijayaragunathan, N. S. Narayanaswamy, and Rajesh Pandian M. (team Resilience from TCS Lab and the Indian Institute of Technology Madras) got an average ratio of 94.57/100
  `github.com/mrprajesh/pace2018`
- **11th place:** Sharat Ibrahimpur (team the65thbit from University of Waterloo) got an average ratio of 94.37/100
  `github.com/sharat1105/PACE2018`
- **12th place:** Saket Saurabh, P. S. Srinivasan, and Prafullkumar Tale (team SSPSPT from the Institute Of Mathematical Sciences, HBNI, Chennai and the International Institute of Information Technology, Bangalore) got an average ratio of 82.61/100
  `github.com/pptale/PACE18`
- **13th place:** Harumi Haraguchi, Hiroshi Arai, Shiyougo Akiyama, and Masaki Kubonoya (team Haraguchi laboratory from Ibaraki University) got an average ratio of 80.73/100
  `github.com/Harulabo/pace2018`

Eleven more teams participated in this track but did not finalize their submission by sharing a link to their code. Table 3 shows that the second team (reko) was more often the one reporting a best solution. However, the first team (CIMAT Team) was more consistently very close to those best solutions.

**Table 3** For the first four teams: their final score, how many times they computed the best solution, and how many times their solution is $x\%$ close to the best solution, on the private set.

| Team name | Cumul. ratio | # Best | # $\leqslant 0.1\%$ | # $\leqslant 0.5\%$ | # $\leqslant 1\%$ | # $\leqslant 2\%$ | # $\leqslant 3\%$ |
|---|---|---|---|---|---|---|---|
| CIMAT Team | 99.93 | 34 | 49 | 95 | 100 | 100 | 100 |
| reko | 99.85 | 78 | 81 | 88 | 93 | 99 | 100 |
| MJG | 99.80 | 24 | 58 | 87 | 95 | 98 | 100 |
| CUiB | 99.72 | 19 | 48 | 82 | 89 | 97 | 100 |

## 5 PACE organization

In September 2018, the PACE 2018 Program Committee transferred to the Steering Committee. The Steering Committee and the PACE 2018 Program Committee are as follows.

| | | |
|---|---|---|
| **Steering committee:** | Édouard Bonnet | ENS Lyon |
| | Holger Dell | Saarland Informatics Campus |
| | Thore Husfeldt | ITU Copenhagen & Lund University |
| | Bart M. P. Jansen (chair) | Eindhoven University of Technology |
| | Petteri Kaski | Aalto University |
| | Christian Komusiewicz | Philipps-Universität Marburg |
| | Frances A. Rosamond | University of Bergen |
| | Florian Sikora | Paris-Dauphine University |
| **Track A, B, C:** | Édouard Bonnet | ENS Lyon |
| | Florian Sikora | Paris-Dauphine University |

The Program Committee of PACE 2019 will be chaired by Johannes Fichte (TU Dresden) and Markus Hecher (TU Vienna).

## 6 Conclusion

We thank all the participants for their enthusiasm and look forward to the next PACE. We are particularly happy that this edition attracted many people outside the parameterized complexity community, and wish that this will continue for the future editions.

We welcome anyone who is interested to add their name to the mailing list on the website [36] to receive PACE updates and join the discussion. In particular, plans for PACE 2019 will be posted there.

## References

1 Michael Abseher, Nysret Musliu, and Stefan Woltran. htd - A Free, Open-Source Framework for (Customized) Tree Decompositions and Beyond. In Domenico Salvagnin and Michele Lombardi, editors, *Proceedings of the 14th International Conference on Integration of AI and OR Techniques in Constraint Programming (CPAIOR '17)*, volume 10335 of *Lecture Notes in Computer Science*, pages 376–386. Springer, 2017. doi:10.1007/978-3-319-59776-8_30.

2 Max Bannach and Sebastian Berndt. Practical Access to Dynamic Programming on Tree Decompositions. In *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, pages 6:1–6:13, 2018. doi:10.4230/LIPIcs.ESA.2018.6.

3 Max Bannach, Sebastian Berndt, and Thorsten Ehlers. Jdrasil: A Modular Library for Computing Tree Decompositions. In Costas S. Iliopoulos, Solon P. Pissis, Simon J. Puglisi, and Rajeev Raman, editors, *Proceedings of the 16th International Symposium on Experimental Algorithms (SEA 2017)*, volume 75 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:21, Dagstuhl, Germany, 2017. doi:10.4230/LIPIcs.SEA.2017.28.

4 Max Bannach, Sebastian Berndt, Thorsten Ehlers, and Dirk Nowotka. SAT-encodings of tree decompositions. *SAT COMPETITION 2018*, page 72, 2018.

5 Sebastian Berndt. Computing Tree Width: From Theory to Practice and Back. In *Conference on Computability in Europe*, pages 81–88. Springer, 2018.

6 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets möbius: fast subset convolution. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 67–74, 2007. doi:10.1145/1250790.1250801.

7 Johannes Blum and Sabine Storandt. Computation and Growth of Road Network Dimensions. In *International Computing and Combinatorics Conference*, pages 230–241. Springer, 2018.

8 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.

9 Jarosław Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6:1–6:33, 2013. doi:10.1145/2432622.2432628.

10 Miroslav Chlebík and Janka Chlebíková. The Steiner tree problem on graphs: Inapproximability results. *Theor. Comput. Sci.*, 406(3):207–214, 2008. doi:10.1016/j.tcs.2008.06.046.

11 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

12 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving Connectivity Problems Parameterized by Treewidth in Single Exponential Time. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159, 2011. doi:10.1109/FOCS.2011.23.

13 Holger Dell, Thore Husfeldt, Bart M. P. Jansen, Petteri Kaski, Christian Komusiewicz, and Frances A. Rosamond. The First Parameterized Algorithms and Computational Experiments Challenge. In Jiong Guo and Danny Hermelin, editors, *Proceedings of the 11th International Symposium on Parameterized and Exact Computation (IPEC 2016)*, volume 63 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:9, Dagstuhl, Germany, 2017. doi:10.4230/LIPIcs.IPEC.2016.30.

14 Holger Dell, Christian Komusiewicz, Nimrod Talmon, and Mathias Weller. The PACE 2017 Parameterized Algorithms and Computational Experiments Challenge: The Second Iteration. In *12th International Symposium on Parameterized and Exact Computation, IPEC 2017, September 6-8, 2017, Vienna, Austria*, pages 30:1–30:12, 2017. doi:10.4230/LIPIcs.IPEC.2017.30.

15 Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization Lower Bounds Through Colors and IDs. *ACM Trans. Algorithms*, 11(2):13:1–13:20, 2014. doi:10.1145/2650261.

16 Stuart E. Dreyfus and Robert A. Wagner. The Steiner problem in graphs. *Networks*, 1(3):195–207, 1971.

17 Eugene F. Dumitrescu, Allison L. Fisher, Timothy D. Goodrich, Travis S. Humble, Blair D. Sullivan, and Andrew L. Wright. Benchmarking treewidth as a practical component of tensor-network–based quantum simulation. *arXiv preprint arXiv:1807.04599*, 2018.

18 Pavel Dvořák, Andreas Emil Feldmann, Dušan Knop, Tomáš Masařík, Tomáš Toufar, and Pavel Veselý. Parameterized Approximation Schemes for Steiner Trees with Small Number of Steiner Vertices. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, volume 96 of *LIPIcs*, pages 26:1–26:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.STACS.2018.26.

19 Ranel E. Erickson, Clyde L. Monma, and Arthur F. Veinott Jr. Send-and-Split Method for Minimum-Concave-Cost Network Flows. *Math. Oper. Res.*, 12(4):634–664, 1987. doi:10.1287/moor.12.4.634.

20 Johannes K Fichte, Markus Hecher, Neha Lodha, and Stefan Szeider. An SMT Approach to Fractional Hypertree Width. *technical report*, 2018.

21 Johannes Klaus Fichte, Markus Hecher, Michael Morak, and Stefan Woltran. Answer Set Solving with Bounded Treewidth Revisited. In Marcello Balduccini and Tomi Janhunen, editors, *Proceedings of 14th International Conference on Logic Programming and Nonmonotonic Reasoning - (LPNMR 2017)*, volume 10377 of *Lecture Notes in Computer Science*, pages 132–145. Springer, 2017. doi:10.1007/978-3-319-61660-5_13.

22 Johannes Klaus Fichte, Markus Hecher, Michael Morak, and Stefan Woltran. DynASP2.5: Dynamic programming on tree decompositions in action. In *Proceedings of the 12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*, Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, 2017. doi:10.4230/LIPIcs.IPEC.2017.17.

23 Johannes Klaus Fichte, Markus Hecher, Stefan Woltran, and Markus Zisser. Weighted Model Counting on the GPU by Exploiting Small Treewidth. In *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, pages 28:1–28:16, 2018. doi:10.4230/LIPIcs.ESA.2018.28.

24 Bernhard Fuchs, Walter Kern, Daniel Mölle, Stefan Richter, Peter Rossmanith, and Xinhui Wang. Dynamic Programming for Minimum Steiner Trees. *Theory Comput. Syst.*, 41(3):493–500, 2007. doi:10.1007/s00224-007-1324-4.

25 Serge Gaspers, Joachim Gudmundsson, Mitchell Jones, Julián Mestre, and Stefan Rümmele. Turbocharging Treewidth Heuristics. In Jiong Guo and Danny Hermelin, editors, *Proceedings of the 11th International Symposium on Parameterized and Exact Computation (IPEC 2016)*, volume 63 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:13, Dagstuhl, Germany, 2017. doi:10.4230/LIPIcs.IPEC.2016.13.

26 Martin Josef Geiger. Implementation of a Metaheuristic for the Steiner Tree Problem in Graphs. Mendeley Data. v1. doi:10.17632/yf9vpkgwdr.1.

27 Edgar N. Gilbert and Henry O. Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968.

28 L.W. van der Graaff. Dynamic programming on Nice Tree Decompositions. Master's thesis, Utrecht University, 2015. URL: https://dspace.library.uu.nl/handle/1874/309652.

29 Stefan Hougardy, Jannik Silvanus, and Jens Vygen. Dijkstra meets Steiner: A fast exact goal-oriented Steiner tree algorithm. *Math. Program. Comput.*, 9(2):135–202, 2017. doi:10.1007/s12532-016-0110-1.

30 Frank K. Hwang, Dana S. Richards, and Pawel Winter. *The Steiner tree problem*, volume 53. Elsevier, 1992.

31 Yoichi Iwata. Linear-Time Kernelization for Feedback Vertex Set. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*,

volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 68:1–68:14, 2017. doi:10.4230/LIPIcs.ICALP.2017.68.

32 Krzysztof Kiljan and Marcin Pilipczuk. Experimental Evaluation of Parameterized Algorithms for Feedback Vertex Set. *arXiv preprint arXiv:1803.00925*, 2018.

33 Neha Lodha, Sebastian Ordyniak, and Stefan Szeider. SAT-encodings for special treewidth and pathwidth. In *Proceedings of the 20th International Conference on Theory and Applications of Satisfiability Testing (SAT 2017)*, volume 10491 of *Lecture Notes in Computer Science*, pages 429–445. Springer, 2017. doi:10.1007/978-3-319-66263-3_27.

34 Daniel Mölle, Stefan Richter, and Peter Rossmanith. Enumerate and Expand: Improved Algorithms for Connected Vertex Cover and Tree Cover. *Theory Comput. Syst.*, 43(2):234–253, 2008. doi:10.1007/s00224-007-9089-3.

35 Networks project, 2017. URL: http://www.thenetworkcenter.nl.

36 Parameterized Algorithms and Computational Experiments website, 2015–2018. URL: https://pacechallenge.org.

37 Daniel Rehfeldt. A generic approach to solving the Steiner tree problem and variants. Master's thesis, Technische Universität Berlin, Germany, 2015. URL: http://nbn-resolving.de/urn:nbn:de:0297-zib-57817.

38 Gabriel Robins and Alexander Zelikovsky. Tighter bounds for graph Steiner tree approximation. *SIAM Journal on Discrete Mathematics*, 19(1):122–134, 2005.

39 Ondřej Suchý. Extending the Kernel for Planar Steiner Tree to the Number of Steiner Vertices. *Algorithmica*, 79(1):189–210, 2017. doi:10.1007/s00453-016-0249-1.

40 Hisao Tamaki. Positive-Instance Driven Dynamic Programming for Treewidth. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms (ESA 2017)*, volume 87 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 68:1–68:13, Dagstuhl, Germany, 2017. doi:10.4230/LIPIcs.ESA.2017.68.

41 Tom C. van der Zanden and Hans L. Bodlaender. Computing Treewidth on the GPU. In *Proceedings of the 12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*, Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, 2017. doi:10.4230/LIPIcs.IPEC.2017.29.

42 Rim van Wersch and Steven Kelk. ToTo: An open database for computation, storage and retrieval of tree decompositions. *Discrete Applied Mathematics*, 217:389–393, 2017. doi:10.1016/j.dam.2016.09.023.