Structural Parameters, Tight Bounds, and Parameterized Approximation

Emmanouil Vasilakis

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'Université Paris-Dauphine

Structural Parameters, Tight Bounds, and Parameterized Approximation

Soutenue par
Emmanouil VASILAKIS
Le 06 février 2026

École doctorale n°543
SDOSE

Spécialité
Informatique

Composition du jury :

Cristina BAZGAN Professeure, Université Paris Dauphine – PSL	*Présidente*
Bruno ESCOFFIER Professeur, Sorbonne Université	*Rapporteur*
Stefan KRATSCH Professeur, Humboldt-Universität zu Berlin	*Rapporteur*
Pierre ABOULKER Maître de Conférences, École Normale Supérieure – PSL	*Examinateur*
Christophe PAUL Directeur de Recherche CNRS, Université de Montpellier	*Examinateur*
Michael LAMPIS Maître de Conférences HDR, Université Paris Dauphine – PSL	*Directeur de thèse*

Dauphine | PSL
UNIVERSITÉ PARIS

Abstract

This thesis studies NP-hard graph problems through structural parameterization and fine-grained analysis. For our problems of study we (i) locate the boundary between fixed-parameter tractability and W[1]-hardness, by giving improved hardness reductions and designing new FPT algorithms; (ii) match natural DP algorithms with tight SETH- or pw-SETH-lower bounds; (iii) obtain tight ETH-lower bounds for parameterizations beyond treewidth; and (iv) show that adding approximation into the mix can in some cases overcome exact barriers, yet in some other cases it cannot. On our way to these results we develop new techniques for both hardness reductions and algorithm design that may be of independent interest. In particular, we introduce (i) a new variant of Unary Bin Packing that allows for improved W[1]-hardness results; (ii) techniques for obtaining improved lower bounds for problems parameterized by tree-depth and vertex cover; and (iii) a simple construction for designing FPT approximation schemes. We leverage these techniques to obtain a number of new results for several well-studied NP-hard graph problems.

Keywords: Parameterized complexity · Fine-grained complexity · Approximation algorithms · Treewidth

Résumé

Cette thèse étudie des problèmes sur les graphes NP-difficiles au travers de la paramétrisation structurelle et d'une analyse fine. Pour les problèmes considérés, nous (i) localisons la frontière entre la tractabilité à paramètres fixes (FPT) et la W[1]-dureté, en proposant des réductions d'intractabilité améliorées et en concevant de nouveaux algorithmes FPT ; (ii) faisons correspondre des algorithmes naturels de programmation dynamique à des bornes inférieures serrées fondées sur SETH ou pw-SETH ; (iii) établissons des bornes inférieures serrées sous ETH pour des paramétrisations au-delà de la largeur d'arbre ; et (iv) montrons que l'approximation permet, dans certains cas, de franchir des barrières du calcul exact, tandis que dans d'autres elle ne le peut pas. Pour atteindre ces résultats, nous développons de nouvelles techniques de réductions de dureté et de conception d'algorithmes, susceptibles d'intérêt indépendant. En particulier, nous introduisons (i) une nouvelle variante de Unary Bin Packing qui conduit à des résultats de W[1]-dureté renforcés ; (ii) des techniques pour obtenir des bornes inférieures améliorées pour des problèmes paramétrés par la profondeur d'arbre et la couverture de sommets ; et (iii) une construction simple pour concevoir des schémas d'approximation FPT. Nous mettons à profit ces techniques pour obtenir un ensemble de nouveaux résultats sur plusieurs problèmes sur les graphes NP-difficiles bien étudiés.

Mots clés: Complexité paramétrée · Complexité fine · Algorithmes d'approximation · Largeur d'arbre

Acknowledgements

I would first like to thank all the members of the committee. My thanks go to Cristina Bazgan, Christophe Paul, and Pierre Aboulker for agreeing to take part in the jury. I am especially grateful to Bruno Escoffier and Stefan Kratsch for serving as rapporteurs, and for their careful reading of my dissertation and their detailed reports.

A special and warm thank you goes to my supervisor, Michael Lampis. Since the very first day I arrived in France, you have supported and guided me with great generosity. You always gave me space to vent about my everyday problems,[1] while also being consistently available to discuss new ideas and open questions. Thank you for believing in me, for your care, and for constantly pushing me to become a better researcher. I deeply appreciate the time, honesty, and effort you have invested in me over these years, and I feel very grateful and lucky to have had you as my advisor.

There are several other people I would especially like to thank. A warm thank you to my undergraduate advisor, Aris Pagourtzis. Thank you for believing in me from the very beginning, and for helping me pursue my dream of doing a PhD abroad. Next, I would like to thank Dimitris Fotakis, who has been a major influence on my decision to pursue a research career in theoretical computer science. Lastly, I would like to thank Nikos Melissinos, a close collaborator and friend across so many countries over all these years.

LAMSADE is a peculiar place. Even though things can be chaotic, it is full of people who are welcoming, supportive, and always willing to help. I was continually struck by everyone's kindness (and, occasionally, by French bureaucracy). Even if the offices are not exactly beautiful (and I admit I compare them to the PhD offices wherever I visit), LAMSADE became a place I could call home while being far away from home. Paradoxically, even though Lucas calls me the biggest LAMSADE hater,[2] I am sure that I will miss it. Merci beaucoup à toutes et à tous!

During my PhD, I had the chance to visit several places and collaborate with many different people. Thanks to Yota, Andreas, Dušan, Édouard, and Robert for your hospitality; I am deeply thankful for these research visits and discussions. Speaking of travels, arguably one of the best experiences of my PhD has been my time at JamCoders in Kingston. Thanks to Sam, Zaria, Adrianna, Piyush, Xavier, Lydia, Joy, Bruno, Hanna, and Frank, I had an amazing time, and I am very grateful for all the great moments we shared during this special summer.

One of the most important parts of my life during the past few years in Paris has been the people I met and the friends I made. Their support, kindness, and friendship have been invaluable.

[1] Sorry if I did that too often!
[2] Perhaps I do vent too often!

Andreas, I am really glad I met you as soon as I arrived in this hectic city, and I am thankful for all the great times we had together; even though I was sad when you left Paris, I am very happy that you are now enjoying life in Utrecht. I have to admit that my eyes get teary when I think of living in a city without Dimitris, and I am extremely grateful for all the amazing moments we have shared together. I cannot imagine what everyday life would have been without Christos: biking to random bakeries, spontaneous trips around Europe, champagne tasting on a Monday morning... none of these would have been possible without you. I am grateful to Athina for making me smile every single day and for making me genuinely happy.

There are so many more people that I would like to thank, but I will mention just a few more. Iakove, I still believe that 2026 will be a great year for us. Vasili, I admire your dedication, and I firmly believe that you will achieve great things. Kosta, to be honest, I am sometimes astounded by your passion for Olympiakos. Geramise, the library at Fondation Hellénique is not the same without you. Gianni, I truly miss your poems. Alex, I admire you a lot!

I have been fortunate to have had several great friends over the years, both in Greece and abroad. I deeply appreciate each and every one of you, and I am extremely grateful for the moments we have shared together. On a side note, I love the random get-togethers around Europe that we have managed to pull off over the years.

Finally, I would like to thank my family. Your constant love, support, and encouragement have been essential throughout this journey. I am deeply grateful for everything you have done for me, and I could not have reached this milestone without you by my side.

Looking back, I am overwhelmed by the amount of support I have received from so many people. More than anything, I am grateful to those who stood by me and helped make my thesis defense such a special moment, one that I will remember for the rest of my life. I feel very lucky to be surrounded by people who care about me.

Funding. I gratefully acknowledge the financial support of Université Paris Dauphine – PSL through my contrat doctoral and of the A.G. Leventis Foundation towards my PhD studies. I also gratefully acknowledge the support of the Barrande Fellowship, the GDR ROD, and the PSL Erasmus+ Doctoral Mobility Grant for part of my research visits.

Publications During the PhD

This chapter lists the author's publications produced during the PhD.

Publications Included in This Thesis

Chapter 3. Michael Lampis and Manolis Vasilakis. *Structural Parameterizations for Two Bounded Degree Problems Revisited. ACM Transactions on Computation Theory* **16**(3): 17:1–17:51, 2024. https://doi.org/10.1145/3665156

Chapter 4. Michael Lampis and Manolis Vasilakis. *Structural Parameterizations for Induced and Acyclic Matching.* In *Proceedings of the 51st International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2025)*, LNCS, volume 16124, pages 360–376, Springer, Cham, 2026. https://doi.org/10.1007/978-3-032-11835-6_26

Chapter 5. Michael Lampis and Manolis Vasilakis. *Parameterized Maximum Node-Disjoint Paths.* In *Proceedings of the 20th International Symposium on Parameterized and Exact Computation (IPEC 2025)*, LIPIcs, volume 358, pages 3:1–3:15, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2025. https://doi.org/10.4230/LIPIcs.IPEC.2025.3

Chapter 6. Tesshu Hanaka, Michael Lampis, Manolis Vasilakis, and Kanae Yoshiwatari. *Parameterized Vertex Integrity Revisited.* In *Proceedings of the 49th International Symposium on Mathematical Foundations of Computer Science (MFCS 2024)*, LIPIcs, volume 306, pages 58:1–58:14, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024. https://doi.org/10.4230/LIPIcs.MFCS.2024.58

Chapter 7. Dušan Knop, Nikolaos Melissinos, and Manolis Vasilakis. *Parameterized Critical Node Cut Revisited.* https://arxiv.org/abs/2506.23363

Other Publications During the PhD Not Included in This Thesis

Entries are grouped by type (**J**ournal, **C**onference) and ordered in reverse chronological order within each group. Although not included in this manuscript, these papers also investigate the parameterized complexity of diverse graph problems under structural parameterizations and present both algorithmic and hardness results.

J1. Tatsuya Gima, Eun Jung Kim, Noleen Köhler, Nikolaos Melissinos, and Manolis Vasilakis. *Bandwidth Parameterized by Cluster Vertex Deletion Number. Algorithmica* **87**(8): 1146–1177, 2025. https://doi.org/10.1007/s00453-025-01315-x

J2. Michael Lampis, Nikolaos Melissinos, and Manolis Vasilakis. *Parameterized Max Min Feedback Vertex Set. SIAM Journal on Discrete Mathematics* **39**(3): 1587–1620, 2025. https://doi.org/10.1137/23M1605247

C1. Foivos Fioravantes, Dušan Knop, Nikolaos Melissinos, Michal Opler, and Manolis Vasilakis. *Exact Algorithms for Distance to Unique Vertex Cover.* In *Proceedings of the 40th AAAI Conference on Artificial Intelligence (AAAI 2026)*, AAAI Press, volume 40, pages 14217–14224, 2026. https://doi.org/10.1609/aaai.v40i17.38435

C2. Tesshu Hanaka, Michael Lampis, Nikolaos Melissinos, Edouard Nemery, Hirotaka Ono, and Manolis Vasilakis. *Structural Parameters for Steiner Orientation.* In *Proceedings of the 36th International Symposium on Algorithms and Computation (ISAAC 2025)*, LIPIcs, volume 359, pages 38:1–38:14, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2025. https://doi.org/10.4230/LIPIcs.ISAAC.2025.38

C3. Yudai Egami, Tatsuya Gima, Tesshu Hanaka, Yasuaki Kobayashi, Michael Lampis, Valia Mitsou, Edouard Nemery, Yota Otachi, Manolis Vasilakis, and Daniel Vaz. *Broadcasting Under Structural Restrictions.* In *Proceedings of the 50th International Symposium on Mathematical Foundations of Computer Science (MFCS 2025)*, LIPIcs, volume 345, pages 42:1–42:18, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2025. https://doi.org/10.4230/LIPIcs.MFCS.2025.42

C4. Michael Lampis, Valia Mitsou, Edouard Nemery, Yota Otachi, Manolis Vasilakis, and Daniel Vaz. *Parameterized Spanning Tree Congestion.* In *Proceedings of the 50th International Symposium on Mathematical Foundations of Computer Science (MFCS 2025)*, LIPIcs, volume 345, pages 65:1–65:20, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2025. https://doi.org/10.4230/LIPIcs.MFCS.2025.65

C5. Michael Lampis and Manolis Vasilakis. *Structural Parameterizations for Two Bounded Degree Problems Revisited.* In *Proceedings of the 31st Annual European Symposium on Algorithms (ESA 2023)*, LIPIcs, volume 274, pages 77:1–77:16, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2023. https://doi.org/10.4230/LIPIcs.ESA.2023.77

Note: Conference version of the ToCT 2024 journal paper on which Chapter 3 is based.

C6. Michael Lampis, Nikolaos Melissinos, and Manolis Vasilakis. *Parameterized Max Min Feedback Vertex Set.* In *Proceedings of the 48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, LIPIcs, volume 272, pages 62:1–62:15, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2023. https://doi.org/10.4230/LIPIcs.MFCS.2023.62

Note: Conference version of J2.

C7. Tatsuya Gima, Eun Jung Kim, Noleen Köhler, Nikolaos Melissinos, and Manolis Vasilakis. *Bandwidth Parameterized by Cluster Vertex Deletion Number.* In *Proceedings of the 18th International Symposium on Parameterized and Exact Computation (IPEC 2023)*, LIPIcs, volume 285, pages 21:1–21:15, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2023. https://doi.org/10.4230/LIPIcs.IPEC.2023.21

Note: Conference version of J1.

Contents

1. Introduction

Theoretical computer science lies at the interface of mathematics and computer science, offering a rigorous framework to explore the power and limits of computation. It provides the formal languages and tools necessary to understand computational models, algorithmic strategies, and complexity classes [269]. At its core, the field forms the mathematical backbone of computer science, enabling the precise analysis of algorithms and the systematic study of computational feasibility. A central goal of theoretical computer science is to delineate what can be computed efficiently and what lies beyond our algorithmic reach. These questions lead to computational complexity, where problems are classified according to the resources, most notably time and space, required to solve them [146]. Such classifications not only provide insights into the theoretical feasibility of solving problems, but also inform the design of algorithms that attempt to circumvent computational hardness.

At the heart of complexity theory lies the famous P versus NP problem [82], widely regarded as one of the most important open questions in computer science. Roughly speaking, the class P consists of problems solvable in polynomial time by deterministic algorithms, whereas NP contains those problems for which a solution can be *verified* in polynomial time given a suitable certificate. Although $\mathrm{P} \subseteq \mathrm{NP}$ is immediate, the converse inclusion remains an intriguing open question, that is, whether every efficiently verifiable problem is also efficiently solvable, or in other words, whether $\mathrm{P} = \mathrm{NP}$. A positive resolution would yield efficient algorithms for a vast number of problems across optimization, artificial intelligence, cryptography, and other domains [134]. Conversely, a negative resolution would certify the existence of intrinsically hard problems for which there are no efficient algorithms. Despite decades of intensive research, the problem remains open and is widely conjectured to have a negative answer [133]. Its significance is underscored by its inclusion in the Clay Mathematics Institute's list of Millennium Prize Problems.

Many practically relevant computational problems are known to be NP-hard [146], meaning they are among the hardest problems in NP in the sense that, if any one of them is solvable in polynomial time, then every problem in NP is. There are numerous such NP-hard problems on *graphs*, and a central pursuit in algorithmic graph theory is to understand when the *structural properties* of an instance permit efficient computation, and to delineate precise boundaries where even highly specialized algorithms provably cannot do better. Several measures of structure, such as *treewidth* [192] and *tree-depth* [249], have emerged as quantitative lenses capturing different notions of structure, e.g., "how tree-like" or "how star-like" a graph is, and have close ties with areas as diverse as mathematical logic [85, 86, 249].

Central to this quest of understanding computational hardness and algorithmic tractabil-

ity with respect to the structure of the input graph is the field of *parameterized complexity* [92, 103, 123]. Pioneered by Rodney G. Downey and Michael R. Fellows in the early 1990s, this framework provides a refined analysis of computational problems by introducing one or more secondary inputs, called *parameters*, alongside the main input size. The guiding insight is that, in fact, the intractability of many problems can be attributed to specific structural features captured by these parameters, rather than to the overall input size. They observed that many NP-hard problems become efficiently solvable when certain (not necessarily structural) parameters are small, even if the overall input size is large, admitting algorithms with running times that depend on these parameters, of the form $n^{\mathcal{O}(k)}$ or even $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$, where k is the parameter and n is the overall input size. With respect to graph problems, this insight translates to the observation that many NP-hard problems become tractable on specific graph classes such as trees or cluster graphs, thus *quantifying* the distance of the input graph to such a tractable case may be the "right" notion of hardness.

While parameterized complexity provides a powerful framework for designing efficient algorithms, it does not offer a way to argue about their optimality, apart from just a coarse categorization. In fact, this issue is not specific to parameterized complexity, but rather a general limitation of classical complexity theory, which classifies problems into broad complexity classes such as P and NP, without providing fine-grained insights into their exact computational requirements. To address this limitation, the field of *fine-grained complexity* has emerged as a powerful framework that refines classical complexity theory by focusing on exact running time bounds and conditional lower bounds [53]. Under assumptions such as the *Exponential Time Hypothesis* and the *Strong Exponential Time Hypothesis* [172, 173], one can derive tight lower bounds for problems in P and beyond. This framework has been particularly successful in the context of parameterized complexity, allowing us to establish tight lower bounds for parameterized problems, showing that algorithms of running times say $n^{o(k)}$ or $(2-\varepsilon)^k \cdot n^{\mathcal{O}(1)}$ would refute the respective hypothesis.

Yet another avenue towards coping with computational hardness is to relax the requirement of finding an *optimal* solution. In this direction, *approximation algorithms* seek to find solutions that are close to optimal within a guaranteed factor, yet require significantly less computational resources and run in polynomial time [282, 284]. Their performance is quantified by an *approximation ratio*, the ratio between the cost of the algorithm's output and the cost of an optimal solution. Approximation algorithms are particularly useful in the context of NP-hard problems, where finding exact solutions is computationally prohibitive. This is a well-established field with a rich body of literature, that continues to evolve with new techniques and insights.

Scope of this Thesis. This dissertation investigates the algorithmic landscape of various NP-hard graph problems, through the lens of structural parameterization and fine-grained analysis. In a nutshell, we study when and how measures of structure, most prominently treewidth, enable exact or approximate algorithms, with running times whose dependence on the parameter is optimal. Our case studies extend over various problems with different

complexity profiles. Across these problems, we develop and analyze dynamic programming frameworks on structural decompositions, rounding-based and color-coding techniques for parameterized approximation schemes, and lower-bound frameworks grounded in hypotheses coming from fine-grained complexity. The unifying objective is to pinpoint optimal parameter dependencies, to identify barriers that preclude efficient parameterized algorithms, and, when exact computation is provably stuck, to design approximation schemes that recover near-optimal solutions while taking advantage of the structure of the input, or, alternatively, to prove that even this approach is unlikely to succeed.

Regarding the remainder of Chapter 1, Section 1.1 provides the necessary background on these paradigms and basic definitions, followed by a high-level overview of our contributions in Section 1.2, and a discussion of the cross-cutting insights that emerge from them in Section 1.3.

1.1. Background and Motivation

Parameterized Complexity. Since the early 1990s, the parameterized viewpoint of Downey and Fellows has reframed how we reason about intractability by separating the influence of a carefully chosen secondary measure from the raw input size [92, 103, 123]. As already mentioned, the intuition behind parameterized complexity is that the size of the input is too coarse a measure of complexity, and that by having an appropriately chosen parameter as a secondary measure, we can obtain efficient algorithms for hard problems by exploiting their structure. To formalize this intuition, we say that a problem is *fixed-parameter tractable* (FPT) with respect to a parameter k and belongs to the class FPT if it can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$, where f is a computable function independent of the input size n. This allows efficient algorithms for instances where k is small, regardless of the size of the input; notice that for NP-hard problems, which are the focus of this thesis, this allows us to restrict the (unavoidable under P $\neq$ NP) superpolynomial blow-up in the running time to a function depending solely on k. Downey and Fellows introduced a hierarchy of complexity classes (FPT $\subseteq$ W[1] $\subseteq \cdots \subseteq$ XP) to further classify parameterized problems with respect to their tractability. In particular, they defined the class W[1] as the parameterized analog of NP, so that if a problem is W[1]-hard then it presumably does not admit an algorithm with running time $f(k) \cdot n^{\mathcal{O}(1)}$ for any computable function f, under the assumption that FPT $\neq$ W[1].[1]

In its early days, parameterized complexity mostly focused on *solution-size parameterization*, where the parameter k is the size of the solution sought.[2] This is a natural choice for many problems, such as Vertex Cover or Clique, where we seek a vertex cover of size at most k or a clique of size at least k, respectively. In fact, notice that the classical brute-force algorithm for Vertex Cover that goes over all uncovered edges and branches on which one of its endpoints to include runs in time $2^k \cdot n^{\mathcal{O}(1)}$, which is fixed-parameter tractable. In contrast, the brute-force algorithm for Clique that goes over all k-subsets runs in time $n^{\mathcal{O}(k)}$, which is not fixed-parameter tractable. This simple

[1] We postpone any further definitions to Chapter 2 to avoid overloading this introductory chapter.
[2] This is often called the *natural parameterization*.

observation already raises the question: is the observed gap in running times merely an artifact of our current techniques for Clique, or does it point to a more fundamental separation between the two parameterized problems? Trying to answer this question, Downey and Fellows established many foundational results in that direction, introducing the so-called *W-hierarchy* in order to appropriately place parameterized problems into distinct complexity classes. These results laid the groundwork for a rich theory of parameterized complexity, and spurred a flurry of research into the parameterized complexity of various problems with respect to solution-size parameterization [103, 123]. However, it soon became apparent that solution-size parameterization is not always the most appropriate choice, especially for problems where the solution size is not a good indicator of the problem's complexity.[3] This led to the development of *structural parameterization*, where the parameter k is a measure of the *structure* of the input graph.

Beyond the now-standard FPT versus W-hierarchy stratification, a sustained algorithmic narrative has emerged around exploiting *structure* in graphs, with the paradigm of parameterized complexity being particularly effective in this context. The prime example is undoubtedly *treewidth*, denoted by tw, born out of the Graph Minors project of Robertson and Seymour [261, 262] and now ubiquitous in algorithms and logic (see, e.g., [85, 192]).[4] As a parameter, treewidth has played a pivotal role in the development of the whole field, uncovering a wealth of parameterized algorithms for graph problems, and leading to connections with other areas such as model checking [85] and algebraic techniques [281]. For many NP-hard graph problems, the textbook recipe – bottom-up dynamic programming on a nice tree decomposition – yields single-exponential algorithms of the form $c^{\mathrm{tw}} \cdot n^{\mathcal{O}(1)}$ for a small constant c, see, e.g., [92, Chapter 7]. However, things are not always so rosy. For some problems, the fastest known algorithms have slightly-superexponential dependence on tw, that is, of the form $\mathrm{tw}^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$, and this seems to be unavoidable under standard fine-grained complexity hypotheses [225]. Nevertheless, the field has seen a flurry of activity in recent years, and new techniques have been developed to push the boundaries of what is possible. As an example, we mention the application of various algebraic convolution techniques and state representations to speed up algorithms for problems parameterized by treewidth and to tighten the constants in the base of the exponent in the running time [37, 95, 281] (see also [92, Section 11.1]).

Perhaps the most striking development in this direction over the past few years has been the progress on problems with *connectivity constraints*. In a breakthrough result of Cygan, Nederlof, Ma. Pilipczuk, Mi. Pilipczuk, van Rooij, and Wojtaszczyk [94], they introduced the *Cut & Count* technique and showed how to obtain randomized $c^{\mathrm{tw}} \cdot n^{\mathcal{O}(1)}$ algorithms for a large class of problems involving connectivity constraints, such as Steiner Tree, Connected Vertex Cover, and Feedback Vertex Set, where the constant c is often as small as 2 or 3. This result came as a surprise at the time, as it was widely believed that such problems required $\mathrm{tw}^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$ time in order to account for the connectivity of any partial solution. In follow-up works, the

[3] E.g., k-Coloring asks whether the input graph admits a proper k-coloring, and is NP-hard for every fixed $k \geq 3$ [181].

[4] Interestingly, the notion of treewidth has been rediscovered multiple times [31, 162].

Cut & Count technique has been further adapted to other structural parameterizations, including *clique-width* [47, 48, 168], *tree-depth* [166, 248], and *cutwidth* [45, 46], while other techniques have been developed to obtain *deterministic* single-exponential algorithms for problems with connectivity constraints, albeit at the expense of increasing the base of the exponent [29, 30, 40]; this increase persists even after several improvements [131]. Avoiding this increase in the base of the exponent while retaining determinism remains a tantalizing open question.

The discussion so far might imply that treewidth is the be-all and end-all of structural parameterization, but this is far from the truth. There are various other structural parameters that have been studied in the literature, and altogether they form a rich hierarchy based on their generality.[5] From the most restrictive to the most general parameterizations, determining the exact running time of problems is a central question. On a coarse level, determining the tractability border between FPT and W[1]-hardness for all such parameters is a natural goal, and finding the most general parameterization that still allows for FPT algorithms is a key challenge. Such "price of generality" phenomena are known across several pairs of structural parameters: there are problems that are FPT for treewidth yet W[1]-hard for clique-width [128], FPT for pathwidth yet W[1]-hard for treewidth [24], and FPT for tree-depth yet W[1]-hard for pathwidth [159]. On a finer level, improving the running time for a specific parameterization is a worthwhile pursuit that unravels the intricacies of the problem. In this context, a central question is to determine the optimal parametric dependence of the running time of a problem under its various parameterizations, for which tools from the area of fine-grained complexity are indispensable.

Fine-Grained (Parameterized) Complexity. Classical complexity theory is *coarse*: it separates P from NP, but says little about the *best* exponent or the *right* base of it in the running time. The field of *fine-grained complexity* has emerged to address this gap by enabling conditional fine-grained lower bounds that allow us to argue about the optimality of algorithms in the strong sense [53]. There are multiple hypotheses that have been proposed in this context, with the most prominent ones being the *Exponential Time Hypothesis* (ETH) and its Strong variant (SETH) [172, 173]. These constitute strong assumptions about the hardness of the Satisfiability problem, and in fact it holds that SETH implies ETH, which in turn implies P $\neq$ NP. Having them as a basis, one can derive tight lower bounds for problems in P and beyond: SETH-based tight lower bounds are by now known for a plethora of computational problems [91], such as Edit Distance (in P) [14] and Subset Sum (NP-hard) [1].

On a high level, fine-grained complexity offers strong evidence for the optimality of known algorithms, explaining the persistence of certain running-time barriers. In the context of parameterized complexity, those hypotheses offer a way to establish oftentimes tight lower bounds for problems parameterized by various structural (or natural) parameters that lie in FPT or even beyond. Such lower bounds complement the coarse-grained classifications of parameterized complexity (e.g., FPT versus W[1]-hardness), offering a

[5] We refer the reader to Chapter 2 and in particular Section 2.4.

more nuanced understanding of the limits of parameterized algorithms. Perhaps the most well-known such lower bound is the one for k-Clique, where it is known that the trivial $n^{\mathcal{O}(k)}$ algorithm is optimal under the ETH, as an $n^{o(k)}$ algorithm would refute it [63]. By now, there is a rich body of work that establishes tight lower bounds for various combinations of problems and parameters, primarily under the ETH and SETH. This has led to a more refined analysis of the complexity landscape, where we know for instance that, assuming the ETH, the optimal algorithms for problems parameterized by treewidth tw have running times ranging from $\mathrm{tw}^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$ [225] to $2^{2^{\mathcal{O}(\mathrm{tw})}} \cdot n^{\mathcal{O}(1)}$ [135, 212, 237], and even to $n^{\mathcal{O}(2^{\mathrm{tw}})}$ [6].

The seminal works of Lokshtanov, Marx, and Saurabh [224, 225] laid the foundation for this line of research. In particular, in [225] they introduced a framework for establishing tight lower bounds for problems that are fixed-parameter tractable parameterized by treewidth, yet the fastest known algorithms have slightly-superexponential parametric dependence, that is, $\mathrm{tw}^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$. As for [224], the authors established tight lower bounds for problems parameterized by treewidth with single-exponential parametric dependence under the SETH, showing that the base of the exponent in the running time is optimal, even if one considers the (more restrictive) parameterization by pathwidth. This influential work has by now become a standard tool rendering lower bounds of this form fairly standard [50, 89, 90, 93, 94, 105, 106, 114, 126, 154, 163, 187, 252], while it has been also extended to other parameters such as clique-width [45, 47, 48, 144, 168, 186, 208], tree-depth [167], and cutwidth [45, 46, 155, 156, 279], and allows us to establish tight lower bounds for a variety of problems under different parameterizations. Towards this quest of finding the optimal running time of a problem under its various parameterizations, a recurring theme is that most lower bounds for treewidth in fact apply for the (more restrictive) parameterization by pathwidth pw as well. It was only recently that the first natural problem that is W[1]-hard for the first and FPT for the second parameterization was discovered [24], while in the context of problems that are FPT by treewidth there is no problem for which the increased generality of treewidth is known to deteriorate the running time. It is an open question whether this is a coincidence or a deeper phenomenon; the prime candidate for a problem that perhaps admits a strictly faster algorithm for the parameterization by pathwidth is Hamiltonicity, which is known to admit a SETH-tight $(2+\sqrt{2})^{\mathrm{pw}} \cdot n^{\mathcal{O}(1)}$ algorithm [93], yet the fastest known algorithm parameterized by treewidth is $4^{\mathrm{tw}} \cdot n^{\mathcal{O}(1)}$ [94].

In a very recent work, Lampis [210] postulated another fine-grained hypothesis, coined *Primal Pathwidth SETH* (pw-SETH). Interestingly, he showed that this hypothesis is implied by *both* SETH and the Set Cover Conjecture [91], while he used it to establish tight lower bounds not only for parameterized problems that are fixed-parameter tractable with single-exponential parametric dependence, but even for problems that presumably lie outside of FPT. Apart from the fact that this hypothesis is implied by two well-established conjectures, its most interesting aspect is that it allows us to establish the *equivalence* of improving over various problems parameterized by linear width measures; as an example, a $(2-\varepsilon)^{\mathrm{pw}} \cdot n^{\mathcal{O}(1)}$ algorithm for Independent Set would imply a $(3-\varepsilon')^{\mathrm{pw}} \cdot n^{\mathcal{O}(1)}$ algorithm for Dominating Set [210].

Approximation Algorithms and Parameterized Approximation. So far we have been implicitly referring to *decision problems*, whose output is either "yes" or "no", depending on whether there exists a feasible solution. In contrast, an *optimization problem* seeks to find the feasible solution that either maximizes or minimizes an underlying objective function. An *optimal* solution is one for which the problem's objective function attains its extremal value, being the minimum for minimization and the maximum for maximization problems.

Sometimes demanding optimality is too stringent; this is particularly relevant for NP-hard problems, where finding an optimal solution is computationally prohibitive under P $\neq$ NP. In an attempt to circumvent this hardness, one may seek *approximately optimal* solutions for optimization problems, and the field of *approximation algorithms* studies how to find such approximately optimal solutions efficiently, that is, in polynomial time [282, 284]. In particular, it seeks algorithms that run in polynomial time and return solutions with a guaranteed *approximation ratio*. Roughly, we say that an algorithm has *approximation ratio* ρ if for every instance it returns a feasible solution of objective value within ratio ρ to the optimal. As a textbook example, the classic Vertex Cover problem, albeit being NP-hard, admits a simple polynomial-time 2-approximation algorithm based on finding a maximal matching of the graph [83, Theorem 35.1]. Some problems seem to be more amenable to approximation than others; there are even problems that admit what is known as a *polynomial-time approximation scheme* (PTAS), which is an algorithm that, for every $\varepsilon > 0$, runs in polynomial time (for fixed ε) and returns a $(1+\varepsilon)$-approximate solution for minimization problems (or $(1-\varepsilon)$ for maximization problems instead).[6] On the other hand, not all problems are equally approximable, with some being hard to approximate even within a factor of $n^{1-\varepsilon}$ for any constant $\varepsilon > 0$, unless P = NP [294]. These limits are often established via the celebrated *PCP Theorem* [12, 99], which provides the theoretical underpinning for inapproximability results. Thus, when exact algorithms are out of reach and classical approximation hits PCP barriers, one may ask whether *structure* can still be exploited.

This naturally leads to *parameterized approximation*, which combines structural (or solution-size) parameterization with approximation guarantees [115, 202, 236]. In particular, it addresses problems that are hard to solve exactly even under severe restrictions on the input, or where classical approximation guarantees are weak. The goal is to design algorithms that run ideally in FPT time and produce solutions with provable approximation guarantees, that overcome the limitations established under each approach individually. Here, given a parameter k (which may or may not be a structural measure such as treewidth) we seek algorithms running in time FPT in k and producing solutions with a proven ratio. Such algorithms may achieve an approximation ratio ρ in time $f(k) \cdot n^{\mathcal{O}(1)}$, or even offer *efficient FPT-approximation schemes* (FPT-AS) that output $(1+\varepsilon)$-approximate solutions in time $f(k,\varepsilon) \cdot n^{\mathcal{O}(1)}$ for all $\varepsilon > 0$. This hybrid approach has found success in a variety of settings, particularly for graph problems [21, 74, 109, 116, 127, 174, 177, 189, 207, 227, 228, 229, 230], and continues to grow as a versatile tool in algorithm design. On the hardness side, as an analog

[6] With respect to the running time, a PTAS behaves like an XP algorithm parameterized by ε.

to the PCP theorem, the *Parameterized Inapproximability Hypothesis* (PIH) was introduced by Lokshtanov, Ramanujan, Saurabh, and Zehavi [228] and has been used to prove inapproximability results in the parameterized setting for several canonical problems, e.g. [68, 69, 81, 185, 251].[7] In a breakthrough result, it was recently shown that PIH is implied by the ETH [158], with several follow-up works establishing further improvements [15, 157]. There has also been a long line of research on proving hardness of approximation results for canonical W[1]- and W[2]-complete problems such as Clique, Set Cover, and Dominating Set in the parameterized setting, see e.g. [66, 183, 184, 218, 219, 220, 221] and the references therein, while several works have focused on establishing parameterized inapproximability results under the weaker FPT $\neq$ W[1] conjecture [27, 35, 65, 66, 285, 286]; see also the recent survey of Chen and Lin [67].

1.2. Our Contributions

Our central goal in this thesis is to expand the framework of structurally parameterized complexity and fine-grained analysis by contributing *reusable* techniques that clarify how structural parameters govern algorithmic tractability. We position this work within the broader quest to chart the algorithmic landscape of NP-hard graph problems using tools from parameterized complexity, fine-grained complexity, and approximation algorithms. Concretely, we both *develop* new techniques, enabling sharper upper and lower bounds under standard structural measures, and *apply* this enriched toolkit to a set of canonical graph problems to obtain tight upper and lower bounds, structurally parameterized (in)approximability results, and precise borders between tractability and hardness. We believe that our contributions not only advance the state of the art in the specific problems we study, but also provide a foundation for future research in this area and that our techniques can be lifted to other problems beyond those studied here.

We pursue two complementary facets. First, we delineate the precise boundaries where problems transition from being fixed-parameter tractable to being W[1]-hard, thereby identifying where efficient parameterized algorithms are out of reach. Second, we study the optimality of various parameterized algorithms via fine-grained lower bounds under various hypotheses from fine-grained complexity, such as the ETH, SETH, and pw-SETH, matching them against the running times of our algorithms whenever possible. When exact computation is out of reach, we further explore the potential of approximation algorithms to find near-optimal solutions efficiently, and in particular we design FPT approximation schemes that leverage the structure of the input graph, or in some cases prove that no such schemes can exist.

FPT vs. W[1]-hardness. In our exploration of the algorithmic landscape, we pay particular attention to the distinction between fixed-parameter tractability (FPT) and W[1]-hardness. We aim to delineate the precise boundaries where problems transition from being FPT to W[1]-hard, providing a clearer understanding of the challenges involved

[7] A lot of these inapproximability results are phrased under the *Gap-ETH Hypothesis* [100, 235], which implies the PIH [101].

in designing efficient algorithms for these problems. Our key technical contribution here is the introduction of a novel technique that allows us to establish W[1]-hardness results for problems under very restrictive structural parameterizations. This technique is based on the Unary Bin Packing problem [179]; revisiting its W[1]-hardness proof, we notice that in fact this hardness persists even if every item may be placed in exactly two bins. Coupled with pseudopolynomial algorithms for Subset Sum [22], this "two-choice unary packing" variant lets us transfer W[1]-hardness to graph-cut problems while tightly controlling parameters such as feedback edge set number and maximum degree. This yields hardness under structural parameterizations that were previously out of reach (see Chapters 6 and 7), and we expect the technique to be broadly reusable beyond the problems treated here. Overall, this facet provides new avenues to certify W[1]-hardness in settings where structure is severely constrained, thereby clarifying where FPT is presumably unattainable (unless FPT = W[1]).

Fine-grained Lower Bounds. In addition to establishing W[1]-hardness results, we also investigate the implications of the ETH as well as of the (pw-)SETH on the parameterized complexity of our problems. To this end, we develop a variety of new techniques and constructions that allow us to establish tight lower bounds under the ETH, both for problems that are known to be FPT and for those that are W[1]-hard. In this context, we devise a recursive construction that allows us to obtain ETH-tight lower bounds for W[1]-hard problems parameterized by tree-depth, taking full advantage of the recursive nature of this parameterization. The core idea lies into designing so-called *copy gadgets*, and carefully combining them in a recursive manner to keep the tree-depth of the whole construction linear. This allows us to establish tight lower bounds for problems parameterized by tree-depth improving over previous reductions that result in (at least) quadratic tree-depth growth (see Chapters 3 and 5). For FPT regimes, we give a new application of the technique of d-detecting families introduced by Bonamy et al. [49], previously used to establish tight lower bounds for the natural parameterization of the Multicoloring problem, to obtain tight lower bounds for problems parameterized by vertex cover. We believe that this technique is of independent interest and is a promising approach that can be applied to other problems as well (see Chapter 3). Finally, leveraging the (pw-)SETH, we certify that single-exponential treewidth algorithms have optimal bases even under more restrictive parameterizations such as pathwidth (see Chapters 3 and 4). Although the overarching strategy is by now standard, the concrete reductions and gadget orchestration are problem-specific and require us to overcome significant technical challenges. Overall, our contributions in this area provide a deeper understanding of the complexity landscape of our problems and highlight the importance of fine-grained complexity analysis in the study of parameterized algorithms.

Parameterized Approximation. Lastly, we explore the potential of approximation algorithms to find near-optimal solutions efficiently, especially when exact computation is out of reach. In particular, we introduce a three-step construction that, despite its simplicity, allows us to design efficient FPT approximation schemes that bypass the

limitations of exact computation. We apply this technique in Chapter 5 to obtain an efficient FPT approximation scheme for the MaxNDP problem parameterized by tree-depth, making use of win/win arguments and a careful analysis of the running time of the algorithm. We find this technique to be of independent interest and believe that it can be applied to other problems as well. Moreover, we identify structural frontiers where such schemes cannot exist, thereby demarcating the scope of parameterized approximation.

Our overarching objective is to provide a cohesive picture of how structural parameters mediate tractability: we pinpoint parametric running-time dependencies, identify barriers that preclude efficient parameterized algorithms, and investigate when approximation can overcome those barriers. Our contributions span Chapters 3 to 7, each focusing on a particular problem or class of problems, and we synthesize the cross-cutting insights that emerge from these chapters in Section 1.3. We now give a high-level overview of our contributions, deferring technical details to the respective chapters.

Chapter 3. Here we revisit two well-studied problems, Bounded Degree Vertex Deletion and Defective Coloring, where the input is a graph G and a target degree Δ and we are asked either to edit or partition the graph so that the maximum degree becomes bounded by Δ; those are generalizations of the well-known Vertex Cover and Coloring problems which correspond to the case $\Delta = 0$. Both Bounded Degree Vertex Deletion and Defective Coloring are known to be parameterized intractable for the most well-known structural parameters, such as treewidth. We obtain a comprehensive picture of the complexity of both problems with respect to various structural parameters, focusing on the fine-grained complexity of both problems with respect to treewidth, tree-depth, and vertex cover number. In particular, we obtain tight lower bounds for both problems with respect to the parameterizations by tree-depth, vertex cover number, as well as treewidth plus Δ, under the ETH and SETH. To obtain these results, we develop a variety of new techniques and constructions that may be of independent interest. Specifically, we develop a recursive construction that allows us to obtain ETH-tight lower bounds for tree-depth, as well as a new application of the technique of d-detecting families [49] to obtain tight lower bounds for vertex cover number. Our results settle the complexity of both problems with respect to the aforementioned parameters. The results of this chapter have been published in [214, 215].

Chapter 4. Next, we revisit the structurally parameterized complexity of Induced Matching and Acyclic Matching, two problems where we seek to find a maximum matching whose endpoints induce, respectively, a matching and a forest. Chaudhary and Zehavi [61] recently studied these problems parameterized by treewidth. We resolve several of the problems left open in their work and extend their results as follows: (i) for Acyclic Matching, they gave an algorithm of running time $6^{\mathrm{tw}}n^{\mathcal{O}(1)}$ and a lower bound of $(3-\varepsilon)^{\mathrm{tw}}n^{\mathcal{O}(1)}$ (under the SETH); we close this gap by, on the one hand giving a more careful analysis of their algorithm showing that its complexity is actually $5^{\mathrm{tw}}n^{\mathcal{O}(1)}$, and on the other giving a pw-SETH-based lower bound showing that this running time cannot be improved (even for pathwidth), (ii) for Induced Matching we show that

their $3^{\mathrm{tw}}n^{\mathcal{O}(1)}$ algorithm is optimal under the pw-SETH (in fact improving over this for pathwidth or even for cutwidth is *equivalent* to falsifying the pw-SETH) by adapting our reduction for Bounded Degree Vertex Deletion from Chapter 3, (iii) for Induced Matching we give an FPT algorithm parameterized by clique-width with running time $3^{\mathrm{cw}}n^{\mathcal{O}(1)}$, which is optimal under the pw-SETH from our previous result; in doing so we make use of the idea of *state representations* introduced by van Rooij, Bodlaender, and Rossmanith [281]. The results of this chapter have been published in [217], with the paper receiving the *Best Student Paper Award* of WG 2025.

Chapter 5. Moving on, we study the Maximum Node-Disjoint Paths problem, the natural optimization version of the famous Node-Disjoint Paths problem, where we are given an undirected graph G, k (demand) pairs of vertices (s_i, t_i), and an integer ℓ, and are asked whether there exist at least ℓ vertex-disjoint paths in G whose endpoints are given pairs. This problem has been intensely studied from both the approximation and parameterized complexity point of view and is notably known to be intractable by standard structural parameters, such as tree-depth, as well as the combined parameter ℓ plus pathwidth. We present several results improving and clarifying this state of the art, with an emphasis towards FPT approximation. Our main result is a dichotomy regarding the structurally parameterized approximability of the problem. Specifically, we show via a simple three-step algorithm based on color-coding and win/win arguments that for the parameterization by tree-depth the problem admits an *efficient FPT approximation scheme*, returning a $(1-\varepsilon)$-approximate solution in time $f(\mathrm{td}, \varepsilon)n^{\mathcal{O}(1)}$. On the other hand, we show that under the Parameterized Inapproximability Hypothesis no such scheme is possible for the parameterization by pathwidth, even in time $f(\mathrm{pw}, \varepsilon)n^{g(\varepsilon)}$. In other words, we precisely determine the parameter border where the problem transitions from "hard but approximable" to "inapproximable". The results of this chapter have been published in [216].

Chapter 6. In this chapter, we study two closely related cut problems that seek to minimize the size of the maximum connected component of a graph under a tight deletion budget, Vertex Integrity and Component Order Connectivity. Apart from answering several open questions from the literature, our main result is to show that both problems remain W[1]-hard even when parameterized by the combined parameter $\mathrm{fes} + \Delta + \mathrm{pw}$, thus significantly improving over the previous W[1]-hardness results. To obtain this result, we develop a new reduction technique that may be of independent interest. In particular, we notice that the known reduction that establishes W[1]-hardness for the Unary Bin Packing problem parameterized by the number of bins [179] holds even when the input is restricted to instances where each item has to choose between *exactly two* possible bins. This observation, along with the fact that the input numbers are given in unary, allows us to leverage pseudopolynomial-time algorithms of Subset Sum and obtain a polynomial-time reduction to a variant of Component Order Connectivity that we eventually reduce to our problems of interest. The results of this chapter have been published in [164].

Chapter 7. Finally, we study how to sparsify the global connectivity of a graph under a tight deletion budget. Given a graph G and integers $k, x \geq 0$, Critical Node Cut (CNC) asks whether we can delete at most k vertices so that the number of remaining unordered pairs of connected vertices is at most x. Previous work has established that CNC is W[1]-hard parameterized by $k + \mathrm{tw}$ [5], and our main result is to significantly strengthen this result. In particular, we prove W[1]-hardness for the combined parameter $k+\mathrm{fes}+\Delta+\mathrm{pw}$, by making use of the same reduction technique as in Chapter 6. Although the construction of our reduction is identical, the analysis is quite more involved as we need to carefully account for the number of connected pairs in the remaining graph, which is a *global* property, in contrast to the locality of the size of a component. Beyond this main result, we also obtain several complementary positive results. Among these, we give an FPT approximation scheme that, for any $\varepsilon > 0$, computes a $(1+\varepsilon)$-approximate solution in time $(\mathrm{tw}/\varepsilon)^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$, by leveraging a technique introduced by Lampis [207]. Finally, we show that CNC admits no polynomial kernel when parameterized by vertex cover number, unless standard assumptions fail.

1.3. Synthesis and Takeaways

We now synthesize the cross-cutting insights that emerge from Section 1.2. Across five chapters, we obtain a coherent picture of *optimal DP regimes*, *approximation rescues*, and *hardness walls* under various structural parameterizations. Concretely, we (1) separate between parameters that enable fixed-parameter tractability and those that do not (under FPT $\neq$ W[1]); (2) pin down optimal running times for several structural parameterizations under fine-grained hypotheses; and (3) deploy approximation to overcome exact-computation barriers. Along the way, we develop techniques that reappear across chapters, providing a common language that connects DP tables, lower bounds, and approximation. We now discuss these themes in more detail.

As regards the FPT vs. W[1]-hardness stratification, we delineate the complexity border for the problems we study. In particular, we identify parameters that allow for FPT algorithms and parameters for which such algorithms are unlikely, thereby clarifying where efficient parameterized computation is out of reach. A key ingredient is a new reduction framework that yields W[1]-hardness even under restrictive structural parameterizations. We apply it to show W[1]-hardness for Vertex Integrity, Component Order Connectivity, and Critical Node Cut parameterized by the combined parameter $\mathrm{fes}+\Delta+\mathrm{pw}$, which substantially improves upon previous hardness results and, we believe, is of independent utility beyond the problems treated here. On the positive, we present several FPT algorithms using a range of tools, including state representations (Chapter 4), color-coding and branching (Chapter 5), and win/win arguments (Chapter 6). Taken together, these results assemble a clear, technique-driven picture of the FPT vs. W[1]-hardness frontier for our problem set.

A second recurring theme is the use of fine-grained hypotheses to obtain tight lower bounds that match natural dynamic programs. For Bounded Degree Vertex Deletion, Defective Coloring, Induced Matching, and Acyclic Matching, simple

bag-wise DP states, augmented when needed with more sophisticated techniques such as subset convolution, Cut & Count, or FFT, lead to single-exponential $c^{\mathrm{tw}}n^{\mathcal{O}(1)}$ algorithms. We prove that these algorithms are optimal under SETH or pw-SETH; in fact, improving the base c even under the more restrictive parameterization by pathwidth would refute the underlying hypothesis. This pins down the exact exponent and formally justifies that the "obvious" tables are best possible in the rigorous, fine-grained sense. Methodologically, we develop reductions and constructions that both leverage problem-specific structure and reuse components introduced in Chapter 3. For Induced Matching specifically, we strengthen the reduction for Bounded Degree Vertex Deletion to obtain a tight lower bound under the (weaker) pw-SETH and even establish equivalence for both the parameterizations by pathwidth and cutwidth. Furthermore, towards our SETH-based lower bound for Defective Coloring, we design an extensive toolkit of gadgets that we expect to be applicable to other parameterizations of the problem.

We also introduce techniques for tight ETH-based lower bounds under structural parameters beyond treewidth. In particular, we design a recursive construction that, by fully exploiting the recursive nature of tree-depth, rules out $n^{o(\mathrm{td})}$-time algorithms for Bounded Degree Vertex Deletion, Defective Coloring, and MaxNDP under the ETH. This is a novel approach for obtaining tight lower bounds for tree-depth, and it highlights that, despite being significantly more restrictive than treewidth, tree-depth still forbids sublinear-in-td exponents in the targeted problems. Moreover, we give a new application of the technique of d-detecting families [49] to derive tight ETH-based $\mathrm{vc}^{o(\mathrm{vc})}n^{\mathcal{O}(1)}$ lower bounds for Bounded Degree Vertex Deletion and Defective Coloring parameterized by vertex cover number. This technique was previously used to establish tight lower bounds for the natural parameterization of the Multicoloring problem [49], and here we exhibit its applicability to lower bounds for structural parameterizations by vertex cover number. We view both the tree-depth recursion and the detecting-family application as broadly reusable techniques that are of independent interest and can be applied to other problems as well.

Yet another theme explored in this thesis is the use of approximation to overcome exact-computation barriers. We identify structural frontiers where efficient FPT approximation schemes can and cannot exist, thereby demarcating the scope of parameterized approximation. For MaxNDP, we give an efficient FPT approximation scheme under the parameterization by tree-depth, while, under the Parameterized Inapproximability Hypothesis, we show that no such scheme exists for the parameterization by pathwidth. For the former result, our proof template is a simple three-step construction that, despite its simplicity, yields efficient schemes precisely where exact methods stall. The construction is modular and, we believe, transferable to other problems. For Critical Node Cut, we obtain an FPT approximation scheme parameterized by treewidth by leveraging a technique of Lampis [207]; it is easy to see that we can apply the same technique to obtain similar approximation guarantees for the problems studied in Chapter 6.

In summary, we (i) settle the tractability boundary under different structural parameters (FPT vs. W[1]-hardness); (ii) match natural DP upper bounds with tight fine-grained lower bounds; (iii) prove tight lower bounds for structural parameters beyond treewidth; and (iv) when exact computation provably fails, add approximation into the mix in order

to recover efficient algorithms. These themes interlock across chapters and set up the technical developments that follow.

2. Preliminaries

In this chapter we record the notation, definitions, and standing hypotheses used throughout the dissertation. We assume basic familiarity with algorithms [83], graphs [98], and (classical) complexity [11].

2.1. Notation and Basic Definitions

General Notation. We use $\mathbb{N}$ to denote the set of non-negative integers, while $\mathbb{Z}$ denotes the set of all integers. For $x, y \in \mathbb{Z}$, let $[x, y] = \{z \in \mathbb{Z} : x \leq z \leq y\}$ denote the set of integers in-between x and y, while $[x] = [1, x]$. For a set S and $c \in \mathbb{N}$ let $\binom{S}{c}$ denote the set of subsets of S of size c, i.e., $\binom{S}{c} = \{S' \subseteq S : |S'| = c\}$. For a (multi)set of positive integers S, let $\Sigma(S)$ denote the sum of its elements. Given a (partial) function $f\colon A \to B$, for every $b \in B$ we denote by $f^{-1}(b)$ the set of preimages of b under f, that is, $f^{-1}(b) = \{a \in A : f(a) = b\}$. Notice that in the case of partial functions it is not necessary for f to be defined over the whole set A. For a (partial) function $f\colon A \to B$, an element $a \in A$ and a value $b \in B$, we write $f[a \mapsto b]$ for the (partial) function obtained by updating (or inserting) the image of a to b. Standard $\mathcal{O}^\star(\cdot)$ notation is used to suppress polynomial factors in the input size.

Graphs. We use standard graph notation [98]. Formally, a graph G consists of a finite vertex set $V(G)$ and finite edge set $E(G) \subseteq V(G) \times V(G)$, that is, we write $G = (V(G), E(G))$. We denote an edge $\{u, v\} \in E(G)$, also by uv. Following standard graph notation, we use n and m to denote the number of vertices $|V|$ and the number of edges $|E|$ respectively. Unless explicitly stated, the graphs in this thesis are undirected, simple, and unweighted. For a vertex $v \in V(G)$, we denote by $N_G(v) = \{u \in V(G) : uv \in E(G)\}$ its open neighborhood, while $N_G[v] = N_G(v) \cup \{v\}$ denotes its closed neighborhood. We define the degree of v as the size of its open neighborhood, that is, $\deg_G(v) = |N_G(v)|$. We write $\Delta(G)$ for the maximum degree of G and $\mathtt{cc}(G)$ for the set of its connected components. For $S \subseteq V$, $G[S]$ is the subgraph induced by S, and $G - S$ abbreviates $G[V \setminus S]$; we write $G - v$ if $S = \{v\}$. In an analogous way, for $F \subseteq E$, $G - F$ is the graph obtained by removing the edges in F from G.

2.2. A Primer on Parameterized Complexity

In this section we briefly review the basics of parameterized complexity. The notions below suffice for the results referenced throughout the thesis; for a thorough exposition see [92, 103, 123].

An instance of a *parameterized problem* is a pair (x, k), where x is the main input (with size $n := |x|$) and $k \in \mathbb{N}$ is the parameter. We say that a parameterized problem is *fixed-parameter tractable* (FPT) and belongs to the class FPT if it admits an algorithm of running time $f(k) \cdot n^{\mathcal{O}(1)}$ for some computable function f. We say that a parameterized problem is *slice-wise polynomial time* (XP) and belongs to the class XP if it admits an algorithm of running time $n^{g(k)}$ for some computable function g. It holds that FPT $\subsetneq$ XP [103].

Let Π_1 and Π_2 be two parameterized problems. A *parameterized reduction* (also known as an *FPT reduction*) from Π_1 to Π_2 is an algorithm that, given an instance (x, k) of Π_1, produces an instance (x', k') of Π_2 such that:

- (x, k) is a yes-instance of Π_1 if and only if (x', k') is a yes-instance of Π_2,
- $k' \leq g(k)$ for some computable function g,
- the running time of the algorithm is $f(k) \cdot |x|^{\mathcal{O}(1)}$ for some computable function f.

Notice that if Π_2 is fixed-parameter tractable (FPT), then so is Π_1.

We will use parameterized reductions to show hardness results for parameterized problems. In particular, we define the class W[1] as the set of parameterized problems that are FPT-reducible to k-Clique parameterized by k.[1] It holds that FPT $\subseteq$ W[1] $\subseteq$ XP [103], and it is widely believed that FPT $\neq$ W[1], thus showing that a problem is W[1]-hard is strong evidence against fixed-parameter tractability [92, 103]. We also mention in passing that the ETH (see Section 2.5) implies that FPT $\neq$ W[1], while one can easily show that if P = NP then FPT = W[1], which, by contraposition, implies that FPT $\neq$ W[1] $\implies$ P $\neq$ NP. We omit any further details on the W-hierarchy as we will not need them in this dissertation, and we refer the interested reader to [103, 123].

Another complexity class considered in this dissertation is XNLP. This class was recently introduced by Bodlaender, Groenland, Nederlof, and Swennenhuis [41], and it consists of parameterized problems decidable by a nondeterministic algorithm that, on inputs of size n with parameter k, uses time $f(k) \cdot n^{\mathcal{O}(1)}$ and space $f(k) \cdot \log n$, for some computable function f. It holds that W[1] $\subseteq$ XNLP $\subseteq$ XP, and in fact XNLP-hardness implies W[t]-hardness for all $t \in \mathbb{N}$. In order to show XNLP-hardness results, we need to introduce the notion of *parameterized logspace reductions*, which are simply parameterized reductions that use $\mathcal{O}(g(k) + \log |x|)$ space, where g is a computable function. In that case, if Π_1 is XNLP-hard and there is a parameterized logspace reduction from Π_1 to Π_2, then Π_2 is also XNLP-hard [41].

A *kernelization* for a parameterized decision problem Π is a polynomial-time algorithm that maps any instance (x, k) of Π to an equivalent instance (x', k') of Π such that $|x'| \leq h(k)$ and $k' \leq g(k)$ for some computable functions h, g. The output (x', k') is a *kernel*, and its *size* is $|x'|$; if $h(k)$ is polynomial in k, we speak of a *polynomial kernel*. A parameterized problem is in FPT if and only if it admits a (not necessarily polynomial) kernel [92, 132].

[1] The k-Clique problem asks, given a graph G and an integer k, whether G contains a clique of size at least k.

2.3. A Primer on Approximation Algorithms

In this section we briefly review the basics of approximation algorithms. The notions below suffice for the results referenced throughout the thesis; for a thorough exposition see [282, 284].

An *optimization problem* Π consists of a set of instances $\mathcal{I}$, a set of feasible solutions $\mathcal{S}(I)$ for each instance $I \in \mathcal{I}$, and an objective function val defined on pairs (I, S) with $I \in \mathcal{I}$ and $S \in \mathcal{S}(I)$. For an instance $I \in \mathcal{I}$, the optimal objective value is

$$\mathrm{OPT}(I) = \begin{cases} \max\{\mathrm{val}(I,S) : S \in \mathcal{S}(I)\} & \text{(maximization)}, \\ \min\{\mathrm{val}(I,S) : S \in \mathcal{S}(I)\} & \text{(minimization)}. \end{cases}$$

When the context is clear, we write OPT for $\mathrm{OPT}(I)$.

For a minimization problem, an algorithm $\mathcal{A}$ is a *ρ-approximation* (with $\rho \geq 1$) if, for every instance I, it outputs a feasible solution S with $\mathrm{val}(I,S) \leq \rho \cdot \mathrm{OPT}(I)$. For a maximization problem, $\mathcal{A}$ is a *ρ-approximation* (with $\rho \leq 1$) if it outputs S with $\mathrm{val}(I,S) \geq \rho \cdot \mathrm{OPT}(I)$. A *polynomial-time approximation scheme* (PTAS) for Π is a family $\{\mathcal{A}_\varepsilon\}_{\varepsilon>0}$ such that, for every fixed $\varepsilon > 0$, $\mathcal{A}_\varepsilon$ runs in time $n^{\mathcal{O}(1)}$, where n denotes the input size, and returns a $(1+\varepsilon)$-approximate solution for minimization (resp., $(1-\varepsilon)$ for maximization). An *efficient PTAS* (EPTAS) is a PTAS whose running time is $f(\varepsilon) \cdot n^{\mathcal{O}(1)}$, i.e., the exponent of n is independent of ε.

When the instance comes with a parameter k, a *fixed-parameter tractable approximation scheme* (FPT-AS) is a family $\{\mathcal{A}_{k,\varepsilon}\}$ that returns a $(1 \pm \varepsilon)$ approximate solution in time $f(k) \cdot n^{\mathcal{O}(1)}$ for every fixed $\varepsilon > 0$. An *efficient* FPT-AS has running time $f(k,\varepsilon) \cdot n^{\mathcal{O}(1)}$, i.e., the super-polynomial dependence is confined to k and ε, while the dependence on n remains polynomial.

We next recall *gap-preserving reductions*, a standard tool for inapproximability. Since definitions differ slightly between maximization and minimization, we focus on the former, which is the case used in Chapter 5. Let I_1 be an instance of a maximization problem Π_1 with $|I_1| = n$, which we reduce to an instance I_2 of a maximization problem Π_2 with $|I_2| = n'$. A gap-preserving reduction from Π_1 to Π_2 is specified by functions $(k(n), k'(n'), c(n), c'(n'))$ such that

- $\mathrm{OPT}_{\Pi_1}(I_1) \geq k \implies \mathrm{OPT}_{\Pi_2}(I_2) \geq k'$, and
- $\mathrm{OPT}_{\Pi_1}(I_1) < k/c \implies \mathrm{OPT}_{\Pi_2}(I_2) < k'/c'$.

2.4. Graph Parameters

A recurring theme in this dissertation is that the *structure* of the input graph often governs the tractability and the exact running time of our algorithms. To make this precise, we quantify structure via a handful of graph parameters. Each parameter emphasizes a different obstruction to "tree-likeness" (or to other forms of simplicity), and distinct parameters can lead to markedly different algorithmic behaviors. Throughout, we will evaluate and compare these measures, using them both to design dynamic programs and

approximation schemes and to state fine-grained lower bounds. Figure 2.1 summarizes the relationships among the presented parameters. In the remainder of this section we first give formal definitions, and then record some standard relationships.

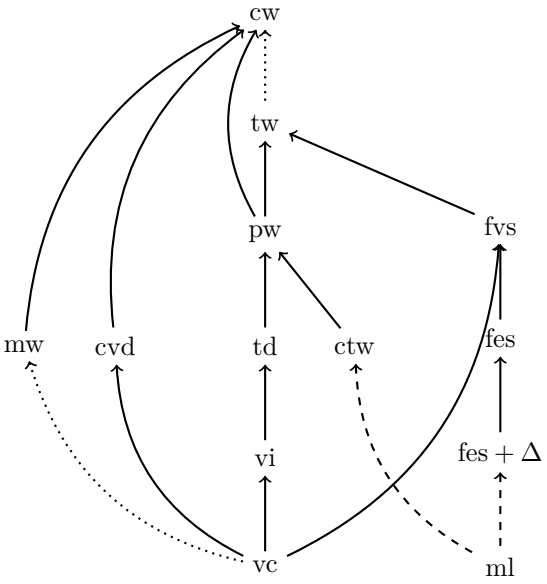


Figure 2.1.: The hierarchy of the introduced graph parameters.

2.4.1. Definitions

Treewidth (tw**).** The *treewidth* of a graph is a measure of how close the graph is to being a tree. A *tree-decomposition* $\mathcal{T}$ of a graph G is a pair $(T, \{X_t : t \in V(T)\})$ where T is a tree and each node t of T is assigned a *bag* $X_t \subseteq V(G)$, such that: (i) every vertex $v \in V(G)$ is contained in some bag X_t; (ii) for every edge $\{u, v\} \in E(G)$ there exists $t \in V(T)$ with $\{u, v\} \subseteq X_t$; and (iii) for every $v \in V(G)$, the set $\{t \in V(T) : v \in X_t\}$ induces a connected subtree of T. The *width* is $\max_{t \in V(T)} |X_t| - 1$; the *treewidth* $\mathrm{tw}(G)$ is the minimum width over all tree-decompositions of G. A tree-decomposition is *nice* if each node of T has one of the following types: (i) *Leaf* nodes t where $X_t = \varnothing$; (ii) *Introduce* nodes t with a unique child t' such that $X_t = X_{t'} \cup \{v\}$ for some $v \in V(G)$; (iii) *Forget* nodes t with a unique child t' such that $X_{t'} = X_t \cup \{v\}$ for some $v \in V(G)$; (iv) *Join* nodes t with exactly two children t_1, t_2 such that $X_t = X_{t_1} = X_{t_2}$. It is known that given any tree decomposition one can in linear time construct a nice tree decomposition of the same width. Computing a tree-decomposition of minimum width is in FPT when parameterized by the treewidth [38, 192, 198], and even more efficient algorithms exist for obtaining near-optimal tree-decompositions [197].

Pathwidth (pw**).** A *path-decomposition* is a tree-decomposition whose underlying tree is a path. Its width is defined as above, and the *pathwidth* $\mathrm{pw}(G)$ is the minimum such

width [43, 192].

Tree-depth (td**).** The *tree-depth* of a graph G is the minimum height of a rooted tree T on the same set of vertices such that every connected pair of vertices in G adheres to the descendant/ancestor relation in T. Equivalently,

$$\mathrm{td}(G) = \begin{cases} 1 & \text{if } |V(G)| = 1, \\ \max_{C \in \mathtt{cc}(G)} \mathrm{td}(C) & \text{if } G \text{ is disconnected}, \\ 1 + \min_{v \in V(G)} \mathrm{td}(G - v) & \text{otherwise.} \end{cases}$$

Computing the tree-depth of a graph is NP-hard, but it lies in FPT when parameterized by the tree-depth itself [247, 260].

Clique-width (cw**).** A *k-expression* builds a labeled graph from the following operations using labels in $[k]$: (i) creation $i(v)$ of a single vertex v with label i; (ii) disjoint union $\oplus$; (iii) $\eta_{i,j}$ which adds all edges between labels i and j ($i \neq j$); (iv) $\rho_{i \to j}$ which relabels all i-labeled vertices to j. The *clique-width* $\mathrm{cw}(G)$ is the minimum k such that G arises (after forgetting labels) from a k-expression [87]. An expression is *irredundant* if $\eta_{i,j}$ is never applied when an i-j edge already exists; It is known that any clique-width expression can be transformed in linear time into an irredundant one of the same width [87]. For a labeled graph G and $v \in V(G)$, let $\mathrm{lab}_G(v)$ denote the label of v in G, while $\mathrm{lab}_G^{-1}(i) = \{v \in V(G) : \mathrm{lab}_G(v) = i\}$ denotes the set of vertices of G of label i. Even though computing the clique-width of a graph is NP-hard [120], there exist algorithms that compute $f(\mathrm{cw}(G))$-expressions in FPT time [130, 255].

Cutwidth (ctw**).** The *cutwidth* of a graph G is the minimum integer k such that there exists a linear ordering $v_1, \ldots, v_n$ of the vertices of G with the following property: for every $i \in [n-1]$, the number of edges with one endpoint in $\{v_1, \ldots, v_i\}$ and the other endpoint in $\{v_{i+1}, \ldots, v_n\}$ is at most k. The cutwidth of a graph can be computed in FPT time when parameterized by the cutwidth itself [147, 275].

Modular-width (mw**).** A *module* of a graph $G = (V, E)$ is a set of vertices $M \subseteq V$ such that for all $x \in V \setminus M$ we have that x is either adjacent to all vertices of M or to none. The *modular-width* of a graph $G = (V, E)$ is the smallest integer k such that, either $|V| \leq k$, or V can be partitioned into at most $k' \leq k$ sets $V_1, \ldots, V_{k'}$, with the following two properties: (i) for all $i \in [k']$, V_i is a module of G, (ii) for all $i \in [k']$, $G[V_i]$ has modular width at most k [143]. We can compute the modular-width of a graph and a corresponding decomposition in linear time [274].

Vertex Integrity (vi**).** Formally, the *vertex integrity* $\mathrm{vi}(G)$ of a graph G is defined as

$$\mathrm{vi}(G) = \min_{S \subseteq V(G)} \{|S| + \max_{C \in \mathtt{cc}(G-S)} |V(C)|\}.$$

Intuitively, the vertex integrity measures the minimum number of vertices that need to be removed from G in order to break it into small connected components. Computing the vertex integrity of a graph is NP-hard, but it lies in FPT when parameterized by the vertex integrity itself [104].

Vertex Cover (vc). A *vertex cover* of a graph G is a set of vertices $S \subseteq V(G)$ such that $G - S$ is an independent set, i.e., every edge in $E(G)$ is incident to at least one vertex in S. The *vertex cover number* of G is denoted $\mathrm{vc}(G)$ and is the minimum size of a vertex cover in G. Computing the vertex cover number is well-known to be NP-hard, but it is also known to lie in FPT when parameterized by the vertex cover itself [92].

Cluster Vertex Deletion Number (cvd). The *cluster vertex deletion number* of a graph G is the minimum size of a set of vertices $S \subseteq V(G)$ such that $G - S$ is a disjoint union of cliques [102]. We denote the cluster vertex deletion number of G as $\mathrm{cvd}(G)$. Computing the cluster vertex deletion number is well-known to be NP-hard, but it is also known to lie in FPT when parameterized by the natural parameter [277].

Feedback Vertex Set (fvs). A *feedback vertex set* of a graph G is a set of vertices $S \subseteq V(G)$ such that $G - S$ is a forest. The *feedback vertex set number* of G is denoted $\mathrm{fvs}(G)$ and is the minimum size of a feedback vertex set in G. Computing the vertex cover number is well-known to be NP-hard, but it is also known to lie in FPT when parameterized by the natural parameter [92].

Feedback Edge Set (fes). A *feedback edge set* of a graph G is a set of edges $F \subseteq E(G)$ such that $G - F$ is a forest. The *feedback edge set number* of G is denoted $\mathrm{fes}(G)$ and is the minimum size of a feedback edge set in G. The feedback edge set number of a graph can be computed in polynomial time [98].

Max-leaf Number (ml). The *max-leaf number* of a graph G is the maximum number of leaves over all spanning trees of G. We denote the max-leaf number of G as $\mathrm{ml}(G)$. Computing the max-leaf number of a graph is NP-hard [146, ND2], but it admits a polynomial-time 2-approximation algorithm [270] and is FPT parameterized by the natural parameter [292].

2.4.2. Relationships Between Parameters

Having defined our structural parameters, we now explain how to read Figure 2.1. An arrow $A \to B$ means *generalization*: for every k, the class $\{G : A(G) \leq k\}$ is contained in $\{G : B(G) \leq \mathcal{O}(k)\}$. Intuitively, keeping the (asymptotic) bound the same, B covers at least as many graphs as A, hence B is "more general" than A. Dashed and dotted arrows carry the same meaning with a possible quadratic and exponential blow-up respectively, i.e., $B(G) \leq \mathcal{O}((A(G))^2)$ and $B(G) \leq 2^{\mathcal{O}(A(G))}$. If there is *no* arrow between two

parameters, they are incomparable: there exist graph families witnessing one parameter bounded while the other is unbounded.

The arrows are algorithmically useful as they induce parameter-preserving translations. If $A \to B$, then the identity map is an fpt-reduction from a problem parameterized by A to the same problem parameterized by B (since $B \leq \mathcal{O}(f(A))$ on every instance, for a function f). Consequently, *negative* results (e.g., W[1]-hardness) *propagate upwards* along arrows: hard for $A \Rightarrow$ hard for every B reachable from A. Symmetrically, *positive* results (FPT algorithms, kernels, FPT-AS) *propagate downwards*: FPT by $B \Rightarrow$ FPT by every A with $A \to B$. For example, an FPT algorithm parameterized by tw immediately yields FPT for the more restrictive parameters pw and fvs (because $\text{tw} \leq \text{pw}$ and $\text{tw} \leq \text{fvs} + 1$), while W[1]-hardness for fes propagates to fvs (since $\text{fvs} \leq \text{fes}$).

Viewed through this lens, several results in the sequel pinpoint the *frontier of generality* for each problem: the highest parameter in the map for which the problem remains in FPT before it turns W[1]-hard. Tracing this frontier, and, where possible, calibrating optimal bases in the running time along it, is a recurring theme of this thesis.

We now argue about the relationships in Figure 2.1. Most are standard; we provide references where available and give brief proof sketches otherwise. We additionally argue that $\text{ctw} \leftrightarrow \text{pw} + \Delta$.

Linear dominations (solid arrows). For every graph G:

- $\text{tw}(G) \leq \text{pw}(G)$, since path-decompositions are tree-decompositions.
- $\text{pw}(G) \leq \text{td}(G)$, by induction on td: remove a vertex whose deletion reduces td by one and add it to all bags of a path-decomposition of $G - v$ (glue components when G is disconnected).
- $\text{td}(G) \leq \text{vi}(G)$, by induction on vi using a separator S whose removal leaves components of total size at most $\text{vi}(G) - |S|$.
- $\text{vi}(G) \leq \text{vc}(G) + 1$, since a vertex cover is a separator that leaves an edgeless graph (all components are singletons).
- $\text{tw}(G) \leq \text{fvs}(G) + 1$, by placing a minimum feedback vertex set S in every bag of a tree-decomposition of the forest $G - S$.
- $\text{fvs}(G) \leq \text{fes}(G)$, trivial.
- $\text{cvd}(G) \leq \text{vc}(G)$, as deleting a vertex cover yields a cluster graph.
- $\text{cw}(G) \leq \text{mw}(G)$: if $V(G)$ is partitioned into modules $V_1, \ldots, V_k$ and each $G[V_i]$ has a k-expression, then relabel each $G[V_i]$ so all its vertices share label i, take the disjoint union, and add all required joins between labels corresponding to adjacent modules (edges are either complete or absent between any pair of modules) [209].
- $\text{cw}(G) \leq \text{cvd}(G) + 3$, by giving each vertex in a cluster deletion set a distinct label, constructing cliques one by one, and using an auxiliary junk label.

- $\mathrm{pw}(G) \leq \mathrm{ctw}(G)$ [39, Theorem 47].
- $\mathrm{cw}(G) \leq \mathrm{pw}(G) + 2$, by assigning distinct labels to vertices in each bag of a path-decomposition and using one junk label for forgotten vertices while introducing along the path.

Quadratic dominations (dashed arrows).

- A graph G is a subdivision of a graph on $\mathcal{O}(\mathrm{ml}(G))$ vertices (see [190] and Lemma 6.17). Hence $\mathrm{fes}(G), \mathrm{ctw}(G) = \mathcal{O}((\mathrm{ml}(G))^2)$ and $\Delta(G) = \mathcal{O}(\mathrm{ml}(G))$, as all of them are invariant under subdivisions.

Exponential dominations (dotted arrows).

- $\mathrm{cw}(G) \leq 4 \cdot 2^{\mathrm{tw}(G)-1} + 1$ [84].
- $\mathrm{mw}(G) \leq 2^{\mathrm{vc}(G)} + \mathrm{vc}(G)$, since each vertex outside a minimum vertex cover has a neighborhood contained in the cover, yielding at most $2^{\mathrm{vc}(G)}$ distinct modules.

Strictness (witness families). All arrows above are strict: the target parameter is not bounded by any $\mathcal{O}(\cdot)$-function of the source. Representative families:

- $\mathrm{cw} \nrightarrow \mathrm{tw}$. Cliques K_n satisfy $\mathrm{cw}(K_n) = 2$ while $\mathrm{tw}(K_n) = n - 1$.
- $\mathrm{tw} \nrightarrow \mathrm{pw}$. Complete binary trees have $\mathrm{tw} = 1$ but $\mathrm{pw} = \Omega(\log n)$ [265].
- $\mathrm{pw} \nrightarrow \mathrm{td}$. Paths P_n have $\mathrm{pw}(P_n) = 1$ but $\mathrm{td}(P_n) = \Omega(\log n)$.
- $\mathrm{td} \nrightarrow \mathrm{vi}$. A forest of n stars $K_{1,n}$ has $\mathrm{td} = 2$ but $\mathrm{vi} = \Omega(n)$.
- $\mathrm{vi} \nrightarrow \mathrm{vc}$. Subdividing every edge of a star $K_{1,n}$ yields constant vi but $\mathrm{vc} = \Omega(n)$.
- $\mathrm{cw} \nrightarrow \mathrm{cvd}$. $K_{n,n}$ has $\mathrm{cw} = 2$ but $\mathrm{cvd} = \Omega(n)$.
- $\mathrm{cvd} \nrightarrow \mathrm{vc}$. Cliques K_n satisfy $\mathrm{cvd}(K_n) = 0$ while $\mathrm{vc}(K_n) = n - 1$.
- $\mathrm{cw} \nrightarrow \mathrm{mw}$. A path P_n has $\mathrm{cw}(P_n) \leq 3$ but no nontrivial modules for $n \geq 4$.
- $\mathrm{mw} \nrightarrow \mathrm{vc}$. Cliques K_n satisfy $\mathrm{mw}(K_n) = 1$ while $\mathrm{vc}(K_n) = n - 1$.
- $\mathrm{tw} \nrightarrow \mathrm{fvs}$. A disjoint union of n triangles has $\mathrm{tw} = 2$ but $\mathrm{fvs} = n$.
- $\mathrm{fvs} \nrightarrow \mathrm{vc}$. Paths P_n have $\mathrm{fvs}(P_n) = 0$ but $\mathrm{vc}(P_n) = \Omega(n)$.
- $\mathrm{fvs} \nrightarrow \mathrm{fes}$. $K_{2,n}$ has $\mathrm{fvs}(K_{2,n}) = 1$ and $\mathrm{fes}(K_{2,n}) = \Omega(n)$.
- $\mathrm{fes} \nrightarrow \mathrm{fes} + \Delta$. A star $K_{1,n}$ has $\mathrm{fes} = 0$ and $\Delta = n$.
- $\mathrm{fes} + \Delta \nrightarrow \mathrm{ml}$. A path where each vertex gets one leaf has $\mathrm{fes} = 0$, $\Delta = 3$, and $\mathrm{ml} = \Omega(n)$.
- $\mathrm{pw} \nrightarrow \mathrm{ctw}$. A star $K_{1,n}$ has $\mathrm{pw} = 1$ and $\mathrm{ctw} = \Omega(n)$.
- $\mathrm{ctw} \nrightarrow \mathrm{ml}$. A path where each vertex gets one leaf has $\mathrm{ctw} \leq 3$ but $\mathrm{ml} = \Omega(n)$.

Incomparabilities (no arrow in the figure). For the following pairs (A, B) it holds that neither A bounds B nor B bounds A by any function.

- The disjoint union of n triangles has small vi (hence also small td and pw), ctw, cvd, and mw, but fvs $= n$ (and thus fes $\geq n$). A complete binary tree has fes $= 0$, $\Delta = 3$, and pw $= \Omega(\log n)$ (and thus td, vi, ctw $= \Omega(\log n)$), as well as cvd, mw $= \Omega(\log n)$. Hence fes, fes $+ \Delta$, and fvs are incomparable to pw, td, vi, ctw, cvd, and mw.
- A star $K_{1,n}$ has vc $= 1$ (hence small vi, td, cvd, and mw) but ctw $= \Omega(n)$ (and therefore ml $= \Omega(\sqrt{n})$). A path P_n has td, cvd, mw $= \Omega(\log n)$ (hence large vc) but ml $= 2$ (and ctw $= 1$). Hence ml and ctw are incomparable to vc, vi, td, cvd, and mw.
- A clique K_n has mw $= 1$ and cvd $= 0$, but tw $= n - 1$ (and thus pw, td, vi, vc $= \Omega(n)$). Conversely, subdividing every edge of $K_{1,n}$ twice yields vi $= 4$ but mw, cvd $= \Omega(n)$. Hence mw and cvd are incomparable to vi, td, pw, and tw.

Cutwidth vs. Pathwidth plus Maximum Degree. We now argue that ctw $\leftrightarrow$ pw $+ \Delta$. We have already argued that pw $\leq$ ctw. To see that $\Delta \leq 2$ctw, consider a linear ordering of G achieving cutwidth ctw, and notice that for any vertex v, all its incident edges must cross the cut right before or right after v in the ordering. Lastly, it holds that ctw $\leq$ pw $\cdot \Delta$ [43, Theorem 49].

2.5. Complexity Assumptions and Hypotheses

Here we collect the complexity assumptions used throughout our hardness proofs and fix notation for the starting problems from which we reduce. We first recall the canonical satisfiability problem k-SAT.

k-SAT

Instance:	A boolean formula in conjunctive normal form (CNF) with at most k literals per clause.
Goal:	Determine whether there exists an assignment to the variables that satisfies all clauses.

ETH. The *Exponential Time Hypothesis* (ETH) posits that there exists a constant $c > 0$ such that no algorithm solves 3-SAT on n variables and m clauses in time $2^{c(n+m)} \cdot (n+m)^{\mathcal{O}(1)}$. We will often rely on a weaker form stating that 3-SAT with n variables and m clauses cannot be solved in time $2^{o(n+m)}$.

We next formalize the k-Multicolored Clique problem, which repeatedly serves as a starting point for reductions establishing ETH-based lower bounds and W[1]-hardness. k-Multicolored Clique parameterized by k is the canonical W[1]-complete problem; moreover, it admits no $f(k)\, n^{o(k)}$-time algorithm, for any computable f, unless ETH fails [63, 92].

k-Multicolored Clique

Instance: A graph $G = (V, E)$ and a partition of V into k independent sets $V_1, \ldots, V_k$, each of size n.
Goal: Determine whether G contains a k-clique.

SETH. The *Strong Exponential Time Hypothesis* (SETH) states that for every $\varepsilon > 0$ there exists a constant $k \geq 3$ (depending only on ε) such that no algorithm solves k-SAT on n variables in time $(2 - \varepsilon)^n \cdot n^{\mathcal{O}(1)}$.

We also introduce a closely related constraint-satisfaction problem that will be useful in SETH-based reductions.

q-CSP-B

Instance: A Constraint Satisfaction (CSP) instance with n variables and m constraints. The variables take values in a set Y of size B, i.e. $|Y| = B$. Each constraint involves at most q variables and is given as a list of satisfying assignments for these variables, where a satisfying assignment is a q-tuple of values from the set Y given to each of the q variables.
Goal: Determine whether there exists an assignment of the variables that satisfies all the constraints.

Theorem 2.1 ([208])**.** *For any $B \geq 2$ it holds that, if the SETH is true, then for all $\varepsilon > 0$, there exists a q such that n-variable q-CSP-B cannot be solved in time $\mathcal{O}^\star((B - \varepsilon)^n)$.*

pw-SETH. The *primal pathwidth SETH* (pw-SETH) states that, for all $\varepsilon > 0$, 3-SAT requires time at least $(2 - \varepsilon)^{\mathrm{pw}} n^{\mathcal{O}(1)}$, where pw is the pathwidth of the primal graph of the input formula.[2] In other words, the pw-SETH posits that the simple DP algorithm which solves 3-SAT in time $\mathcal{O}^\star(2^{\mathrm{pw}})$ is best possible. Beating this algorithm seems to encapsulate the difficulty of improving upon simple DP algorithms in general and indeed the pw-SETH is *equivalent* to many tight lower bounds for problems parameterized by linear structure widths (e.g., pathwidth or linear clique-width) [210]. The pw-SETH also has the advantage of being implied by several other standard assumptions, such as the Set Cover Conjecture and the SETH for circuits of depth εn [211], making lower bound results based on the pw-SETH seem more believable.

We also recall some notions and results from [210] that we will use. Informally, reductions in this context start from a CSP instance over an alphabet whose size matches the desired base of the lower bound, as in SETH-based reductions [208]. However, rather than reducing from a CSP instance parameterized by the number of variables we reduce from an instance parameterized by the pathwidth of its primal graph. To facilitate the reduction, we use the fact that it is hard (under pw-SETH) to distinguish between

[2]The *primal* (or *Gaifman*) graph of a formula contains a vertex for each variable and an edge when two variables appear in a common constraint.

CSP instances which are satisfiable and instances which are unsatisfiable even if for the majority of variables we are allowed to select multiple assignments, albeit while only modifying values in a monotone way along the given path decomposition.

Definition 2.2 ([210, Definition 3.2])**.** *Suppose we have a CSP instance ψ over an alphabet $[B]$ of size $B \geq 2$, with variable set V, a path decomposition of its primal graph $B_1, \ldots, B_t$, and an injective function b mapping each constraint to the index of a bag that contains all its variables. A* multi-assignment *is a function σ that takes as input a variable $x \in V$ and an index $j \in [t]$ such that $x \in B_j$ and returns a value in $[B]$. We will say that:*

1. *A multi-assignment σ is* satisfying *for ψ if for each constraint c, the assignment $\sigma_c(x) = \sigma(x, b(c))$, that is, the restriction of the multi-assignment to $b(c)$, satisfies the constraint.*

2. *A multi-assignment σ is* monotone *if for all $x \in V$ and $j_1 < j_2$ with $x \in B_{j_1} \cap B_{j_2}$ we have $\sigma(x, j_1) \leq \sigma(x, j_2)$.*

3. *A multi-assignment σ is* consistent *for $x \in V$ if for all $j_1, j_2 \in [t]$ such that $x \in B_{j_1} \cap B_{j_2}$ we have $\sigma(x, j_1) = \sigma(x, j_2)$.*

Corollary 2.3 ([210, Corollary 3.1])**.** *For all $\varepsilon > 0, B \geq 2$ we have the following. Suppose there is an algorithm with the following properties:*

- *it takes as input a 4-CSP instance ψ over an alphabet of size B, a partition of its variables into two sets V_1, V_2, a path decomposition of its primal graph of width p where each bag contains at most $\mathcal{O}(B \log p)$ variables of V_2, and an injective function b mapping each constraint to a bag that contains its variables;*

- *it decides if there exists a monotone satisfying multi-assignment σ, which is consistent for the variables of V_2;*

- *it runs in time $\mathcal{O}((B - \varepsilon)^p |\psi|^{\mathcal{O}(1)})$.*

Then the pw-SETH is false.

PIH. The last hypothesis we introduce is the *Parameterized Inapproximability Hypothesis* (PIH), which we will use in Chapter 5 to prove hardness of approximation results. We first need to formally define the following problem.

Multicolored Densest k-Subgraph

Instance:	A graph $G = (V, E)$ and a partition of V into k independent sets $V_1, \ldots, V_k$, each of size n.
Goal:	Determine the maximum number of edges that are induced by a subgraph of G that contains exactly one vertex per independent set.

Let $E^{i,j} \subseteq E$ denote the set of edges $e = \{u, v\}$ where $u \in V_i$ and $v \in V_j$. In that case, let $s_i = |\{j \in [k] : E^{i,j} \neq \varnothing\}|$ and $\sigma = \sum_{i \in [k]} s_i/2$. Assuming that OPT denotes said maximum value, the *Parameterized Inapproximability Hypothesis* (PIH) [228] states that there exists a constant $0 < c < 1$ such that no $f(k)n^{\mathcal{O}(1)}$ algorithm can distinguish between $\text{OPT} = \sigma$ and $\text{OPT} < c \cdot \sigma$. In fact, one can assume without loss of generality that $1 \leq s_i \leq 3$ for all $i = 1, \ldots, k$ [228, Lemma 4.4]. This hypothesis was very recently proved to hold under the ETH [158] (see also [15, 157]), and is the analogue of the PCP theorem in the setting of parameterized complexity.

3. Problems with Target Degree

Publication note. An extended abstract of the results presented in this chapter appeared in the proceedings of the ESA 2023 [214], while the full version has been published in the ACM Transactions of Computation Theory [215].

Chapter summary. This chapter investigates Bounded Degree Vertex Deletion and Defective Coloring, problems that generalize the classical Vertex Cover and Coloring problems respectively, through the lens of structural parameterization. In particular, we study the parameterizations by treewidth, tree-depth, and vertex cover number of the input graph. We present tight lower bounds for both problems for all considered parameterizations, introducing new techniques and insights along the way.

Parameterized complexity and in particular the study of structural parameters such as treewidth is one of the most well-developed approaches for dealing with NP-hard problems on graphs. Treewidth is of course one of the major success stories of this field, as a plethora of hard problems become fixed-parameter tractable when parameterized by this parameter. Naturally, this success has motivated the effort to trace the limits of the algorithmic power of treewidth by attempting to understand what are the problems for which treewidth-based techniques *cannot* work.

When could the treewidth toolbox fail? One common scenario that seems to be shared by a multitude of problems which are W[1]-hard parameterized by treewidth is when a natural dynamic programming algorithm does exist, but the DP is forced to store for each vertex of a bag in the tree decomposition an arbitrarily large integer – for example a number related to the degree of the vertex. Our goal in this paper is to study situations of this type and pose the natural question of whether one can do better than the "obvious" DP, by obtaining an algorithm with better running time, even at the expense of looking at a parameter more restrictive than treewidth.

Given the above, we focus on two problems which are arguably among the most natural representatives of our scenario: Bounded Degree Vertex Deletion and Defective Coloring. In both problems the input is a graph G and a target degree Δ and we are asked, in the case of Bounded Degree Vertex Deletion to delete a minimum number of vertices so that the remaining graph has degree at most Δ, and in the case of Defective Coloring to partition G into a minimum number of color classes such that each class induces a graph of degree at most Δ. Both problems are well-studied, as they generalize classical problems (Vertex Cover and Coloring respectively) and we review some of the previous work below. However, the most relevant aspect of the two problems for our purposes is the following: (i) both problems admit DP algorithms with complexity of the form $n^{\mathcal{O}(\mathrm{tw})}$ and (ii) both problems are W[1]-hard parameterized by treewidth; in fact, for Defective Coloring, it is even known that assuming the ETH

it cannot be solved in time $n^{o(\mathrm{tw})}$ [25].

Since the $n^{\mathcal{O}(\mathrm{tw})}$ algorithms follow from standard DP techniques, it becomes a natural question whether we can do better. Does a better algorithm exist? Realistically, one could hope for one of two things: either an algorithm which still handles the problem parameterized by treewidth and in view of the aforementioned lower bound only attempts a fine-grained improvement in the running time; or an algorithm which is qualitatively faster at the expense of using a more restricted parameter. Our results give strong negative evidence for both questions: if we parameterize by treewidth (and even by pathwidth) the running time of the standard DP is optimal under the SETH even for all fixed values of the other relevant parameters (Δ and the number of colors χ_d); while if we parameterize by more restrictive parameters, such as tree-depth and vertex cover, we obtain lower bound results (under the ETH) which indicate that the best algorithm is still essentially to run a form of the standard treewidth DP, even in these much more restricted cases. Our results thus paint a complete picture of the structurally parameterized complexity of these two problems and indicate that the standard DP is optimal in a multitude of restricted cases.

Our contribution in more detail. Following standard techniques, the two problems admit DP algorithms with tables of sizes $(\Delta+2)^{\mathrm{tw}}$ and $(\chi_\mathrm{d}(\Delta+1))^{\mathrm{tw}}$ respectively. Our first result is a collection of reductions proving that, assuming the SETH, no algorithm can improve upon these dynamic programs, even for pathwidth. More precisely, we show that no algorithm can solve Bounded Degree Vertex Deletion and Defective Coloring in time $(\Delta+2-\varepsilon)^{\mathrm{pw}}n^{\mathcal{O}(1)}$ and $(\chi_\mathrm{d}(\Delta+1)-\varepsilon)^{\mathrm{pw}}n^{\mathcal{O}(1)}$ respectively, for any $\varepsilon>0$ and for any combination of fixed values of Δ, χ_d (except the combination $\Delta=0$ and $\chi_\mathrm{d}=2$, which trivially makes Defective Coloring polynomial-time solvable). Our reductions follow the general strategy pioneered by Lokshtanov, Marx, and Saurabh [224] and indeed generalize their results for Vertex Cover and Coloring (which already covered the case $\Delta=0$). The main difficulty here is being able to cover all values of the secondary parameters and for technical reasons we are forced to give separate versions of our reductions to cover the case $\Delta=1$ for both problems. Along the way we note that, even though an algorithm with complexity $(\chi_\mathrm{d}\Delta)^{\mathcal{O}(\mathrm{tw})}n^{\mathcal{O}(1)}$ was given for Defective Coloring in [25], it was not known if an algorithm with complexity $(\chi_\mathrm{d}(\Delta+1))^{\mathrm{tw}}n^{\mathcal{O}(1)}$ (that is, with a quasi-linear dependence on the table size) is possible. For completeness, we settle this by providing an algorithm of this running time, using the FFT technique proposed by Cygan and Pilipczuk [95]. Taking also into account the Bounded Degree Vertex Deletion algorithm of running time $(\Delta+2)^{\mathrm{tw}}n^{\mathcal{O}(1)}$ given by van Rooij [280], we have exactly matching upper and lower bounds for both problems, for both treewidth and pathwidth.

Given that the results above show rather conclusively that the standard DP is the best algorithm for parameters treewidth and pathwidth, we then move on to a more restricted case: tree-depth. We recall that graphs of tree-depth k are a proper subclass of graphs of pathwidth k, therefore one could reasonably hope to obtain a better algorithm for this parameter. This hope may further be supported by the fact that known lower bounds do

not match the complexity of the standard algorithm. More precisely, the W[1]-hardness reduction given for Bounded Degree Vertex Deletion parameterized by tree-depth by Ganian, Klute, and Ordyniak [145] has a quartic blow-up, thus only implying that no $n^{o(\sqrt[4]{\mathrm{td}})}$ algorithm is possible; while the reduction given for Defective Coloring in [25] has a quadratic blow-up, only implying that no $n^{o(\sqrt{\mathrm{td}})}$ algorithm is possible (in both cases under the ETH). Our contribution is to show that both reductions can be replaced by more efficient reductions which are *linear* in the parameter; we thus establish that neither problem can be solved in time $n^{o(\mathrm{td})}$, implying that the treewidth-based algorithm remains (qualitatively) optimal even in this restricted case. One interesting aspect of our reductions is that, rather than using a modulator to a low tree-depth graph, which is common in such reductions, we use a recursive construction that leverages the full power of the parameter and may be of further use in tightening other lower bounds for the parameter tree-depth.

Finally, we move on to a more special case, parameterizing both problems by vertex cover. Both problems are FPT for this parameter and, since vertex cover is very restrictive as a parameter, one would hope that, finally, we should be able to obtain an algorithm that is more clever than the treewidth-based DP. Somewhat disappointingly, the known FPT algorithms for both problems have complexity $\mathrm{vc}^{\mathcal{O}(\mathrm{vc})}n^{\mathcal{O}(1)}$ [25], and the super-exponential dependence on the parameter is due to the fact that both algorithms are simple win/win arguments which, in one case, just execute the standard treewidth DP. We show that this is justified, as neither problem can be solved in time $\mathrm{vc}^{o(\mathrm{vc})}n^{\mathcal{O}(1)}$ (under the ETH), meaning that the algorithm that blindly executes the treewidth-based DP in some cases is still (qualitatively) best possible. We obtain our result by applying the technique of d-detecting families, introduced by Bonamy, Kowalik, Pilipczuk, Socala, and Wrochna [49]. Our results indicate that parameterization by vertex cover is a domain where this promising, but currently under-used, technique may find more applications in parameterized complexity.

Related work. Both Bounded Degree Vertex Deletion and Defective Coloring are well-studied problems with a rich literature. Bounded Degree Vertex Deletion finds application in a multitude of areas, ranging from computational biology [117] to some related problems in voting theory [32, 34], and its dual problem, called s-Plex Detection, has numerous applications in social network analysis [16, 240, 244]. Various approximation algorithms are known [137, 138, 253]. The problem has also been extensively studied under the scope of parameterized complexity. It is W[2]-hard for unbounded values of Δ and parameter k (the value of the optimal) [117], while it admits a linear-size kernel parameterized by k [117, 289], for any fixed $\Delta \geq 0$; numerous FPT algorithms have been presented in the latter setting [244, 250, 288]. FPT approximation algorithms were given for Bounded Degree Vertex Deletion in [207] and [227]. As for Defective Coloring, which also appears in the literature as Improper Coloring, it was introduced almost 40 years ago [8, 88]. The main motivation behind this problem comes from the field of telecommunications, where the colors correspond to available frequencies and the goal is to assign them to communication nodes; a small amount

of interference between neighboring nodes may be tolerable, which is modeled by the parameter Δ. There have been plenty of works on the problem (see [9, 10, 25, 26, 73, 165] and the references therein), especially on unit disk graphs and various classes of grids.

The previous work for both problems that is most relevant to us focuses on their parameterized complexity for structural parameters, such as treewidth. In this setting, Bounded Degree Vertex Deletion was one of the first problems to be discovered to be W[1]-hard parameterized by treewidth [33], though the problem does become FPT parameterized by $\mathrm{tw} + \Delta$ or $\mathrm{tw} + k$. This hardness result was more recently improved by Ganian, Klute, and Ordyniak [145], who showed that Bounded Degree Vertex Deletion is W[1]-hard parameterized by tree-depth and feedback vertex set. Defective Coloring was shown to be W[1]-hard parameterized by tree-depth (and hence pathwidth and treewidth) in [25]. However, [25] gave a hardness reduction for pathwidth that is linear in the parameter, and hence implies a $n^{o(\mathrm{pw})}$ lower bound for Defective Coloring under the ETH, but a hardness reduction for tree-depth that is quadratic (implying only a $n^{o(\sqrt{\mathrm{td}})}$ lower bound). Similarly, the reduction given by [145] for Bounded Degree Vertex Deletion parameterized by tree-depth is quartic in the parameter, as it goes through an intermediate problem (a variant of Subset Sum), implying only a $n^{o(\sqrt[4]{\mathrm{td}})}$ lower bound. Defective Coloring is known to be FPT parameterized by vertex cover using a simple win/win argument which applies the treewidth-based DP in one case (if $\Delta > \mathrm{vc}$, then the graph is always 2-colorable; otherwise the standard DP algorithm runs in FPT time), and the same is true for Bounded Degree Vertex Deletion (if $\Delta \leq \mathrm{vc}$, we can use the aforementioned FPT algorithm for parameters $\mathrm{tw}+\Delta$, else assume that $k < \mathrm{vc}$, as otherwise the problem is trivial, follow the reduction of [33] to Vector Dominating Set and notice that at most vc vertices have degree greater than Δ, thus Vector Dominating Set can be decided in time $\mathrm{vc}^{\mathcal{O}(\mathrm{vc})}n^{\mathcal{O}(1)}$ due to [258]). Hence, the best algorithms for both problems for this parameter have complexity $\mathrm{vc}^{\mathcal{O}(\mathrm{vc})}n^{\mathcal{O}(1)}$.

The fine-grained analysis of the complexity of structural parameterizations, such as by treewidth, is an active field of research. The technique of using the SETH to establish tight running time lower bounds was pioneered by Lokshtanov, Marx, and Saurabh [224]. Since then, tight upper and lower bounds are known for a multitude of problems for parameterizations by treewidth and related parameters, such as pathwidth and clique-width [93, 94, 105, 106, 126, 144, 156, 163, 175, 252, 279]. One difficulty of the results we present here is that we need to present a family of reductions: one for each fixed value of Δ and χ_{d}. There are a few other problems for which families of tight lower bounds are known, such as k-Coloring, for which the correct dependence is k^{tw} for treewidth [224] and $(2^k - 2)^{\mathrm{cw}}$ for clique-width [208] for all $k \geq 3$; distance r-Dominating Set, for which the correct dependence is $(2r+1)^{\mathrm{tw}}$ [50] and $(3r+1)^{\mathrm{cw}}$ [186], for all $r \geq 1$; and distance d-Independent Set, for which the correct dependence is d^{tw} [187]. In all these cases, the optimal algorithm is the "natural" DP, and our results for Bounded Degree Vertex Deletion and Defective Coloring fit this pattern.

Even though the previous work mentioned above may make it seem that our SETH-based lower bounds are not surprising, it is important to stress that it is not a given that the naïve DP should be optimal for our problems. In particular, Bounded De-

Table 3.1.: Lower bounds established in the current work. The results of the first row are under SETH, while all the rest under ETH.

Parameter	Bounded Degree Vertex Deletion	Defective Coloring
pathwidth + Δ	$\mathcal{O}^\star((\Delta+2-\varepsilon)^{\mathrm{pw}})$	$\mathcal{O}^\star((\chi_\mathrm{d}\cdot(\Delta+1)-\varepsilon)^{\mathrm{pw}})$
tree-depth	$n^{o(\mathrm{td})}$	$n^{o(\mathrm{td})}$
vertex cover	$\mathrm{vc}^{o(\mathrm{vc})}n^{\mathcal{O}(1)}$	$\mathrm{vc}^{o(\mathrm{vc})}n^{\mathcal{O}(1)}$

gree Vertex Deletion falls into a general category of (σ,ρ)-domination problems, which were studied recently in [124, 125] (we refer the reader there for the definition of (σ,ρ)-domination). One of the main results of that work was to show that significant improvements over the basic DP are indeed possible in some cases, and in particular when one of σ,ρ is cofinite. Since Bounded Degree Vertex Deletion is the case where $\sigma=\{0,\ldots,\Delta\}$ and $\rho=\mathbb{N}$ (that is, ρ is co-finite), our result falls exactly in the territory left uncharted by [124, 125], where more efficient algorithms could still be found (and where indeed [124, 125] did uncover such algorithms for some values of σ,ρ).

3.1. Preliminaries

For a graph $G=(V,E)$ and two integers $\chi_\mathrm{d}\geq 1,\Delta\geq 0$, we say that G admits a (χ_d,Δ)-coloring if one can partition V into χ_d sets such that the graph induced by each set has maximum degree at most Δ.

Defective Coloring

Instance: A graph G as well as integers $\chi_\mathrm{d}\geq 2$ and $\Delta\geq 0$.
Goal: Determine whether G admits a (χ_d,Δ)-coloring.

Bounded Degree Vertex Deletion

Instance: A graph G as well as integers $k\geq 0$ and $\Delta\geq 0$.
Goal: Determine whether there exists $S\subseteq V(G)$ of size $|S|\leq k$ such that $G-S$ has maximum degree at most Δ.

3.2. Treewidth and Maximum Degree

In this section we present tight lower bounds on the complexity of solving both Bounded Degree Vertex Deletion and Defective Coloring parameterized by the treewidth of the input graph plus the target degree. In the case of Bounded Degree Vertex Deletion, this lower bound matches the algorithm of [124, 125], while for Defective Coloring we develop an algorithm of matching running time in Section 3.2.2.

Both reductions are similar in nature: we start from an instance ϕ of q-CSP-B, and produce an equivalent instance on a graph of pathwidth $\mathrm{pw}=n+\mathcal{O}(1)$, where n denotes

the number of variables of ϕ. An interesting observation however, is that for both problems we have to distinguish between the case where $\Delta = 1$ and $\Delta \geq 2$; the whole construction becomes much more complicated in the latter case.

3.2.1. Bounded Degree Vertex Deletion

In the following, we will present a reduction from q-CSP-B to Bounded Degree Vertex Deletion, for any fixed $\Delta \geq 1$, where $\Delta = B - 2$. In that case, if there exists a $\mathcal{O}^\star((\Delta + 2 - \varepsilon)^{\mathrm{pw}})$ algorithm for Bounded Degree Vertex Deletion, where $\varepsilon > 0$, then there exists a $\mathcal{O}^\star((B - \varepsilon)^n)$ algorithm for q-CSP-B, for any constant q, which due to Theorem 2.1 results in SETH failing.

Our reduction is based on the construction of "long paths" of *Block gadgets*, that are serially connected in a path-like manner. Each such "path" corresponds to a variable of the given CSP, while each column of the construction is associated with one of its constraints. Intuitively, our aim is to embed the B^n possible variable assignments into the $(\Delta + 2)^{\mathrm{tw}}$ states of some optimal dynamic program that would solve the problem on our constructed instance.

Below, we present a sequence of gadgets used in our reduction. The aforementioned block gadgets, which allow a solution to choose among $\Delta + 2$ reasonable choices, are the main ingredient. Notice that these gadgets will differ significantly depending on whether Δ is equal to 1 or not. We connect these gadgets in a path-like manner that ensures that choices remain consistent throughout the construction, and connect constraint gadgets in different "columns" of the constructed grid in a way that allows us to verify if the choice made represents a satisfying assignment, without significantly increasing the graph's pathwidth.

Theorem 3.1. *For any constant $\varepsilon > 0$, there is no $\mathcal{O}^\star((3 - \varepsilon)^{\mathrm{pw}})$ algorithm deciding Bounded Degree Vertex Deletion for $\Delta = 1$, where* pw *denotes the input graph's pathwidth, unless the SETH is false.*

Proof. Fix some positive $\varepsilon > 0$ for which we want to prove the theorem. We will reduce q-CSP-3, for some q that is a constant that only depends on ε, to Bounded Degree Vertex Deletion for $\Delta = 1$ in a way that ensures that if the resulting Bounded Degree Vertex Deletion instance could be solved in time $\mathcal{O}^\star((3-\varepsilon)^{\mathrm{pw}})$, then we would obtain an algorithm for q-CSP-3 that would contradict the SETH due to Theorem 2.1. To this end, let ϕ be an instance of q-CSP-3 of n variables $X = \{x_i : i \in [n]\}$ taking values over the set $Y = [3]$ and m constraints $C = \{c_j : j \in [m]\}$. For each constraint we are given a set of at most q variables which are involved in this constraint and a list of satisfying assignments for these variables, the size of which is denoted by $s \colon C \to [3^q]$, i.e. $s(c_j) \leq 3^q = \mathcal{O}(1)$ denotes the number of satisfying assignments for constraint c_j. We will construct in polynomial time an equivalent instance $\mathcal{I} = (G, k)$ of Bounded Degree Vertex Deletion for $\Delta = 1$, where $\mathrm{pw}(G) \leq n + \mathcal{O}(1)$.

Block and Variable Gadgets. For every variable x_i and every constraint c_j, construct a path of 3 vertices $p^1_{i,j}, p^2_{i,j}$ and $p^3_{i,j}$, which comprises the *block gadget* $\hat{B}_{i,j}$. Intuitively,

we will map the deletion of $p^y_{i,j}$ with an assignment where x_i receives value y. Next, for $j \in [m-1]$, we add an edge between $p^3_{i,j}$ and $p^1_{i,j+1}$, thus resulting in n paths $P_1, \ldots, P_n$ of length $3m$, called *variable gadgets*.

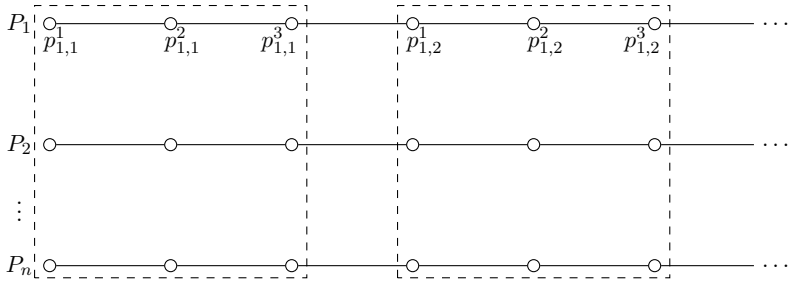


Figure 3.1.: Sequences of block gadgets comprise the variable gadgets.

Constraint Gadget. This gadget is responsible for determining constraint satisfaction, based on the choices made in the rest of the graph. For constraint c_j, construct the *constraint gadget* $\hat{C}_j$ as follows:

- construct a clique of $s(c_j)$ vertices $v^j_1, \ldots, v^j_{s(c_j)}$, and fix an arbitrary one-to-one mapping between those vertices and the satisfying assignments of c_j,
- attach to each vertex v^j_ℓ a leaf l^j_ℓ,
- if variable x_i is involved in the constraint c_j and v^j_ℓ corresponds to an assignment where x_i has value $y \in Y$, add an edge between v^j_ℓ and $p^y_{i,j}$.

Let graph G_0 be the graph containing all variable gadgets P_i as well as all the constraint gadgets $\hat{C}_j$, for $i \in [n]$ and $j \in [m]$. To construct graph G, introduce $\kappa = 2n+1$ copies $G_1, \ldots, G_\kappa$ of G_0, such that they are connected sequentially as follows: for $i \in [n]$ and $j \in [\kappa - 1]$, add an edge between $p^3_{i,m}(G_j)$ and $p^1_{i,1}(G_{j+1})$, where $p^y_{i,j}(G_z)$ denotes the vertex $p^y_{i,j}$ of graph G_z. We refer to the block gadget $\hat{B}_{i,j}$, to the variable gadget P_i, and to the constraint gadget $\hat{C}_j$ of G_z as $\hat{B}^{G_z}_{i,j}$, $P^{G_z}_i$, and $\hat{C}^{G_z}_j$ respectively. Let $\mathcal{P}^i$ denote the path resulting from $P^{G_1}_i, \ldots, P^{G_\kappa}_i$. Set $k = \kappa \cdot k'$, where $k' = \sum_{j=1}^{m}(s(c_j)-1+n) = m \cdot n + \sum_{j=1}^{m}(s(c_j)-1)$, and let $\mathcal{I} = (G, k)$ denote the instance of Bounded Degree Vertex Deletion for $\Delta = 1$.

Lemma 3.2. *If ϕ is satisfiable, then there exists $S \subseteq V(G)$ such that $G - S$ has maximum degree at most 1 and $|S| \leq k$.*

Proof. Let $f\colon X \to Y$ denote an assignment which satisfies all the constraints $c_1, \ldots, c_m$. In that case, let $p^y_{i,j}(G_z) \in S$, where $j \in [m]$ and $z \in [\kappa]$, if $f(x_i) = y$. In other words, from every path $\mathcal{P}^i$, include in S either the first, the second, or the third vertex out of every block gadget $\hat{B}_{i,j}$, depending on the value of $f(x_i)$. Moreover, since f is a satisfying assignment, for every constraint c_j there exists a satisfying assignment which

is a restriction of f to the at most q involved variables of c_j and corresponds to a vertex $v^j_\ell(G_z)$ of $\hat{C}^{G_z}_j$. Let $v^j_{\ell'}(G_z) \in S$ if $\ell' \neq \ell$. It holds that $|S| = \kappa \cdot (mn + \sum_{i=1}^{m}(s(c_j)-1)) = k$.

It remains to prove that $G - S$ has maximum degree at most 1. Consider the constraint gadget $\hat{C}^{G_z}_j$. Any leaf l^j_i has degree either 0 or 1 in $G - S$. Let $v^j_\ell(G_z) \notin S$ denote the vertex of the clique of the constraint gadget which does not belong to S. It holds that $\deg_{G-S}(v^j_\ell(G_z)) = 1$, since

- $l^j_\ell(G_z) \notin S$,
- $v^j_{\ell'}(G_z) \in S$, for $\ell' \neq \ell$,
- $v^j_\ell(G_z)$ has an edge with $p^y_{i,j}(G_z)$, where $y \in Y$, only if x_i is involved in c_j and $v^j_\ell(G_z)$ corresponds to an assignment g where variable x_i has value y. However, since this assignment is a restriction of f, it follows that $g(x_i) = f(x_i) = y$, thus $p^y_{i,j}(G_z) \in S$.

Consequently, there are no edges between vertices of the constraint and the block gadgets in $G - S$. Notice that from every three consecutive vertices in $\mathcal{P}^i$, one belongs to S. Therefore, it holds that if $p^y_{i,j}(G_z) \notin S$, at most one of its neighbors does not belong to S, thus its degree is at most 1 in $G - S$. □

Lemma 3.3. *If there exists $S \subseteq V(G)$ such that $G - S$ has maximum degree at most 1 and $|S| \leq k$, then ϕ is satisfiable.*

Proof. Let $S \subseteq V(G)$ such that $G - S$ has maximum degree at most 1 and $|S| \leq k$. First we will prove that S contains a single vertex $p^y_{i,j}$ from each block gadget $\hat{B}_{i,j}$, as well as vertices v^j_l, where $l \in [s(c_j)] \setminus \{\ell\}$ for some $\ell \in [s(c_j)]$ from each constraint gadget $\hat{C}_j$.

Notice that S contains at least 1 vertex from every block gadget $\hat{B}_{i,j}$, since otherwise the vertex $p^2_{i,j}$ has degree at least 2 in $G - S$. Additionally, S contains at least $s(c_j) - 1$ vertices from every constraint gadget $\hat{C}_j$, since each such gadget has a clique of size $s(c_j)$, every vertex of which has a leaf attached. Therefore, it holds that $|S \cap V(G_z)| \geq k' = m \cdot n + \sum_{j=1}^{m}(s(c_j) - 1)$, for all $z \in [\kappa]$. Since $|S| \leq \kappa \cdot k'$, it follows that $|S \cap V(G_z)| = k'$. Consequently, S contains exactly one vertex per block gadget and exactly $s(c_j) - 1$ vertices per constraint gadget $\hat{C}_j$, which can only be vertices of the clique.

Notice that then it holds that $|S \cap V(G_z)| = k'$, for all $z \in [\kappa]$. We will say that an *inconsistency* occurs in a variable gadget $P^{G_z}_i$ if there exist two consecutive block gadgets $\hat{B}^{G_z}_{i,j}, \hat{B}^{G_z}_{i,j+1}$ such that $p^y_{i,j}, p^{y'}_{i,j+1} \in S$, for $y \neq y'$. Moreover, we will say that G_z is *consistent* if no inconsistency occurs in any of the variable gadgets $P^{G_z}_i$, for $i \in [n]$.

▷ Claim 3.4. There exists $\pi \in [\kappa]$ such that G_π is consistent.

Proof of the claim. Notice that S contains $\kappa \cdot m$ vertices from every path $\mathcal{P}^i$, since the latter is comprised of that many block gadgets. We will prove that every path $\mathcal{P}^i$ may induce at most 2 inconsistencies. In that case, since there are n such paths and $\kappa = 2n + 1$ copies of G_0, due to the pigeonhole principle there exists some G_π with no inconsistencies.

Consider a path $\mathcal{P}^i$ as well as a block gadget $\hat{B}^{G_z}_{i,j}$, for some $z \in [\kappa]$ and $j \in [m]$. Let $N(\hat{B}^{G_z}_{i,j})$ denote the block gadget right of $\hat{B}^{G_z}_{i,j}$, consisting of vertices $n_1(\hat{B}^{G_z}_{i,j})$, $n_2(\hat{B}^{G_z}_{i,j})$ and $n_3(\hat{B}^{G_z}_{i,j})$, where $p^3_{i,j}$ and $n_1(\hat{B}^{G_z}_{i,j})$ are connected via an edge in G. Moreover, let $\hat{B}^{G_{z'}}_{i,j'}$, where either a) $z' = z$ and $j' > j$ or b) $z' > z$ and $j' \in [m]$, denote some block gadget which appears to the right of $\hat{B}^{G_z}_{i,j}$. In that case:

- If $p^1_{i,j}(G_z) \in S$, then $p^1_{i,j'}(G_{z'}) \in S$. It holds for $N(\hat{B}^{G_z}_{i,j})$, since $p^3_{i,j}(G_z)$ has degree 2 in $G - S$ otherwise. Suppose that it holds for all block gadgets after $\hat{B}^{G_z}_{i,j}$ and up to some block gadget $\hat{B}^{G_{z^*}}_{i,j^*}$. Then it holds for $N(\hat{B}^{G_{z^*}}_{i,j^*})$ as well, since $p^3_{i,j^*}(G_{z^*})$ has degree 2 in $G - S$ otherwise.
- If $p^2_{i,j}(G_z) \in S$, then, either $p^1_{i,j'}(G_{z'}) \in S$ or $p^2_{i,j'}(G_{z'}) \in S$. It holds for $N(\hat{B}^{G_z}_{i,j})$, since $n_3(\hat{B}^{G_z}_{i,j}) \in S$ otherwise and it follows that $n_1(\hat{B}^{G_z}_{i,j})$ has degree 2 in $G - S$. Suppose that it holds for all block gadgets after $\hat{B}^{G_z}_{i,j}$ and up to some block gadget $\hat{B}^{G_{z^*}}_{i,j^*}$. Then it holds for $N(\hat{B}^{G_{z^*}}_{i,j^*})$ as well, since $n_3(\hat{B}^{G_{z^*}}_{i,j^*}) \in S$ otherwise and it follows that $n_1(\hat{B}^{G_{z^*}}_{i,j^*})$ has degree 2 in $G - S$.
- Lastly, if $p^3_{i,j}(G_z) \in S$, then S contains a single vertex from every subsequent block gadget $\hat{B}^{G_{z'}}_{i,j'}$, since it contains exactly one vertex per block gadget.

Thus, it follows that every path can induce at most 2 inconsistencies, and since there is a total of n paths, there exists a copy G_π which is consistent. ◁

Now, consider the assignment $f\colon X \to Y$, where $f(x_i) = y$ if $p^y_{i,j}(G_\pi) \in S$. This is a valid assignment, since G_π is consistent and a single vertex is contained in S from each block gadget. We will prove that it satisfies all the constraints. Consider a constraint c_j. Regarding the constraint gadget $\hat{C}^{G_\pi}_j$, it holds that S includes exactly $s(c_j) - 1$ vertices, none of which is a leaf vertex. Let $v^j_\ell(G_\pi) \notin S$ be the only vertex which is part of the clique and does not belong to S. Since $l^j_\ell(G_\pi) \notin S$, it follows that every neighbor of $v^j_\ell(G_\pi)$ in the block gadgets $\hat{B}^{G_\pi}_{i,j}$ belongs to S. In that case, the satisfying assignment corresponding to $v^j_\ell(G_\pi)$ is a restriction of f, thus f satisfies c_j. Since this holds for any j, ϕ is satisfied. □

Lemma 3.5. *It holds that* $\mathrm{pw}(G) \leq n + \mathcal{O}(1)$.

Proof. We will prove the statement by providing a mixed search strategy to clean G using at most this many searchers simultaneously. Since for the mixed search number ms it holds that $\mathrm{pw}(G) \leq \mathrm{ms}(G) \leq \mathrm{pw}(G) + 1$ [273], we will show that $\mathrm{ms}(G) \leq n + 2 \cdot 3^q$ and the statement will follow.

Start with graph G_1. Place $2s(c_1)$ searchers to all the vertices of $\hat{C}^{G_1}_1$, as well as n searchers on vertices $p^1_{i,1}(G_1)$, for $i \in [n]$. In this way, all the edges of the constraint gadget are cleared. Next we will describe the procedure to clear $\hat{B}^{G_1}_{i,1}$. Move the searcher along the edge connecting $p^1_{i,1}(G_1)$ to $p^2_{i,1}(G_1)$ and then along the edge connecting $p^2_{i,1}(G_1)$

and $p^3_{i,1}(G_1)$. Repeat the whole process for all i, thus clearing all block gadgets $\hat{B}^{G_1}_{i,1}$ as well as the edges between those block gadgets and $\hat{C}^{G_1}_1$.

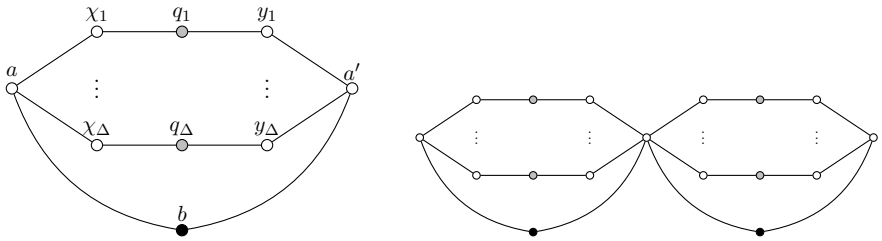
In order to clear the rest of the graph, we first move the searchers from $\hat{C}^{G_z}_j$ to $\hat{C}^{G_z}_{j+1}$ if $j < m$ or to $\hat{C}^{G_{z+1}}_1$ alternatively (possibly introducing new searchers if required), and then proceed by clearing the corresponding block gadgets, by first sliding all searchers from $p^3_{i,j}(G_z)$ to $p^1_{i,j+1}(G_z)$ if $j < m$ or to $p^1_{i,1}(G_{z+1})$ alternatively. By repeating this procedure, in the end we clear all the edges of G by using at most $n + 2 \cdot 3^q = n + \mathcal{O}(1)$ searchers. □

Therefore, in polynomial time we can construct a graph G, of pathwidth $\mathrm{pw}(G) \leq n + \mathcal{O}(1)$ due to Lemma 3.5, such that, due to Lemmas 3.2 and 3.3, deciding whether there exists $S \subseteq V(G)$ of size $|S| \leq k$ and $G - S$ has maximum degree at most 1 is equivalent to deciding whether ϕ is satisfiable. In that case, assuming there exists a $\mathcal{O}^\star((3-\varepsilon)^{\mathrm{pw}(G)})$ algorithm for Bounded Degree Vertex Deletion for $\Delta = 1$, one could decide q-CSP-3 in time $\mathcal{O}^\star((3-\varepsilon)^{\mathrm{pw}(G)}) = \mathcal{O}^\star((3-\varepsilon)^{n+\mathcal{O}(1)}) = \mathcal{O}^\star((3-\varepsilon)^n)$ for any constant q, which contradicts the SETH due to Theorem 2.1. □

Theorem 3.6. *For any constant $\varepsilon > 0$, there is no $\mathcal{O}^\star((\Delta+2-\varepsilon)^{\mathrm{pw}})$ algorithm deciding Bounded Degree Vertex Deletion for $\Delta \geq 2$, where* pw *denotes the input graph's pathwidth, unless the SETH is false.*

Proof. Fix some positive $\varepsilon > 0$ for which we want to prove the theorem. Let $B \geq 4$. We will reduce q-CSP-B, for some q that is a constant that only depends on ε, to Bounded Degree Vertex Deletion for $\Delta = B - 2$ in a way that ensures that if the resulting Bounded Degree Vertex Deletion instance could be solved in time $\mathcal{O}^\star((\Delta+2-\varepsilon)^{\mathrm{pw}})$, then we would obtain an algorithm for q-CSP-B that would contradict the SETH due to Theorem 2.1. To this end, let ϕ be an instance of q-CSP-B of n variables $X = \{x_i : i \in [n]\}$ taking values over the set $Y = [0, \Delta + 1]$ and m constraints $C = \{c_j : j \in [m]\}$, where $\Delta = B - 2$. For each constraint we are given a set of at most q variables which are involved in this constraint and a list of satisfying assignments for these variables, the size of which is denoted by $s\colon C \to [B^q]$, i.e. $s(c_j) \leq B^q = \mathcal{O}(1)$ denotes the number of satisfying assignments for constraint c_j. We will construct in polynomial time an equivalent instance $\mathcal{I} = (G, k)$ of Bounded Degree Vertex Deletion for $\Delta = B - 2$, where $\mathrm{pw}(G) \leq n + \mathcal{O}(1)$.

Block and Variable Gadgets. For every variable x_i and every constraint c_j, construct a *block gadget* $\hat{B}_{i,j}$, as depicted in Figure 3.2a. In order to do so, we introduce vertices a, a', b, χ_i, y_i, and q_i, for $i \in [\Delta]$. Then, we attach Δ leaves on b and $\Delta - 1$ leaves on vertices q_i. Finally, we add edges $\{a, b\}, \{a', b\}, \{a, \chi_i\}, \{a', y_i\}, \{\chi_i, q_i\}$, and $\{y_i, q_i\}$. Intuitively, we map the deletion of b as well as of p vertices out of $\{\chi_i : i \in [\Delta]\}$ and $\Delta - p$ vertices out of $\{y_i : i \in [\Delta]\}$ with an assignment where x_i receives value $p \in [0, \Delta]$, while the deletion of a maps with an assignment where x_i receives value $\Delta + 1$. Next, for $j \in [m-1]$, we serially connect the block gadgets $\hat{B}_{i,j}$ and $\hat{B}_{i,j+1}$ so that the vertex a' of $\hat{B}_{i,j}$ is the vertex a of $\hat{B}_{i,j+1}$, thus resulting in n "paths" $P_1, \ldots, P_n$ consisting of m serially connected block gadgets, called *variable gadgets*. For an illustration see Figure 3.2.



(a) Block gadget when $\Delta \geq 2$. (b) Serially connected block gadgets.

Figure 3.2.: Gray vertices have $\Delta - 1$ and black vertices Δ leaves attached.

Constraint Gadget. This gadget is responsible for determining constraint satisfaction, based on the choices made in the rest of the graph. For constraint c_j, construct the *constraint gadget* $\hat{C}_j$ as follows:

- construct a clique of $s(c_j)$ vertices $v^j_1, \ldots, v^j_{s(c_j)}$, and fix an arbitrary one-to-one mapping between those vertices and the satisfying assignments of c_j,
- attach to each vertex of the clique Δ unnamed leaves,
- if variable x_i is involved in the constraint c_j and v^j_ℓ corresponds to an assignment where x_i has value $p \in Y$, consider two cases:
  (i) If $p \in [0, \Delta]$, then add an edge between v^j_ℓ and vertex b of $\hat{B}_{i,j}$. Moreover, add edges between v^j_ℓ and $\{\chi_i : i \in [p]\} \cup \{y_i : i \in [p+1, \Delta]\}$.
  (ii) On the other hand, if v^j_ℓ corresponds to an assignment where variable x_i takes value $\Delta + 1$, then add an edge between v^j_ℓ and the vertex a of $\hat{B}_{i,j}$.

Let graph G_0 correspond to the graph containing all variable gadgets P_i as well as all the constraint gadgets $\hat{C}_j$, for $i \in [n]$ and $j \in [m]$. We refer to the block gadget $\hat{B}_{i,j}$, to the variable gadget P_i, and to the constraint gadget $\hat{C}_j$ of G_z as $\hat{B}^{G_z}_{i,j}$, $P^{G_z}_i$, and $\hat{C}^{G_z}_j$ respectively. To construct graph G, introduce $\kappa = \kappa_1 \cdot \kappa_2$ copies $G_1, \ldots, G_\kappa$ of G_0, where $\kappa_1 = n + 1$ and $\kappa_2 = (2\Delta + 1)n + 1$, such that they are connected sequentially as follows: for $i \in [n]$ and $j \in [\kappa - 1]$, the vertex a' of $\hat{B}^{G_j}_{i,m}$ is the vertex a of $\hat{B}^{G_{j+1}}_{i,1}$. Let $\mathcal{P}^i$ denote the "path" resulting from $P^{G_1}_i, \ldots, P^{G_\kappa}_i$. Set $k = n + \kappa \cdot k_c$, where $k_c = mn(\Delta + 1) + \sum_{j=1}^{m}(s(c_j) - 1)$, and let $\mathcal{I} = (G, k)$ denote the instance of Bounded Degree Vertex Deletion for $\Delta = B - 2$.

Lemma 3.7. *If ϕ is satisfiable, then there exists $S \subseteq V(G)$ such that $G - S$ has maximum degree at most Δ and $|S| \leq k$.*

Proof. Let $f \colon X \to Y$ be an assignment that satisfies all constraints c_j. We will present a set $S \subseteq V(G)$ of size $|S| \leq k$ such that $G - S$ has maximum degree at most Δ. First, for every constraint gadget $\hat{C}_j$, since f is a satisfying assignment, there exists some vertex v^j_ℓ which corresponds to a restriction of f to the at most q variables involved in c_j. Add in S

every vertex v_l^j for $l \in [s(c_j)] \setminus \{\ell\}$. Consequently, S contains exactly $\kappa \cdot \sum_{j=1}^{m}(s(c_j)-1)$ such vertices. Then, for every $i \in [n]$, consider the following two cases:

- If $f(x_i) = \Delta + 1$, then, for every block gadget $\hat{B}_{i,j}$, include in S the vertices a, a', and q_i, for $i \in [\Delta]$. Since the vertex a' of a block gadget is the vertex a of its subsequent block gadget, S includes $\kappa \cdot m \cdot (\Delta+1)+1$ vertices in this case.

- If on the other hand $f(x_i) = p \in [0, \Delta]$, then, for every block gadget $\hat{B}_{i,j}$, include in S the vertices b, χ_i, and y_j, for $i \in [p]$ and $j \in [p+1, \Delta]$. In this case, S includes $\kappa \cdot m \cdot (\Delta+1)$ vertices.

Consequently, S contains at most $\kappa \cdot \sum_{j=1}^{m}(s(c_j)-1) + n \cdot (\kappa \cdot m \cdot (\Delta+1)+1) = n + \kappa \cdot k_c = k$ vertices.

It remains to prove that $G-S$ has maximum degree at most Δ. Consider any constraint gadget $\hat{C}_j^{G_z}$. Any leaf present has degree either 0 or 1 in $G-S$. Let $v_\ell^j(G_z) \notin S$ denote the only vertex of the clique of the constraint gadget which does not belong to S. This vertex corresponds to a satisfying assignment for c_j, which is a restriction of f. It holds that $\deg_{G-S}(v_\ell^j(G_z)) = \Delta$, since

- none of its Δ neighboring leaves belongs to S,

- $v_{\ell'}^j(G_z) \in S$, for $\ell' \neq \ell$,

- $v_\ell^j(G_z)$ has no neighbors in $\hat{B}_{i,j}^{G_z}$, for all $i \in [n]$: consider the two cases where either a is a neighbor of $v_\ell^j(G_z)$ or vertices b, χ_i, y_j, where $i \in [p]$ and $j \in [p+1, \Delta]$, for some $p \in [0, \Delta]$, are neighbors of $v_\ell^j(G_z)$. In both cases, since $v_\ell^j(G_z)$ corresponds to an assignment g which is a restriction of f, all of its neighbors in $\hat{B}_{i,j}^{G_z}$ belong to S.

Consequently, there are no edges between vertices of the constraint and the block gadgets in $G-S$.

Lastly, for $i \in [n]$, no vertex in a block gadget $\hat{B}_{i,j}$ of $\mathcal{P}^i$ has degree larger than Δ in $G-S$:

- if $f(x_i) = \Delta + 1$, then vertex b has degree Δ, while the vertices χ_i, y_i have degree 0.

- on the other hand, if $f(x_i) = p \neq \Delta + 1$, it holds that all vertices q_i have degree Δ, since one of their neighbors belongs to S. As for the non-deleted vertices χ_i and y_i, they have degree equal to $2 \leq \Delta$. Lastly, any vertex a has degree at most Δ, since it has $\Delta - p$ neighbors due to the block gadget it appears as a and p neighbors due to the block gadget it appears as a'.

This concludes the proof. □

Lemma 3.8. *If there exists $S \subseteq V(G)$ such that $G-S$ has maximum degree Δ and $|S| \leq k$, then ϕ is satisfiable.*

Proof. Let $S \subseteq V(G)$ such that $G - S$ has maximum degree Δ and $|S| \leq k$. For G_z, consider a mapping between subsets of vertices of $\hat{B}_{i,j}^{G_z}$ that belong to S and the value of x_i for some assignment of the variables of ϕ. In particular, S containing vertex b as well as p vertices out of $\{\chi_i : i \in [n]\}$ and $\Delta - p$ out of $\{y_i : i \in [n]\}$ is mapped with an assignment where x_i receives value $p \in [0, \Delta]$, while S containing just vertex a is mapped with an assignment where x_i receives value $\Delta + 1$. Any other subset will correspond to an *invalid* assignment of x_i. We will say that an *inconsistency* occurs in a variable gadget $P_i^{G_z}$ if there exist block gadgets $\hat{B}_{i,j}^{G_z}$ and $\hat{B}_{i,j+1}^{G_z}$, such that the vertices belonging to S from each block gadget map to either invalid or different assignments of x_i. We say that G_z is *consistent* if the deletions occurring in its block gadget $\hat{B}_{i,j}^{G_z}$ map to some non-invalid assignment for x_i, while additionally no inconsistency occurs in its variable gadgets $P_i^{G_z}$, for every $i \in [n]$.

▷ Claim 3.9. There exists $\pi \in [\kappa]$ such that G_π is consistent. Moreover, it holds that S contains vertices v_l^j, where $l \in [s(c_j)] \setminus \{\ell\}$, for some $\ell \in [s(c_j)]$, from $\hat{C}_j^{G_\pi}$.

Proof of the claim. In the following, we will assume that the vertices a' of block gadgets $\hat{B}_{i,j}^{G_z}$, which coincide with the vertices a of $\hat{B}_{i,j+1}^{G_z}$ if $j < m$ and of $\hat{B}_{i,1}^{G_{z+1}}$ alternatively, will not belong to the vertices of $\hat{B}_{i,j}^{G_z}$, so that we do not double count them (with the exception of the block gadget $\hat{B}_{i,m}^{G_\kappa}$). As a consequence, for $z \in [\kappa - 1]$, vertices a' of $\hat{B}_{i,m}^{G_z}$ will not belong to $V(G_z)$ and will belong to $V(G_{z+1})$ instead. We will first prove that $|S \cap V(G_z)| \geq k_c$, for all $z \in [\kappa]$. Notice that from each constraint gadget $\hat{C}_j$, at least $s(c_j) - 1$ vertices belong to S: each such gadget has a clique of size $s(c_j)$, every vertex of which has Δ leaves attached. Furthermore, for each block gadget $\hat{B}_{i,j}$, since b as well as vertices q_i have degree larger than Δ, while no two share any neighbors, $\Delta + 1$ deletions are required. Notice that, since b has degree $\Delta + 2$, even if we delete the vertex a', in the following block gadget b still has degree over Δ. Consequently, from every G_z, at least $\sum_{j=1}^{m}(s(c_j) - 1) + nm(\Delta + 1) = k_c$ vertices are deleted, and $|S \cap V(G_z)| \geq k_c$ follows.

Since $\kappa = \kappa_1 \cdot \kappa_2 = (n + 1) \cdot \kappa_2$, while $|S| \leq n + \kappa \cdot k_c$, it follows that there exist κ_2 sequential copies $G_{z'+1}, \ldots, G_{z'+\kappa_2}$, such that $|S \cap V(G_w)| = k_c$, for $w \in [z' + 1, z' + \kappa_2]$. Consequently, for any such G_w, S contains exactly $s(c_j) - 1$ vertices from each $\hat{C}_j^{G_w}$ as well as exactly $\Delta + 1$ vertices from $\hat{B}_{i,j}^{G_w}$, where $i \in [n]$ and $j \in [m]$. Out of those $\Delta + 1$ vertices, 1 must belong to the set $\{\chi_i, y_i, q_i\}$, for every $i \in [n]$. The last vertex must be either vertex b or one of its neighbors: if that were not the case, then b would have $\Delta + 1$ neighbors among the vertices of $\hat{B}_{i,j}^{G_w}$. Assume that, in the case $b \notin S$, the neighbor of b belonging to S is a: if that is not the case, one can construct a deletion set S' of the same cardinality for which the statement holds, and since the neighborhood of a is a superset of the neighborhood of any leaf attached to b, $G - S'$ has maximum degree at most Δ.

Let $\hat{B}_{i,j}^{G_w}$ be one of those block gadgets and let $N(\hat{B}_{i,j}^{G_w})$ denote the block gadget whose a vertex is the vertex a' of $\hat{B}_{i,j}^{G_w}$. We will consider two cases: either $a \in S$ or $b \in S$.

If $a \in S$, then for any block gadgets $\hat{B}_{i,j'}^{G_{w'}}$, where either a) $w' = w$ and $j' > j$ or b) $w' > w$ and $j' \in [m]$, it holds that $a \in S$: the statement holds for $N(\hat{B}_{i,j}^{G_w}(G_w))$, otherwise the vertex b of $\hat{B}_{i,j}^{G_w}$ has degree $\Delta + 1$ in $G - S$. Suppose that it holds for every

block gadget up to some $\hat{B}^{G_{w^*}}_{i,j^*}$. Then, it holds for $N(\hat{B}^{G_{w^*}}_{i,j^*})$ as well, otherwise the vertex b of $\hat{B}^{G_{w^*}}_{i,j^*}$ has degree $\Delta+1$ in $G-S$.

On the other hand, assume that $b \in S$. We will prove that $\mathcal{P}^i$ may induce at most 2Δ inconsistencies. Notice that as soon as we delete vertex a from some block gadget, then we delete also vertex a from all subsequent block gadgets. Thus, assume in the following that exactly $\Delta+1$ vertices are deleted per block gadget, none of which is vertex a. If for block gadget $\hat{B}^{G_w}_{i,j}$, α_p vertices are included in S from set $\{\chi_i : i \in [n]\}$, β_p from set $\{q_i : i \in [n]\}$ and γ_p from set $\{y_i : i \in [n]\}$, where p is an index denoting the block gadget, then $a'(\hat{B}^{G_w}_{i,j})$ has $\Delta - \gamma_p$ incoming edges from $\{y_i : i \in [n]\}$ in $G-S$, where $a'(\hat{B}^{G_w}_{i,j})$ denotes the vertex a' of $\hat{B}^{G_w}_{i,j}$. Notice that vertices $a'(\hat{B}^{G_w}_{i,j})$ and $a(N(\hat{B}^{G_w}_{i,j}))$ coincide, thus the latter may have at most γ_p incoming edges from the block gadget $N(\hat{B}^{G_w}_{i,j})$, i.e. $\gamma_p + \alpha_{p+1} \geq \Delta$. Since $\alpha_i + \beta_i + \gamma_i = \Delta$, $\gamma_p \geq \gamma_{p+1} + \beta_{p+1}$ and $\alpha_{p+1} \geq \alpha_p + \beta_p$ follow. Each inconsistency then corresponds to either $\alpha_{p+1} > \alpha_p$ or $\gamma_{p+1} < \gamma_p$. Taking into account the fact that $0 \leq \alpha_i, \gamma_i \leq \Delta$, one can infer that the number of possible inconsistencies is at most 2Δ; half of them occur due to the increase of α_p while the other half due to the decrease of γ_p (one may assume that in the worst case these happen independently).

Therefore, the maximum amount of inconsistencies for each of the n paths $\mathcal{P}_i$, without deleting more than $\Delta+1$ vertices per block gadget, is $2\Delta+1$ (the $+1$ being due to the case where $a \in S$).

Now, suppose that $\beta_p > 0$, for some block gadget $\hat{B}_{i,j}$ of index p. Then we will show that $\alpha_p \neq \alpha_{p+1}$. Suppose that this is not the case. Then, it holds that $\gamma_p + \alpha_{p+1} \geq \Delta \iff \gamma_p + \alpha_p \geq \Delta$, which implies that $\beta_p \leq 0$, contradiction.

Since we have $\kappa_2 = (2\Delta+1)n+1$ repetitions of the whole construction, due to the pigeonhole principle, there is a copy G_π, for which all the deletions happening in the block gadgets of $P_i^{G_\pi}$ are mapped to the same non-invalid assignment (this is indeed a non-invalid assignment, since $\beta_p = 0$). Moreover, since for every constraint gadget $\hat{C}_j^{G_\pi}$ exactly $s(c_j)-1$ vertices are included in S, and each such gadget has a clique of size $s(c_j)$, each vertex of which has Δ leaves attached, S cannot contain any leaves from $\hat{C}_j^{G_\pi}$. $\lhd$

Consider the following assignment $f\colon X \to Y$ on the variables of ϕ:

- if $a \in S$, then let $f(x_i) = \Delta+1$,
- alternatively, let $f(x_i) = y$, where $y \in [0,\Delta]$ is equal to the number of vertices of $\{\chi_i : i \in [\Delta]\}$ belonging to S,

where a and χ_i refer to the vertices of the block gadget $\hat{B}^{G_\pi}_{i,j}$ for some $j \in [m]$ (since G_π is consistent, the choice of j does not matter).

We will prove that f satisfies all the constraints. Consider a constraint c_j. Regarding the constraint gadget $\hat{C}_j^{G_\pi}$, it holds that S includes exactly $s(c_j)-1$ vertices, none of which can be a leaf vertex. Let $v_\ell^j \notin S$ be the only non-leaf vertex of $\hat{C}_j^{G_\pi}$ not belonging to S. Since none of its leaves belongs to S, it follows that every neighbor of v_ℓ^j in the

block gadgets $\hat{B}_{i,j}^{G_\pi}$ belongs to S. In that case, notice that the deletion of all the neighbors of v_ℓ^j in the block gadget $\hat{B}_{i,j}^{G_\pi}$ is mapped to an assignment of x_i: if the only neighbor was a, then $f(x_i) = \Delta + 1$, alternatively the neighborhood was comprised of b as well as vertices $\{\chi_i : i \in [p]\} \cup \{y_i : i \in [p+1, \Delta]\}$, which implies that exactly p vertices of $\{\chi_i : i \in [\Delta]\}$ belong to S. Consequently, the satisfying assignment corresponding to v_ℓ^j is a restriction of f, thus f satisfies c_j. Since this holds for any j, ϕ is satisfied. □

Lemma 3.10. *It holds that* $\mathrm{pw}(G) = n + \mathcal{O}(1)$.

Proof. We will prove the statement by providing a mixed search strategy to clean G using at most this many searchers simultaneously. Since for the mixed search number ms it holds that $\mathrm{pw}(G) \leq \mathrm{ms}(G) \leq \mathrm{pw}(G) + 1$ [273], we will show that $\mathrm{ms}(G) \leq n + 3 + 2 \cdot B^q$ and the statement will follow.

Start with graph G_1. Place $2s(c_1)$ searchers to the clique vertices of $\hat{C}_1^{G_1}$ as well as to one leaf per clique vertex, and n searchers on vertices a of block gadgets $\hat{B}_{i,1}^{G_1}$, for $i \in [n]$. By moving the searchers placed on the leaves among the leaf vertices of the constraint gadget, all the edges of the constraint gadget can be cleaned. Next we will describe the procedure to clean $\hat{B}_{i,1}^{G_1}$. Move three extra searchers to vertices b and a' of the block gadget, as well as to a leaf of b. Move the latter among all the different leaves of b. Finally, the searchers placed in b and its leaf can clean the rest of the edges of the gadget: put one on χ_i and the other on a leaf of q_i. Slide the first along the edge connecting χ_i with q_i. Move the latter searcher among all the leaves of q_i, and then place it to y_i. Repeat the whole process for all i, and finally remove all searchers apart from the one placed on a'. By following the same procedure, eventually all block gadgets $\hat{B}_{i,1}^{G_1}$ are cleaned.

In order to clean the rest of the graph, we first move the searchers from $\hat{C}_j^{G_z}$ to $\hat{C}_{j+1}^{G_z}$ if $j < m$ or to $\hat{C}_1^{G_{z+1}}$ alternatively (possibly introducing new searchers if required), clean the latter, and then proceed by cleaning the corresponding block gadgets. By repeating this procedure, in the end we clean all the edges of G by using at most $n+3+2\cdot B^q = n+\mathcal{O}(1)$ searchers. □

Therefore, in polynomial time we can construct a graph G, of pathwidth $\mathrm{pw}(G) \leq n + \mathcal{O}(1)$ due to Lemma 3.10, such that, due to Lemmas 3.7 and 3.8, deciding whether there exists $S \subseteq V(G)$ of size $|S| \leq k$ and $G - S$ has maximum degree at most Δ is equivalent to deciding whether ϕ is satisfiable. In that case, assuming there exists a $\mathcal{O}^\star((\Delta + 2 - \varepsilon)^{\mathrm{pw}(G)})$ algorithm for Bounded Degree Vertex Deletion, one could decide q-CSP-B in time $\mathcal{O}^\star((\Delta + 2 - \varepsilon)^{\mathrm{pw}(G)}) = \mathcal{O}^\star((B - \varepsilon)^{n+\mathcal{O}(1)}) = \mathcal{O}^\star((B - \varepsilon)^n)$, which contradicts the SETH due to Theorem 2.1. □

3.2.2. Defective Coloring

In this subsection, we present tight lower bounds for Defective Coloring parameterized by the treewidth of the input graph plus the target degree. We start by first presenting a variety of useful gadgets employed in our constructions, followed by the lower bounds, and finally an algorithm of matching running time.

Coloring Gadgets

Here we develop various gadgets that will be mainly used in Section 3.2.2. We heavily rely on the constructions presented in [25], some of which we briefly present for the sake of completeness.

Intuitively, our goal is to extend the toolbox of [25], and to construct gadgets that, for any $(\chi_\mathrm{d}, \Delta)$-coloring c of a graph G, where $\chi_\mathrm{d} \geq 2$ and $\Delta \geq 1$, imply the following relationships for vertices $v_1, v_2, c_1, c_2 \in V(G)$ (where c_1, c_2 might coincide):

- $D(v_1, v_2)$: Vertices v_1 and v_2 receive distinct colors, i.e. $c(v_1) \neq c(v_2)$,
- $E(v_1, v_2, c_1, c_2)$: If vertex v_1 receives the color of c_1, then vertex v_2 does not receive the color of c_2, i.e. $c(v_1) = c(c_1) \implies c(v_2) \neq c(c_2)$,
- $I(v_1, v_2, c_1, c_2)$: If vertex v_1 receives the color of c_1, then vertex v_2 receives the color of c_2, i.e. $c(v_1) = c(c_1) \implies c(v_2) = c(c_2)$.

We build upon the results of [25], who presented the *equality gadget* $Q(v_1, v_2, \chi_\mathrm{d}, \Delta)$ as well as the *palette gadget* $P(v_1, v_2, v_3, \chi_\mathrm{d}, \Delta)$.

Lemma 3.11 ([25, Lemma 3.5])**.** *Let $G = (V, E)$ be a graph with $v_1, v_2 \in V$, and let G' be the graph obtained from G by adding to it a copy of $Q(u_1, u_2, \chi_\mathrm{d}, \Delta)$ and identifying u_1 with v_1 and u_2 with v_2. Then, any $(\chi_\mathrm{d}, \Delta)$-coloring of G' must give the same color to v_1, v_2. Furthermore, if there exists a $(\chi_\mathrm{d}, \Delta)$-coloring of G that gives the same color to v_1, v_2, this coloring can be extended to a $(\chi_\mathrm{d}, \Delta)$-coloring of G'.*

Lemma 3.12 ([25, Lemma 3.8])**.** *Let $G = (V, E)$ be a graph with $v_1, v_2, v_3 \in V$, and let G' be the graph obtained from G by adding to it a copy of $P(u_1, u_2, u_3, \chi_\mathrm{d}, \Delta)$ and identifying u_i with v_i for $i \in [3]$. Then, in any $(\chi_\mathrm{d}, \Delta)$-coloring of G', at least two of the vertices of $\{v_1, v_2, v_3\}$ must share a color. Furthermore, if there exists a $(\chi_\mathrm{d}, \Delta)$-coloring of G that gives the same color to two of the vertices of $\{v_1, v_2, v_3\}$, this coloring can be extended to a $(\chi_\mathrm{d}, \Delta)$-coloring of G'.*

Definition 3.13 (Difference Gadget)**.** *For $\chi_\mathrm{d} \geq 2$, $\Delta \geq 0$, $k \geq 0$, let $D(u_1, u_2, \chi_\mathrm{d}, \Delta, k)$ be a graph defined as follows: D contains vertices u_1 and u_2, with the latter having k leaves $l_1, \ldots, l_k$, as well as k copies of the equality gadget $Q(x, y, \chi_\mathrm{d}, \Delta)$, on each of which we identify x with u_1 and y with l_i, for $i \in [k]$.*

Lemma 3.14. *Let $G = (V, E)$ be a graph with $v_1, v_2 \in V$, and let G' be the graph obtained from G by adding to it a copy of $D(u_1, u_2, \chi_\mathrm{d}, \Delta, \Delta + 1)$ and identifying u_1 with v_1 and u_2 with v_2. Then, any $(\chi_\mathrm{d}, \Delta)$-coloring of G' must give different colors to v_1 and v_2. Furthermore, if there exists a $(\chi_\mathrm{d}, \Delta)$-coloring of G that gives different colors to v_1 and v_2, this coloring can be extended to a $(\chi_\mathrm{d}, \Delta)$-coloring of G'.*

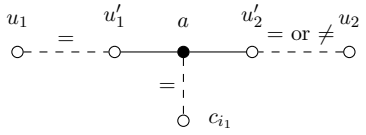
Proof. For the first statement, consider a $(\chi_\mathrm{d}, \Delta)$-coloring $c' \colon V(G') \to [\chi_\mathrm{d}]$ of G'. Notice that due to the properties of the equality gadget [25, Lemma 3.5], it follows that $c'(v_1) = c'(l_i)$ for all $i \in [\Delta + 1]$, where l_i denote the leaves attached to v_2 due to the

difference gadget. In that case, v_2 has at least $\Delta + 1$ neighbors of color $c'(v_1)$, thus $c'(v_2) \neq c'(v_1)$.

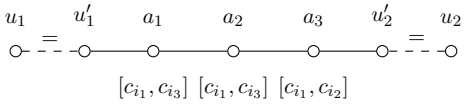
For the second statement, let $c\colon V(G) \to [\chi_\mathrm{d}]$ be a $(\chi_\mathrm{d}, \Delta)$-coloring of G, where $c(v_1) \neq c(v_2)$. In order to extend it to a $(\chi_\mathrm{d}, \Delta)$-coloring of G', color the vertices l_i with the color of v_1, and appropriately color the vertices of the equality gadgets by using [25, Lemma 3.5]. □

Definition 3.15 (Exclusion Gadget)**.** *For $\chi_\mathrm{d} \geq 2$, $\Delta \geq 1$, $C = \{c_i : i \in [\chi_\mathrm{d}]\}$, and (not necessarily distinct) $i_1, i_2 \in [\chi_\mathrm{d}]$, let $E(u_1, u_2, c_{i_1}, c_{i_2}, C, \chi_\mathrm{d}, \Delta)$ be a graph defined as follows:*

- *If either a) $i_1 = i_2$, or b) $i_1 \neq i_2$ and $\chi_\mathrm{d} = 2$, then let E contain vertices $u_1, u_2, u_1', u_2', c_{i_1}$, and a. Add edges between a and u_1', u_2', as well as gadgets $Q(u_1, u_1', \chi_\mathrm{d}, \Delta)$, $Q(c_{i_1}, a, \chi_\mathrm{d}, \Delta)$, and $D(c_{i_1}, a, \chi_\mathrm{d}, \Delta, \Delta - 1)$. If $i_1 = i_2$ add the gadget $Q(u_2, u_2', \chi_\mathrm{d}, \Delta)$, else the gadget $D(u_2, u_2', \chi_\mathrm{d}, \Delta, \Delta + 1)$.*
- *Otherwise, i.e. when $i_1 \neq i_2$ and $\chi_\mathrm{d} > 2$, let E contain vertices $u_1, u_2, u_1', u_2', c_{i_1}, c_{i_2}, c_{i_3}$ and a_1, a_2, a_3, where $i_3 \in [\chi_\mathrm{d}] \setminus \{i_1, i_2\}$. Let vertices $u_1', a_1, a_2, a_3, u_2'$ form a path and add the following gadgets:*
  - $Q(u_1, u_1', \chi_\mathrm{d}, \Delta)$,
  - $Q(u_2, u_2', \chi_\mathrm{d}, \Delta)$,
  - $P(c_{i_1}, c_{i_3}, a_1, \chi_\mathrm{d}, \Delta)$, $D(c_{i_1}, a_1, \chi_\mathrm{d}, \Delta, \Delta)$, $D(c_{i_3}, a_1, \chi_\mathrm{d}, \Delta, \Delta)$,
  - $P(c_{i_1}, c_{i_3}, a_2, \chi_\mathrm{d}, \Delta)$, $D(c_{i_1}, a_2, \chi_\mathrm{d}, \Delta, \Delta)$, $D(c_{i_3}, a_2, \chi_\mathrm{d}, \Delta, \Delta)$,
  - $P(c_{i_1}, c_{i_2}, a_3, \chi_\mathrm{d}, \Delta)$, $D(c_{i_1}, a_3, \chi_\mathrm{d}, \Delta, \Delta)$, $D(c_{i_2}, a_3, \chi_\mathrm{d}, \Delta, \Delta)$.



(a) Either $i_1 = i_2$, or $i_1 \neq i_2$ and $\chi_\mathrm{d} = 2$.



(b) Case where $i_1 \neq i_2$ and $\chi_\mathrm{d} > 2$.

Figure 3.3.: Exclusion gadget $E(u_1, u_2, c_{i_1}, c_{i_2}, C, \chi_\mathrm{d}, \Delta)$. Black vertex a has $\Delta - 1$ leaves l attached, each taking part in an equality gadget $Q(c_{i_1}, l, \chi_\mathrm{d}, \Delta)$. The bracket $[x, y]$ under vertex v denotes that there exists a palette gadget $P(x, y, v, \chi_\mathrm{d}, \Delta)$, as well as that v has 2Δ leaves l attached, with half taking part in a gadget $Q(x, l, \chi_\mathrm{d}, \Delta)$, and the other half in $Q(y, l, \chi_\mathrm{d}, \Delta)$.

Lemma 3.16. *Let $G = (V, E)$ be a graph with $v_1, v_2 \in V$ and $P = \{p^i : i \in [\chi_\mathrm{d}]\} \subseteq V$, and let, for some (not necessarily distinct) $i_1, i_2 \in [\chi_\mathrm{d}]$, G' be the graph obtained from G by adding to it a copy of $E(u_1, u_2, c_{i_1}, c_{i_2}, C, \chi_\mathrm{d}, \Delta)$ and identifying u_1 with v_1, u_2 with v_2, and c_i with p^i. Then, in any $(\chi_\mathrm{d}, \Delta)$-coloring $c'\colon V(G') \to [\chi_\mathrm{d}]$ of G' where a) $c'(p^i) \neq c'(p^j)$ for distinct $i, j \in [\chi_\mathrm{d}]$ and b) $c'(v_1) = c'(p^{i_1})$, it holds that $c'(v_2) \neq c'(p^{i_2})$.*

Furthermore, if there exists a $(\chi_\mathrm{d}, \Delta)$*-coloring* $c\colon V(G) \to [\chi_\mathrm{d}]$ *of* G *where a)* $c(p^i) \neq c(p^j)$ *for distinct* $i, j \in [\chi_\mathrm{d}]$ *and b) either* $c(v_1) \neq c(p^{i_1})$ *or* $c(v_2) \neq c(p^{i_2})$*, this can be extended to a* $(\chi_\mathrm{d}, \Delta)$*-coloring of* G'*.*

Proof. For the first statement, consider a $(\chi_\mathrm{d}, \Delta)$-coloring $c'\colon V(G') \to [\chi_\mathrm{d}]$ of G', where every vertex p^i receives a distinct color. Assume that $c'(v_1) = c'(p^{i_1})$ and consider the following cases:

- If either a) $i_1 = i_2$, or b) $i_1 \neq i_2$ and $\chi_\mathrm{d} = 2$, then vertex a of E has $\Delta - 1$ leaves of color $c'(p^{i_1})$. Moreover, it holds that $c'(u_1') = c'(p^{i_1})$, due to the gadget $Q(u_1, u_1', \chi_\mathrm{d}, \Delta)$. Consequently, $c'(v_2) \neq c'(p^{i_2})$, since otherwise it holds that $c'(u_2') = c'(p^{i_1})$ due to either gadget $Q(u_2, u_2', \chi_\mathrm{d}, \Delta)$ or $D(u_2, u_2', \chi_\mathrm{d}, \Delta, \Delta + 1)$, thus a has $\Delta + 1$ same colored neighbors, which is a contradiction.
- Otherwise, i.e. if $i_1 \neq i_2$ and $\chi_\mathrm{d} > 2$, it follows that a_1 has Δ leaves of color $c'(p^{i_1})$ as well as a neighbor which is connected via an equality gadget with v_1. Consequently, a_1 has $\Delta + 1$ neighbors of color $c'(p^{i_1})$, and due to the palette gadget, it follows that $c'(a_1) = c'(p^{i_3})$. In an analogous way, it follows that $c'(a_2) = c'(p^{i_1})$ and that $c'(a_3) = c'(p^{i_2})$. Assume that $c'(v_2) = c'(p^{i_2})$. Then, a_3 has $\Delta + 1$ neighbors of color $c(p^{i_2})$, due to its Δ leaves as well as vertex u_2', which is connected via an equality gadget with v_2, which leads to a contradiction.

For the second statement, consider a $(\chi_\mathrm{d}, \Delta)$-coloring $c\colon V(G) \to [\chi_\mathrm{d}]$ of G, where every vertex p^i receives a distinct color and additionally either $c(v_1) \neq c(p^{i_1})$ or $c(v_2) \neq c(p^{i_2})$. Consider the following cases:

- If either a) $i_1 = i_2$, or b) $i_1 \neq i_2$ and $\chi_\mathrm{d} = 2$, then let vertex a of E, as well as the latter's $\Delta - 1$ leaves receive color $c(p^{i_1})$. Next, color u_1' with $c(u_1)$. In case $i_1 = i_2$, then let u_2' receive color $c(v_2)$, otherwise it receives color different than $c(v_2)$. In both cases a has at most Δ same colored neighbors, and by using Lemmas 3.11 and 3.14, we can color all the internal vertices of the equality and difference gadgets.
- Otherwise, i.e. if $i_1 \neq i_2$ and $\chi_\mathrm{d} > 2$, consider the following two cases:
  - If $c(v_1) \neq c(p^{i_1})$, then let a_1 receive color $c(p^{i_1})$, a_2 color $c(p^{i_3})$ and a_3 color $c(p^{i_1})$.
  - Alternatively, it holds that $c(v_2) \neq c(p^{i_2})$, and let a_1 receive color $c(p^{i_3})$, a_2 color $c(p^{i_1})$ and a_3 color $c(p^{i_2})$.

  In both cases, all vertices a_1, a_2, a_3 have exactly Δ same colored neighbors, and it remains to color the internal vertices of the equality and palette gadgets using Lemmas 3.11 and 3.12.

This concludes the proof. □

Definition 3.17 (Implication Gadget)**.** *For* $\chi_\mathrm{d} \geq 2$*,* $\Delta \geq 1$*,* $C = \{c_i : i \in [\chi_\mathrm{d}]\}$*, and (not necessarily distinct)* $i_1, i_2 \in [\chi_\mathrm{d}]$*, let* $I(u_1, u_2, c_{i_1}, c_{i_2}, C, \chi_\mathrm{d}, \Delta)$ *be a graph defined as follows:* I *contains vertices* u_1 *and* u_2 *and exclusion gadgets* $E(u_1, u_2, c_{i_1}, c_k, C, \chi_\mathrm{d}, \Delta)$*, for all* $k \in [\chi_\mathrm{d}] \setminus \{i_2\}$*.*

Lemma 3.18. *Let $G = (V, E)$ be a graph with $v_1, v_2 \in V$ and $P = \{p^i : i \in [\chi_d]\} \subseteq V$, and let, for some (not necessarily distinct) $i_1, i_2 \in [\chi_d]$, G' be the graph obtained from G by adding to it a copy of $I(u_1, u_2, c_{i_1}, c_{i_2}, C, \chi_d, \Delta)$ and identifying u_1 with v_1, u_2 with v_2, and c_i with p^i. Then, in any (χ_d, Δ)-coloring $c' \colon V(G') \to [\chi_d]$ of G' where a) $c'(p^i) \neq c'(p^j)$ for distinct $i, j \in [\chi_d]$ and b) $c'(v_1) = c'(p^{i_1})$, it holds that $c'(v_2) = c'(p^{i_2})$. Furthermore, if there exists a (χ_d, Δ)-coloring $c \colon V(G) \to [\chi_d]$ of G where a) $c(p^i) \neq c(p^j)$ for distinct $i, j \in [\chi_d]$ and b) either $c(v_1) \neq c(p^{i_1})$ or $c(v_2) = c(p^{i_2})$, this can be extended to a (χ_d, Δ)-coloring of G'.*

Proof. For the first statement, consider a (χ_d, Δ)-coloring $c' \colon V(G') \to [\chi_d]$ of G', where every vertex p^i receives a distinct color. Assume that $c'(v_1) = c'(p^{i_1})$. Then, due to Lemma 3.16 it follows that $c'(v_2) \neq c'(p^k)$ for all $k \in [\chi_d] \setminus \{i_2\}$, thus $c'(v_2) = c'(p^{i_2})$ follows.

For the second statement, consider a (χ_d, Δ)-coloring $c \colon V(G) \to [\chi_d]$ of G, where every vertex p^i receives a distinct color and additionally either $c(v_1) \neq c(p^{i_1})$ or $c(v_2) = c(p^{i_2})$. Consequently, it holds that either $c(v_1) \neq c(p^{i_1})$ or $c(v_2) \neq c(p^k)$, for all $k \in [\chi_d] \setminus \{i_2\}$, thus from Lemma 3.16 the statement follows. □

The following lemma proves that the use of the previously described gadgets does not increase the pathwidth of the graph by much. Notice that part of it has been proven in [25].

Lemma 3.19. *Let $G = (V, E)$ be a graph and let G' be the graph obtained from G by repeating the following operation: find a copy of one of the following gadgets*

- $Q(u_1, u_2, \chi_d, \Delta)$,
- $P(u_1, u_2, u_3, \chi_d, \Delta)$,
- $D(u_1, u_2, \chi_d, \Delta, k)$,
- $E(u_1, u_2, c_{i_1}, c_{i_2}, C, \chi_d, \Delta)$,
- $I(u_1, u_2, c_{i_1}, c_{i_2}, C, \chi_d, \Delta)$,

remove all its internal vertices from the graph, and add all edges between its endpoints which are not already connected. Then $\mathrm{tw}(G) \leq \max\{\mathrm{tw}(G'), \mathcal{O}(\chi_d)\}$ and $\mathrm{pw}(G) \leq \mathrm{pw}(G') + \mathcal{O}(\chi_d)$.

Proof. The result for the equality and palette gadgets has been shown in [25, Lemma 4.2]. In particular, it is shown that one can obtain a path decomposition of an equality or a palette gadget, named T_Q and T_P respectively, of width χ_d, every bag of which contains vertices u_1, u_2 (and u_3 in the case of the palette gadget).

Difference Gadget. Remember that the difference gadget contains k leaves l_i, attached to vertex u_2. First, as observed in the proof of [25, Lemma 4.2], there is a path decomposition of $Q(u_1, l_i, \chi_\mathrm{d}, \Delta)$ with width χ_d, where every bag contains the vertices u_1 and l_i. In that case, there exists a path decomposition of $D(u_1, u_2, \chi_\mathrm{d}, \Delta)$ of width $\chi_\mathrm{d}+1$: serially connect the path decompositions of $Q(u_1, l_i, \chi_\mathrm{d}, \Delta)$, for $i \in [k]$, and add to all the bags the vertex u_2. Call this path decomposition T_D, and notice that all of its bags contain both vertices u_1 and u_2.

Exclusion Gadget. First consider the case where either $i_1 = i_2$ or $\chi_\mathrm{d} = 2$. Then, the gadget consists of vertices $u_1, u_2, u_1', u_2', c_{i_1}$, and a. Vertex a has an edge with both u_1', u_2', while there exist gadgets $Q(u_1, u_1', \chi_\mathrm{d}, \Delta)$, $Q(c_{i_1}, a, \chi_\mathrm{d}, \Delta)$, and $D(c_{i_1}, a, \chi_\mathrm{d}, \Delta, \Delta-1)$. If $i_1 = i_2$, there also exists the equality gadget $Q(u_2, u_2', \chi_\mathrm{d}, \Delta)$, else the difference gadget $D(u_2, u_2', \chi_\mathrm{d}, \Delta, \Delta+1)$. Construct a path decomposition T_E of width $\mathcal{O}(\chi_\mathrm{d})$ comprised of the following path decompositions:

- a path decomposition of $Q(u_1, u_1', \chi_\mathrm{d}, \Delta)$ of width χ_d, every bag of which contains u_1, u_1',
- a path decomposition of $Q(c_{i_1}, a, \chi_\mathrm{d}, \Delta)$ of width χ_d, every bag of which contains c_{i_1}, a,
- a path decomposition of $D(c_{i_1}, a, \chi_\mathrm{d}, \Delta, \Delta-1)$ of width $\chi_\mathrm{d}+1$, every bag of which contains c_{i_1}, a,
- depending on which case we are, a path decomposition of either $Q(u_2, u_2', \chi_\mathrm{d}, \Delta)$ of width χ_d, or of $D(u_2, u_2', \chi_\mathrm{d}, \Delta, \Delta+1)$ of width $\chi_\mathrm{d}+1$, every bag of which contains vertices u_2, u_2'.

To conclude the construction of T_E, add to every bag of this path decomposition the necessary vertices such that $u_1, u_2, u_1', u_2', c_i$ are contained in every bag, for all $i \in [\chi_\mathrm{d}]$.

For the remaining case, the exclusion gadget consists of equality, palette, and difference gadgets. By connecting path decompositions of each respective gadget in an analogous way, while subsequently adding vertices in order to ensure that every bag contains $u_1, u_2, u_1', u_2', a_1, a_2, a_3, c_i$ for all $i \in [\chi_\mathrm{d}]$, it follows that we get a path decomposition T_E of width $\mathcal{O}(\chi_\mathrm{d})$.

Implication Gadget. Lastly, for the case of the implication gadget, remember that this consists of exclusion gadgets $E(u_1, u_2, c_{i_1}, c_k, C, \chi_\mathrm{d}, \Delta)$, for all $k \in [\chi_\mathrm{d}] \setminus \{i_2\}$. Again, consider a path decomposition of each of those $|C|-1 = \chi_\mathrm{d}-1$ exclusion gadgets resulting in a path decomposition T_I of width $\mathcal{O}(\chi_\mathrm{d})$ (notice that each bag of the decomposition of an exclusion gadget contains vertices $C \cup \{u_1, u_2\}$).

We now take an optimal tree or path decomposition of G', call it T', and construct from it a decomposition of G. Consider a gadget $H \in \{Q, P, D, E, I\}$ that appears in G with endpoints u_1, u_2 (and possibly u_3 and $C = \{c_1, \ldots, c_{\chi_\mathrm{d}}\}$). Since in G' these endpoints form a clique, there is a bag in T' that contains all of them. Let B be the

smallest such bag, that is, the bag that contains the smallest number of vertices. Now, if T' is a tree decomposition, we take T_H and attach it to B. If T' is a path decomposition, we insert in the decomposition immediately after B the decomposition T_H where we have added all vertices of B in all bags of T_H. It is not hard to see that in both cases the decompositions remain valid, and we can repeat this process for every H until we have a decomposition of G. □

Lower Bound

In the following, we will present a reduction from q-CSP-B to Defective Coloring, for any fixed $\Delta \geq 1$ and $\chi_{\mathrm{d}} \geq 2$, where $B = \chi_{\mathrm{d}} \cdot (\Delta + 1)$. In that case, if there exists a $\mathcal{O}^\star((\chi_{\mathrm{d}} \cdot (\Delta + 1) - \varepsilon)^{\mathrm{pw}})$ algorithm for Defective Coloring, where $\varepsilon > 0$, then there exists a $\mathcal{O}^\star((B-\varepsilon)^n)$ algorithm for q-CSP-B, for any constant q, which due to Theorem 2.1 results in SETH failing.

The reduction is similar in nature to the one presented in Section 3.2.1, consisting of "long paths" of serially connected block gadgets, each of which corresponds to a variable of the given CSP, while each column of this construction is associated with one of its constraints.

In the whole section we will use the coloring gadgets presented in Section 3.2.2. As was the case for Bounded Degree Vertex Deletion, we first start with the case where $\Delta = 1$ and then with the case where $\Delta \geq 2$.

Theorem 3.20. *For any constant $\varepsilon > 0$ and for any fixed $\chi_{\mathrm{d}} \geq 2$, there is no $\mathcal{O}^\star((2\chi_{\mathrm{d}} - \varepsilon)^{\mathrm{pw}})$ algorithm deciding whether G admits a $(\chi_{\mathrm{d}}, 1)$-coloring, where* pw *denotes the graph's pathwidth, unless the SETH is false.*

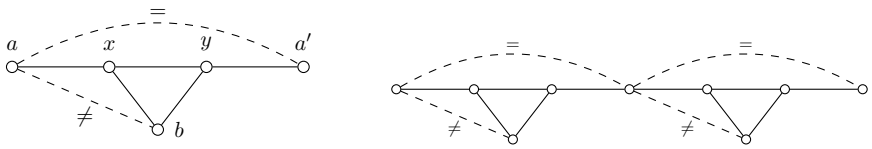
Proof. Fix some positive $\varepsilon > 0$ for which we want to prove the theorem. Let $B = 2\chi_{\mathrm{d}}$. We will reduce q-CSP-B, for some q that is a constant that only depends on ε, to Defective Coloring for $\Delta = 1$ and χ_{d} colors in a way that ensures that if the resulting Defective Coloring instance could be solved in time $\mathcal{O}^\star((2\chi_{\mathrm{d}} - \varepsilon)^{\mathrm{pw}})$, then we would obtain an algorithm for q-CSP-B that would contradict the SETH due to Theorem 2.1. To this end, let ϕ be an instance of q-CSP-B of n variables $X = \{x_i : i \in [n]\}$ taking values over the set $Y = [B]$ and m constraints $C = \{c_j : j \in [m]\}$. For each constraint we are given a set of at most q variables which are involved in this constraint and a list of satisfying assignments for these variables, the size of which is denoted by $s \colon C \to [B^q]$, i.e. $s(c_j) \leq B^q = \mathcal{O}(1)$ denotes the number of satisfying assignments for constraint c_j. We will construct in polynomial time an equivalent instance G of Defective Coloring for $\Delta = 1$ and χ_{d} colors, where $\mathrm{pw}(G) \leq n + \mathcal{O}(1)$.

Since we will repeatedly use the equality, difference, and palette gadgets (see Section 3.2.2), we will use the following convention: whenever v_1, v_2, v_3 are vertices we have already introduced to G, when we say that we add an equality gadget $Q(v_1, v_2)$, a difference gadget $D(v_1, v_2)$, or a palette gadget $P(v_1, v_2, v_3)$, this means that we add to G a copy of $Q(u_1, u_2, \chi_{\mathrm{d}}, \Delta)$, of $D(u_1, u_2, \chi_{\mathrm{d}}, \Delta, \Delta + 1)$, or of $P(u_1, u_2, u_3, \chi_{\mathrm{d}}, \Delta)$ respectively, and then identify $u_1, u_2(, u_3)$ with $v_1, v_2(, v_3)$ respectively.

Palette Vertices. Construct a clique of χ_d vertices $P = \{p^i : i \in [\chi_d]\}$. Attach to vertex p^i a leaf p^i_l, and add equality gadgets $Q(p^i, p^i_l)$, where $i \in [\chi_d]$.

Whenever v_1, v_2 are vertices we have already introduced to G, when we say that we add an exclusion gadget $E(v_1, v_2, v'_1, v'_2)$ or an implication gadget $I(v_1, v_2, v'_1, v'_2)$, this means that we add to G a copy of $E(u_1, u_2, c_1, c_2, C, \chi_d, \Delta)$ or of $I(u_1, u_2, c_1, c_2, C, \chi_d, \Delta)$ respectively and then identify u_1, u_2, c_i with v_1, v_2, p^i respectively, for all $i \in [\chi_d]$.

Block and Variable Gadgets. For every variable x_i and every constraint c_j, construct a *block gadget* $\hat{B}_{i,j}$ as depicted in Figure 3.4a. Dashed lines between u_1 and u_2 imply an equality $Q(u_1, u_2)$ or a difference gadget $D(u_1, u_2)$, and not an edge. If $\chi_d \geq 3$, add palette gadgets $P(a, b, x)$ and $P(a', b, y)$. Next, for $j \in [m-1]$, we serially connect the block gadgets $\hat{B}_{i,j}$ and $\hat{B}_{i,j+1}$ so that the vertex a' of $\hat{B}_{i,j}$ is the vertex a of $\hat{B}_{i,j+1}$, thus resulting in n "paths" $P_1, \ldots, P_n$ consisting of m serially connected block gadgets, called *variable gadgets*. Intuitively, the variable gadget is meant to represent a variable x_i and hence needs to have $2\chi_d$ different viable configurations. These are made up by deciding on a color for a (χ_d choices) and then deciding which of x, y will receive the same color as a (two choices). We will show that the gadget is set up so that exactly one of x, y receives the same color as a (and a').



(a) Block gadget when $\Delta = 1$. (b) Serially connected block gadgets.

Figure 3.4.: Variable gadgets are comprised of serially connected block gadgets.

Constraint Gadget. This gadget is responsible for determining constraint satisfaction, based on the choices made in the rest of the graph. For constraint c_j, construct the *constraint gadget* $\hat{C}_j$ as depicted in Figure 3.5:

- introduce vertices r_w, where $w \in [s(c_j)]$, and add $Q(p^2, r_1)$, as well as $P(p^1, p^2, r_w)$ when $\chi_d \geq 3$, for $w \in [2, s(j)]$,
- for $w \in [s(j)]$, introduce vertices v^j_w, as well as palette gadgets $P(p^1, p^2, v^j_w)$ when $\chi_d \geq 3$, and fix an arbitrary one-to-one mapping between those vertices and the satisfying assignments of c_j,
- introduce vertices k_w, where $w \in [s(c_j)]$, and add $Q(p^2, k_{s(c_j)})$, as well as $P(p^1, p^2, k_w)$ when $\chi_d \geq 3$ for $w \in [s(j) - 1]$,
- add edges between vertex k_w and vertices r_w, v^j_w, as well as $D(k_w, r_{w+1})$,
- if variable x_i is involved in the constraint c_j and v^j_ℓ corresponds to an assignment where x_i has value $s \in Y$, then:

(i) if $s \leq \chi_{\mathrm{d}}$, then add implication gadgets $I(v_\ell^j, a, p^1, p^s)$ and $I(v_\ell^j, x, p^1, p^s)$,

(ii) if on the other hand $\chi_{\mathrm{d}} < s \leq 2\chi_{\mathrm{d}}$, then add implication gadget $I(v_\ell^j, a, p^1, p^{s'})$ and exclusion gadget $E(v_\ell^j, x, p^1, p^{s'})$, for $s' = s - \chi_{\mathrm{d}}$,

where vertices a and x belong to $\hat{B}_{i,j}$.

Intuitively, the constraint gadget is set up in a way that forces, for some $\ell \in [s(c_j)]$, vertex v_ℓ^j to receive color 1, which in turn "activates" the implication and exclusion gadgets we have added to this vertex. This ensures that the assignment encoded by the variable gadgets agrees with the satisfying assignment of c_j represented by v_ℓ^j.

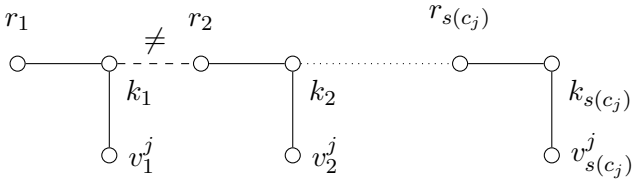


Figure 3.5.: Constraint gadget $\hat{C}_j$ when $\Delta = 1$.

Let graph G_0 correspond to the graph containing all variable gadgets P_i as well as all the constraint gadgets $\hat{C}_j$, for $i \in [n]$ and $j \in [m]$. We refer to the block gadget $\hat{B}_{i,j}$, to the variable gadget P_i, and to the constraint gadget $\hat{C}_j$ of G_z as $\hat{B}_{i,j}^{G_z}$, $P_i^{G_z}$ and $\hat{C}_j^{G_z}$ respectively. To construct graph G, introduce $\kappa = n + 1$ copies $G_1, \ldots, G_\kappa$ of G_0, such that they are connected sequentially as follows: for $i \in [n]$ and $j \in [\kappa - 1]$, the vertex a' of $\hat{B}_{i,m}^{G_j}$ is the vertex a of $\hat{B}_{i,1}^{G_{j+1}}$. Let $\mathcal{P}^i$ denote the "path" resulting from $P_i^{G_1}, \ldots, P_i^{G_\kappa}$.

Lemma 3.21. *For any $\chi_{\mathrm{d}} \geq 2$, if ϕ is satisfiable, then G admits a $(\chi_{\mathrm{d}}, 1)$-coloring.*

Proof. Let $f\colon X \to Y$ denote an assignment which satisfies all the constraints $c_1, \ldots, c_m$. We will describe a $(\chi_{\mathrm{d}}, 1)$-coloring $c\colon V(G) \to [\chi_{\mathrm{d}}]$ of G.

Let $c(p^i) = c(p_l^i) = i$, for $i \in [\chi_{\mathrm{d}}]$. Next, for the vertices of block gadget $\hat{B}_{i,j}^{G_z}$, where $z \in [\kappa]$, $i \in [n]$ and $j \in [m]$, consider the following cases:

1. if $f(x_i) = k$, for $k \in [\chi_{\mathrm{d}}]$, then let $k' \in [\chi_{\mathrm{d}}] \setminus \{k\}$ be an arbitrary color and set $c(a) = c(x) = k$, while $c(b) = c(y) = k'$,

2. if $f(x_i) = \chi_{\mathrm{d}} + k$, for $k \in [\chi_{\mathrm{d}}]$, then let $k' \in [\chi_{\mathrm{d}}] \setminus \{k\}$ be an arbitrary color and set $c(a) = c(y) = k$, while $c(b) = c(x) = k'$.

Regarding the constraint gadgets, let c_j be one of the constraints of ϕ. Since f is a satisfying assignment, there exists at least one vertex among $v_1^j, \ldots, v_{s(j)}^j$ in $\hat{C}_j^{G_z}$, for some $z \in [\kappa]$, mapping to the restriction of f to the variables appearing in c_j. Let v_ℓ^j be one such vertex of minimum index. Then, set $c(v_\ell^j) = 1$, while any other vertex $v_{\ell'}^j$, with $\ell' \neq \ell$, receives color 2. Moreover, let k_w receive color 1 for $w < \ell$ and color 2 for $\ell \leq w \leq s(c_j)$. On the other hand, let r_w receive color 2 for $w \leq \ell$ and color 1 for $\ell < w \leq s(c_j)$.

Lastly, properly color the internal vertices of the equality / palette / difference / exclusion / implication gadgets using Lemmas 3.11, 3.12, 3.14, 3.16, and 3.18. To see that all gadgets are properly colored using these lemmas, observe that any vertex colored so far has at most 1 same-colored neighbor, while the following hold:

- $P = \{p^i : i \in [\chi_{\mathrm{d}}]\}$ consists of χ_{d} vertices, each receiving a distinct color, and for all $i \in [\chi_{\mathrm{d}}]$, $c(p^i) = c(p^i_l)$,
- in all block gadgets, $c(a) = c(a')$, $c(a) \neq c(b)$, and vertices x, y are colored either $c(a)$ or $c(b)$,
- for $z \in [\kappa]$ and $j \in [m]$, in constraint gadget $\hat{C}^{G_z}_j$ it holds that $c(r_1) = c(p^2)$, $c(k_{s(c_j)}) = c(p^2)$, $c(k_w) \neq c(r_{w+1})$ for $w \in [s(c_j) - 1]$, vertices k_w, r_w, v^j_w for $w \in [s(c_j)]$ are colored either $c(p^1)$ or $c(p^2)$, and lastly if $c(v^j_\ell) = c(p^1)$ and x_i denotes a variable appearing in c_j such that vertex v^j_ℓ corresponds to an assignment where x_i receives value $s \in [2\chi_{\mathrm{d}}]$ (which is a restriction of assignment f), then
  - if $s \leq \chi_{\mathrm{d}}$, $c(a) = s$ and $c(x) = s$,
  - else $c(a) = s'$ and $c(x) \neq s'$, for $s' = s - \chi_{\mathrm{d}}$,

  where vertices a, x belong to $\hat{B}^{G_z}_{i,j}$.

This concludes the proof. □

Lemma 3.22. *For any $\chi_{\mathrm{d}} \geq 2$, if G admits a $(\chi_{\mathrm{d}}, 1)$-coloring, then ϕ is satisfiable.*

Proof. Let $c\colon V(G) \to [\chi_{\mathrm{d}}]$ be a $(\chi_{\mathrm{d}}, 1)$-coloring of G. Due to the properties of the equality gadgets, it holds that $c(p^i) = c(p^i_l)$, for all $i \in [\chi_{\mathrm{d}}]$. Since c is a $(\chi_{\mathrm{d}}, 1)$-coloring, it follows that $c(p^i) \neq c(p^j)$, for distinct $i, j \in [\chi_{\mathrm{d}}]$. Assume without loss of generality that $c(p^i) = i$, for $i \in [\chi_{\mathrm{d}}]$.

For G_z, consider a mapping between the coloring of vertices of $\hat{B}^{G_z}_{i,j}$ and the value of x_i for some assignment of the variables of ϕ. In particular, the coloring of both vertices a and x with color $k \in [\chi_{\mathrm{d}}]$ is mapped with an assignment where x_i receives value k, while if only a receives color k, with an assignment where x_i receives value $\chi_{\mathrm{d}} + k$. We will say that an *inconsistency* occurs in a variable gadget $P^{G_z}_i$ if there exist block gadgets $\hat{B}^{G_z}_{i,j}$ and $\hat{B}^{G_z}_{i,j+1}$, such that the coloring of the vertices of each block gadget maps to different assignments of x_i. We say that G_z is *consistent* if no inconsistency occurs in its variable gadgets $P^{G_z}_i$, for every $i \in [n]$.

▷ Claim 3.23. There exists $\pi \in [\kappa]$ such that G_π is consistent.

Proof of the claim. We will prove that every path $\mathcal{P}^i$ may induce at most 1 inconsistency. In that case, since there are n such paths and $\kappa = n+1$ copies of G_0, due to the pigeonhole principle there exists some G_π without any inconsistencies.

Consider a path $\mathcal{P}^i$ as well as a block gadget $\hat{B}^{G_z}_{i,j}$, for some $z \in [\kappa]$ and $j \in [m]$. Let $N(\hat{B}^{G_z}_{i,j})$ denote the block gadget right of $\hat{B}^{G_z}_{i,j}$, i.e. vertex a' of $\hat{B}^{G_z}_{i,j}$ coincides with vertex a of $N(\hat{B}^{G_z}_{i,j})$. Moreover, let $\hat{B}^{G_{z'}}_{i,j'}$, where either a) $z' = z$ and $j' > j$ or b) $z' > z$ and

$j' \in [m]$, denote some block gadget which appears to the right of $\hat{B}_{i,j}^{G_z}$. For every block gadget, due to the properties of the equality gadget, it holds that $c(a) = c(a')$, therefore the color of vertex a is the same for all block gadgets belonging to the same path $\mathcal{P}^i$. Consider the following two cases regarding the vertices of $\hat{B}_{i,j}^{G_z}$:

- If $c(a) \neq c(x)$, it follows that $c(a) = c(y)$, since alternatively b would have 2 same colored neighbors. Then, it holds that a', which is the vertex a of $N(\hat{B}_{i,j}^{G_z})$, has a same colored neighbor in $\hat{B}_{i,j}^{G_z}$, thus for the vertex x of $N(\hat{B}_{i,j}^{G_z})$ it follows that $c(x) \neq c(a)$, and inductively, it follows that $c(a) \neq c(x)$ for all block gadgets $\hat{B}_{i,j'}^{G_{z'}}$.
- If $c(a) = c(x)$, it follows that $c(a) \neq c(y)$, since $c(x)$ cannot have two same colored neighbors. Then, it holds that a', which is the vertex a of $N(\hat{B}_{i,j}^{G_z})$, has no same colored neighbor in $\hat{B}_{i,j}^{G_z}$. Consequently, for the vertices a and x of $N(\hat{B}_{i,j}^{G_z})$ it follows that either $c(a) = c(x)$ or $c(a) \neq c(x)$. The same holds for all block gadgets $\hat{B}_{i,j'}^{G_{z'}}$.

Thus, it follows that every path can induce at most 1 inconsistency, and since there is a total of n paths, there exists a copy G_π which is consistent. ◁

Consider an assignment $f\colon X \to Y$ as follows. Let a and x denote vertices of the block gadget $\hat{B}_{i,j}^{G_\pi}$, where $c(a) = k \in [\chi_{\mathrm{d}}]$. Then, set $f(x_i) = k$ if $c(x) = k$, and $f(x_i) = \chi_{\mathrm{d}} + k$ otherwise. Notice that one of the above cases holds for every block gadget, thus all variables x_i are assigned a value and f is well defined.

It remains to prove that this assignment satisfies all constraints. Consider the constraint gadget $\hat{C}_j^{G_\pi}$, where $j \in [m]$. We first prove that $c(v_\ell^j) = 1$, for some $\ell \in [s(j)]$. Assume that this is not the case. Then it follows that every vertex k_w has two neighbors of color 2, consequently $c(k_w) = 1$, for all $w \in [s(j)]$. However, due to $Q(p^2, k_{s(j)})$, it follows that $c(k_{s(j)}) = 2$, which is a contradiction. Let v_ℓ^j such that $c(v_\ell^j) = 1$. In that case, due to the implication/exclusion gadgets involving v_ℓ^j, it follows that, if variable x_i is involved in the constraint c_j and v_ℓ^j corresponds to an assignment where x_i has value $s \in Y$, then

(i) if $s = k$, where $k \in [\chi_{\mathrm{d}}]$, then $c(a) = k = c(x)$,

(ii) if $s = \chi_{\mathrm{d}} + k$, where $k \in [\chi_{\mathrm{d}}]$, then $c(a) = k \neq c(x)$,

where vertices a and x belong to $\hat{B}_{i,j}^{G_\pi}$. However, in that case, the assignment that corresponds to v_ℓ^j is a restriction of f, thus f satisfies the constraint c_j. Since $j = 1, \ldots, m$ was arbitrary, this concludes the proof that f is a satisfying assignment for ϕ. □

Lemma 3.24. *It holds that* $\mathrm{pw}(G) \leq n + \mathcal{O}(1)$.

Proof. Due to Lemma 3.19, it holds that $\mathrm{pw}(G) = \mathrm{pw}(G' - P) + 3\chi_{\mathrm{d}}$, where G' is the graph we obtain from G by removing all the equality / palette / difference / exclusion / implication gadgets and add all edges between their endpoints which are not already connected. It therefore suffices to show that $\mathrm{pw}(G' - P) = n + \mathcal{O}(1)$.

We will do so by providing a mixed search strategy to clean $G' - P$ using at most this many searchers simultaneously. Since for the mixed search number ms it holds that $\mathrm{pw}(G'-P) \leq \mathrm{ms}(G'-P) \leq \mathrm{pw}(G'-P)+1$ [273], we will show that $\mathrm{ms}(G'-P) \leq n+5+B^q$ and the statement will follow.

Start with graph G_1. Place $s(c_1)+1$ searchers to the vertices r_1 and v_w^1 of $\hat{C}_1^{G_1}$, for $w \in [s(c_1)]$, as well as n searchers on vertices a of block gadgets $\hat{B}_{i,1}^{G_1}$, for $i \in [n]$. By moving the searcher placed on r_1 along the path formed by $k_1, r_2, k_2, \ldots$, all the edges of the constraint gadget can be cleaned. Next we will describe the procedure to clean $\hat{B}_{i,1}^{G_1}$. Move four extra searchers to all other vertices of $\hat{B}_{i,1}^{G_1}$, namely x, y, b, a'. Afterwards, remove the searchers from vertices a, x, y, b. Repeat the whole procedure for all $i \in [n]$.

In order to clean the rest of the graph, we first move the searchers from $\hat{C}_j^{G_z}$ to $\hat{C}_{j+1}^{G_z}$ if $j < m$ or to $\hat{C}_1^{G_{z+1}}$ alternatively (possibly introducing new searchers if required), clean the latter, and then proceed by cleaning the corresponding block gadgets. By repeating this procedure, in the end we clean all the edges of $G' - P$ by using at most $n + 5 + B^q = n + \mathcal{O}(1)$ searchers. □

Therefore, in polynomial time, we can construct a graph G, of pathwidth $\mathrm{pw}(G) \leq n + \mathcal{O}(1)$ due to Lemma 3.24, such that, due to Lemmas 3.21 and 3.22, deciding whether G admits a $(\chi_\mathrm{d}, 1)$-coloring is equivalent to deciding whether ϕ is satisfiable. In that case, assuming there exists a $\mathcal{O}^\star((2\chi_\mathrm{d} - \varepsilon)^{\mathrm{pw}(G)})$ algorithm for Defective Coloring for χ_d colors and $\Delta = 1$, then for $B = 2\chi_\mathrm{d}$, one could decide q-CSP-B in time $\mathcal{O}^\star((2\chi_\mathrm{d} - \varepsilon)^{\mathrm{pw}(G)}) = \mathcal{O}^\star((B-\varepsilon)^{n+\mathcal{O}(1)}) = \mathcal{O}^\star((B-\varepsilon)^n)$, which contradicts the SETH due to Theorem 2.1. □

Theorem 3.25. *For any constant $\varepsilon > 0$ and for any fixed $\chi_\mathrm{d} \geq 2$, $\Delta \geq 2$, there is no $\mathcal{O}^\star((\chi_\mathrm{d} \cdot (\Delta+1) - \varepsilon)^{\mathrm{pw}})$ algorithm deciding whether G admits a $(\chi_\mathrm{d}, \Delta)$-coloring, where* pw *denotes the graph's pathwidth, unless the SETH is false.*

Proof. Fix some positive $\varepsilon > 0$ for which we want to prove the theorem. Let $B = \chi_\mathrm{d} \cdot (\Delta+1)$. We will reduce q-CSP-B, for some q that is a constant that only depends on ε, to Defective Coloring for maximum degree $\Delta \geq 2$ and χ_d colors in a way that ensures that if the resulting Defective Coloring instance could be solved in time $\mathcal{O}^\star((\chi_\mathrm{d} \cdot (\Delta+1) - \varepsilon)^{\mathrm{pw}})$, then we would obtain an algorithm for q-CSP-B that would contradict the SETH due to Theorem 2.1. To this end, let ϕ be an instance of q-CSP-B of n variables $X = \{x_i : i \in [n]\}$ taking values over the set $Y = [0, B-1]$ and m constraints $C = \{c_j : j \in [m]\}$. For each constraint we are given a set of at most q variables which are involved in this constraint and a list of satisfying assignments for these variables, the size of which is denoted by $s \colon C \to [B^q]$, i.e. $s(c_j) \leq B^q = \mathcal{O}(1)$ denotes the number of satisfying assignments for constraint c_j. We will construct in polynomial time an equivalent instance G of Defective Coloring for $\Delta \geq 2$ and χ_d colors, where $\mathrm{pw}(G) \leq n + \mathcal{O}(1)$.

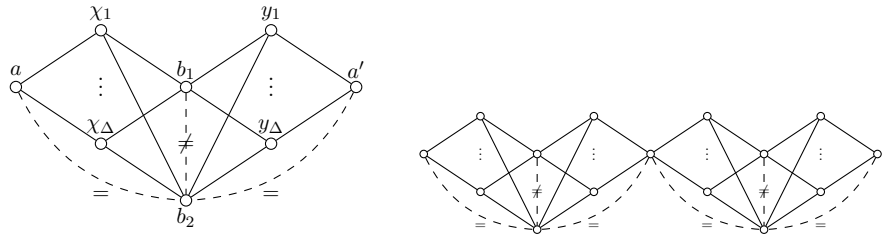
Since we will repeatedly use the equality, difference, and palette gadgets (see Section 3.2.2), we will use the following convention: whenever v_1, v_2, v_3 are vertices we have already introduced to G, when we say that we add an equality gadget $Q(v_1, v_2)$, a

difference gadget $D(v_1, v_2, \delta)$ or a palette gadget $P(v_1, v_2, v_3)$, this means that we add to G a copy of $Q(u_1, u_2, \chi_\mathrm{d}, \Delta)$, of $D(u_1, u_2, \chi_\mathrm{d}, \Delta, \delta)$ or of $P(u_1, u_2, u_3, \chi_\mathrm{d}, \Delta)$ respectively and then identify $u_1, u_2(, u_3)$ with $v_1, v_2(, v_3)$ respectively.

Palette Vertices. Construct a clique of χ_d vertices $P = \{p^i : i \in [\chi_\mathrm{d}]\}$. For $i \in [\chi_\mathrm{d}]$, attach to vertex p^i leaves p^i_l, for $l \in [\Delta]$, and add equality gadgets $Q(p^i, p^i_l)$.

Whenever v_1, v_2 are vertices we have already introduced to G, when we say that we add an exclusion gadget $E(v_1, v_2, v'_1, v'_2)$ or an implication gadget $I(v_1, v_2, v'_1, v'_2)$, this means that we add to G a copy of $E(u_1, u_2, c_1, c_2, C, \chi_\mathrm{d}, \Delta)$ or of $I(u_1, u_2, c_1, c_2, C, \chi_\mathrm{d}, \Delta)$ respectively and then identify u_1, u_2, c_i with v_1, v_2, p^i respectively, for all $i \in [\chi_\mathrm{d}]$.

Block and Variable Gadgets. For every variable x_i and every constraint c_j, construct a *block gadget* $\hat{B}_{i,j}$ as depicted in Figure 3.6a. In order to do so, we introduce vertices a, a', b_1, b_2, χ_i and y_i, for $i \in [\Delta]$. Then, we add gadgets $Q(a, b_2)$, $Q(b_2, a')$ and $D(b_1, b_2, \Delta + 1)$. Finally, we add edges $\{a, \chi_i\}$, $\{a', y_i\}$, as well as between vertices b_1, b_2 and every vertex χ_i, y_i. Moreover, if $\chi_\mathrm{d} \geq 3$, we add palette gadgets $P(b_1, b_2, \chi_i)$ and $P(b_1, b_2, y_i)$, for all $i \in [\Delta]$. Next, for $j \in [m-1]$, we serially connect the block gadgets $\hat{B}_{i,j}$ and $\hat{B}_{i,j+1}$ so that the vertex a' of $\hat{B}_{i,j}$ is the vertex a of $\hat{B}_{i,j+1}$, thus resulting in n "paths" $P_1, \ldots, P_n$ consisting of m serially connected block gadgets, called *variable gadgets*. Intuitively, the variable gadget is meant to represent a variable x_i and hence needs to have $\chi_\mathrm{d}(\Delta + 1)$ different viable configurations. These are made up by deciding on a color for a (χ_d choices) and then deciding how many vertices of $\{\chi_i : i \in [\Delta]\}$ will receive the same color as a ($\Delta + 1$ choices).



(a) Block gadget when $\Delta \geq 2$. (b) Serially connected block gadgets.

Figure 3.6.: Variable gadgets are comprised of serially connected block gadgets.

Constraint Gadget. This gadget is responsible for determining constraint satisfaction, based on the choices made in the rest of the graph. For constraint c_j, construct the *constraint gadget* $\hat{C}_j$ as depicted in Figure 3.7:

- introduce vertices r_w, where $w \in [s(c_j)]$, and add $Q(p^2, r_1)$, as well as $P(p^1, p^2, r_w)$ when $\chi_\mathrm{d} \geq 3$, for $w \in [2, s(j)]$
- for $w \in [s(j)]$, introduce vertices v^j_w, as well as palette gadgets $P(p^1, p^2, v^j_w)$ when $\chi_\mathrm{d} \geq 3$, and fix an arbitrary one-to-one mapping between those vertices and the

satisfying assignments of c_j,

- introduce vertices k_w, where $w \in [s(c_j)]$, and add gadgets $D(p^2, k_w, \Delta - 1)$, $Q(p^2, k_{s(c_j)})$, as well as $P(p^1, p^2, k_w)$ when $\chi_\mathrm{d} \geq 3$ for $w \in [s(j) - 1]$
- add edges between vertex k_w and vertices r_w, v_w^j, as well as $D(k_w, r_{w+1}, \Delta + 1)$,
- if variable x_i is involved in the constraint c_j and v_ℓ^j corresponds to an assignment where x_i has value $s \in Y$, where $s = (\Delta + 1) \cdot (k - 1) + \delta$, for $k \in [\chi_\mathrm{d}]$ and $\delta \in [0, \Delta]$, then add implication gadgets $I(v_\ell^j, a, p^1, p^k)$, $I(v_\ell^j, \chi_{i_1}, p^1, p^k)$ and exclusion gadgets $E(v_\ell^j, \chi_{i_2}, p^1, p^k)$, for $i_1 \in [\delta]$ and $i_2 \in [\delta + 1, \Delta]$, where vertices a and χ belong to $\hat{B}_{i,j}$.

Intuitively, the constraint gadget is set up in a way that forces, for some $\ell \in [s(c_j)]$, vertex v_ℓ^j to receive color 1, which in turn "activates" the implication and exclusion gadgets we have added to this vertex. This ensures that the assignment encoded by the variable gadgets agrees with the satisfying assignment of c_j represented by v_ℓ^j.

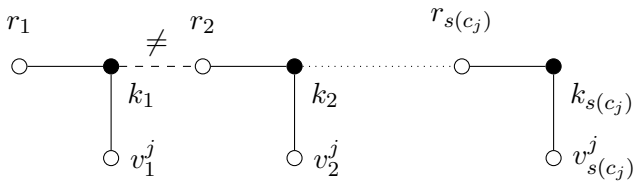


Figure 3.7.: Constraint gadget. For every black vertex k_w, there exists a gadget $D(p^2, k_w, \Delta - 1)$.

Let graph G_0 correspond to the graph containing all variable gadgets P_i as well as all the constraint gadgets $\hat{C}_j$, for $i \in [n]$ and $j \in [m]$. We refer to the block gadget $\hat{B}_{i,j}$, to the variable gadget P_i, and to the constraint gadget $\hat{C}_j$ of G_z as $\hat{B}_{i,j}^{G_z}$, $P_i^{G_z}$, and $\hat{C}_j^{G_z}$ respectively. To construct graph G, introduce $\kappa = n \cdot \Delta + 1$ copies $G_1, \ldots, G_\kappa$ of G_0, such that they are connected sequentially as follows: for $i \in [n]$ and $j \in [\kappa - 1]$, the vertex a' of $\hat{B}_{i,m}^{G_j}$ is the vertex a of $\hat{B}_{i,1}^{G_{j+1}}$. Let $\mathcal{P}^i$ denote the "path" resulting from $P_i^{G_1}, \ldots, P_i^{G_\kappa}$.

Lemma 3.26. *For any $\chi_\mathrm{d} \geq 2$, if ϕ is satisfiable, then G admits a $(\chi_\mathrm{d}, \Delta)$-coloring.*

Proof. Let $f \colon X \to Y$ denote an assignment which satisfies all the constraints $c_1, \ldots, c_m$. We will describe a $(\chi_\mathrm{d}, \Delta)$-coloring $c \colon V(G) \to [\chi_\mathrm{d}]$ of G.

Let $c(p^i) = c(p_l^i) = i$, for $i \in [\chi_\mathrm{d}]$ and $l \in [\Delta]$. Next, for the vertices of block gadget $\hat{B}_{i,j}^{G_z}$, where $z \in [\kappa]$, $i \in [n]$ and $j \in [m]$, if $f(x_i) = (\Delta + 1) \cdot (k - 1) + \delta$, for $k \in [\chi_\mathrm{d}]$ and $\delta \in [0, \Delta]$, then let

- $c(a) = c(\chi_{i_1}) = c(y_{i_2}) = k$, where $i_1 \in [\delta]$ and $i_2 \in [\delta + 1, \Delta]$,
- $c(b) = c(\chi_{i_1}) = c(y_{i_2}) = k'$, for some arbitrary $k' \in [\chi_\mathrm{d}] \setminus \{k\}$, where $i_1 \in [\delta + 1, \Delta]$ and $i_2 \in [\delta]$.

Regarding the constraint gadgets, let c_j be one of the constraints of ϕ. Let the $\Delta-1$ leaves attached to vertex k_w receive color 2, for $w \in [s(c_j)]$. Since f is a satisfying assignment, there exists at least one vertex among $v^j_1, \ldots, v^j_{s(j)}$ in $\hat{C}^{G_z}_j$, for $z \in [\kappa]$, mapped to a restriction of f. Let v^j_ℓ be one such vertex of minimum index. Then, let $c(v^j_\ell) = 1$, while any other vertex $v^j_{\ell'}$, with $\ell' \neq \ell$, receives color 2. Moreover, let k_w receive color 1 for $w < \ell$ and color 2 for $\ell \leq w \leq s(c_j)$. On the other hand, let r_w receive color 2 for $w \leq \ell$ and color 1 for $\ell < w \leq s(c_j)$.

Lastly, properly color the internal vertices of the equality / palette / difference / exclusion / implication gadgets using Lemmas 3.11, 3.12, 3.14, 3.16, and 3.18. To see that all gadgets are properly colored using these lemmas, observe that any vertex colored so far has at most Δ same-colored neighbors, while the following hold:

- $P = \{p^i : i \in [\chi_\mathrm{d}]\}$ consists of χ_d vertices, each receiving a distinct color, and for all $i \in [\chi_\mathrm{d}]$, $l \in [\Delta]$, $c(p^i) = c(p^i_l)$,
- in all block gadgets, $c(a) = c(b_2)$, $c(b_2) = c(a')$, $c(b_1) \neq c(b_2)$, and vertices χ_i, y_i for $i \in [\Delta]$ are colored either $c(b_1)$ or $c(b_2)$,
- for $z \in [\kappa]$ and $j \in [m]$, in constraint gadget $\hat{C}^{G_z}_j$ it holds that $c(r_1) = c(p^2)$, $c(k_{s(c_j)}) = c(p^2)$, $c(k_w) \neq c(r_{w+1})$ for $w \in [s(c_j) - 1]$, vertices k_w, r_w, v^j_w for $w \in [s(c_j)]$ are colored either $c(p^1)$ or $c(p^2)$ while the leaves attached to k_w all receive color $c(p^2)$, and lastly if $c(v^j_\ell) = c(p^1)$ and x_i denotes a variable appearing in c_j such that vertex v^j_ℓ corresponds to an assignment where x_i receives value $s = (\Delta+1)\cdot(k-1)+\delta$ for $k \in [\chi_\mathrm{d}]$ and $\delta \in [0, \Delta]$ (which is a restriction of assignment f), then $c(a) = c(\chi_{i_1}) = k$ and $c(\chi_{i_2}) \neq k$, for $i_1 \in [\delta]$ and $i_2 \in [\delta + 1, \Delta]$, where vertices $a, \chi_{i_1}, \chi_{i_2}$ belong to $\hat{B}^{G_z}_{i,j}$.

This concludes the proof. □

Lemma 3.27. *For any $\chi_\mathrm{d} \geq 2$, if G admits a $(\chi_\mathrm{d}, \Delta)$-coloring, then ϕ is satisfiable.*

Proof. Let $c\colon V(G) \to [\chi_\mathrm{d}]$ be a $(\chi_\mathrm{d}, \Delta)$-coloring of G. Due to the properties of the equality gadgets, it holds that $c(p^i) = c(p^i_l)$, for all $i \in [\chi_\mathrm{d}], l \in [\Delta]$. Since c is a $(\chi_\mathrm{d}, \Delta)$-coloring, it follows that $c(p^i) \neq c(p^j)$, for distinct $i, j \in [\chi_\mathrm{d}]$. Assume without loss of generality that $c(p^i) = i$, for $i \in [\chi_\mathrm{d}]$.

For G_z, consider a mapping between the coloring of vertices of $\hat{B}^{G_z}_{i,j}$ and the value of x_i for some assignment of the variables of ϕ. In particular, the coloring of vertex a as well as of $\delta \in [0, \Delta]$ vertices of $\{\chi_i : i \in [\Delta]\}$ with color $k \in [\chi_\mathrm{d}]$ is mapped to an assignment where x_i receives value $(\Delta + 1) \cdot (k - 1) + \delta$. We will say that an *inconsistency* occurs in a variable gadget $P^{G_z}_i$ if there exist block gadgets $\hat{B}^{G_z}_{i,j}$ and $\hat{B}^{G_z}_{i,j+1}$, such that the coloring of the vertices of each block gadget maps to different assignments of x_i. We say that G_z is *consistent* if no inconsistency occurs in its variable gadgets $P^{G_z}_i$, for every $i \in [n]$.

Notice that, for the vertices of the block gadget $\hat{B}^{G_z}_{i,j}$, it holds that vertices χ_{i_1}, y_{i_2}, for $i_1, i_2 \in [\Delta]$, are colored either $c(a)$ or $c(b_1)$, and since there are 2Δ such vertices in total, exactly half of them are colored with each color, since otherwise either b_1 or b_2 have more than Δ same colored neighbors. We will now prove the following claim.

▷ Claim 3.28. There exists $\pi \in [\kappa]$ such that G_π is consistent.

Proof of the claim. We will prove that every path $\mathcal{P}^i$ may induce at most Δ inconsistencies. In that case, since there are n such paths and $\kappa = n \cdot \Delta + 1$ copies of G_0, due to the pigeonhole principle there exists some G_π without any inconsistencies.

Consider a path $\mathcal{P}^i$ as well as a block gadget $\hat{B}_{i,j}^{G_z}$, for some $z \in [\kappa]$ and $j \in [m]$. Let $N(\hat{B}_{i,j}^{G_z})$ denote the block gadget right of $\hat{B}_{i,j}^{G_z}$, i.e. vertex a' of $\hat{B}_{i,j}^{G_z}$ coincides with vertex a of $N(\hat{B}_{i,j}^{G_z})$. Moreover, let $\hat{B}_{i,j'}^{G_{z'}}$, where either a) $z' = z$ and $j' > j$ or b) $z' > z$ and $j' \in [m]$, denote some block gadget which appears to the right of $\hat{B}_{i,j}^{G_z}$. For every block gadget, due to the properties of the equality gadget, it holds that $c(a) = c(a')$, therefore the color of vertex a is the same for all block gadgets belonging to the same path $\mathcal{P}^i$. For the vertices of $\hat{B}_{i,j}^{G_z}$, assume that exactly δ vertices χ_{i_1} are colored with color $c(a)$. We will prove that then, in every gadget $\hat{B}_{i,j'}^{G_{z'}}$, at most δ vertices χ_{i_1} are colored with color $c(a)$. For the base of the induction, notice that in $\hat{B}_{i,j}^{G_z}$, exactly $\Delta - \delta$ vertices y_{i_2} are colored with color $c(a)$. Thus, in $N(\hat{B}_{i,j}^{G_z})$, at most δ vertices χ_{i_1} receive color $c(a)$, since vertex a' of $\hat{B}_{i,j}^{G_z}$ coincides with vertex a of $N(\hat{B}_{i,j}^{G_z})$. Assume that this is the case for some gadget $\hat{B}_{i,j'}^{G_{z'}}$ to the right of $\hat{B}_{i,j}^{G_z}$. Then, since there are at least $\Delta - \delta$ vertices y_{i_2} of $\hat{B}_{i,j}^{G_z}$ receiving color $c(a)$, it follows that there are at most δ vertices χ_{i_1} of $N(\hat{B}_{i,j'}^{G_{z'}})$ receiving color $c(a)$. Consequently, it follows that every path can induce at most Δ inconsistencies, and since there is a total of n paths, there exists a copy G_π which is consistent. ◁

Let $f\colon X \to Y$ be an assignment such that $f(x_i) = (\Delta + 1) \cdot (k - 1) + \delta$, where, for the vertices of the block gadget $\hat{B}_{i,j}^{G_\pi}$, it holds that $c(a) = k \in [\chi_{\mathrm{d}}]$ and exactly $\delta \in [0, \Delta]$ vertices of $\{\chi_i : i \in [\Delta]\}$ are of color k.

It remains to prove that this is an assignment that satisfies all constraints. Consider the constraint gadget $\hat{C}_j^{G_\pi}$, where $j \in [m]$. We first prove that $c(v_\ell^j) = 1$, for some $\ell \in [s(j)]$. Assume that this is not the case. Then it follows that every vertex k_w has $\Delta + 1$ neighbors of color 2 (remember that due to $D(p^2, k_i, \Delta - 1)$, k_w has $\Delta - 1$ neighboring leaves of color $c(p^2)$), consequently $c(k_w) \neq 2$, for every $w \in [s(j)]$. However, due to $Q(p^2, k_{s(j)})$, it follows that $c(k_{s(j)}) = 2$, which is a contradiction. Let v_ℓ^j such that $c(v_\ell^j) = 1$. In that case, due to the implication and exclusion gadgets involving v_ℓ^j, it follows that, if variable x_i is involved in the constraint c_j and v_ℓ^j corresponds to an assignment where x_i has, for $k \in [\chi_{\mathrm{d}}]$ and $\delta \in [0, \Delta]$, value $(\Delta + 1) \cdot (k - 1) + \delta$, then in $\hat{B}_{i,j}^{G_\pi}$, vertex a as well as exactly δ vertices of $\{\chi_i : i \in [\Delta]\}$ have color k. In that case, the assignment corresponding to v_ℓ^j is a restriction of f, thus f satisfies constraint c_j. Since $j = 1, \ldots, m$ was arbitrary, this concludes the proof that f is a satisfying assignment for ϕ. □

Lemma 3.29. *It holds that* $\mathrm{pw}(G) \leq n + \mathcal{O}(1)$.

Proof. Due to Lemma 3.19, it holds that $\mathrm{pw}(G) = \mathrm{pw}(G' - P) + 3\chi_{\mathrm{d}}$, where G' is the graph we obtain from G by removing all the equality / palette / difference / exclusion / implication gadgets and add all edges between their endpoints which are not already connected. It therefore suffices to show that $\mathrm{pw}(G' - P) = n + \mathcal{O}(1)$.

We will do so by providing a mixed search strategy to clean $G' - P$ using at most this many searchers simultaneously. Since for the mixed search number ms it holds that $\mathrm{pw}(G'-P) \leq \mathrm{ms}(G'-P) \leq \mathrm{pw}(G'-P)+1$ [273], we will show that $\mathrm{ms}(G'-P) \leq n+5+B^q$ and the statement will follow.

Start with graph G_1. Place $s(c_1)+1$ searchers to the vertices r_1 and v_w^1 of $\hat{C}_1^{G_1}$, for $w \in [s(c_1)]$, as well as n searchers on vertices a of block gadgets $\hat{B}_{i,1}^{G_1}$, for $i \in [n]$. By moving the searcher placed on r_1 along the path formed by $k_1, r_2, k_2, \ldots$, all the edges of the constraint gadget can be cleaned. Next we will describe the procedure to clean $\hat{B}_{i,1}^{G_1}$. Move four extra searchers to vertices a', b_1, b_2, χ_1 of $\hat{B}_{i,1}^{G_1}$. Move the latter searcher to all other vertices χ_p and y_p, thus successfully cleaning $\hat{B}_{i,1}^{G_1}$. Lastly, remove all the searchers from $\hat{B}_{i,1}^{G_1}$ apart from the one present on vertex a'. Repeat the whole procedure for all $i \in [n]$.

In order to clean the rest of the graph, we first move the searchers from $\hat{C}_j^{G_z}$ to $\hat{C}_{j+1}^{G_z}$ if $j < m$ or to $\hat{C}_1^{G_{z+1}}$ alternatively (possibly introducing new searchers if required), clean the latter, and then proceed by cleaning the corresponding block gadgets. By repeating this procedure, in the end we clean all the edges of $G' - P$ by using at most $n + 5 + B^q = n + \mathcal{O}(1)$ searchers. □

Therefore, in polynomial time, we can construct a graph G, of pathwidth $\mathrm{pw}(G) \leq n + \mathcal{O}(1)$ due to Lemma 3.29, such that, due to Lemmas 3.26 and 3.27, deciding whether G admits a $(\chi_\mathrm{d}, \Delta)$-coloring is equivalent to deciding whether ϕ is satisfiable. In that case, assuming there exists a $\mathcal{O}^\star((\chi_\mathrm{d} \cdot (\Delta+1) - \varepsilon)^{\mathrm{pw}(G)})$ algorithm for Defective Coloring, then for $B = \chi_\mathrm{d} \cdot (\Delta+1)$, one could decide q-CSP-B in time $\mathcal{O}^\star((\chi_\mathrm{d} \cdot (\Delta+1) - \varepsilon)^{\mathrm{pw}(G)}) = \mathcal{O}^\star((B-\varepsilon)^{n+\mathcal{O}(1)}) = \mathcal{O}^\star((B-\varepsilon)^n)$, which contradicts the SETH due to Theorem 2.1. □

Algorithm

Here we present an algorithm for Defective Coloring parameterized by the treewidth of the input graph plus the target degree Δ. The algorithm uses standard techniques, and closely follows the approach previously sketched in the $(\chi_\mathrm{d}\Delta)^{\mathcal{O}(\mathrm{tw})} n^{\mathcal{O}(1)}$ algorithm of [25, 26]. The novelty is the use of a convolution technique presented in [95] in order to speed up the computation in the case of the join nodes.

Theorem 3.30. *Given an instance $\mathcal{I} = (G, \chi_\mathrm{d}, \Delta)$ of Defective Coloring, as well as a nice tree decomposition of G of width* tw*, there exists an algorithm that decides $\mathcal{I}$ in time $\mathcal{O}^\star((\chi_\mathrm{d} \cdot (\Delta+1))^{\mathrm{tw}})$.*

Proof. To avoid confusion, we will refer to the vertices of the nice tree decomposition as *nodes*, and to the vertices of G as vertices. Let $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ denote the nice tree decomposition of G, where by r we denote the root node. For a node t of T, let $X_t^\downarrow$ denote the union of all the bags present in the subtree rooted at t, including X_t. Moreover, let $s \colon V(T) \to [\mathrm{tw}+1]$ such that $s(t) = |X_t|$, i.e. $s(t)$ denotes the size of the bag X_t, and assume that $X_t = \{v_1^t, \ldots, v_{s(t)}^t\}$.

Assuming there exists a $(\chi_\mathrm{d}, \Delta)$-coloring of G, there are $\chi_\mathrm{d} \cdot (\Delta+1)$ different possibilities for every vertex $v \in V(G)$, since it belongs to one of the χ_d color classes with some

degree $\delta \in [0, \Delta]$ in the corresponding subgraph. Therefore, for each node t of T, we consider tuples $z_i^t = (w_1^{t,i}, \ldots, w_{|X_t|}^{t,i})$, where each $w_j^{t,i}$ is a pair $w_j^{t,i} = (c_j^{t,i}, \delta_j^{t,i})$ such that $c_j^{t,i} \in [\chi_\mathrm{d}]$ and $\delta_j^{t,i} \in [0, \Delta]$, used to encode this information for vertex v_j^t, for a total of $(\chi_\mathrm{d} \cdot (\Delta + 1))^{s(t)}$ tuples per node t.

In that case, for node t, let $S[t, z_i^t]$, where $i \in [(\chi_\mathrm{d} \cdot (\Delta + 1))^{s(t)}]$, denote the number of $(\chi_\mathrm{d}, \Delta)$-colorings of graph $G[X_t^\downarrow]$, where vertex v_j^t receives color $c_j^{t,i}$ and has exactly $\delta_j^{t,i}$ same-colored neighbors in $X_t^\downarrow \setminus X_t$. If $s(t) = 0$, then let $S[t, \varnothing]$ be equal to the $(\chi_\mathrm{d}, \Delta)$-colorings of graph $G[X_t^\downarrow]$. Then, in order to find the number of $(\chi_\mathrm{d}, \Delta)$-colorings of G, it suffices to compute $S[r, \varnothing]$.

Notice that each such tuple z_i^t induces χ_d sets of same colored vertices $V_c^{t,i} = \{v_j^t \in X_t : c_j^{t,i} = c\}$, for $c \in [\chi_\mathrm{d}]$. For a tuple z_i^t to be considered, it must hold that,

$$\forall v_j^t \in X_t, v_j^t \in V_c^{t,i} \implies |N(v_j^t) \cap V_c^{t,i}| + \delta_j^{t,i} \leq \Delta, \tag{3.1}$$

or in other words, that if some vertex has δ same colored neighboring vertices in the subgraph which do not belong to the bag, it should have at most $\Delta - \delta$ same colored neighbors inside the bag.

Leaf Node. If node t is a leaf, then $X_t = \{v_1^t\}$ and

$$S[t, (c_1^{t,i}, \delta_1^{t,i})] = \begin{cases} 1, & \text{if } \delta_1^{t,i} = 0, \\ 0, & \text{otherwise} \end{cases}$$

since no matter what color vertex v_1^t is assigned, $X_t^\downarrow \setminus X_t$ is empty, thus it cannot have any same colored neighbors.

Introduce Node. Suppose t is an introduce node with child node t_1 such that $X_t = X_{t_1} \cup \{v_{s(t)}^t\}$ for $v_{s(t)}^t \notin X_{t_1}$, where for $j \in [s(t_1)]$, vertices v_j^t and $v_j^{t_1}$ coincide. Consider a tuple $z_i^t = (w_1^{t,i}, \ldots, w_{s(t)}^{t,i})$ of node t. We will compute the value $S[t, z_i^t]$. First, we verify that Equation (3.1) is satisfied (if not we can put the value 0 as answer). Then,

$$S[t, z_i^t] = \begin{cases} S[t_1, z_{i_1}^{t_1}], & \text{if } \delta_{s(t)}^{t,i} = 0, \\ 0, & \text{otherwise,} \end{cases}$$

where $z_{i_1}^{t_1} = (w_1^{t_1,i_1}, \ldots, w_{s(t_1)}^{t_1,i_1})$ is the tuple of node t_1 where $c_j^{t,i} = c_j^{t_1,i_1}$ and $\delta_j^{t,i} = \delta_j^{t_1,i_1}$, for all $j \in [s(t_1)]$. Intuitively, vertex $v_{s(t)}^t$ cannot have any neighbors in $X_t^\downarrow \setminus X_t$, therefore the only valid value for $\delta_{s(t)}^{t,i}$ is 0.

Forget Node. Suppose t is a forget node with child node t_1 such that $X_t = X_{t_1} \setminus \{v_{s(t_1)}^{t_1}\}$ for $v_{s(t_1)}^{t_1} \in X_{t_1}$, where for $j \in [s(t)]$, vertices v_j^t and $v_j^{t_1}$ coincide. Consider a tuple

$z_i^t = (w_1^{t,i}, \ldots, w_{s(t)}^{t,i})$ of node t. We will compute the value $S[t, z_i^t]$. First, we verify that Equation (3.1) is satisfied (if not we can put the value 0 as answer). Then,

$$S[t, z_i^t] = \sum_{i_1 \in \mathcal{R}_1(i)} S[t_1, z_{i_1}^{t_1}],$$

where $i_1 \in \mathcal{R}_1(i)$ if, for all $j \in [s(t)]$,

- $c_j^{t,i} = c_j^{t_1,i_1}$ and
- if $v_{s(t_1)}^{t_1} \in N(v_j^{t_1})$ and $c_{s(t_1)}^{t_1,i_1} = c_j^{t_1,i_1}$, then $\delta_j^{t,i} = \delta_j^{t_1,i_1} + 1$, otherwise $\delta_j^{t,i} = \delta_j^{t_1,i_1}$.

In this case, we consider all possibilities regarding the forgotten vertex, taking into account how it affects the rest of the vertices; if it was a same-colored neighbor of another vertex of the bag, then the latter's same-colored neighbors in the subtree, excluding the bag, increased by one.

Join Node. Suppose t is a join node with children nodes t_1, t_2 such that $X_t = X_{t_1} = X_{t_2}$, where for $j \in [s(t)]$, vertices $v_j^t, v_j^{t_1}$ and $v_j^{t_2}$ coincide. Consider a tuple $z_i^t = (w_1^{t,i}, \ldots, w_{s(t)}^{t,i})$ of node t. We will compute the value $S[t, z_i^t]$. First, we verify that Equation (3.1) is satisfied (if not we can put the value 0 as answer). Then,

$$S[t, z_i^t] = \sum_{(i_1,i_2) \in \mathcal{R}_2(i)} S[t_1, z_{i_1}^{t_1}] \cdot S[t_2, z_{i_2}^{t_2}],$$

where $(i_1, i_2) \in \mathcal{R}_2(i)$ if, for all $j \in [s(t)]$,

- $c_j^{t,i} = c_j^{t_1,i_1} = c_j^{t_2,i_2}$ and
- $\delta_j^{t_1,i_1} + \delta_j^{t_2,i_2} = \delta_j^{t,i}$, where $0 \leq \delta_j^{t_1,i_1}, \delta_j^{t_2,i_2} \leq \delta_j^{t,i}$.

Intuitively, in a join node, for every vertex in the bag, we should take into account its same-colored neighbors in both of its children subtrees, excluding the vertices of the bag, as well as consider all possibilities regarding how these neighbors are partitioned in those subtrees.

Notice that table S has at most $(\chi_{\mathrm{d}} \cdot (\Delta + 1))^{\mathrm{tw}+1}$ cells. Moreover, by employing dynamic programming, we can fill all of its cells with a bottom-up approach. In order to check Equation (3.1), $n^{\mathcal{O}(1)}$ time is required. Then, each tuple of a leaf or introduce node can be computed in constant time, while each tuple of forget nodes in time $\chi_{\mathrm{d}} \cdot (\Delta + 1)$. However, in the case of join nodes, the time required per tuple is $\mathcal{O}((\Delta + 1)^{2\mathrm{tw}})$, since we need to take into account all possible values the degree of each vertex may have in each child node. In order to circumvent this, we employ a technique based on FFT introduced in [95]. This allows us to compute, for a given join node, the values of the table for all of its $\mathcal{O}((\chi_{\mathrm{d}} \cdot (\Delta + 1))^{\mathrm{tw}})$ tuples in time $\mathcal{O}^\star((\chi_{\mathrm{d}} \cdot (\Delta + 1))^{\mathrm{tw}})$.

Faster Join Computations. Let t be a join node with children nodes t_1 and t_2. First, we fix a coloring on the vertices of X_t, thus choosing among the $\chi_{\mathrm{d}}^{s(t)} = \mathcal{O}(\chi_{\mathrm{d}}^{\mathrm{tw}})$ different choices. We will describe how to compute $S[t, z_i^t]$ for any tuple z_i^t respecting said coloring. In the following, for every z_i^t considered, we assume that it respects this coloring.

Let, for tuple z_i^t, $\Sigma(z_i^t) = \sum_{j=1}^{s(t)} \delta_j^{t,i}$ denote its *sum*. Since $\delta_j^{t,i} \in [0, \Delta]$ for all $j \in [s(t)]$, it follows that $\Sigma(z_i^t) \in [0, s(t) \cdot \Delta]$. For every tuple $z_i^{t_p}$ of t_1 and t_2, where $p \in \{1, 2\}$, we will construct an identifier $i(z_i^{t_p})$ as follows:

$$i(z_i^{t_p}) = \sum_{j=1}^{s(t)} (\Delta+1)^{j-1} \cdot \delta_j^{t_p,i} \leq \Delta \cdot \sum_{j=1}^{s(t)} (\Delta+1)^{j-1} = \Delta \cdot \frac{1-(\Delta+1)^{s(t)}}{1-(\Delta+1)} = (\Delta+1)^{s(t)} - 1.$$

Next, we introduce polynomials $P_{\Sigma_q}^{t_p}(x)$, where $p \in \{1, 2\}$ and $\Sigma_q \in [0, s(t) \cdot \Delta]$, such that $P_{\Sigma_q}^{t_p}(x)$ is comprised of monomials $S[t_p, z_i^{t_p}] \cdot x^{i(z_i^{t_p})}$, for every tuple $z_i^{t_p}$ such that $\Sigma(z_i^{t_p}) = \Sigma_q$. Subsequently, by using FFT, we compute the polynomial $P_{\Sigma_1}^{t_1} \cdot P_{\Sigma_2}^{t_2}$, for all $\Sigma_1, \Sigma_2 \in [0, s(t) \cdot \Delta]$. Since the multiplication of two polynomials of degree n requires $\mathcal{O}(n \log n)$ time [243], and we perform $(s(t) \cdot \Delta + 1)^2 = \mathcal{O}(n^2)$ such multiplications on polynomials of degree at most $(\Delta+1)^{s(t)} - 1$, it follows that in total $\mathcal{O}(n^2 \cdot (\Delta+1)^{s(t)} \cdot s(t) \log(\Delta+1)) = \mathcal{O}^\star((\Delta+1)^{s(t)})$ time is required.

▷ Claim 3.31. For $z_{i_1}^{t_1}, z_{i_2}^{t_2}$, let $i(z_{i_1}^{t_1}) + i(z_{i_2}^{t_2}) = \sum_{j=1}^{s(t)+1} a_j \cdot (\Delta+1)^{j-1}$, where $0 \leq a_j \leq \Delta$. Then, the following are equivalent:

- $\Sigma(z_{i_1}^{t_1}) + \Sigma(z_{i_2}^{t_2}) = \sum_{j=1}^{s(t)} a_j$,
- for all $j \in [s(t)]$, $\delta_j^{t_1,i_1} + \delta_j^{t_2,i_2} \leq \Delta$.

Proof of the claim. Express $i(z_{i_1}^{t_1})$, $i(z_{i_2}^{t_2})$, as well as $\Sigma(z_{i_1}^{t_1}) + \Sigma(z_{i_2}^{t_2})$ as numbers d^1, d^2 and a respectively in the $(\Delta+1)$-ary system, where each of their digits d_j^1, d_j^2, a_j is a number between 0 and Δ. Then, $\delta_j^{t_1,i_1}$, $\delta_j^{t_2,i_2}$, and a_j correspond to the j-th digit of d^1, d^2, and a respectively.

Now, assume that we add the numbers d^1 and d^2, and let $s_j = d_j^1 + d_j^2$ if $d_j^1 + d_j^2 \leq \Delta$, while $s_j = 1 + d_j^1 + d_j^2 - (\Delta+1) < d_j^1 + d_j^2$ otherwise, where we assumed that $\Delta > 0$. Notice that it holds that $\sum_{j=1}^{s(t)} s_j \geq \sum_{j=1}^{s(t)} a_j$. In that case, assuming $\Sigma(z_{i_1}^{t_1}) + \Sigma(z_{i_2}^{t_2}) = \sum_{j=1}^{s(t)} a_j$ implies that $\sum_{j=1}^{s(t)} d_j^1 + \sum_{j=1}^{s(t)} d_j^2 \leq \sum_{j=1}^{s(t)} s_j$, which in turn implies that $s_j = d_j^1 + d_j^2 \implies d_j^1 + d_j^2 \leq \Delta$ for all j.

On the other hand, if $d_j^1 + d_j^2 \leq \Delta$ for every j, it follows that $a_j = d_j^1 + d_j^2$, and thus $\sum_{j=1}^{s(t)} d_j^1 + \sum_{j=1}^{s(t)} d_j^2 = \sum_{j=1}^{s(t)} a_j \implies \Sigma(z_{i_1}^{t_1}) + \Sigma(z_{i_2}^{t_2}) = \sum_{j=1}^{s(t)} a_j$. ◁

As a consequence of Claim 3.31, we can easily distinguish whether a monomial of $P_{\Sigma_{q_1}}^{t_1} \cdot P_{\Sigma_{q_2}}^{t_2}$ corresponds to a tuple of t occurring from the addition of tuples of t_1 and t_2 of sum Σ_{q_1} and Σ_{q_2} respectively. Moreover, any such tuple of t is encoded by some monomial, whose coefficient dictates the number of different ways this tuple can occur

from pairs of tuples of such sums. Therefore, in order to identify the number of ways a tuple of t may occur, it suffices to add all the coefficients of the corresponding monomial in all $\mathcal{O}(n^2)$ polynomial multiplications performed.

Lastly, in order to compute the value of S for the rest of the tuples of t, it suffices to repeat the whole procedure for all different colorings of X_t, thus resulting in $\chi_{\mathrm{d}}^{s(t)}$ iterations.

In the end, in order to compute the value of S for all $\mathcal{O}((\chi_{\mathrm{d}}(\Delta+1))^{\mathrm{tw}})$ tuples of t, we need $\mathcal{O}^\star(\chi_{\mathrm{d}}^{\mathrm{tw}} \cdot (\Delta+1)^{\mathrm{tw}})$ time.

Complexity. For the final complexity of the algorithm, notice that, for a node t, it holds that in order to compute the value of S for all of its $\mathcal{O}((\chi_{\mathrm{d}} \cdot (\Delta+1))^{\mathrm{tw}})$ tuples, we require time:

- $\mathcal{O}(1)$ per tuple, if t is a leaf or an introduction node,
- $\mathcal{O}(\chi_{\mathrm{d}}(\Delta+1))$ per tuple, if t is a forget node,
- $\mathcal{O}^\star((\chi_{\mathrm{d}}(\Delta+1))^{\mathrm{tw}})$ for all tuples if t is a join node.

Therefore, the total running time of the algorithm is upper bounded by $\mathcal{O}^\star((\chi_{\mathrm{d}}(\Delta+1))^{\mathrm{tw}})$. □

3.3. Tree-depth Lower Bounds

In this section we present tight lower bounds on the complexity of solving Bounded Degree Vertex Deletion and Defective Coloring, when parameterized by the tree-depth of the input graph. As in previous reductions, we start from a k-Multicolored Clique instance $G = (V, E)$, where the vertices are partitioned into k sets V_i, for $i \in [k]$. Our main technical contribution is a recursive construction which allows us to keep the tree-depth of the constructed graph linear with respect to k, thereby tightening previously known lower bounds. In the following we provide a high level sketch of the new ingredients of our construction. For an illustration we refer to Figure 3.8.

For every set V_i we design a simple choice gadget $\hat{C}_i$ which encodes the choice of a vertex in V_i. We also design a simple "copy" gadget, which, using a constant number of extra vertices per copy, allows us to produce multiple choice gadgets which encode the same value. In previous reductions ([25, 145]), we would now construct for each of the k main choice gadgets at most $k-1$ copies, and then for each $i_1, i_2 \in [k]$ we would select a distinct pair of copies of $\hat{C}_{i_1}, \hat{C}_{i_2}$ and add some machinery on these copies to verify that the choices for these groups encode the endpoints of an edge. This approach naturally leads to a graph with tree-depth $\mathcal{O}(k^2)$, and as a matter of fact it establishes hardness even for more restrictive parameters: as [145] points out, if we remove the $\mathcal{O}(k^2)$ vertices that ensure that the copies of the choice gadgets encode the same values, we obtain a collection of graphs of constant tree-depth, i.e. the parameter is in fact modulator size to constant tree-depth, rather than tree-depth itself.

The new ingredient in our approach is to observe that if we allow our reduction to use the full power of tree-depth as a parameter, we can avoid the quadratic blow-up in this construction. Consider the slightly more general problem, where we have two intervals $I_1, I_2 \subseteq [k]$ and we want to construct some machinery that checks if for each $i_1 \in I_1$ and $i_2 \in I_2$ our choices for V_{i_1}, V_{i_2} are valid, that is, encode the endpoints of an edge. On a high level, this is the problem we want to solve for $I_1 = I_2 = [k]$, and suppose we have some base gadget for the case $|I_1| = |I_2| = 1$. We now observe that one way to solve the general case is recursive: we cut the two intervals in half, say $I_1 = I_1^L \cup I_1^H$ and $I_2 = I_2^L \cup I_2^H$, and then check the same condition for each pair in $(I_1^L, I_2^L), (I_1^L, I_2^H), (I_1^H, I_2^L), (I_1^H, I_2^H)$. To this end, we make two copies of each choice gadget, thus constructing $\mathcal{O}(k)$ new separator vertices, but reducing to four instances of the same problem where all interval sizes have been cut in half. As a result, to calculate the tree-depth of such a construction we get a recurrence of the form $T(k) \leq \mathcal{O}(k) + T(k/2)$, which in the end gives tree-depth $\mathcal{O}(k)$. Observe that this technique manages to produce better results than previous reductions exactly because we are exploiting the full power of tree-depth: we construct an instance that has ck vertices whose removal, rather than breaking the graph down into trivial components, gives components which (recursively) have $ck/2$ vertices whose removal produces even simpler components, and so on, through a recursion depth of height $\log k$. In other words, unlike previous reductions, we crucially rely on the recursive definition of tree-depth.

3.3.1. Bounded Degree Vertex Deletion

Theorem 3.32. *For any computable function f, if there exists an algorithm that solves* Bounded Degree Vertex Deletion *in time $f(\mathrm{td})n^{o(\mathrm{td})}$, where* td *denotes the tree-depth of the input graph, then the ETH is false.*

Proof. Let (G, k) be an instance of k-Multicolored Clique, such that every vertex of G has a self loop, i.e. $\{v, v\} \in E(G)$ for all $v \in V(G)$. Recall that we assume that G is given to us partitioned into k independent sets $V_1, \ldots, V_k$, where $V_i = \{v_1^i, \ldots, v_n^i\}$. Assume without loss of generality that $k = 2^z$ for some $z \in \mathbb{N}$ (one can do so by adding dummy independent sets connected to all the other vertices of the graph). Moreover, let $E^{i_1,i_2} \subseteq E(G)$ denote the edges of G with one endpoint in V_{i_1} and the other in V_{i_2}. Set $\Delta = n^3$. We will construct in polynomial time a graph H of tree-depth $\mathrm{td}(H) = \mathcal{O}(k)$ and size $|V(H)| = k^{\mathcal{O}(1)} \cdot n^{\mathcal{O}(1)}$, such that there exists $S \subseteq V(H)$, $|S| \leq k'$, and $H - S$ has maximum degree at most Δ, for some k', if and only if G has a k-clique.

Choice Gadget. For an independent set V_i, we construct the *choice gadget* $\hat{C}_i$ as depicted in Figure 3.9a. We first construct independent sets $\hat{C}_i^p = \{v_1^{i,p}, \ldots, v_n^{i,p}\}$, where $p \in \{h, l\}$. Afterwards, we connect $v_j^{i,h}$ and $v_j^{i,l}$ with a vertex q_j^i, and add to the latter $\Delta - 1$ leaves. Intuitively, we will consider a one-to-one mapping between the vertex v_j^i of V_i belonging to a supposed k-clique of G and the deletion of exactly j vertices of $\hat{C}_i^l$ and $n - j$ vertices of $\hat{C}_i^h$.

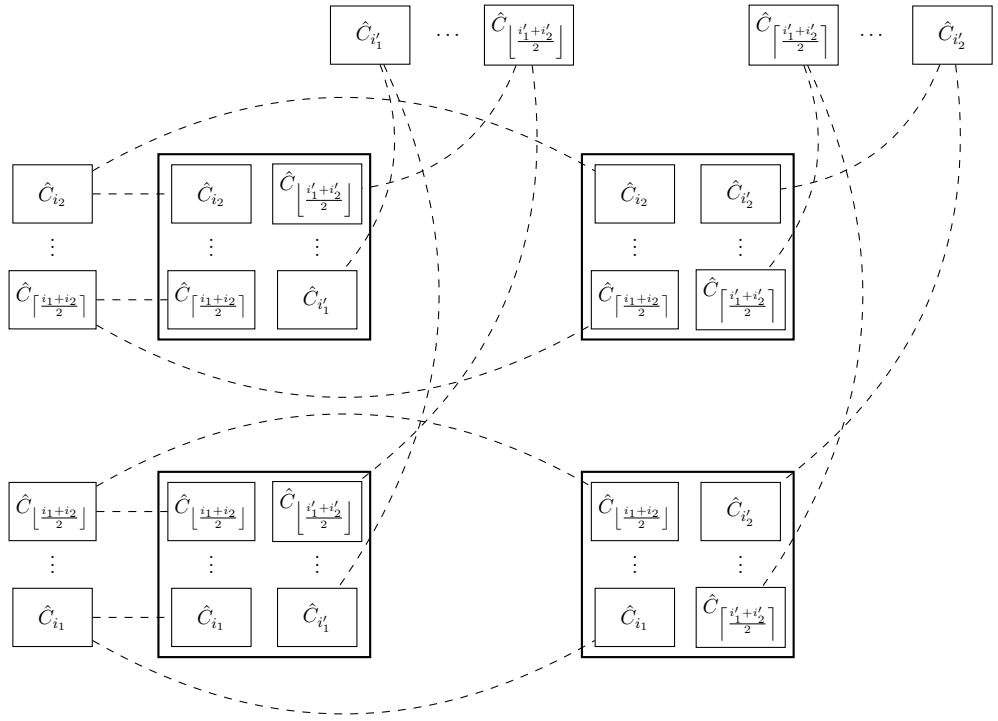


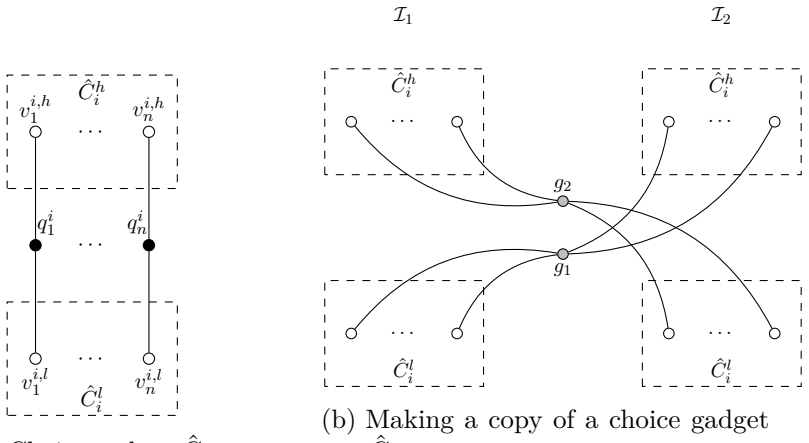
Figure 3.8.: Illustration where $I_1 = [i_1, i_2]$ and $I_2 = [i'_1, i'_2]$. Dashed lines denote copies, while the rectangles denote the reduced instances.

Copy Gadget. Given two instances $\mathcal{I}_1$, $\mathcal{I}_2$ of a choice gadget $\hat{C}_i$, when we say that we connect them with a *copy gadget*, we introduce two vertices g_1 and g_2, attach to each of those $\Delta - n$ leaves, and lastly add an edge between g_1 (respectively, g_2) and the vertices of $\hat{C}_i^l$ of instance $\mathcal{I}_1$ (respectively, $\mathcal{I}_2$), as well as the vertices of $\hat{C}_i^h$ of instance $\mathcal{I}_2$ (respectively, $\mathcal{I}_1$), as depicted in Figure 3.9b.

Edge Gadget. Let $e = \{v_{j_1}^{i_1}, v_{j_2}^{i_2}\} \in E^{i_1,i_2}$ be an edge of G. Construct the *edge gadget* $\hat{E}_e$ as depicted in Figure 3.10, where every vertex c_j^i has Δ leaves attached.

Adjacency Gadget. For $i_1 \leq i_2$ and $i'_1 \leq i'_2$, we define the *adjacency gadget* $\hat{A}(i_1, i_2, i'_1, i'_2)$ as follows:

- Consider first the case when $i_1 = i_2$ and $i'_1 = i'_2$. Let the adjacency gadget contain instances of the edge gadgets $\hat{E}_e$, for $e \in E^{i_1,i'_1}$, the choice gadgets $\hat{C}_{i_1}$ and $\hat{C}_{i'_1}$, as well as vertices $\ell^l_{i_1,i'_1}$, $\ell^h_{i_1,i'_1}$, $r^l_{i_1,i'_1}$ and $r^h_{i_1,i'_1}$. Add edges between
  - $\ell^l_{i_1,i'_1}$ and $\hat{C}^l_{i_1}$,
  - $\ell^h_{i_1,i'_1}$ and $\hat{C}^h_{i_1}$,
  - $r^l_{i_1,i'_1}$ and $\hat{C}^l_{i'_1}$,
  - $r^h_{i_1,i'_1}$ and $\hat{C}^h_{i'_1}$.



(a) Choice gadget $\hat{C}_i$.

(b) Making a copy of a choice gadget $\hat{C}_i$.

Figure 3.9.: Black vertices have $\Delta - 1$ and gray vertices $\Delta - n$ leaves attached.

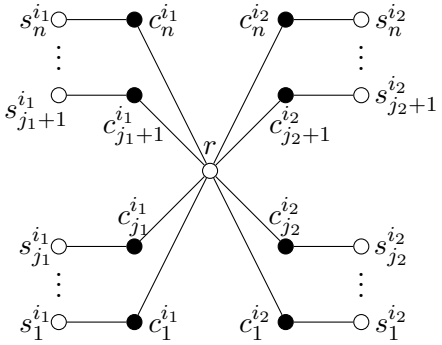


Figure 3.10.: Edge gadget $\hat{E}_e$ for $e = \{v_{j_1}^{i_1}, v_{j_2}^{i_2}\}$. Black vertices have Δ leaves attached.

If $e = \{v_{j_1}^{i_1}, v_{j_2}^{i'_1}\} \in E^{i_1,i'_1}$, then add the following edges adjacent to $\hat{E}_e$:

- $\ell^l_{i_1,i'_1}$ with $s^{i_1}_\kappa$, for $\kappa \in [j_1]$,
- $r^l_{i_1,i'_1}$ with $s^{i'_1}_\kappa$, for $\kappa \in [j_2]$,
- $\ell^h_{i_1,i'_1}$ with $s^{i_1}_\kappa$, for $\kappa \in [j_1+1, n]$,
- $r^h_{i_1,i'_1}$ with $s^{i'_1}_\kappa$, for $\kappa \in [j_2+1, n]$.

Let $\tau(x)$, where $x \in \{\ell^l_{i_1,i'_1}, \ell^h_{i_1,i'_1}, r^l_{i_1,i'_1}, r^h_{i_1,i'_1}\}$, denote the number of neighbors of x belonging to some edge gadget. Attach $\Delta - \tau(x)$ leaves to vertex x. For an illustration see Figure 3.11.

- Now consider the case when $i_1 < i_2$ and $i'_1 < i'_2$. Then, let $\hat{A}(i_1, i_2, i'_1, i'_2)$ contain choice gadgets $\hat{C}_i$ and $\hat{C}_{i'}$, where $i \in [i_1, i_2]$ and $i' \in [i'_1, i'_2]$, which we will refer to as the *original choice gadgets* of $\hat{A}(i_1, i_2, i'_1, i'_2)$, as well as the adjacency gadgets

– $\hat{A}\big(i_1, \lfloor \frac{i_1+i_2}{2} \rfloor, i'_1, \lfloor \frac{i'_1+i'_2}{2} \rfloor\big)$,
– $\hat{A}\big(i_1, \lfloor \frac{i_1+i_2}{2} \rfloor, \lceil \frac{i'_1+i'_2}{2} \rceil, i'_2\big)$,
– $\hat{A}\big(\lceil \frac{i_1+i_2}{2} \rceil, i_2, i'_1, \lfloor \frac{i'_1+i'_2}{2} \rfloor\big)$,
– $\hat{A}\big(\lceil \frac{i_1+i_2}{2} \rceil, i_2, \lceil \frac{i'_1+i'_2}{2} \rceil, i'_2\big)$.

Lastly, we connect with a copy gadget any choice gadgets $\hat{C}_i$ and $\hat{C}_{i'}$ appearing in said adjacency gadgets, with the corresponding original choice gadget $\hat{C}_i$ and $\hat{C}_{i'}$. Notice that then, every original choice gadget is taking part in two copy gadgets. For a high level illustration see Figure 3.8.

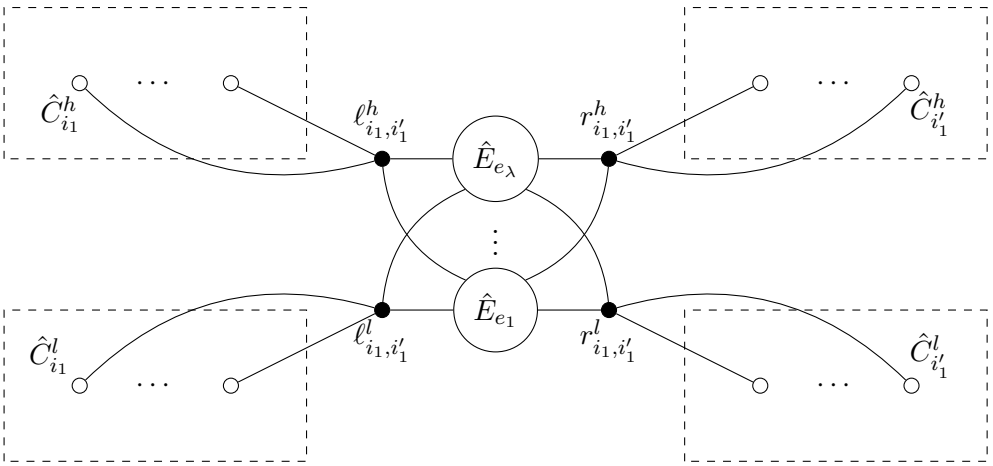


Figure 3.11.: Adjacency gadget $\hat{A}(i_1, i_1, i'_1, i'_1)$, where $E^{i_1,i'_1} = \{e_i : i \in [\lambda]\}$. Black vertices have leaves attached.

Let graph H be the adjacency gadget $\hat{A}(1, k, 1, k)$. Notice that it holds that $|V(H)| = (n \cdot k)^{\mathcal{O}(1)}$. Let $\beta = 2k(2k-1)$, and set $k' = 2(|E(G)| - kn) \cdot 2n + kn \cdot 2n + 2\binom{k}{2} + k + n \cdot \beta$.

Lemma 3.33. *H has the following properties:*

- *The number of instances of choice gadgets present in H is β,*
- *The number of instances of edge gadget $\hat{E}_e$ present in H, where $e = \{v^{i_1}_{j_1}, v^{i_2}_{j_2}\} \in E(G)$, is one if $i_1 = i_2$, and two otherwise.*

Proof. For the first statement, notice that the number of instances of choice gadgets is given by the recursive formula $T(k) = 2k + 4T(k/2)$, where $T(1) = 2$. In that case, it follows that

$$T(k) = \sum_{i=0}^{\log k} \left(4^i \cdot 2 \cdot \frac{k}{2^i}\right) = 2k \sum_{i=0}^{\log k} 2^i = 2k(2k-1) = \beta.$$

For the second statement, first we will prove that for every adjacency gadget $\hat{A}(i_1, i_2, i'_1, i'_2)$ appearing in H, it holds that $i_2 - i_1 = i'_2 - i'_1 = 2^c - 1$, for some $c \in \mathbb{N}$. The statement holds for $\hat{A}(1, k, 1, k)$, as well as when $i_2 - i_1 = i'_2 - i'_1 = 0$. Suppose that it holds for some $\hat{A}(i_1, i_2, i'_1, i'_2)$, i.e. $i_2 - i_1 = i'_2 - i'_1 = 2^c - 1 > 0$, for some $c \in \mathbb{N}$. Then, it follows

that $\left\lfloor \frac{i_1+i_2}{2} \right\rfloor - i_1 = \lfloor i_2 - 2^{c-1} + 0.5 \rfloor - i_1 = i_2 - i_1 - 2^{c-1} = 2^{c-1} - 1$. Moreover, it follows that $i_2 - \left\lceil \frac{i_1+i_2}{2} \right\rceil = i_2 - \lceil i_1 + 2^{c-1} - 0.5 \rceil = i_2 - (i_1 + 2^{c-1}) = 2^{c-1} - 1$. Therefore, the stated property holds.

In that case, for some $\hat{A}(i_1, i_2, i'_1, i'_2)$, in every step of the recursion, intervals $[i_1, i_2]$ and $[i'_1, i'_2]$ are partitioned in the middle, and an adjacency gadget is considered for each of the four combinations. In that case, starting from $\hat{A}(1, k, 1, k)$, there is a single way to produce every adjacency gadget $\hat{A}(i_1, i_1, i_2, i_2)$, where $i_1, i_2 \in [k]$. Consider an edge gadget $\hat{E}_e$, where $e = \{v^{i_1}_{j_1}, v^{i_2}_{j_2}\} \in E(G)$. There are two cases:

- if $i_1 = i_2$, then this gadget appears only in the adjacency gadget $\hat{A}(i_1, i_1, i_1, i_1)$,
- alternatively, it appears in both adjacency gadgets $\hat{A}(i_1, i_1, i_2, i_2)$ and $\hat{A}(i_2, i_2, i_1, i_1)$.

This concludes the proof. □

Lemma 3.34. *It holds that* $\mathrm{td}(H) = \mathcal{O}(k)$.

Proof. Let $T(\kappa)$ denote the tree-depth of $\hat{A}(i_1, i_2, i'_1, i'_2)$ in the case when $i_2 - i_1 = i'_2 - i'_1 = \kappa$.

First, notice that, for $i_1, i_2 \in [k]$, the tree-depth of $\hat{A}(i_1, i_1, i_2, i_2)$ is less than 8: remove vertices $\ell^l_{i_1,i_2}$, $\ell^h_{i_1,i_2}$, $r^l_{i_1,i_2}$ and $r^h_{i_1,i_2}$, and all remaining connected components are trees of height at most 3. Consequently, $T(1) \leq 8$.

Now, consider the adjacency gadget $\hat{A}(i_1, i_2, i'_1, i'_2)$, where $i_2 - i_1 = i'_2 - i'_1 = \kappa$. This is comprised of adjacency gadgets

- $\hat{A}\Big(i_1, \left\lfloor \frac{i_1+i_2}{2} \right\rfloor, i'_1, \left\lfloor \frac{i'_1+i'_2}{2} \right\rfloor\Big)$,
- $\hat{A}\Big(\left\lceil \frac{i_1+i_2}{2} \right\rceil, i_2, i'_1, \left\lfloor \frac{i'_1+i'_2}{2} \right\rfloor\Big)$,
- $\hat{A}\Big(i_1, \left\lfloor \frac{i_1+i_2}{2} \right\rfloor, \left\lceil \frac{i'_1+i'_2}{2} \right\rceil, i'_2\Big)$,
- $\hat{A}\Big(\left\lceil \frac{i_1+i_2}{2} \right\rceil, i_2, \left\lceil \frac{i'_1+i'_2}{2} \right\rceil, i'_2\Big)$,

as well as of exactly 2κ original choice gadgets, each of which is connected with two copy gadgets to other instances of choice gadgets present in the adjacency gadgets. By removing all vertices g_1 and g_2 of the copy gadgets (see Figure 3.9b), all the original choice gadgets as well as the adjacency gadgets are disconnected. Therefore, it holds that $T(\kappa) \leq 8\kappa + T(\kappa/2)$, thus, it follows that $T(k) \leq 8 \sum_{i=0}^{\log k} \frac{k}{2^i} = \mathcal{O}(k)$. □

Lemma 3.35. *If G contains a k-clique, then there exists $S \subseteq V(H)$, with $|S| \leq k'$, such that $H - S$ has maximum degree at most Δ.*

Proof. Let $\mathcal{V} \subseteq V(G)$ be a k-clique of G, consisting of vertices $v^i_{s(i)} \in V_i$, for $i \in [k]$. We will construct a deletion set $S \subseteq V(H)$ as follows:

- Let S contain vertices $v^{i,h}_{j_1}$ and $v^{i,l}_{j_2}$, for $j_1 \in [s(i)]$ and $j_2 \in [s(i)+1, n]$, from every instance of the choice gadget $\hat{C}_i$.

- Let $\hat{E}_e$ be the edge gadget of edge e, where $e = \{v_{j_1}^{i_1}, v_{j_2}^{i_2}\}$. If $v_{j_1}^{i_1}, v_{j_2}^{i_2} \in \mathcal{V}$, then let S include from $\hat{E}_e$ the vertices r, $s_j^{i_1}$ and $s_j^{i_2}$, where $j \in [n]$. Alternatively, let S include from $\hat{E}_e$ all the vertices $c_j^{i_1}$ and $c_j^{i_2}$, where $j \in [n]$.

The edges for which the first case holds are exactly $\binom{k}{2} + k$. Due to Lemma 3.33, it holds that $\hat{E}_e$ appears once in H if e is a self loop and twice if not, while exactly β instances of choice gadgets are present in H. In the end, S contains $2(|E(G)|-kn)\cdot 2n+kn\cdot 2n+2\binom{k}{2}+k$ vertices due to the edge gadgets, plus $\beta \cdot n$ vertices due to the choice gadgets, thus $|S| = k'$ follows.

It remains to prove that $H - S$ has maximum degree at most Δ. One can easily verify that every vertex g_1 and g_2 of a copy gadget has degree exactly Δ. Moreover, every vertex q_j^i in the instances of choice gadgets has exactly one neighbor in S. For every vertex c_j^i in an edge gadget, either itself or its neighboring vertices r and s_j^i belong to S. Lastly, for $i_1, i_2 \in [k]$, let $v_{s(i_1)}^{i_1}, v_{s(i_2)}^{i_2} \in \mathcal{V}$, where e denotes the edge connecting them. Then, for ℓ_{i_1,i_2}^l, it holds that it has exactly $\Delta - \tau(\ell_{i_1,i_2}^l) + v_{s(i_1)}^{i_1} + \tau(\ell_{i_1,i_2}^l) - v_{s(i_1)}^{i_1} = \Delta$ neighbors in $H - S$, due to its leaves, its neighbors in $\hat{C}_{i_1}^l$ and its neighbors in all the edge gadgets. In an analogous way, one can show that ℓ_{i_1,i_2}^h, r_{i_1,i_2}^l and r_{i_1,i_2}^h all have degree Δ in $H - S$. □

Lemma 3.36. *If there exists $S \subseteq V(H)$, with $|S| \leq k'$, such that $H - S$ has maximum degree at most Δ, then G contains a k-clique.*

Proof. Let $S \subseteq V(H)$, with $|S| \leq k'$, such that $H - S$ has maximum degree at most Δ. Due to Lemma 3.33, it holds that H contains exactly β instances of choice gadgets, while the edge gadget $\hat{E}_e$, where $e = \{v_{j_1}^{i_1}, v_{j_2}^{i_2}\}$, appears once if $i_1 = i_2$ and twice otherwise. Notice that S must contain at least n vertices per choice gadget, since every vertex q_j^i has degree $\Delta + 1$, while no two share any neighbors, for a total of at least $n \cdot \beta$ vertices. Moreover, S contains at least $2n$ vertices per edge gadget, since vertices c_j^i are of degree $\Delta + 2$ and share only a single neighbor. Notice that there are $2(|E(G)| - kn) + kn$ instances of edge gadgets, for a total of $2(|E(G)| - kn) \cdot 2n + kn \cdot 2n$ vertices.

▷ Claim 3.37. There exists a single edge gadget in $\hat{A}(i_1, i_1, i_2, i_2)$ from which S contains exactly $2n + 1$ vertices, for every $i_1, i_2 \in [k]$.

Proof of the claim. Since S contains at least n and $2n$ vertices per choice and edge gadget respectively, it follows that we are left with an additional budget of at most $2\binom{k}{2} + k$. The number of adjacency gadgets $\hat{A}(i_1, i_1, i_2, i_2)$ is $2\binom{k}{2}+k$, due to Lemma 3.33. Consequently, we will prove for every such adjacency gadget, S contains $2n + 2n \cdot |E^{i_1,i_2}| + 1$ (remember that each such gadget has 2 choice gadgets as well as $|E^{i_1,i_2}|$ edge gadgets). In fact, we will prove that for each such adjacency gadget, there exists a single edge gadget from which S contains $2n + 1$ vertices.

First, we will prove that S contains exactly $2n + 2n \cdot |E^{i_1,i_2}| + 1$ vertices per adjacency gadget $\hat{A}(i_1, i_1, i_2, i_2)$. If it contains less, then it necessarily holds that it contains n vertices from each of the two choice gadgets, as well as $2n|E^{i_1,i_2}|$ vertices c_j^i from all the

edge gadgets of the adjacency gadget. In that case, both vertices $\ell^l_{i_1,i_2}$ and $\ell^h_{i_1,i_2}$ must have no neighbors from $\hat{C}^l_{i_1}$ and $\hat{C}^h_{i_1}$ respectively, which cannot be the case since only n vertices of $\hat{C}_{i_1}$ belong to S. On the other hand, if S contains more than $2n + 2n \cdot |E^{i_1,i_2}| + 1$ vertices from some adjacency gadget $\hat{A}(i_1, i_1, i_2, i_2)$, then it follows that there exists another adjacency gadget $\hat{A}(i'_1, i'_1, i'_2, i'_2)$ from which it contains less than that many vertices, contradiction.

Now, suppose that for some adjacency gadget $\hat{A}(i_1, i_1, i_2, i_2)$, there is no edge gadget from which S contains $2n + 1$ vertices. In that case, from every edge gadget, all the vertices c^i_j belong to S, and consequently, for every vertex $\ell^l_{i_1,i_2}$, $\ell^h_{i_1,i_2}$, $r^l_{i_1,i_2}$ and $r^h_{i_1,i_2}$, it holds that either all of its neighbors in the choice gadgets belong to S or that it itself does. Since at most one of those vertices may belong to S, this leads to a contradiction. As a consequence of the above, it follows that S contains exactly n vertices per choice gadget of $\hat{A}(i_1, i_1, i_2, i_2)$, as well as all vertices c^i_j from all but one edge gadgets present. For the extra edge gadget, we can assume that S contains vertex r as well as all the vertices s^i_j (if that is not the case, there exists some deletion set S' of same cardinality for which this holds, since only vertices s^i_j have an edge with vertices outside of the edge gadget). ◁

Consequently, S contains exactly n vertices per choice gadget, as well as $2n \cdot |E^{i_1,i_2}| + 1$ vertices from the edge gadgets of the adjacency gadget $\hat{A}(i_1, i_1, i_2, i_2)$. Notice that no vertex g_1 and g_2 of a copy gadget belongs to S, and both have at most n neighbors in $H - S$ from the corresponding parts of the choice gadgets (see Figure 3.9b). In that case, it follows that only vertices $v^{i,l}_j$ and $v^{i,h}_j$ belong to S from the choice gadgets. Additionally, it follows that, in Figure 3.9b, the number of vertices of $\hat{C}^j_i$ belonging to S of instance $\mathcal{I}_1$ is the same as the one of instance $\mathcal{I}_2$, for $j \in \{l, h\}$. Therefore, we conclude that the number of vertices belonging to S from $\hat{C}^l_i$ (respectively, $\hat{C}^h_i$) is the same in all the instances of the choice gadget $\hat{C}_i$.

Let $\mathcal{V} \subseteq V(G)$ be a set of cardinality k, containing vertex $v^i_{s(i)} \in V_i$ if, for choice gadget $\hat{C}_i$, it holds that $|S \cap \hat{C}^h_i| = s(i)$ and $|S \cap \hat{C}^l_i| = n - s(i)$. Notice that $\mathcal{V} \cap V_i \neq \varnothing$, for all $i \in [k]$. We will prove that $\mathcal{V}$ is a clique.

Let $v^{i_1}_{s(i_1)}, v^{i_2}_{s(i_2)}$ belong to $\mathcal{V}$. Consider the adjacency gadget $\hat{A}(i_1, i_1, i_2, i_2)$. We will prove that it contains an edge gadget $\hat{E}_e$, where $e = \{v^{i_1}_{s(i_1)}, v^{i_2}_{s(i_2)}\}$. Consider the vertices $\ell^h_{i_1,i_2}$, $\ell^l_{i_1,i_2}$, $r^h_{i_1,i_2}$ and $r^l_{i_1,i_2}$. It holds that x has $\Delta - \tau(x)$ leaves, as well as $\tau(x)$ neighbors in the edge gadgets, where $x \in \{\ell^h_{i_1,i_2}, \ell^l_{i_1,i_2}, r^h_{i_1,i_2}, r^l_{i_1,i_2}\}$. Moreover,

- $\ell^h_{i_1,i_2}$ has $n - s(i_1)$ neighbors due to $\hat{C}^h_{i_1}$ in $H - S$,
- $\ell^l_{i_1,i_2}$ has $s(i_1)$ neighbors due to $\hat{C}^l_{i_1}$ in $H - S$,
- $r^h_{i_1,i_2}$ has $n - s(i_2)$ neighbors due to $\hat{C}^h_{i_2}$ in $H - S$,
- $r^l_{i_1,i_2}$ has $s(i_1)$ neighbors due to $\hat{C}^l_{i_2}$ in $H - S$.

Notice that from all but one edge gadgets, S contains all the c^i_j vertices. Since all x have degree at most Δ in $H - S$, it follows that there exists an edge gadget $\hat{E}_{e'}$, where

- $\ell^h_{i_1,i_2}$ has at least $n - s(i_1)$ neighbors in,
- $\ell^l_{i_1,i_2}$ has at least $s(i_1)$ neighbors in,
- $r^h_{i_1,i_2}$ has at least $n - s(i_2)$ neighbors in,
- $r^l_{i_1,i_2}$ has at least $s(i_1)$ neighbors in,

and from which all the vertices s^i_j belong to S. The only case this may happen is when $e' = e$, thus there exists such an edge in G.

Since this holds for any two vertices belonging to $\mathcal{V}$, it follows that G has a k-clique. □

Therefore, in polynomial time, we can construct a graph H, of tree-depth $\mathrm{td} = \mathcal{O}(k)$ due to Lemma 3.34, such that, due to Lemmas 3.35 and 3.36, deciding whether there exists $S \subseteq V(H)$ of size $|S| \leq k'$ and $H - S$ has maximum degree at most $\Delta = n^3$ is equivalent to deciding whether G has a k-clique. In that case, assuming there exists a $f(\mathrm{td})|V(H)|^{o(\mathrm{td})}$ algorithm for Bounded Degree Vertex Deletion, where f is any computable function, one could decide k-Multicolored Clique in time $f(\mathrm{td})|V(H)|^{o(\mathrm{td})} = g(k) \cdot n^{o(k)}$, for some computable function g, which contradicts the ETH. □

3.3.2. Defective Coloring

The proof will closely follow the hardness proof presented in [25]. Since we will repeatedly use the equality and palette gadgets (see Section 3.2.2), we will use the following convention: whenever v_1, v_2 are two vertices we have already introduced to the constructed graph H, when we say that we add an equality gadget $Q(v_1, v_2)$, this means that we add to H a copy of $Q(u_1, u_2, \chi_\mathrm{d}, \Delta)$ and then identify u_1, u_2 with v_1, v_2 respectively (similarly for palette gadgets).

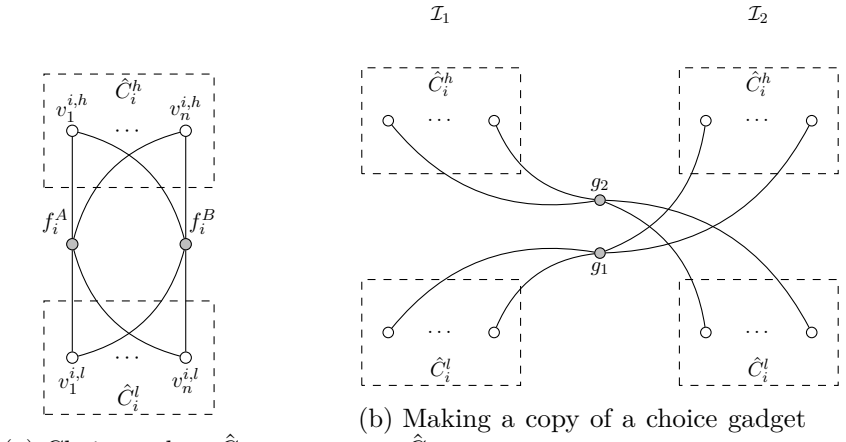
Theorem 3.38. *For any fixed $\chi_\mathrm{d} \geq 2$, if there exists an algorithm that solves Defective Coloring in time $f(\mathrm{td})n^{o(\mathrm{td})}$, where f is any computable function and* td *denotes the tree-depth of the input graph, then the ETH is false.*

Proof. Let (G, k) be an instance of k-Multicolored Clique, such that every vertex of G has a self loop, i.e. $\{v, v\} \in E(G)$ for all $v \in V(G)$. Recall that we assume that G is given to us partitioned into k independent sets $V_1, \ldots, V_k$, where $V_i = \{v^i_1, \ldots, v^i_n\}$, and each independent set has size exactly n. Assume without loss of generality that $k = 2^z$ for some $z \in \mathbb{N}$ (one can do so by adding dummy independent sets connected to all the other vertices of the graph). Moreover, let $E^{i_1,i_2} \subseteq E(G)$ denote the edges of G with one endpoint in V_{i_1} and the other in V_{i_2}. Set $\Delta = 2(|E(G)| - kn) + kn - (2\binom{k}{2} + k)$. We will construct in polynomial time a graph H of tree-depth $\mathrm{td}(H) = \mathcal{O}(k)$ such that H admits a $(\chi_\mathrm{d}, \Delta)$-coloring if and only if G has a k-clique.

Palette Vertices. Construct two vertices p^A and p^B, which we call main palette vertices, and add an edge connecting them. Next, construct vertices p^A_i and p^B_i, for $i \in [\Delta]$, add equality gadgets $Q(p^j, p^j_i)$ as well as edges between p^j and p^j_i, where $j \in \{A, B\}$ and $i \in [\Delta]$.

Choice Gadget. For an independent set V_i, we construct the *choice gadget* $\hat{C}_i$ as depicted in Figure 3.12a. We first construct independent sets $\hat{C}_i^p = \{v_j^{i,p} : j \in [n]\}$, where $p \in \{h, l\}$. We will refer to these vertices as *choice vertices*. Next, we introduce vertices f_i^A and f_i^B, connected with all choice vertices, and add equality gadgets $Q(p^A, f_i^A)$ and $Q(p^B, f_i^B)$. Finally, we attach to f_i^A (respectively, f_i^B) $\Delta - n$ leaves $l_j^{f_i^A}$ (respectively, $l_j^{f_i^B}$), for $j \in [\Delta - n]$, and add equality gadgets $Q(p^A, l_j^{f_i^A})$ (respectively, $Q(p^B, l_j^{f_i^B})$). If $\chi_{\mathrm{d}} \geq 3$, then we add a palette gadget $P(p^A, p^B, v_j^{i,q})$ for all choice vertices $v_j^{i,q}$. Intuitively, we consider a one-to-one mapping between the vertex v_j^i of V_i belonging to a supposed k-clique of G and the coloring of exactly j vertices of $\hat{C}_i^l$ and $n - j$ of $\hat{C}_i^h$ with the same color as the one used to color p^A.

Copy Gadget. Given two instances $\mathcal{I}_1$, $\mathcal{I}_2$ of a choice gadget $\hat{C}_i$, when we say that we connect them with a *copy gadget*, we introduce two vertices g_1 and g_2 and add equality gadgets $Q(p^A, g_1)$ and $Q(p^A, g_2)$. Moreover, we add an edge between g_1 (respectively, g_2) and the vertices of $\hat{C}_i^l$ of instance $\mathcal{I}_1$ (respectively, $\mathcal{I}_2$), as well as the vertices of $\hat{C}_i^h$ of instance $\mathcal{I}_2$ (respectively, $\mathcal{I}_1$), as depicted in Figure 3.12b. Lastly, we attach to each of g_1 and g_2 $\Delta - n$ leaves $l_j^{g_1}$ and $l_j^{g_b}$ respectively, where $j \in [\Delta - n]$, and add equality gadgets $Q(p^A, l_j^{g_1})$ and $Q(p^A, l_j^{g_2})$.



(a) Choice gadget $\hat{C}_i$.

(b) Making a copy of a choice gadget $\hat{C}_i$.

Figure 3.12.: Gray vertices have $\Delta - n$ leaves attached.

Edge Gadget. Let $e = \{v_{j_1}^{i_1}, v_{j_2}^{i_2}\} \in E^{i_1,i_2}$ be an edge of G. Construct the *edge gadget* $\hat{E}_e$ as depicted in Figure 3.13, where vertex c_e has Δ leaves l_j^e, $j \in [\Delta]$ attached, and add equality gadgets $Q(p^B, l_j^e)$ for each such leaf. Moreover, if $\chi_{\mathrm{d}} \geq 3$, then add a palette gadget $P(p^A, p^B, v)$, for every vertex $v \in \{c_e, s_j^{i_1}, s_j^{i_2}\}$.

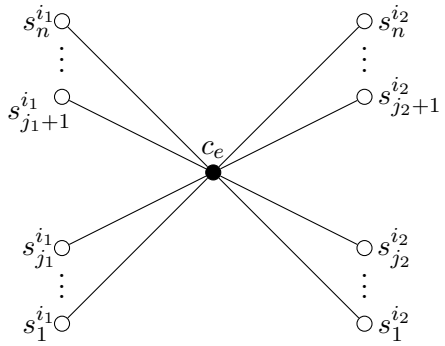


Figure 3.13.: Edge gadget $\hat{E}_e$ for $e = \{v_{j_1}^{i_1}, v_{j_2}^{i_2}\}$. The black vertex has Δ leaves attached.

Adjacency Gadget. For $i_1 \leq i_2$ and $j_1 \leq j_2$, we define the *adjacency gadget* $\hat{A}(i_1, i_2, j_1, j_2)$ as follows:

- Consider first the case when $i_1 = i_2$ and $i_1' = i_2'$. Let the adjacency gadget contain instances of the edge gadgets $\hat{E}_e$, for $e \in E^{i_1,i_1'}$, the choice gadgets $\hat{C}_{i_1}$ and $\hat{C}_{i_1'}$, as well as vertices $\ell^l_{i_1,i_1'}$, $\ell^h_{i_1,i_1'}$, $r^l_{i_1,i_1'}$ and $r^h_{i_1,i_1'}$. Add edges between
  - $\ell^l_{i_1,i_1'}$ and $\hat{C}^l_{i_1}$,
  - $\ell^h_{i_1,i_1'}$ and $\hat{C}^h_{i_1}$,
  - $r^l_{i_1,i_1'}$ and $\hat{C}^l_{i_1'}$,
  - $r^h_{i_1,i_1'}$ and $\hat{C}^h_{i_1'}$.

  If $e = \{v_{j_1}^{i_1}, v_{j_2}^{i_1'}\} \in E^{i_1,i_1'}$, then add the following edges adjacent to $\hat{E}_e$:
  - $\ell^l_{i_1,i_1'}$ with $s_\kappa^{i_1}$, for $\kappa \in [j_1]$,
  - $\ell^h_{i_1,i_1'}$ with $s_\kappa^{i_1}$, for $\kappa \in [j_1+1, n]$,
  - $r^l_{i_1,i_1'}$ with $s_\kappa^{i_1'}$, for $\kappa \in [j_2]$,
  - $r^h_{i_1,i_1'}$ with $s_\kappa^{i_1'}$, for $\kappa \in [j_2+1, n]$.

  For every vertex $x \in \{\ell^l_{i_1,i_1'}, \ell^h_{i_1,i_1'}, r^l_{i_1,i_1'}, r^h_{i_1,i_1'}\}$, add an equality gadget $Q(p^A, x)$ and attach $\Delta - n$ leaves. Lastly, for each leaf l, add an equality gadget $Q(p^A, l)$. For an illustration see Figure 3.14.
- Now consider the case when $i_1 < i_2$ and $i_1' < i_2'$. Then, let $\hat{A}(i_1, i_2, i_1', i_2')$ contain choice gadgets $\hat{C}_i$ and $\hat{C}_{i'}$, where $i \in [i_1, i_2]$ and $i' \in [i_1', i_2']$, which we will refer to as the *original choice gadgets* of $\hat{A}(i_1, i_2, i_1', i_2')$, as well as the adjacency gadgets
  - $\hat{A}\Big(i_1, \Big\lfloor \frac{i_1+i_2}{2} \Big\rfloor, i_1', \Big\lfloor \frac{i_1'+i_2'}{2} \Big\rfloor\Big)$,
  - $\hat{A}\Big(i_1, \Big\lfloor \frac{i_1+i_2}{2} \Big\rfloor, \Big\lceil \frac{i_1'+i_2'}{2} \Big\rceil, i_2'\Big)$,
  - $\hat{A}\Big(\Big\lceil \frac{i_1+i_2}{2} \Big\rceil, i_2, i_1', \Big\lfloor \frac{i_1'+i_2'}{2} \Big\rfloor\Big)$,
  - $\hat{A}\Big(\Big\lceil \frac{i_1+i_2}{2} \Big\rceil, i_2, \Big\lceil \frac{i_1'+i_2'}{2} \Big\rceil, i_2'\Big)$.

  Lastly, we connect with a copy gadget any choice gadgets $\hat{C}_i$ and $\hat{C}_{i'}$ appearing in said adjacency gadgets, with the corresponding original choice gadget $\hat{C}_i$ and $\hat{C}_{i'}$.

Notice that then, every original choice gadget is taking part in two copy gadgets. For a high level illustration see Figure 3.8.

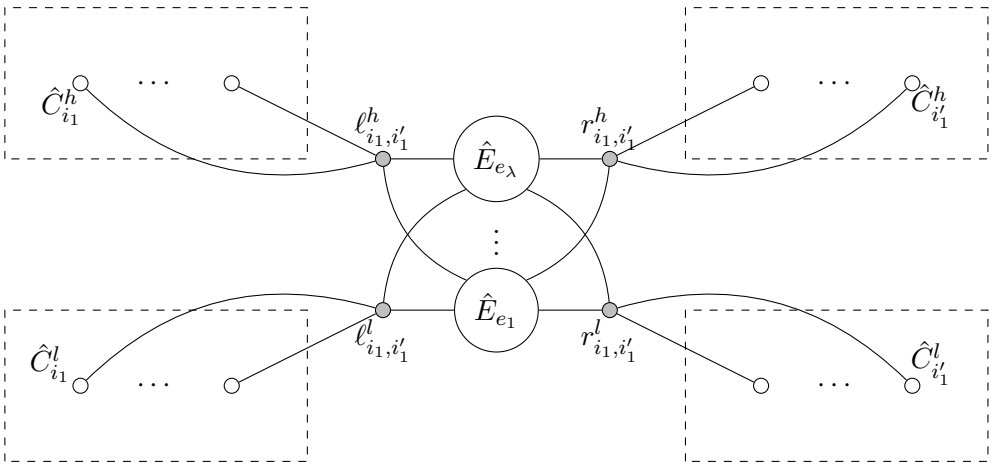


Figure 3.14.: Adjacency gadget $\hat{A}(i_1, i_1, i'_1, i'_1)$, where $E^{i_1,i'_1} = \{e_i : i \in [\lambda]\}$. Gray vertices have $\Delta - n$ leaves attached.

To construct graph H, first construct the adjacency gadget $\hat{A}(1, k, 1, k)$. Then, introduce a vertex u, which has an edge with the vertex c_e of $\hat{E}_e$, for all $e \in E(G)$. Lastly, add an equality gadget $Q(p^A, u)$.

Lemma 3.39. *The number of instances of edge gadget $\hat{E}_e$ present in H, where $e = \{v^{i_1}_{j_1}, v^{i_2}_{j_2}\} \in E(G)$, is one if $i_1 = i_2$, and two otherwise.*

Proof. First we will prove that for every adjacency gadget $\hat{A}(i_1, i_2, i'_1, i'_2)$ appearing in H, it holds that $i_2 - i_1 = i'_2 - i'_1 = 2^c - 1$, for some $c \in \mathbb{N}$. The statement holds for $\hat{A}(1, k, 1, k)$, as well as when $i_2 - i_1 = i'_2 - i'_1 = 0$. Suppose that it holds for some $\hat{A}(i_1, i_2, i'_1, i'_2)$, i.e. $i_2 - i_1 = i'_2 - i'_1 = 2^c - 1 > 0$, for some $c \in \mathbb{N}$. Then, it follows that $\left\lfloor \frac{i_1+i_2}{2} \right\rfloor - i_1 = \lfloor i_2 - 2^{c-1} + 0.5 \rfloor - i_1 = i_2 - i_1 - 2^{c-1} = 2^{c-1} - 1$. Moreover, it follows that $i_2 - \left\lceil \frac{i_1+i_2}{2} \right\rceil = i_2 - \lceil i_1 + 2^{c-1} - 0.5 \rceil = i_2 - (i_1 + 2^{c-1}) = 2^{c-1} - 1$. Therefore, the stated property holds.

In that case, for some $\hat{A}(i_1, i_2, i'_1, i'_2)$, in every step of the recursion, intervals $[i_1, i_2]$ and $[i'_1, i'_2]$ are partitioned in the middle, and an adjacency gadget is considered for each of the four combinations. In that case, starting from $\hat{A}(1, k, 1, k)$, there is a single way to produce every adjacency gadget $\hat{A}(i_1, i_1, i_2, i_2)$, where $i_1, i_2 \in [k]$. Consider an edge gadget $\hat{E}_e$, where $e = \{v^{i_1}_{j_1}, v^{i_2}_{j_2}\} \in E(G)$. There are two cases:

- if $i_1 = i_2$, then this gadget appears only in the adjacency gadget $\hat{A}(i_1, i_1, i_1, i_1)$,
- alternatively, it appears in both adjacency gadgets $\hat{A}(i_1, i_1, i_2, i_2)$ and $\hat{A}(i_2, i_2, i_1, i_1)$.

This concludes the proof. □

Lemma 3.40. *Let $\mathcal{I}_1, \mathcal{I}_2$ be two instances of $\hat{C}_i$ connected by a copy gadget. Then, for any (χ_d, Δ)-coloring $c\colon V(H) \to [\chi_d]$ of H, it holds that*

- *the number of vertices of color $c(p^A)$ of $\hat{C}_i^l$ (respectively, $\hat{C}_i^h$) of $\mathcal{I}_1$ is equal to the number of vertices of color $c(p^A)$ of $\hat{C}_i^l$ (respectively, $\hat{C}_i^h$) of $\mathcal{I}_2$,*
- *the number of vertices of color $c(p^B)$ of $\hat{C}_i^l$ (respectively, $\hat{C}_i^h$) of $\mathcal{I}_1$ is equal to the number of vertices of color $c(p^B)$ of $\hat{C}_i^l$ (respectively, $\hat{C}_i^h$) of $\mathcal{I}_2$.*

Proof. Due to the palette gadgets, any choice vertex of $\hat{C}_i$ is colored either $c(p^A)$ or $c(p^B)$. Due to the properties of the equality gadgets, it follows that vertex f_i^A (respectively, f_i^B) is of color $c(p^A)$ (respectively, $c(p^B)$), and has $\Delta - n$ same colored neighboring leaves. Consequently, exactly n choice vertices of $\hat{C}_i$ receive color $c(p^A)$ and exactly n color $c(p^B)$.

Let g_1, g_2 be the vertices present in the copy gadget connecting instances $\mathcal{I}_1, \mathcal{I}_2$. It follows that each of them has at most n neighbors of color $c(p^A)$ apart from its leaves. In that case, for $j \in \{l, h\}$, if the number of vertices of color $c(p^A)$ of $\hat{C}_i^j$ of instance $\mathcal{I}_1$ differs from the respective number of instance $\mathcal{I}_2$, either g_1 or g_2 has more than Δ same colored neighbors, contradiction. □

Lemma 3.41. *It holds that* $\mathrm{td}(H) = \mathcal{O}(k)$.

Proof. We first observe that all equality and palette gadgets added to the graph have at most one endpoint outside of $\{p^A, p^B\}$. Hence, by [25, Lemmata 3.6 and 3.9], we can conclude that $\mathrm{td}(H) = \mathrm{td}(H' \setminus \{p^A, p^B\}) + \chi_d + 1$, where H' is the graph we obtain from H by removing all the equality and palette gadgets. It therefore suffices to show that $\mathrm{td}(H') = \mathcal{O}(k)$. We first remove vertex u.

Now, let $T(\kappa)$ denote the tree-depth of $\hat{A}(i_1, i_2, i_1', i_2')$ in the case when $i_2 - i_1 = i_2' - i_1' = \kappa$. First, notice that, for $i_1, i_2 \in [k]$, the tree-depth of $\hat{A}(i_1, i_1, i_2, i_2)$ is at most 8: remove vertices ℓ_{i_1,i_2}^l, ℓ_{i_1,i_2}^h, r_{i_1,i_2}^l and r_{i_1,i_2}^h, resulting in the choice gadgets becoming disconnected with the edge gadgets. Then, it suffices to remove the vertices f_i^A, f_i^B, f_j^A and f_j^B from the choice gadgets, while the edge gadgets are trees of height 2. Consequently, $T(1) \leq 8$.

Now, consider the adjacency gadget $\hat{A}(i_1, i_2, i_1', i_2')$, where $i_2 - i_1 = i_2' - i_1' = \kappa$. This is comprised of adjacency gadgets

- $\hat{A}\Big(i_1, \Big\lfloor \frac{i_1+i_2}{2} \Big\rfloor, i_1', \Big\lfloor \frac{i_1'+i_2'}{2} \Big\rfloor\Big)$,
- $\hat{A}\Big(\Big\lceil \frac{i_1+i_2}{2} \Big\rceil, i_2, i_1', \Big\lfloor \frac{i_1'+i_2'}{2} \Big\rfloor\Big)$,
- $\hat{A}\Big(i_1, \Big\lfloor \frac{i_1+i_2}{2} \Big\rfloor, \Big\lceil \frac{i_1'+i_2'}{2} \Big\rceil, i_2'\Big)$,
- $\hat{A}\Big(\Big\lceil \frac{i_1+i_2}{2} \Big\rceil, i_2, \Big\lceil \frac{i_1'+i_2'}{2} \Big\rceil, i_2'\Big)$,

as well as of exactly 2κ original choice gadgets, each of which is connected with two copy gadgets to other instances of choice gadgets present in the adjacency gadgets. By removing all vertices g_1 and g_2 of the copy gadgets (see Figure 3.12b), all the original choice gadgets as well as the adjacency gadgets are disconnected. Therefore, it holds that $T(\kappa) \leq 8\kappa + T(\kappa/2)$, thus, it follows that $T(k) \leq 8 \sum_{i=0}^{\log k} \frac{k}{2^i} = \mathcal{O}(k)$, i.e. $\mathrm{td}(H') = \mathcal{O}(k)$. □

Lemma 3.42. *For any $\chi_{\mathrm{d}} \geq 2$, if G contains a k-clique, then H admits a $(\chi_{\mathrm{d}}, \Delta)$-coloring.*

Proof. Let $\mathcal{V} \subseteq V(G)$ be a k-clique of G, consisting of vertices $v^i_{s(i)} \in V_i$, for $i \in [k]$. We will produce a $(\chi_{\mathrm{d}}, \Delta)$-coloring of H as follows:

- Vertex p^A receives color 1 and vertex p^B color 2.
- All vertices for which we have added an equality gadget with one endpoint identified with p^A (respectively, p^B) take color 1 (respectively, 2).
- We use [25, Lemma 3.5] to properly color the internal vertices of the equality gadgets.
- For choice gadget $\hat{C}_i$, we color the vertices of $\{v^{i,h}_j \in \hat{C}^h_i : j \in [s(i)]\} \cup \{v^{i,l}_j \in \hat{C}^l_i : j \in [s(i)+1, n]\}$ with color 1, while we color the vertices of $\{v^{i,l}_j \in \hat{C}^l_i : j \in [s(i)]\} \cup \{v^{i,h}_j \in \hat{C}^h_i : j \in [s(i)+1, n]\}$ with color 2.
- For every $e = \{v^{i_1}_{s(i_1)}, v^{i_2}_{s(i_2)}\}$ that is contained in the clique, we color all vertices $s^{i_1}_j, s^{i_2}_j$ of $\hat{E}_e$, for $j \in [n]$, with color 1, and the vertex c_e with color 2. For all other edges e' not belonging to the clique, we use the opposite coloring, coloring vertices $s^{i_1}_j, s^{i_2}_j$ with color 2 and vertex $c_{e'}$ with color 1.
- We use [25, Lemma 3.8] to properly color the internal vertices of palette gadgets, since all palette gadgets that we add use either color 1 or color 2 twice in their endpoints.

This completes the coloring.

It remains to prove that this is indeed a $(\chi_{\mathrm{d}}, \Delta)$-coloring. We first note that by Lemmata 3.5 and 3.8 of [25], internal vertices of equality and palette gadgets are properly colored. Vertices p^A, p^B have exactly Δ neighbors with the same color. Vertices f^A_i and f^B_i have exactly n neighbors of the same color among vertices of $\hat{C}^l_i \cup \hat{C}^h_i$, thus exactly Δ neighbors of the same color overall. The same holds for every vertex g_1 and g_2 of a copy gadget. Choice vertices have a constant number of neighbors of the same color (due to vertices $f^A_i, f^B_i, g_1, g_2, \ell^l_{i_1,i_2}, \ell^h_{i_1,i_2}, r^l_{i_1,i_2}, r^h_{i_1,i_2}$). The vertex u has exactly $\deg_H(u) - (2\binom{k}{2} + k) = \Delta$ neighbors with color 1, since from Lemma 3.39, it follows that for adjacency gadget $\hat{A}(i, i, j, j)$, if $i \neq j$, then it appears twice in H and once otherwise, while the clique contains $\binom{k}{2}$ edges $e = \{v^{i_1}_{s(i_1)}, v^{i_2}_{s(i_2)}\}$ where $i_1 \neq i_2$ and k where $i_1 = i_2$. Finally, for the vertices $\ell^l_{i_1,i_2}, \ell^h_{i_1,i_2}, r^l_{i_1,i_2}, r^h_{i_1,i_2}$, we note that $\ell^h_{i_1,i_2}$ (respectively, $\ell^l_{i_1,i_2}$) has exactly $s(i_1)$ (respectively, $n - s(i_1)$) neighbors with color 1 among the choice vertices. Moreover, $\ell^h_{i_1,i_2}$ (respectively, $\ell^l_{i_1,i_2}$) has exactly $n - s(i_1)$ (respectively, $s(i_1)$) neighbors with color 1 in the edge gadgets, those belonging to $\hat{E}_e$, where $e = \{v^{i_1}_{s(i_1)}, v^{i_2}_{s(i_2)}\}$. In a similar way, one can show that $r^l_{i_1,i_2}$ and $r^h_{i_1,i_2}$ have exactly Δ neighbors of the same color. □

Lemma 3.43. *For any $\chi_{\mathrm{d}} \geq 2$, if H admits a $(\chi_{\mathrm{d}}, \Delta)$-coloring, then G contains a k-clique.*

Proof. Assume $c\colon V(H) \to [\chi_{\mathrm{d}}]$ is a $(\chi_{\mathrm{d}}, \Delta)$-coloring of H. It holds that $c(p^A) \neq c(p^B)$, since each vertex has Δ neighboring leaves of the same color due to the properties of the equality gadget. Assume without loss of generality that $c(p^A) = 1$ and $c(p^B) = 2$. For every instance of a choice gadget $\hat{C}_i$, it holds that half of its choice vertices are colored with color 1 and the other half with color 2: due to the palette gadgets, every vertex is colored with one of these two colors, and if more than n were colored with the same color, one of f_i^A, f_i^B would have more than Δ same colored neighbors. Moreover, due to Lemma 3.40, it follows that on every instance of the choice gadget $\hat{C}_i$ present in H, the number of vertices colored with color 1 in $\hat{C}_i^l$ (respectively, $\hat{C}_i^h$) is the same. We will first prove the following claim.

▷ Claim 3.44. For every $i_1, i_2 \in [k]$, there exists a single edge gadget $\hat{E}_e$ present in the adjacency gadget $\hat{A}(i_1, i_1, i_2, i_2)$ where $c(c_e) = 2$.

Proof of the claim. First, notice that due to Lemma 3.39, u has a total of $2(E(G)-kn)+kn$ neighbors. Consequently, at least $2(E(G) - kn) + kn - \Delta = 2\binom{k}{2} + k$ of its neighbors are colored with color 2, since u is of color 1 due to the equality gadget $Q(p^A, u)$. Notice that this is also the number of adjacency gadgets $\hat{A}(i_1, i_1, i_2, i_2)$, since every such gadget appears twice if $i_1 \neq i_2$ and once otherwise. Assume that for some adjacency gadget $\hat{A}(i_1, i_1, i_2, i_2)$ there is an edge gadget $\hat{E}_e$ such that $c(c_e) = 2$. In that case, c_e has Δ same colored neighboring leaves, therefore all vertices $s_j^{i_1}, s_j^{i_2}$ of $\hat{E}_e$ are colored with color 1 (if $\chi_{\mathrm{d}} \geq 3$, this is a consequence of the palette gadget attached to those vertices). If there existed a second edge gadget $\hat{E}_{e'}$ present in $\hat{A}(i_1, i_1, i_2, i_2)$ with $c(c_{e'}) = 2$, then it follows that $\ell^l_{i_1,i_2}$ and $\ell^h_{i_1,i_2}$ have in total $2n$ neighbors colored with color 1 and belonging to the edge gadgets. Consequently, since both already have $\Delta - n$ neighboring same colored leaves, it follows that each has exactly n neighbors from the edge gadgets which are colored with color 1, and no such neighbor from the choice gadget $\hat{C}_{i_1}$, which results in a contradiction, since exactly n choice vertices of $\hat{C}_{i_1}$ are colored with color 1. Consequently, for u to have at least $2\binom{k}{2} + k$ neighbors with color 2, it follows that each adjacency gadget $\hat{A}(i_1, i_1, i_2, i_2)$ has a single edge gadget $\hat{E}_e$ such that $c(c_e) = 2$. ◁

Let $\mathcal{V} \subseteq V(G)$ be a set of cardinality k, containing vertex $v^i_{s(i)} \in V_i$ if, for choice gadget $\hat{C}_i$, it holds that exactly $s(i)$ vertices of $\hat{C}_i^h$ and $n - s(i)$ vertices of $\hat{C}_i^l$ are colored with color 1. Notice that $\mathcal{V} \cap V_i \neq \varnothing$, for every i. We will prove that $\mathcal{V}$ is a clique.

Let $v^{i_1}_{s(i_1)}, v^{i_2}_{s(i_2)}$ belong to $\mathcal{V}$. Consider the adjacency gadget $\hat{A}(i_1, i_1, i_2, i_2)$. We will prove that it contains an edge gadget $\hat{E}_e$, where $e = \{v^{i_1}_{s(i_1)}, v^{i_2}_{s(i_2)}\}$. Consider the vertices $\ell^h_{i_1,i_2}$, $\ell^l_{i_1,i_2}$, $r^h_{i_1,i_2}$ and $r^l_{i_1,i_2}$. Each one of them has attached $\Delta - n$ neighboring leaves of color 1. Moreover, it holds that

- $\ell^h_{i_1,i_2}$ has $s(i_1)$ same colored neighbors from $\hat{C}^h_{i_1}$,
- $\ell^l_{i_1,i_2}$ has $n - s(i_1)$ same colored neighbors from $\hat{C}^l_{i_1}$,

- $r^h_{i_1,i_2}$ has $s(i_2)$ same colored neighbors from $\hat{C}^h_{i_2}$,
- $r^l_{i_1,i_2}$ has $n - s(i_2)$ same colored neighbors from $\hat{C}^l_{i_2}$.

Notice that due to Claim 3.44, there exists a single edge gadget $\hat{E}_{e'}$, where $e' = \{v^{i_1}_{j_1}, v^{i_2}_{j_2}\}$, for which $c(c_{e'}) = 2$ holds, thus all vertices $s^{i_1}_j, s^{i_2}_j$ have color 1. For c to be a $(\chi_{\mathrm{d}}, \Delta)$-coloring, it holds that $\ell^l_{i_1,i_2}$ (respectively, $\ell^h_{i_1,i_2}$) has at most $s(i_1)$ (respectively, $n - s(i_1)$) neighbors from the set $\{s^{i_1}_j : j \in [n]\}$. Consequently, it follows that $\ell^l_{i_1,i_2}$ (respectively, $\ell^h_{i_1,i_2}$) has exactly $s(i_1)$ (respectively, $n - s(i_1)$) neighbors from said set. Similarly, it follows that $r^l_{i_1,i_2}$ (respectively, $r^h_{i_1,i_2}$) has exactly $s(i_2)$ (respectively, $n - s(i_2)$) neighbors from the set $\{s^{i_2}_j : j \in [n]\}$. Consequently, $e' = \{v^{i_1}_{j_1}, v^{i_2}_{j_2}\}$, thus there exists such an edge in G.

Since this holds for any two vertices belonging to $\mathcal{V}$, it follows that G has a k-clique. □

Therefore, in polynomial time, we can construct a graph H, of tree-depth $\mathrm{td} = \mathcal{O}(k)$ due to Lemma 3.41, such that, due to Lemmas 3.42 and 3.43, deciding whether H admits a $(\chi_{\mathrm{d}}, \Delta)$-coloring is equivalent to deciding whether G has a k-clique. In that case, assuming there exists a $f(\mathrm{td})|V(H)|^{o(\mathrm{td})}$ algorithm for Defective Coloring, where f is any computable function, one could decide k-Multicolored Clique in time $f(\mathrm{td})|V(H)|^{o(\mathrm{td})} = g(k)n^{o(k)}$, for some computable function g, which contradicts the ETH. □

3.4. Vertex Cover Lower Bounds

In this section we present lower bounds on the complexity of solving Bounded Degree Vertex Deletion and Defective Coloring when parameterized by the vertex cover number of the input graph. In both cases we start from a 3-SAT instance of n variables, and produce an equivalent instance where the input graph has vertex cover $\mathcal{O}(n/\log n)$, hence any algorithm solving the latter problem in time $\mathrm{vc}^{o(\mathrm{vc})}n^{\mathcal{O}(1)}$ would refute the ETH. As a consequence of the above, already known algorithms for both of these problems are essentially optimal. We start by presenting some necessary tools used in both reductions, and then prove the stated results.

3.4.1. Preliminary Tools

We first define a constrained version of 3-SAT, called (3,4)-XSAT. This variant is closely related to the (3,4)-SAT problem [276] which asks whether a given formula ϕ is satisfiable, where ϕ is a 3-SAT formula each clause of which contains exactly 3 different variables and each variable occurs in at most 4 clauses. As observed by Bonamy et al. [49], a corollary of Tovey's work [276] is that there is no $2^{o(n)}$ algorithm for (3,4)-SAT unless the ETH is false, where n denotes the number of variables of the formula. Here we prove an analogous lower bound for (3,4)-XSAT. Subsequently, by closely following Lemma 3.2 from [49], we present a way to partition the formula's variables and clauses into groups

such that variables appearing in clauses of the same clause group belong to different variable groups.

(3,4)-XSAT

Instance: A 3-SAT formula ϕ every clause of which contains exactly 3 distinct variables and each variable appears in at most 4 clauses.

Goal: Determine whether there exists an assignment to the variables of ϕ such that each clause has exactly one True literal.

Theorem 3.45. *(3,4)-XSAT cannot be decided in time $2^{o(n)}$, where n denotes the number of variables of the input formula, unless the ETH fails.*

Proof. We will closely follow a known reduction from 3SAT to 1-IN-3-SAT, which is a simplification of Schaefer's work [264]. Let ϕ be a (3,4)-SAT formula of n variables and $m \leq \frac{4}{3} \cdot n$ clauses. Let V denote the set of its variables, and C the set of its clauses. We will construct an equivalent instance ϕ' of (3,4)-XSAT, where V' denotes the set of its variables and C' the set of its clauses, as follows:

- for every variable $x \in V$, introduce a variable $x \in V'$,
- for every clause $c_i \in C$, introduce variables $\alpha_i, \beta_i, \gamma_i, \delta_i \in V'$,
- for every clause $c_i = x \vee y \vee z$ of ϕ, introduce clauses $c_i^1 = \neg x \vee \alpha_i \vee \beta_i$, $c_i^2 = y \vee \beta_i \vee \gamma_i$ and $c_i^3 = \neg z \vee \gamma_i \vee \delta_i$ in ϕ'.

Notice that ϕ' is a valid (3,4)-XSAT instance, since every one of its $3m$ clauses contains exactly 3 different variables, while all of its $n + 4m$ variables appear in at most 4 clauses. In the following we prove that ϕ is satisfiable if and only if there exists an assignment to the variables of ϕ' such that each of its clauses has exactly one True literal.

For the forward direction, let $f \colon V \to \{T, F\}$ be a satisfying assignment for ϕ. Consider an assignment $f' \colon V' \to \{T, F\}$ such that $f'(x) = f(x)$, for all $x \in V$. It remains to determine the value of f' for any variable belonging to $V' \setminus V$. Let $c_i = x \vee y \vee z$ be a clause of ϕ. Since f is a satisfying assignment, it holds that at least one of x, y, z has a truthful assignment. If $f(y) = T$ then let $f'(\beta_i) = f'(\gamma_i) = F$, while $f'(\alpha_i) = f(x)$ and $f'(\delta_i) = f(z)$. On the other hand, if $f(y) = F$ then one of the following cases holds:

(i) $f(x) = f(z) = T$. Then, set $f'(\alpha_i) = T$, $f'(\beta_i) = F$, $f'(\gamma_i) = T$, and $f'(\delta_i) = F$.

(ii) $f(x) = T$ and $f(z) = F$. Then, set $f'(\alpha_i) = F$, $f'(\beta_i) = T$, $f'(\gamma_i) = F$, and $f'(\delta_i) = F$.

(iii) $f(x) = F$ and $f(z) = T$. Then, set $f'(\alpha_i) = F$, $f'(\beta_i) = F$, $f'(\gamma_i) = T$, and $f'(\delta_i) = F$.

Notice that f' is an assignment on V' such that all clauses of ϕ' have exactly one True literal.

For the converse direction, let $f' \colon V' \to \{T, F\}$ be an assignment such that every clause of ϕ' has exactly one True literal, and let $f \colon V \to \{T, F\}$ be the restriction of f' to V, i.e. $f(x) = f'(x)$, for all $x \in V$. We will prove that f is a satisfying assignment for ϕ. Indeed, assume there exists a clause $c_i = x \vee y \vee z$ of ϕ which is not satisfied by f. In that case, $f(x) = f(y) = f(z) = F$, and since f is a restriction of f' it follows that

- $f'(\alpha_i) = f'(\beta_i) = F$ due to c_i^1,
- $f'(\beta_i) \neq f'(\gamma_i)$ due to c_i^2 and
- $f'(\gamma_i) = f'(\delta_i) = F$ due to c_i^3,

which is a contradiction.

Lastly, assume there exists a $2^{o(|V'|)}$ algorithm for (3,4)-XSAT. Then, since $|V'| = n + 4m$, (3,4)-SAT could be decided in $2^{o(n)}$, thus the ETH fails. □

We proceed by proving that, given a (3,4)-XSAT instance, we can partition the variables and clauses of the formula into groups such that variables appearing in clauses of the same clause group belong to different variable groups.

Lemma 3.46. *Let ϕ be an instance of (3,4)-XSAT, where V denotes the set of its n variables and C the set of its clauses. Moreover, let $b \leq \sqrt{n}$. One can produce in time $n^{\mathcal{O}(1)}$ a partition of ϕ's variables into n_V disjoint sets $V_1, \ldots, V_{n_V}$ of size at most b as well as a partition of its clauses into n_C disjoint sets $C_1, \ldots, C_{n_C}$ of size at most $\sqrt{n}$, for some integers $n_V = \mathcal{O}(n/b)$ and $n_C = \mathcal{O}(\sqrt{n})$, such that, for any $i \in [n_C]$, any two variables appearing in clauses of C_i belong to different variable subsets, while any variable appears in at most 1 clause of C_i.*

Proof. We will first partition the variable set V into disjoint subsets $V_1, \ldots, V_{n_V}$, where $n_V = \mathcal{O}(n/b)$, such that $|V_i| \leq b$, and variables appearing in the same clause belong to different V_i's. Consider the graph G_1, which has a vertex per variable of ϕ, while two vertices have an edge if there exists a clause in ϕ that both corresponding variables appear in, i.e. G_1 is the primal graph of ϕ. G_1 does not have any loops, since no clause contains repeated variables. Since every variable of ϕ occurs in at most four clauses and since those clauses contain two other variables, the maximum degree of G_1 is at most 8. Hence, G_1 can be greedily colored with 9 colors, thus inducing a partition into different colored groups of size $n_1, \ldots, n_9$ respectively, where $n_1 + \ldots + n_9 = n$. Subsequently, we refine said partition so that every group has size at most b, resulting in at most

$$n_V = \sum_{i=1}^{9} \left\lceil \frac{n_i}{b} \right\rceil \leq 9 + \sum_{i=1}^{9} \frac{n_i}{b} = 9 + \frac{n}{b}$$

groups $V_1, \ldots, V_{n_V}$. Notice that it holds that, variables appearing in the same clause belong to different V_i's, since any two such variables are adjacent in G_1 and thus get different colors.

Next, we partition the clause set C into disjoint subsets $C_1, \ldots, C_{n_C}$, where $n_C = \mathcal{O}(\sqrt{n})$, such that $|C_i| \leq \sqrt{n}$, and any two variables appearing in clauses of a group

C_i belong to different variable groups. For this, consider the graph G_2 with clauses as vertices and with an edge between clauses if they contain variables from the same variable group. G_2 has no loops, since any variables occurring in the same clause belong to different V_i's. Since every clause contains exactly 3 variables, each variable group has size at most b, and every variable occurs in at most 4 clauses, the maximum degree of G_2 is at most $12b$. We can therefore color G_2 greedily with $12b+1$ colors. Similarly as before, we refine said partition into $n_C \leq 12b + 1 + |C|/\sqrt{n}$ subsets $C_1, \ldots, C_{n_C}$ of size at most $\sqrt{n}$ each. By the construction of the coloring, it follows that if a variable $v \in V_i$ appears in some clause $c \in C_j$, then no variable belonging to V_i appears in any clause in $C_j \setminus \{c\}$, while no other variable of V_i appears in c. Moreover, since $|C| \leq \frac{4}{3} \cdot n$ and $b \leq \sqrt{n}$, it follows that $n_C = \mathcal{O}(\sqrt{n})$. □

Definition 3.47. *A* d-detecting family *for a finite set* U *is a family* $\mathcal{F} \subseteq 2^U$ *of subsets of* U *such that, for every two functions* $f, g \colon U \to \{0, \ldots, d-1\}$ *where* $f \neq g$*, there exists* $S \in \mathcal{F}$ *such that* $\sum_{x \in S} f(x) \neq \sum_{x \in S} g(x)$.

Lindström [223] has provided a deterministic construction of sublinear d-detecting families, while Bonamy et al. [49] were the first to use them in the context of computational complexity, proving tight lower bounds for the Multicoloring problem under the ETH. The following theorem will be crucial towards proving the stated lower bounds.

Theorem 3.48 ([223])**.** *For every constant* $d \in \mathbb{N}$ *and finite set* U*, there is a* d*-detecting family* $\mathcal{F}$ *on* U *of size* $\frac{2|U|}{\log_d |U|} \cdot (1 + o(1))$*. Moreover,* $\mathcal{F}$ *can be constructed in time polynomial in* $|U|$.

3.4.2. Bounded Degree Vertex Deletion

Theorem 3.49. *There is no* $\mathrm{vc}^{o(\mathrm{vc})} n^{\mathcal{O}(1)}$ *time algorithm for Bounded Degree Vertex Deletion, where* vc *denotes the size of the minimum vertex cover of the input graph, unless the ETH fails.*

Proof. Let ϕ be an instance of (3,4)-XSAT of n variables. Assume without loss of generality that n is a power of 4 (this can be achieved by adding dummy variables to the instance if needed). Making use of Lemma 3.46 one can obtain in time $n^{\mathcal{O}(1)}$ the following:

- a partition of ϕ's variables into subsets $V_1, \ldots, V_{n_V}$, where $|V_i| \leq \log n$ and $n_V = \mathcal{O}(n/\log n)$,
- a partition of ϕ's clauses into subsets $C_1, \ldots, C_{n_C}$, where $|C_i| \leq \sqrt{n}$ and $n_C = \mathcal{O}(\sqrt{n})$,

where any two variables occurring in clauses of the same clause subset belong to different variable subsets. For $i \in [n_C]$, let $\{C_{i,1}, \ldots, C_{i,n^i_{\mathcal{F}}}\}$ be a 4-detecting family of subsets of C_i for some $n^i_{\mathcal{F}} = \mathcal{O}(\sqrt{n}/\log n)$, produced in time $n^{\mathcal{O}(1)}$ due to Theorem 3.48. Moreover, let $n_{\mathcal{F}} = \max_{i \in [n_C]} n^i_{\mathcal{F}}$. Define $\Delta = n^3$ and $k = n_V$. We will construct a graph $G = (V, E)$

such that there exists $S \subseteq V(G)$ of size $|S| \leq k$ and $G - S$ has maximum degree at most Δ if and only if there exists an assignment such that every clause of ϕ has exactly one True literal.

Choice Gadget. For each variable subset V_i we define the choice gadget graph G_i as follows:

- introduce vertices κ_i, λ_i, and v_j^i, where $j \in [n]$,
- add edges $\{\kappa_i, v_j^i\}$ and $\{\lambda_i, v_j^i\}$ for all $j \in [n]$,
- attach sufficiently many leaves to κ_i and λ_i such that their degree is $\Delta + 1$.

Let $\mathcal{V}_i = \{v_j^i : j \in [n]\}$, for $i = 1, \ldots, n_V$. We fix an arbitrary one-to-one mapping so that every vertex of $\mathcal{V}_i$ corresponds to a different assignment for the variables of V_i. Since $2^{|V_i|} \leq n$, there are sufficiently many vertices to uniquely encode all the different assignments of V_i. Let $\mathcal{V} = \mathcal{V}_1 \cup \ldots \cup \mathcal{V}_{n_V}$ denote the set of all such vertices.

Clause Gadget. For $i \in [n_C]$, let C_i be a clause subset and $\{C_{i,1}, \ldots, C_{i,n_{\mathcal{F}}^i}\}$ its 4-detecting family. For every subset $C_{i,j}$ of the 4-detecting family introduce vertices $c_{i,j}$ and $c'_{i,j}$. Add an edge between $c_{i,j}$ and v_q^p if there exists variable $x \in V_p$ such that x appears in some clause $c \in C_{i,j}$, and v_q^p corresponds to an assignment of V_p that satisfies c. Due to Lemma 3.46, $c_{i,j}$ has exactly $|C_{i,j}| \cdot \frac{3n}{2}$ such edges: there are exactly $3|C_{i,j}|$ different variables appearing in clauses of $C_{i,j}$, each belonging to a different variable subset, and for each such variable, half the assignments of the corresponding variable subset result in the satisfaction of the corresponding clause of $C_{i,j}$. Attach to $c_{i,j}$ a sufficient number of leaves such that its total degree is $\Delta + |C_{i,j}|$. Moreover, for $v \in \mathcal{V}$, let $v \in N(c'_{i,j})$ if $v \notin N(c_{i,j})$. Notice that then, it holds that $N(c_{i,j}) \cup N(c'_{i,j}) \supseteq \mathcal{V}$, while $N(c_{i,j}) \cap N(c'_{i,j}) = \varnothing$. Lastly, attach to $c'_{i,j}$ a sufficient number of leaves such that its total degree is $\Delta + (k - |C_{i,j}|)$.

Let $\mathcal{I} = (G, \Delta, k)$ be an instance of Bounded Degree Vertex Deletion.

Lemma 3.50. *It holds that* $\mathrm{vc}(G) = \mathcal{O}(n/\log n)$.

Proof. Notice that the deletion of all vertices κ_i, λ_i, $c_{i,j}$ and $c'_{i,j}$ induces an independent set. Therefore,

$$\mathrm{vc}(G) \leq 2n_V + 2\sum_{i=1}^{n_C} n_{\mathcal{F}}^i \leq 2n_V + 2n_C \cdot n_{\mathcal{F}} = \mathcal{O}\left(\frac{n}{\log n} + \sqrt{n} \cdot \frac{\sqrt{n}}{\log n}\right) = \mathcal{O}\left(\frac{n}{\log n}\right),$$

and the statement follows. □

Lemma 3.51. *If ϕ is a Yes instance of (3,4)-XSAT, then $\mathcal{I}$ is a Yes instance of* Bounded Degree Vertex Deletion.

Proof. Let $f\colon V_1 \cup \ldots \cup V_{n_V} \to \{T, F\}$ be an assignment such that every clause of ϕ has exactly 1 True literal. Let S contain from each $\mathcal{V}_i$ the vertex corresponding to this assignment restricted to V_i. It holds that $|S| = n_V = k$. We will prove that $G - S$ has maximum degree at most Δ. First, notice that in $G - S$, any vertex v_p^q has at most $2 + n_C \cdot n_{\mathcal{F}} \leq \Delta$ neighbors, while any vertex κ_i and λ_i has degree Δ, since $|S \cap \mathcal{V}_i| = 1$ for all $i \in [n_V]$. Lastly, for each vertex $c_{i,j}$ corresponding to a clause set $C_{i,j}$, it holds that, since f is an assignment where every clause of ϕ has exactly 1 True literal, and due to Lemma 3.46, S contains exactly $|C_{i,j}|$ neighbors of $c_{i,j}$. In that case, the remaining $k - |C_{i,j}|$ vertices belonging to S are neighbors of $c'_{i,j}$. □

Lemma 3.52. *If $\mathcal{I}$ is a Yes instance of* Bounded Degree Vertex Deletion*, then ϕ is a Yes instance of (3,4)-XSAT.*

Proof. Let $S \subseteq V(G)$, where $|S| \leq k$ and $G - S$ has maximum degree at most Δ. We will first prove that S contains a single vertex from every set $\mathcal{V}_i$.

▷ Claim 3.53. $|S \cap \mathcal{V}_i| = 1$, for all $i \in [n_V]$.

Proof of the claim. We will first prove that $|S \cap V(G_i)| = 1$, for all $i \in [n_V]$. Suppose that this is not the case. Then, since $|S| \leq n_V$, it holds that there exists some i such that $|S \cap V(G_i)| = 0$. In that case, $\deg_{G-S}(\kappa_i) = \Delta + 1$, contradiction. Lastly, suppose there exists i such that, $v \in S$ and $v \in V(G_i) \setminus \mathcal{V}_i$. Since $\deg_G(\kappa_i) = \deg_G(\lambda_i) = \Delta + 1$, and $v \notin N(\kappa_i) \cap N(\lambda_i)$, it follows that one of κ_i, λ_i does not belong in S and has degree $\Delta + 1$ in $G - S$, contradiction. ◁

Now consider the assignment $h\colon V_1 \cup \ldots \cup V_{n_V} \to \{T, F\}$ of the variables of ϕ depending on which vertex of $\mathcal{V}_i$ belongs to S.

Let C_i, where $i \in [n_C]$, be a clause subset resulting from the partition due to Lemma 3.46. Let $f\colon C_i \to \{0, 1, 2, 3\}$ be the function assigning to each clause $c \in C_i$ the number of vertices $v \in S$ such that $v \in \mathcal{V}_j$ corresponds to an assignment $V_j \to \{T, F\}$ that satisfies c. Notice that since every clause contains 3 literals and $|\mathcal{V}_i \cap S| = 1$, $f(c) \leq 3$ follows. It holds that $\sum_{c \in C_{i,j}} f(c) = |C_{i,j}|$, for any $j \in [n^i_{\mathcal{F}}]$: $|S \cap N(c_{i,j})| \geq |C_{i,j}|$, $|S \cap N(c'_{i,j})| \geq k - |C_{i,j}|$, while $N(c_{i,j}) \cap N(c'_{i,j}) = \varnothing$ and $|S| = k$.

Now consider $g\colon C_i \to \{0, 1, 2, 3\}$ to be the constant function $g \equiv 1$. Notice that $\sum_{c \in C_{i,j}} f(c) = \sum_{c \in C_{i,j}} g(c)$, for all $j \in [n^i_{\mathcal{F}}]$. Since $\{C_{i,1}, \ldots, C_{i,n^i_{\mathcal{F}}}\}$ is a 4-detecting family, this implies that $f = g$. Thus, for every clause $c \in C_i$, it holds that $f(c) = 1$, meaning that there exists a single vertex in S which corresponds to a partial assignment that satisfies c. Since $i = 1, \ldots, n_C$ was arbitrary, this concludes the proof that h is a satisfying assignment for ϕ. □

Therefore, in polynomial time, we can construct a graph G with vertex cover number $\mathrm{vc} = \mathcal{O}(n/\log n)$ due to Lemma 3.50, such that, due to Lemmas 3.51 and 3.52, deciding whether there exists $S \subseteq V(G)$ of size $|S| \leq k$ and $G - S$ has maximum degree at most Δ is equivalent to deciding if there exists an assignment such that every clause of ϕ has

exactly one True literal. In that case, assuming there exists a $\mathrm{vc}^{o(\mathrm{vc})}n^{\mathcal{O}(1)}$ algorithm for Bounded Degree Vertex Deletion, one could decide (3,4)-XSAT in time

$$\mathrm{vc}^{o(\mathrm{vc})}n^{\mathcal{O}(1)} = \left(\frac{n}{\log n}\right)^{o(n/\log n)} n^{\mathcal{O}(1)} = 2^{(\log n - \log\log n)o(n/\log n)} = 2^{o(n)},$$

which contradicts the ETH due to Theorem 3.45. □

3.4.3. Defective Coloring

Theorem 3.54. *For any fixed $\chi_\mathrm{d} \geq 2$, if there exists an algorithm that solves Defective Coloring in time $\mathrm{vc}^{o(\mathrm{vc})}n^{\mathcal{O}(1)}$, where* vc *denotes the size of the minimum vertex cover of the input graph, then the ETH is false.*

Proof. Let ϕ be an instance of (3,4)-XSAT of n variables. Assume without loss of generality that n is a power of 16 (this can be achieved by adding dummy variables to the instance if needed). Making use of Lemma 3.46, one can obtain in time $n^{\mathcal{O}(1)}$ the following:

- a partition of ϕ's variables into subsets $V_1, \ldots, V_{n_V}$, where $|V_i| \leq \log \sqrt[4]{n}$ and $n_V = \mathcal{O}(n/\log n)$,
- a partition of ϕ's clauses into subsets $C_1, \ldots, C_{n_C}$, where $|C_i| \leq \sqrt{n}$ and $n_C = \mathcal{O}(\sqrt{n})$,

where any two variables occurring in clauses of the same clause subset belong to different variable subsets. For $i \in [n_C]$, let $\{C_{i,1}, \ldots, C_{i,n^i_\mathcal{F}}\}$ be a 4-detecting family of subsets of C_i for some $n^i_\mathcal{F} = \mathcal{O}(\sqrt{n}/\log n)$, produced in time $n^{\mathcal{O}(1)}$ due to Theorem 3.48. Moreover, let $n_\mathcal{F} = \max_{i\in[n_C]} n^i_\mathcal{F}$. Define $\Delta = n_V + n_C \cdot n_\mathcal{F}$. We will construct a graph $G = (V, E)$ such that G admits a $(\chi_\mathrm{d}, \Delta)$-coloring if and only if there exists an assignment such that every clause of ϕ has exactly one True literal.

Main Palette Vertices. Introduce vertices b_i and r_j, where $i \in [\Delta + 1]$ and $j \in [2\Delta + 1]$, such that they comprise a complete bipartite graph (i.e. all edges $\{b_i, r_j\}$ are present). We will refer to those as the *main palette vertices*.

Choice Gadget. For each variable subset V_i we define the choice gadget graph G_i as follows:

- introduce vertices κ_i, λ_i, and v^i_j, where $j \in [\sqrt[4]{n}]$,
- add edges $\{\kappa_i, v^i_j\}$ and $\{\lambda_i, v^i_j\}$, for all $j \in [\sqrt[4]{n}]$,
- add edges $\{\kappa_i, r_p\}$ and $\{\kappa_i, b_q\}$, for $p \in [\Delta + 1]$ and $q \in [\Delta - 1]$,
- add edges $\{\lambda_i, b_p\}$ and $\{\lambda_i, r_q\}$, for $p \in [\Delta + 1]$ and $q \in [\Delta - (\sqrt[4]{n} - 1)]$.

Let $\mathcal{V}_i = \{v_j^i : j \in [\sqrt[4]{n}]\}$, for $i = 1, \ldots, n_V$. We fix an arbitrary one-to-one mapping so that every vertex of $\mathcal{V}_i$ corresponds to a different assignment for the variables of V_i. Since $2^{|V_i|} \leq \sqrt[4]{n}$, there are sufficiently many vertices to uniquely encode all the different assignments of V_i. Let $\mathcal{V} = \mathcal{V}_1 \cup \ldots \cup \mathcal{V}_{n_V}$ denote the set of all such vertices.

Clause Gadget. For $i \in [n_C]$, let C_i be a clause subset and $\{C_{i,1}, \ldots, C_{i,n_{\mathcal{F}}^i}\}$ its 4-detecting family. For every subset $C_{i,j}$ of the 4-detecting family, introduce vertices $c_{i,j}$ and $c'_{i,j}$. Add an edge between $c_{i,j}$ and v_q^p if there exists variable $x \in V_p$ such that x occurs in some clause $c \in C_{i,j}$, and v_q^p corresponds to an assignment of V_p that satisfies c. Due to Lemma 3.46, $c_{i,j}$ has exactly $|C_{i,j}| \cdot \frac{3\sqrt[4]{n}}{2}$ such edges: there are exactly $3|C_{i,j}|$ different variables appearing in clauses of $C_{i,j}$, each belonging to a different variable subset, and for each such variable, half the assignments of the corresponding variable subset result in the satisfaction of the corresponding clause of $C_{i,j}$. Additionally, add edges $\{c_{i,j}, r_p\}$ and $\{c_{i,j}, b_q\}$, where $p \in [\Delta + 1]$ and $q \in [\Delta - |C_{i,j}|]$. Regarding $c'_{i,j}$, add an edge $\{c'_{i,j}, v\}$ for all $v \in \mathcal{V}$ such that $v \in N(c_{i,j})$. Finally, add edges $\{c'_{i,j}, b_p\}$ and $\{c'_{i,j}, r_q\}$, where $p \in [\Delta + 1]$ and $q \in [\Delta - (|C_{i,j}| \cdot \frac{3\sqrt[4]{n}}{2} - |C_{i,j}|)]$.

Secondary Palette Vertices. If $\chi_{\mathrm{d}} \geq 3$, then introduce independent sets $P_i = \{p_j^i : j \in [i \cdot \Delta + 1]\}$, for $i \in [3, \chi_{\mathrm{d}}]$. We will refer to the vertices of $P = P_3 \cup \ldots \cup P_{\chi_{\mathrm{d}}}$ as *secondary palette vertices*. Add an edge from every secondary palette vertex $p_j^i \in P_i$ to all vertices introduced except those belonging to P_i.

Lemma 3.55. *It holds that* $\mathrm{vc}(G) = \mathcal{O}(n/\log n)$.

Proof. Notice that the deletion of all main and secondary palette vertices as well as of all vertices κ_i, λ_i, $c_{i,j}$ and $c'_{i,j}$ induces an independent set. Therefore,

$$\mathrm{vc}(G) \leq \sum_{i=1}^{\chi_{\mathrm{d}}}(i \cdot \Delta + 1) + 2n_V + 2\sum_{i=1}^{n_C} n_{\mathcal{F}}^i \leq \chi_{\mathrm{d}}^2 \cdot \Delta + \chi_{\mathrm{d}} + 2n_V + 2n_C \cdot n_{\mathcal{F}} = \mathcal{O}\Big(\frac{n}{\log n}\Big),$$

and the statement follows. □

Lemma 3.56. *For any $\chi_{\mathrm{d}} \geq 2$, if ϕ is a Yes instance of (3,4)-XSAT, then G has a $(\chi_{\mathrm{d}}, \Delta)$-coloring.*

Proof. Let $f \colon V_1 \cup \ldots \cup V_{n_V} \to \{T, F\}$ be an assignment such that every clause of ϕ has exactly 1 True literal. If $\chi_{\mathrm{d}} \geq 3$, color every secondary palette vertex of P_i with color i, for $i \in [3, \chi_{\mathrm{d}}]$, thus resulting in that many independent sets of distinct colors. Refer to the (remaining) two colors as blue and red and consider the following coloring:

- vertices b_i, κ_i, and $c_{i,j}$ are colored blue,
- vertices r_i, λ_i, and $c'_{i,j}$ are colored red,
- for every $\mathcal{V}_i$, a single vertex corresponding to the assignment f restricted to V_i is colored blue, while the rest are colored red.

Let B and R be the subsets of $V(G)$ colored blue and red respectively. We will prove that $G[B]$ and $G[R]$ have maximum degree at most Δ. Regarding vertices b_i and r_i, notice that each has at most $n_V + \sum_{i=1}^{n_C} n^i_{\mathcal{F}} \leq \Delta$ same colored neighbors. Regarding vertices κ_i and λ_i, it holds that $|\mathcal{V}_i \cap B| = 1$ and $|\mathcal{V}_i \cap R| = \sqrt[4]{n} - 1$, thus each has degree equal to Δ in $G[B]$ and $G[R]$ respectively. For every $v \in \mathcal{V}_i$, it holds that it has at most $1 + \sum_{i=1}^{n_C} n^i_{\mathcal{F}} \leq \Delta$ same colored neighbors, due to either κ_i plus the vertices $c_{i,j}$ in case it is colored blue or λ_i plus the vertices $c'_{i,j}$ otherwise. Lastly, for each vertex $c_{i,j}$ corresponding to a clause set $C_{i,j}$, it holds that, since f is an assignment where every clause of ϕ has exactly 1 True literal, and due to Lemma 3.46, exactly $|C_{i,j}|$ neighbors of $c_{i,j}$ in $\mathcal{V}$ are colored blue. It that case, the rest of its neighbors in $\mathcal{V}$ are colored red. Consequently, it follows that each $c_{i,j}$ and $c'_{i,j}$ has exactly Δ same colored neighbors. □

Lemma 3.57. *For any $\chi_{\mathrm{d}} \geq 2$, if G has a $(\chi_{\mathrm{d}}, \Delta)$-coloring, then ϕ is a Yes instance of (3,4)-XSAT.*

Proof. Let $q^* \colon V(G) \to [\chi_{\mathrm{d}}]$ be a $(\chi_{\mathrm{d}}, \Delta)$-coloring of G. We start with the following claim.

▷ Claim 3.58. The restriction of q^* to $V(G - P)$ is a $(2, \Delta)$-coloring of $G - P$.

Proof of the claim. If $\chi_{\mathrm{d}} = 2$ this is true since $P = \varnothing$. Assume that $\chi_{\mathrm{d}} \geq 3$, and let for $i \in [3, \chi_{\mathrm{d}}]$, $G_i = G - \bigcup_{k=i+1}^{\chi_{\mathrm{d}}} P_k$, while $G_{\chi_{\mathrm{d}}} = G$. We will prove that if there exists a (i, Δ)-coloring for G_i, then the restriction of the same coloring to G_{i-1} is a $(i-1, \Delta)$-coloring of G_{i-1}, for $i \in [3, \chi_{\mathrm{d}}]$. The statement holds for χ_{d}: notice that, since $P_{\chi_{\mathrm{d}}}$ contains $\chi_{\mathrm{d}} \cdot \Delta + 1$ vertices, there exist at least $\Delta + 1$ vertices of the same color. In that case, since all those vertices are connected to any other vertex of G not belonging to $P_{\chi_{\mathrm{d}}}$, it follows that at most $\chi_{\mathrm{d}} - 1$ colors are used in the rest of the graph, thus the restriction of q^* to $V(G - P_{\chi_{\mathrm{d}}})$ is a $(i-1, \Delta)$-coloring of $G - P_{\chi_{\mathrm{d}}}$. Now assume that there exists a (i, Δ)-coloring for G_i, where $i \in [3, \chi_{\mathrm{d}} - 1]$. Then, notice that, since P_i contains $i \cdot \Delta + 1$ vertices, there exist at least $\Delta + 1$ vertices of the same color. In that case, since all those vertices are connected to any other vertex of G_i not belonging to P_i, it follows that at most $i - 1$ colors are used in the rest of the graph, thus the restriction of q^* to $V(G_{i-1})$ is a $(i-1, \Delta)$-coloring of G_{i-1}. ◁

Let $q \colon V(G - P) \to \{b, r\}$ be the restriction of q^* to $G - P$, where $q(v)$ denotes the (blue or red) color of every vertex v of $G - P$. Assume without loss of generality that $q(b_1) = b$.

- We first prove that $q(b_i) = b$, for all $i \in [\Delta + 1]$. Assume that this is not the case, i.e. there exists i such that $q(b_i) = r$. It holds that $r_j \in N(b_1) \cap N(b_i)$, for all $j \in [2\Delta + 1]$, and since every vertex r_j is colored either blue or red, it follows that there exist at least $\Delta + 1$ such vertices of the same color. In that case, either b_1 or b_i has $\Delta + 1$ same colored neighbors, contradiction.

- Since every vertex r_i, λ_i, and $c'_{i,j}$ has $\Delta + 1$ blue colored neighbors, it follows that $q(r_i) = q(\lambda_i) = q(c'_{i,j}) = r$.

- Since every vertex κ_i and $c_{i,j}$ has $\Delta + 1$ red colored neighbors, it follows that $q(\kappa_i) = q(c_{i,j}) = b$.
- In that case, any r_i has at most $n_V + \sum_{i=1}^{n_C} n^i_{\mathcal{F}} < \Delta$ same colored neighbors due to the vertices λ_i and $c'_{i,j}$. Symmetrically, any b_i has at most $n_V + \sum_{i=1}^{n_C} n^i_{\mathcal{F}} < \Delta$ same colored neighbors.

For every subset $\mathcal{V}_i$, it holds that exactly one vertex is colored blue: if all vertices were colored red, then λ_i would have in total $\Delta - (\sqrt[4]{n} - 1) + \sqrt[4]{n} > \Delta$ red neighbors, while if more than one vertices, say $p > 1$, were blue then κ_i would have $\Delta - 1 + p > \Delta$ blue neighbors.

Now consider the assignment $h\colon V_1 \cup \ldots \cup V_{n_V} \to \{T, F\}$ of the variables of ϕ depending on which vertex of $\mathcal{V}_i$ is colored blue. Let C_i, where $i \in [n_C]$ be a clause subset resulting from the partition due to Lemma 3.46. Let $f\colon C_i \to \{0, 1, 2, 3\}$ be the function assigning to each clause $c \in C_i$ the number of blue vertices $v \in \mathcal{V}$ such that $v \in \mathcal{V}_j$ corresponds to an assignment $V_j \to \{T, F\}$ that satisfies c. Notice that since every clause contains 3 literals, while from every subset $\mathcal{V}_i$ exactly one vertex is colored blue, $f(c) \leq 3$ follows. It holds that $\sum_{c \in C_{i,j}} f(c) = |C_{i,j}|$, for any $j \in [n^i_{\mathcal{F}}]$: $c_{i,j}$ and $c'_{i,j}$ have the same neighborhood in $\mathcal{V}$ of size $|C_{i,j}| \cdot \frac{3\sqrt[4]{n}}{2}$, and out of those vertices, at most $|C_{i,j}|$ may be blue while at most $|C_{i,j}| \cdot \frac{3\sqrt[4]{n}}{2} - |C_{i,j}|$ may be red.

Now consider $g\colon C_i \to \{0, 1, 2, 3\}$ to be the constant function $g \equiv 1$. Notice that $\sum_{c \in C_{i,j}} f(c) = \sum_{c \in C_{i,j}} g(c)$, for all $j \in [n^i_{\mathcal{F}}]$. Since $\{C_{i,1}, \ldots, C_{i,n^i_{\mathcal{F}}}\}$ is a 4-detecting family, this implies that $f = g$. Thus, for every clause $c \in C_i$, it holds that $f(c) = 1$, meaning that there exists a single blue vertex in $\mathcal{V}$ which corresponds to a partial assignment that satisfies c. Since $i = 1, \ldots, n_C$ was arbitrary, this concludes the proof that h is a satisfying assignment for ϕ. □

Therefore, in polynomial time, we can construct a graph G, with vertex cover number $\mathrm{vc} = \mathcal{O}(n/\log n)$ due to Lemma 3.55, such that, due to Lemmas 3.56 and 3.57, deciding whether G admits a $(\chi_{\mathrm{d}}, \Delta)$-coloring is equivalent to deciding if there exists an assignment such that every clause of ϕ has exactly one True literal. In that case, assuming there exists a $\mathrm{vc}^{o(\mathrm{vc})} n^{\mathcal{O}(1)}$ algorithm for Defective Coloring, one could decide (3,4)-XSAT in time

$$\mathrm{vc}^{o(\mathrm{vc})} n^{\mathcal{O}(1)} = \left(\frac{n}{\log n}\right)^{o(n/\log n)} n^{\mathcal{O}(1)} = 2^{(\log n - \log\log n) o(n/\log n)} = 2^{o(n)},$$

which contradicts the ETH due to Theorem 3.45. □

4. Induced and Acyclic Matching

Publication note. An extended abstract of the results presented in this chapter appeared in the proceedings of the WG 2025 [217]. The corresponding paper received the *Best Student Paper Award*.

Chapter summary. This chapter investigates Induced Matching and Acyclic Matching under structural width measures. In particular, we present tight pw-SETH-based lower bounds for both problems parameterized by pathwidth, that render known algorithms for both problems optimal. Furthermore, we give an FPT algorithm for Induced Matching parameterized by clique-width that matches the aforementioned lower bound under the pw-SETH, which makes use of advanced techniques such as state representations and FFT.

Given a graph G, a *matching* $M \subseteq E(G)$ of G is a subset of its edges such that every vertex of G is incident with at most one edge in M. Matchings are central in Computer Science [232], and problems involving them find numerous practical applications such as matching kidney donors to recipients and clients to server clusters to name a few [234]. Even though finding a matching of maximum cardinality in general graphs is well-known to be polynomial-time solvable [242], further demanding that the graph induced by the vertices of the matching adheres to a specific constraint oftentimes renders the problem NP-hard. Recently, Chaudhary and Zehavi [61] studied a plethora of such NP-hard problems under the lens of parameterized complexity, providing a variety of algorithms and posing some open questions, some of which we answer in this work.

We first define the problems in question. Let $G = (V, E)$ be a graph and $M \subseteq E$ a matching of G, where $V_M \subseteq V$ denotes the set of vertices of G incident with edges in M. We say that matching M is *induced* (resp., *acyclic*) if $G[V_M]$ is 1-regular (resp., acyclic). In that case, given a graph G and an integer ℓ, Induced Matching (resp., Acyclic Matching) asks whether G has an induced (resp., acyclic) matching of size at least ℓ. Originally introduced as the "risk-free" marriage problem [271], Induced Matching has been extensively studied due to its numerous applications (see [152]). This problem and its variations also have connections to various problems such as Maximum Feasible Subsystem [56, 111] and storylines extraction [203]. As for Acyclic Matching, it was introduced by Goddard et al. [151] along with some other variations of Maximum Matching. Both problems are known to be NP-hard [54, 151, 271], thus in this paper we examine them under the perspective of parameterized complexity. Most relevant to us are the results concerning their complexity under structural parameterizations. Both problems are known to be FPT by the treewidth tw of the input graph [61, 161, 245], with the state-of-the-art being due to Chaudhary and Zehavi [61] who proposed algorithms of running time $\mathcal{O}^\star(3^{\mathrm{tw}})$ and $\mathcal{O}^\star(6^{\mathrm{tw}})$ for Induced Matching and Acyclic Matching respectively. Furthermore, the authors complement

their algorithms with $\mathcal{O}^\star((\sqrt{6}-\varepsilon)^{\mathrm{pw}})$ and $\mathcal{O}^\star((3-\varepsilon)^{\mathrm{pw}})$ SETH-based lower bounds respectively, where pw denotes the pathwidth of the input graph. Lastly, as mentioned in [195], Induced Matching is expressible in MSO_1 logic, thus it is FPT parameterized by clique-width due to standard metatheorems [86].

Our Contribution. We start with the Induced Matching problem. Our first result is to show the optimality of the $\mathcal{O}^\star(3^{\mathrm{tw}})$ algorithm of Chaudhary and Zehavi [61]. As a matter of fact, for the parameterizations by the pathwidth pw and the cutwidth ctw of the input graph, we show that for all $\varepsilon > 0$, obtaining an $\mathcal{O}^\star((3-\varepsilon)^{\mathrm{pw}})$ or an $\mathcal{O}^\star((3-\varepsilon)^{\mathrm{ctw}})$ algorithm is *equivalent* to falsifying the pw-SETH, a weakening of the SETH that was recently postulated by Lampis [210]. Note that this implies that an $\mathcal{O}^\star((3-\varepsilon)^{\mathrm{pw}})$ algorithm would falsify not just the standard form of the SETH (for k-CNF formulas), but (among others) also the SETH for circuits of depth εn [211] and the Set Cover Conjecture [91]. Our reduction follows along the lines of the one for Bounded Degree Vertex Deletion of Theorem 3.1, though adapted to the machinery of the pw-SETH. We remark that concurrently and independently to our work, Bojikian, Chekan, and Kratsch [46] recently showed that an $\mathcal{O}^\star((3-\varepsilon)^{\mathrm{ctw}})$ algorithm for Induced Matching would falsify the SETH. In comparison, our result is stronger in two aspects, as not only do we base the hardness on the weaker pw-SETH, but we in fact show the equivalence to it. Moving on, we consider the parameterization by the clique-width cw and develop an (optimal due to our previous result) $\mathcal{O}^\star(3^{\mathrm{cw}})$ algorithm based on dynamic programming. Although obtaining an algorithm of running time $\mathcal{O}^\star(6^{\mathrm{cw}})$ is quite straightforward, improving this to our obtained $\mathcal{O}^\star(3^{\mathrm{cw}})$ running time requires some extra machinery. In particular, to bypass the bottleneck of the computation time in the union nodes, we employ ideas from [44, 281] and consider *merged states* while solving the counting version of the problem. Doing so allows us to subsequently use the FFT technique introduced by Cygan and Pilipczuk [95] and populate the whole table in quasilinear time. Next we move towards the Acyclic Matching problem. We first provide a more careful analysis of the recent algorithm by Chaudhary and Zehavi [61] and show that in fact it runs in time $\mathcal{O}^\star(5^{\mathrm{tw}})$. Our main result however is to show that improving over this, even for the parameterization by pathwidth, results in the pw-SETH failing.

Related Work. Both Induced Matching and Acyclic Matching have been extensively studied with respect to their polynomial-time solvability in specific graph classes. We mention in passing that both problems remain NP-hard even in very restricted graph classes; for the former, such results are known for bipartite graphs of maximum degree 3 [233] and planar graphs of maximum degree 4 [193], while for the latter for planar bipartite graphs of maximum degree 3 [161]. For further results in specific graph classes we refer to [51, 52, 54, 55, 58, 97, 152, 160, 191, 193, 195, 201, 233, 293] for Induced Matching and to [20, 161, 256, 257] for Acyclic Matching. Exact exponential-time algorithms for Induced Matching have been proposed in the literature [59, 291], while both problems have also been studied with respect to their (in)approximability;

see [18, 56, 57, 70, 71, 107, 111, 139, 222, 254, 259] and [19, 20, 140, 256] respectively.

From the viewpoint of Parameterized Complexity, both Induced Matching and Acyclic Matching are W[1]-hard parameterized by the natural parameter ℓ even in bipartite graphs, yet become FPT for various graph classes, including planar graphs [96, 161, 245]; as a matter of fact, in case of Induced Matching in planar graphs, various kernelization algorithms have been developed [113, 180]. Recently, Chaudhary and Zehavi [62] also provided parameterized inapproximability results. Below guarantee parameterizations have also been studied [62, 194, 246, 290].

Regarding structural parameterizations, both problems are FPT parameterized by the treewidth of the input graph, denoted by tw. For Induced Matching, Moser and Sikdar [245] developed a $\mathcal{O}^\star(4^{\mathrm{tw}})$-time DP algorithm which was subsequently improved to $\mathcal{O}^\star(3^{\mathrm{tw}})$ by Chaudhary and Zehavi [61] who also showed a $\mathcal{O}^\star((\sqrt{6}-\varepsilon)^{\mathrm{pw}})$ lower bound under the SETH (pw denotes the pathwidth). As for Acyclic Matching, Hajebi and Javadi [161] noticed that the problem is expressible in MSO_2, thus it is FPT parameterized by treewidth due to Courcelle's theorem [85]; Chaudhary and Zehavi [61] later provided an explicit algorithm of complexity $\mathcal{O}^\star(6^{\mathrm{tw}})$ as well as a $\mathcal{O}^\star((3-\varepsilon)^{\mathrm{pw}})$ lower bound under the SETH. For structural parameters apart from treewidth, it is known that Induced Matching is expressible in MSO_1 logic [195], thus FPT parameterized by clique-width due to standard metatheorems [86], and Acyclic Matching is FPT parameterized by modular-width [161], while neither problem admits (under standard assumptions) a polynomial kernel parameterized by either the vertex cover number or the distance to clique of the input graph [62, 153].

4.1. Induced Matching

In this section we consider the Induced Matching problem. Chaudhary and Zehavi [61] developed an algorithm of running time $\mathcal{O}^\star(3^{\mathrm{tw}})$, where tw denotes the treewidth of the input graph, and left as an open question whether this is optimal. As our first result, in Theorem 4.1 we show that this is indeed the case, as obtaining an algorithm of running time $\mathcal{O}^\star((3-\varepsilon)^{\mathrm{pw}})$, or even $\mathcal{O}^\star((3-\varepsilon)^{\mathrm{ctw}})$, is equivalent to falsifying the pw-SETH. Next, we develop a DP algorithm of running time $\mathcal{O}^\star(3^{\mathrm{cw}})$ for the problem, where cw denotes the clique-width of the input graph. Since $\mathrm{cw}(G) \leq \mathrm{pw}(G)+2$ for any graph G, our algorithm is optimal under the pw-SETH due to Theorem 4.1.

4.1.1. Lower Bound

We first state the main theorem of this section, which we then prove in Lemmas 4.2 and 4.5.

Theorem 4.1. *The following are equivalent:*

- *there exists an algorithm deciding* Induced Matching *in time* $\mathcal{O}^\star((3-\varepsilon)^{\mathrm{pw}})$*, where* pw *denotes the pathwidth of the input graph, for some* $\varepsilon > 0$*,*

- *there exists an algorithm deciding* Induced Matching *in time* $\mathcal{O}^\star((3-\varepsilon)^{\mathrm{ctw}})$, *where* ctw *denotes the cutwidth of the input graph, for some* $\varepsilon > 0$,
- *the pw-SETH is false.*

Lemma 4.2. *If the pw-SETH is false, then there exists* $\varepsilon > 0$ *and an algorithm that solves* Induced Matching *in time* $\mathcal{O}^\star((3-\varepsilon)^{\mathrm{pw}})$, *where* pw *denotes the pathwidth of the input graph.*

Proof. Recall that in the MaxW-CSP problem we are given a CSP instance of arity r and over an alphabet $\mathcal{B}$, as well as a weight function $w\colon \mathcal{B} \to \mathbb{N}$ and an integer w_t, and we want to decide whether there exists a satisfying assignment whose total weight is at least w_t. The weight of an assignment is defined as the sum of the weights of the assignments of its variables.

Given an instance (G, ℓ) of Induced Matching and a path decomposition of $G = (V, E)$ of width pw, we will reduce it to an instance ψ of MaxW-CSP of arity $r = \mathcal{O}(1)$ and over an alphabet $\mathcal{B}$ of size 3, where the pathwidth of the primal graph of ψ is $\mathrm{pw} + \mathcal{O}(1)$. If the pw-SETH is false, then by [210, Theorem 3.2] there exist $\varepsilon > 0$ such that there is an algorithm that decides ψ in time $\mathcal{O}((|\mathcal{B}|-\varepsilon)^{\mathrm{pw}+\mathcal{O}(1)}|\psi|^{\mathcal{O}(1)})$, which allows us to obtain the desired running time for Induced Matching.

Suppose we have a nice path decomposition of G of width pw with the bags numbered $B_1, \ldots, B_t$. Let the vertices of G be numbered, that is, $V = \{v_1, \ldots, v_n\}$. We can assume that there exists an injective function $b\colon E \to [t]$ which maps each edge of G to the index of a bag that contains both of its endpoints (b can be made injective by repeating bags if necessary). Furthermore, we can assume that b maps edges to bags which are not introducing new vertices, again by repeating bags. We construct a CSP instance ψ over an alphabet $\mathcal{B} = \{0, 1, 2\}$ as follows:

1. For each $v_i \in V$ we introduce t variables $x_{i,j}$ for $j \in [t]$.
2. For each $v_i \in V$ and $j \in [t-1]$ we add the following constraints:
   - $x_{i,j} \neq 2 \implies x_{i,j+1} = x_{i,j}$,
   - $x_{i,j} = 2 \implies x_{i,j+1} \neq 0$,
   - If no edge e has $b(e) = j$ or for some edge e we have $b(e) = j$ but e is not incident on v_i, then we also add the constraint $x_{i,j+1} = x_{i,j}$.
3. For each $v_i \in V$, if the last bag containing v_i is B_j, then for all $j' \in [j, t-1]$ we add the constraint $x_{i,j'+1} = x_{i,j'}$.
4. For each $e \in E$ let $j = b(e)$ and $e = \{v_{i_1}, v_{i_2}\}$ with $v_{i_1}, v_{i_2} \in B_j$. We add the constraints
   - $(x_{i_1,j} = 0) \vee (x_{i_2,j} = 0) \vee (x_{i_1,j} = x_{i_2,j} = 2)$,
   - $(x_{i_1,j} = 0) \vee (x_{i_2,j} = 0) \implies ((x_{i_1,j+1} = x_{i_1,j}) \wedge (x_{i_2,j+1} = x_{i_2,j}))$,
   - $(x_{i_1,j} = x_{i_2,j} = 2) \implies (x_{i_1,j+1} = x_{i_2,j+1} = 1)$.

5. For each $v_i \in V$ we add the constraints $x_{i,t} \neq 2$ and $x_{i,1} \neq 1$.

Intuitively, we have a one-to-one mapping between assignments of the variables of ψ and the induced matchings of G so that (i) $x_{i,j} = 0$ if v_i does not belong to the considered induced matching, (ii) $x_{i,j} = 1$ if v_i belongs to the considered induced matching and its incident edge has been introduced by bag B_j, and (iii) $x_{i,j} = 2$ if v_i belongs to the considered induced matching and its incident edge is introduced *after* bag B_j.

We define a weight function $w\colon \mathcal{B} \to \{0, 1\}$ such that $w(1) = w(2) = 1$ and $w(0) = 0$. The target weight is $w_t = 2\ell \cdot t$, where ℓ is the target value for the given Induced Matching instance.

▷ Claim 4.3. If G has an induced matching of size ℓ, then ψ has a satisfying assignment of weight at least $2\ell \cdot t$.

Proof of the claim. Let $M \subseteq E(G)$ be an induced matching of G of size ℓ. For each $v_i \notin V_M$ we assign to $x_{i,j}$ the value 0, for all $j \in [t]$. As for the vertices of V_M, let $e \in M$ with $e = \{v_{i_1}, v_{i_2}\}$ and $b(e) = j$. Then we assign to both $x_{i_1,j'}, x_{i_2,j'}$ the value 2 for all $j' \in [j]$, and the value 1 for all $j' \in [j + 1, t]$.

It is easy to see that this assignment is of weight $|V_M| \cdot t \geq 2\ell \cdot t$. We argue that it is also satisfying. It is easy to see that the constraints introduced in Steps 2, 3, and 5 are satisfied. To see that the constraints introduced in Step 4 are also satisfied, let $e \in E$ with $e = \{v_{i_1}, v_{i_2}\}$ and $b(e) = j$. If $e \in M$, then the corresponding constraints can be easily seen to be satisfied. Assume that $e \notin M$. Since M is an induced matching, it holds that at most one endpoint of e belongs to V_M, consequently it holds that $(x_{i_1,j} = 0) \vee (x_{i_2,j} = 0)$ and the constraints are once again satisfied. ◁

▷ Claim 4.4. If ψ has a satisfying assignment of weight at least $2\ell \cdot t$, then G has an induced matching of size ℓ.

Proof of the claim. Let $i \in [n]$ such that ψ assigns to $x_{i,t}$ the value 0. In that case, due to the constraints in Step 1, it follows that ψ assigns to $x_{i,j}$ the value 0 for all $j \in [t]$. Let $I \subseteq [n]$ such that $x_{i,t}$ gets value 1 by ψ; due to the discussion so far as well as the constraints introduced in Step 5, it follows that $|I| \geq 2\ell$. It suffices to show that $G' = G[\{v_i : i \in I\}]$ is an 1-regular subgraph of G. Let $i \in I$. First assume that there is no edge incident with v_i in G'. In that case, for any edge $e = \{v_i, v_{i'}\}$ in G it holds that $i' \notin I$, that is, $x_{i',j}$ receives value 0 for all $j \in [t]$. Consequently, by the constraints of Steps 2, 4, and 5, $x_{i,t}$ gets value 2, a contradiction. Now assume that there are more than one edges incident with v_i in G', and let $e_1, e_2 \in E$ such that $e_1 = \{v_i, v_{i_1}\}$ and $e_2 = \{v_i, v_{i_2}\}$ with $i_1, i_2 \in I$, while $b(e_1) = j_1$ and $b(e_2) = j_2 > j_1$. In that case, due to the constraints in Steps 2, 3, and 4 it holds that $x_{i,j}$ gets value 1 for all $j \in [j_1 + 1, t]$, in which case it follows that $x_{i,j_2} = 1$ and $x_{i_2,j_2} \neq 0$, falsifying the constraint $(x_{i,j_2} = 0) \vee (x_{i_2,j_2} = 0) \vee (x_{i,j_2} = x_{i_2,j_2} = 2)$, a contradiction. ◁

Finally, let us argue about the pathwidth of ψ. Take the path decomposition of G and replace in each B_j each vertex $v_i \in B_j$ with the variable $x_{i,j}$. For each $e \in E$ such that $e = \{v_{i_1}, v_{i_2}\}$ and $b(e) = j$ also place $x_{i_1,j+1}, x_{i_2,j+1}$ in B_j. This covers the constraints of

Step 4, increasing the width by at most 2 (since b is injective). For each $j \in [t-1]$ we insert between B_j and B_{j+1} a sequence of bags which start with B_j and at each step add a variable $x_{i,j+1}$ and then remove $x_{i,j}$, one by one. Finally, to cover the remaining variables and constraints of Steps 2, 3, and 5, it suffices for each $v_i \in V$ that appears in the interval $B_{j_1}, \ldots, B_{j_2}$ to insert to the left of B_{j_1} a sequence of bags that contain all of B_{j_1} and a path decomposition of the path formed by $x_{i,1}, \ldots, x_{i,j_1}$, and similarly after B_{j_2}. □

For the other direction, we reuse the gadgets of the reduction of Theorem 3.1 which proves that Bounded Degree Vertex Deletion when $\Delta = 1$ cannot be solved in $\mathcal{O}^\star((3-\varepsilon)^{\mathrm{pw}})$ under the SETH; the latter problem asks, given a graph G, $\Delta \geq 0$, and $k > 0$, whether there exists a set $S \subseteq V(G)$ of size at most k such that the maximum degree of $G - S$ is at most Δ.

Lemma 4.5. *If there exists $\varepsilon > 0$ and an algorithm that solves* Induced Matching *in time $\mathcal{O}^\star((3-\varepsilon)^{\mathrm{ctw}})$, where* ctw *denotes the cutwidth of the input graph, then the pw-SETH is false.*

Proof. We present a reduction from the 4-CSP problem of Corollary 2.3 to Induced Matching. We are given a CSP instance ψ whose variables take values from [3], a partition of its variables into two sets V_1, V_2, a path decomposition $B_1, \ldots, B_t$ of ψ of width p such that each bag contains $\mathcal{O}(\log p)$ variables of V_2, and an injective function b mapping each constraint to a bag that contains its variables. We want to construct an Induced Matching instance (G, ℓ) with $\mathrm{ctw}(G) = p + o(p)$, such that if ψ is satisfiable, then G has an induced matching of size ℓ, while if G has an induced matching of size ℓ, ψ admits a monotone satisfying multi-assignment which is consistent for V_2. We assume that the variables of ψ are numbered $X = \{x_1, \ldots, x_n\}$, the constraints are $c_1, \ldots, c_m$, and that the given path decomposition is nice. We construct G as follows.

Block and Variable Gadgets. For every variable x_i and every bag with $x_i \in B_j$, where $j \in [j_1, j_2]$, construct a *block gadget* $\hat{B}_{i,j}$. Here we consider two cases, depending on whether $x_i \in V_1$ or $x_i \in V_2$.

If $x_i \in V_1$ then we construct the block gadget $\hat{B}_{i,j}$ as a path on vertices $p_1^{i,j}, p_2^{i,j}, p_3^{i,j}, p_4^{i,j}$. Furthermore, for all $j \in [j_1, j_2 - 1]$, we identify the vertices $p_4^{i,j}, p_1^{i,j+1}$ so that they are the same vertex. The resulting path is called the *variable gadget* of x_i, and is denoted by $\hat{P}_i$. See also Figure 4.1.

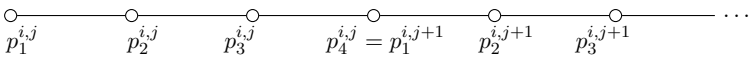


Figure 4.1.: Part of the construction of the variable gadget $\hat{P}_i$, where $x_i \in V_1$.

As for the case where $x_i \in V_2$, then the block gadget $\hat{B}_{i,j}$ is a clique on vertices $p^{i,j}, p_1^{i,j}, p_2^{i,j}, p_3^{i,j}$. Furthermore, for all $j \in [j_1, j_2 - 1]$ and $k \in [3]$, we add all edges between $p_k^{i,j}$ and $\{p_1^{i,j+1}, p_2^{i,j+1}, p_3^{i,j+1}\} \setminus \{p_k^{i,j+1}\}$, and we call the *variable gadget* $\hat{P}_i$ this sequence of serially connected block gadgets. See also Figure 4.2.

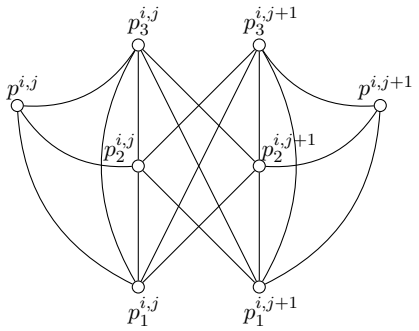


Figure 4.2.: Part of the construction of the variable gadget $\hat{P}_i$, where $x_i \in V_2$.

Constraint Gadget. This gadget is responsible for determining constraint satisfaction, based on the choices made in the rest of the graph. Let c be a constraint of ψ, where $b(c) = j$ and c involves variables $x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4} \in B_j$. Consider the set $\mathcal{S}_c$ of the at most 3^4 satisfying assignments of c. We construct the *constraint gadget* $\hat{C}_c$ as follows. For each $\sigma \in \mathcal{S}_c$ construct a vertex $y_{c,\sigma}$. Additionally construct a vertex w_c, and add edges so that w_c along with the vertices $y_{c,\sigma}$, for $\sigma \in \mathcal{S}_c$, form a clique. For each $\sigma \in \mathcal{S}_c$ and $\alpha \in [4]$, consider the following cases:

- if $x_{i_\alpha} \in V_1$ and $\sigma(x_{i_\alpha}) = 1$, then connect $y_{c,\sigma}$ to $p_3^{i,j}$,
- if $x_{i_\alpha} \in V_1$ and $\sigma(x_{i_\alpha}) = 2$, then connect $y_{c,\sigma}$ to $p_1^{i,j}$ and $p_4^{i,j}$,
- if $x_{i_\alpha} \in V_1$ and $\sigma(x_{i_\alpha}) = 3$, then connect $y_{c,\sigma}$ to $p_2^{i,j}$,
- if $x_{i_\alpha} \in V_2$ and $\sigma(x_{i_\alpha}) = k \in [3]$, then connect $y_{c,\sigma}$ to vertices $\{p_1^{i,j}, p_2^{i,j}, p_3^{i,j}\} \setminus \{p_k^{i,j}\}$.

This completes the construction of G. We set $\ell = L_1 + L_2 + m$, where L_i denotes the number of block gadgets constructed due to variables belonging to V_i, and m is the number of constraints of ψ. In that case, (G, ℓ) is the constructed instance of Induced Matching.

▷ Claim 4.6. If ψ is satisfiable, then G has an induced matching of size ℓ.

Proof of the claim. Let σ be a satisfying assignment for ψ. For each $x_i \in V_2$ we select into our solution M the edge $\{p^{i,j}, p^{i,j}_{\sigma(x_i)}\}$, for all $j \in [t]$ with $x_i \in B_j$. For each $x_i \in V_1$ and $j \in [t]$ such that $x_i \in B_j$ we select into M the edge $\{p^{i,j}_{\sigma(x_i)}, p^{i,j}_{\sigma(x_i)+1}\}$. Finally, for each constraint c, we select into M the edge $\{y_{c,\sigma_c}, w_c\}$, where $\sigma_c \in \mathcal{S}_c$ is the restriction of σ to the variables of c.

M is a matching of size ℓ. It remains to argue that it is an induced matching. To this end, observe that among the vertices of V_M, none of the vertices belonging to a constraint gadget is neighbors with vertices of V_M that belong to a block gadget. The statement then easily follows. ◁

▷ Claim 4.7. If G has an induced matching of size ℓ, then ψ admits a monotone satisfying multi-assignment which is consistent for V_2.

Proof of the claim. Let M be an induced matching of G of size at least ℓ. Let c be a constraint of ψ. We assume without loss of generality that either M does not contain any edge incident with vertices of $\hat{C}_c$, or there exists a $\sigma_c \in \mathcal{S}_c$ such that $\{y_{c,\sigma_c}, w_c\} \in M$. Indeed, assume there exists $e \in M$ such that $y_{c,\sigma_0} \in e$ for some $\sigma_0 \in \mathcal{S}_c$, while $w_c \notin e$. Then it holds that V_M contains no other vertex of $\hat{C}_c$, as its vertices form a clique. Consequently, $(M \setminus \{e\}) \cup \{\{y_{c,\sigma_0}, w_c\}\}$ is also an induced matching of the same size.

Now consider a variable gadget $\hat{P}_i$. First consider the case where $x_i \in V_1$. It is easy to see that M contains at most 1 edge per block gadget, with all edges having both of their endpoints in the same block gadget. Now assume that $x_i \in V_2$ and $x_i \in B_j$ for all $j \in [j_1, j_2]$. Assume that there exists an edge $e \in M$ such that $e = \{p_k^{i,j}, p_{k'}^{i,j+1}\}$, with $j \in [j_1, j_2 - 1]$ and $k, k' \in [3]$. Then, it follows that no other vertex of $\hat{B}_{i,j}$ and $\hat{B}_{i,j+1}$ belongs to V_M, and exchanging $\{p_k^{i,j}, p_{k'}^{i,j+1}\}$ with edges $\{p_k^{i,j}, p^{i,j}\}$ and $\{p_{k'}^{i,j+1}, p^{i,j+1}\}$ results in an induced matching of larger size. In the following assume that for M it holds that any edge incident with vertices of $\hat{B}_{i,j}$ has both of its endpoints in said gadget. Notice that M contains at most one edge from each such block gadget.

Consequently, it follows that M contains at most $L_1 + L_2 + m = \ell$ edges, where L_i denotes the number of block gadgets constructed due to variables belonging to V_i, and m is the number of constraints of ψ. It follows that $|M| = \ell$, and M contains exactly one edge per block gadget as well as exactly one edge per constraint gadget.

Consider a multi-assignment $\sigma \colon X \times [t] \to [3]$ over all pairs $x_i \in X$ and $j \in [t]$ with $x_i \in B_j$. In particular, consider two cases. If $x_i \in V_1$ and $\{p_k^{i,j}, p_{k+1}^{i,j}\} \in M$, then $\sigma(x_i, j) = k \in [3]$. For the case where $x_i \in V_2$, if $\{p_k^{i,j}, p^{i,j}\} \in M$, then $\sigma(x_i, j) = k \in [3]$.

We argue that σ is consistent for all $x_i \in V_2$. Fix $x_i \in V_2$ and $j \in [j_1, j_2 - 1]$, where B_{j_1} and B_{j_2} denote the first and last bag that contain x_i. Notice that $\{p_k^{i,j}, p^{i,j}\} \in M$ implies that $p_{k'}^{i,j+1} \notin V_M$ where $k \in [3]$ and $k' \in [3] \setminus \{k\}$, thus $\sigma(x_i, j) = \sigma(x_i, j')$ for all $j, j' \in [j_1, j_2]$.

We next argue that σ is monotone for $x_i \in V_1$, where $x_i \in B_j$ for all $j \in [j_1, j_2]$. Let $j \in [j_1, j_2 - 1]$. If $\{p_3^{i,j}, p_4^{i,j}\} \in M$, then it follows that $p_2^{i,j+1} \notin V_M$, implying that $\{p_3^{i,j+1}, p_4^{i,j+1}\} \in M$. If on the other hand $\{p_2^{i,j}, p_3^{i,j}\} \in M$, then it follows that $p_1^{i,j+1} \notin V_M$, implying that either $\{p_2^{i,j+1}, p_3^{i,j+1}\} \in M$ or $\{p_3^{i,j+1}, p_4^{i,j+1}\} \in M$.

Finally, we argue that σ is satisfying. Consider a constraint c with $b(c) = j$, involving four variables from $V_{i_1}, V_{i_2}, V_{i_3}, V_{i_4}$. Let $\sigma_c \in \mathcal{S}_c$ such that $\{y_{c,\sigma_c}, w_c\} \in M$. We claim that σ_c must be consistent with our assignment. Indeed, if σ_c assigns value $k \in [3]$ to $x_i \in V_2$, then it follows that $(\{p_1^{i,j}, p_2^{i,j}, p_3^{i,j}\} \setminus \{p_k^{i,j}\}) \notin V_M$, thus $\{p_k^{i,j}, p^{i,j}\} \in M$. On the other hand, if σ_c assigns value $k \in [3]$ to $x_i \in V_1$, then

- if $k = 1$, then $p_3^{i,j} \notin V_M$, implying that $\{p_1^{i,j}, p_2^{i,j}\} \in M$,
- if $k = 2$, then $p_1^{i,j}, p_4^{i,j} \notin V_M$, implying that $\{p_2^{i,j}, p_3^{i,j}\} \in M$,
- if $k = 3$, then $p_2^{i,j} \notin V_M$, implying that $\{p_3^{i,j}, p_4^{i,j}\} \in M$.

This completes the proof. ◁

▷ Claim 4.8. It holds that the cutwidth of G is at most $p + \mathcal{O}(\log p)$.

Proof of the claim. Recall that $B_1, \ldots, B_t$ denotes the path decomposition of the primal graph of ψ. We construct a linear arrangement π of $V(G)$ by starting with an empty linear arrangement and then proceed as follows by always adding the vertices to the very right. First we iterate over $j = 1, \ldots, t$. For a fixed value of j, we next iterate over all $x_i \in B_j$. For a fixed $x_i \in B_j$, we place all unplaced[1] vertices of $\hat{B}_{i,j}$: if $x_i \in V_1$, then we place them consecutively, that is, like $p_1^{i,j}, p_2^{i,j}, p_3^{i,j}, p_4^{i,j}$; otherwise, i.e., if $x_i \in V_2$, we place them in an arbitrary order. After processing all $x_i \in B_j$, if there exists a constraint c with $b(c) = j$, then we add all vertices of $\hat{C}_c$ in an arbitrary order. This completes the description of π.

We now argue that the cutwidth of π is $p + \mathcal{O}(\log p)$. Consider an arbitrary cut. We proceed to bound the number of edges crossing this cut:

- Since every bag of the path decomposition of ψ has at most p variables of V_1, there are at most p edges whose endpoints both belong to a variable gadget due to a variable of V_1.
- Since every bag of the path decomposition of ψ has $\mathcal{O}(\log p)$ variables of V_2, there are $\mathcal{O}(\log p)$ edges whose endpoints both belong to a variable gadget due to a variable of V_2.
- Since every constraint gadget contains at most $(3^4)^2$ edges and no two constraint gadgets are connected, there are at most 3^8 edges whose endpoints both belong to a constraint gadget.
- Since there are at most $3^4 \cdot 8$ edges connecting a constraint gadget with the block gadgets, while any vertex of a block gadget is connected with at most 2 constraint gadgets, there are at most $3^4 \cdot 16$ edges whose one endpoint belongs a block gadget and the other to a constraint gadget.

Summing over the number of those edges results in the stated upper bound. ◁

Lemma 4.5 now follows by Claims 4.6 to 4.8. □

4.1.2. Algorithm

In this section we prove the following theorem.

Theorem 4.9. *There is an algorithm that, given a graph G and an irredundant clique-width expression ψ of G of width* cw*, counts the number of induced matchings of G of each size $\ell \in [0, n/2]$ in time $\mathcal{O}^\star(3^{\mathrm{cw}})$.*

[1] Note that for $j \geq 2$ and $x_i \in B_j \cap V_1$, the vertex $p_1^{i,j}$ is already placed into the linear arrangement as it coincides with $p_4^{i,j-1}$.

Proof. Before describing our algorithm we start with some definitions and notation. Given a tuple σ, we refer to its w-th entry by $\sigma(w)$. Let H be a graph and $S \subseteq V(H)$ a subset of its vertices. We say that a vertex $v \in S$ is *unsaturated* in $H[S]$ if its degree is $\deg_{H[S]}(v) = 0$, else if $\deg_{H[S]}(v) = 1$ it is *saturated* in $H[S]$ instead. We say that S is a *partial matching* of H if the maximum degree of $H[S]$ is at most 1, in which case all of its vertices are either saturated or unsaturated in $H[S]$. The *size of a partial matching* S of H is equal to the number of its vertices, that is, $|S|$. Finally, we say that a partial matching S of H is *extensible* if there exists an induced matching M of G such that $V_M \cap V(H) = S$.

Let $\mathcal{H}$ denote the set of all cw-labeled graphs generated by subexpressions of ψ. For $H \in \mathcal{H}$ and $i \in [\mathrm{cw}]$, we define the partial function $\mathrm{sgn}_{H,i} \colon 2^{V(H)} \to \{0,1,2\}$ for $S \subseteq V(H)$ as follows:

$$\mathrm{sgn}_{H,i}(S) = \begin{cases} 0 & \text{if } S \cap \mathrm{lab}_H^{-1}(i) = \varnothing, \\ 1 & \text{if } S \cap \mathrm{lab}_H^{-1}(i) \neq \varnothing \text{ and } \deg_{H[S]}(v) = 1 \text{ for all } v \in S \cap \mathrm{lab}_H^{-1}(i), \\ 2 & \text{if } S \cap \mathrm{lab}_H^{-1}(i) = \{v\} \text{ and } \deg_{H[S]}(v) = 0. \end{cases}$$

We further define the tuple $\mathrm{sgn}_H(S) = \Big(\mathrm{sgn}_{H,1}(S), \ldots, \mathrm{sgn}_{H,\mathrm{cw}}(S)\Big)$ to be the *signature* of S in H if $\mathrm{sgn}_{H,i}(S)$ is well-defined for all $i \in [\mathrm{cw}]$. Notice that $\mathrm{sgn}_H(S)$ is only defined for a subset of the partial matchings of H, while there are at most 3^{cw} different signatures. For improved readability, we sometimes refer to tuples $\sigma \in \{0,1,2\}^{\mathrm{cw}}$ as signatures. We first argue that for any induced matching M of G, the signature of $V_M \cap V(H)$ is well-defined for all $H \in \mathcal{H}$.

▷ Claim 4.10. Let M be an induced matching of G. For all $H \in \mathcal{H}$ it holds that $\mathrm{sgn}_H(V_M \cap V(H))$ is well-defined.

Proof of the claim. Notice that for all $H \in \mathcal{H}$ it holds that $H[V_M \cap V(H)]$ is a subgraph of $G[V_M]$, therefore the maximum degree of $H[V_M \cap V(H)]$ is at most 1. Consequently, $V_M \cap V(H)$ is a partial matching of H. Fix $H \in \mathcal{H}$ and let $S = V_M \cap V(H)$. It suffices to prove that $\mathrm{sgn}_{H,i}(S)$ is well-defined for all $i \in [\mathrm{cw}]$.

For the sake of contradiction, assume that $\mathrm{sgn}_{H,i}(S)$ is not well-defined for some $i \in [\mathrm{cw}]$. Since S is a partial matching of H, that can only be the case when $S \cap \mathrm{lab}_H^{-1}(i)$ contains at least two vertices, say v, v', and at least one of them, say v, is unsaturated in $H[S]$, that is, $\deg_{H[S]}(v) = 0$. Let $u \in V_M$ such that $\{v,u\} \in M$. Since $\deg_{H[S]}(v) = 0$ it holds that either $u \notin V(H)$ or $\{v,u\} \notin E(H)$. This means that at some point further in the clique-expression there exists a join operation with $H_1, H_2 \in \mathcal{H}$ and $H_2 = \eta_{i',j'}(H_1)$ that adds all edges between vertices of labels i' and j', with v, v' having label i' and u having label j'. In that case however, it holds that the degree of $u \in V_M \cap V(H_2)$ in $H_2[V_M \cap V(H_2)]$ is at least 2 as it is adjacent to both v and v', a contradiction. ◁

Our algorithm proceeds by dynamic programming along ψ in a bottom-up fashion. In particular, for every $H \in \mathcal{H}$ it stores a table $\mathrm{DP}_H[\cdot,\cdot]$ where, on a high-level, for $k \in [0,n]$ and $\sigma \in \{0,1,2\}^{\mathrm{cw}}$, $\mathrm{DP}_H[k,\sigma]$ contains the number of partial matchings S of H of size k and signature σ, that is, with $|S| = k$ and $\mathrm{sgn}_H(S) = \sigma$. We now proceed to describe

how to populate the DP tables, and in Lemmas 4.11, 4.13, and 4.15 we prove that they indeed contain the number of partial matchings of appropriate size and signature.

Singleton $H = i(v)$. For $H = i(v)$, notice that $\mathrm{lab}_H^{-1}(i) = \{v\}$ and $\mathrm{lab}_H^{-1}(w) = \varnothing$ for all $w \in [\mathrm{cw}] \setminus \{i\}$. Consequently, for $k \in [0, n]$ and $\sigma \in \{0,1,2\}^{\mathrm{cw}}$ we set

$$\mathrm{DP}_H[k,\sigma] = \begin{cases} 1 & \text{if } k = 0 \text{ and } \sigma(w) = 0 \text{ for all } w \in [\mathrm{cw}], \\ 1 & \text{if } k = 1 \text{ and } \sigma(w) = 0 \text{ for all } w \in [\mathrm{cw}] \setminus \{i\} \text{ and } \sigma(i) = 2, \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

Notice that all but two entries in $\mathrm{DP}_H[\cdot,\cdot]$ are 0, and by Equation (4.1) we can fill the table $\mathrm{DP}_H[\cdot,\cdot]$ in time $\mathcal{O}^\star(3^{\mathrm{cw}})$.

Joining labels with edges, $H = \eta_{i,j}(H')$. For each pair of distinct $i, j \in [\mathrm{cw}]$, we define a partial function $f_{i,j} \colon \{0,1,2\}^{\mathrm{cw}} \to \{0,1,2\}^{\mathrm{cw}}$. In particular, for $\sigma' \in \{0,1,2\}^{\mathrm{cw}}$ we set $f_{i,j}(\sigma') = \sigma$ such that $\sigma(w) = \sigma'(w)$ for all $w \in [\mathrm{cw}] \setminus \{i,j\}$ and

$$(\sigma(i), \sigma(j)) = \begin{cases} (\sigma'(i),\, \sigma'(j)) & \text{if } 0 \in \{\sigma'(i),\, \sigma'(j)\}, \\ (1,1) & \text{if } \sigma'(i) = \sigma'(j) = 2. \end{cases}$$

For a fixed size $k \in [0, n]$ and signature $\sigma \in \{0,1,2\}^{\mathrm{cw}}$ we set the value of $\mathrm{DP}_H[k,\sigma]$ to be

$$\mathrm{DP}_H[k,\sigma] = \sum_{\sigma' \in f_{i,j}^{-1}(\sigma)} \mathrm{DP}_{H'}[k,\sigma']. \tag{4.2}$$

If $f_{i,j}^{-1}(\sigma) = \varnothing$ we set $\mathrm{DP}_H[k,\sigma] = 0$. Notice that computing $f_{i,j}^{-1}(\sigma)$ requires $\mathcal{O}(\mathrm{cw})$ time, thus by Equation (4.2) we can fill the table $\mathrm{DP}_H[\cdot,\cdot]$ in time $\mathcal{O}^\star(3^{\mathrm{cw}})$.

Lemma 4.11. *Let $H = \eta_{i,j}(H')$. Assume that for all $k \in [0,n]$ and $\sigma \in \{0,1,2\}^{\mathrm{cw}}$, $\mathrm{DP}_{H'}[k,\sigma]$ is equal to the number of partial matchings S of H' such that $|S| = k$ and $\mathrm{sgn}_{H'}(S) = \sigma$. Then, $\mathrm{DP}_H[k,\sigma]$ is equal to the number of partial matchings S of H such that $|S| = k$ and $\mathrm{sgn}_H(S) = \sigma$.*

Proof. We show that the partial function $f_{i,j}$ describes precisely the effect of the operation $\eta_{i,j}$ on the signature of a partial matching S.

▷ Claim 4.12. Let $S \subseteq V(H')$ such that $\mathrm{sgn}_{H'}(S) = \sigma'$. Then, $\mathrm{sgn}_H(S) = \sigma$ if and only if $f_{i,j}(\sigma') = \sigma$.

Proof of the claim. First, notice that the labels of the vertices remain the same between H' and H, therefore it holds that $\mathrm{lab}_H(v) = \mathrm{lab}_{H'}(v)$ for all $v \in V(H')$. This implies that $S \cap \mathrm{lab}_H^{-1}(w) = S \cap \mathrm{lab}_{H'}^{-1}(w)$ for all $w \in [\mathrm{cw}]$. Since $H[S]$ is obtained from $H'[S]$ by adding all edges between vertices of label i and j, it follows that $\deg_{H[S]}(v) = \deg_{H'[S]}(v)$ for all $v \in S \setminus (\mathrm{lab}_H^{-1}(i) \cup \mathrm{lab}_H^{-1}(j))$, thus $\mathrm{sgn}_{H,w}(S) = \mathrm{sgn}_{H',w}(S)$ for all $w \in [\mathrm{cw}] \setminus \{i,j\}$. In that case, $\mathrm{sgn}_H(S)$ and $f_{i,j}(\sigma')$ agree on all labels $w \in [\mathrm{cw}] \setminus \{i,j\}$. It remains to argue for labels i and j.

Assume first that $0 \in \{\sigma'(i), \sigma'(j)\}$. It is easy to see that in that case the graphs $H[S]$ and $H'[S]$ are the same, consequently $\operatorname{sgn}_H(S) = \operatorname{sgn}_{H'}(S)$ follows, and by the definition of partial function $f_{i,j}$ we have that $\operatorname{sgn}_H(S) = f_{i,j}(\sigma')$.

Next, assume that $\sigma'(i) = \sigma'(j) = 2$. Then, it holds that both $S \cap \operatorname{lab}_H^{-1}(i)$ and $S \cap \operatorname{lab}_H^{-1}(i)$ are comprised of a single vertex that is of degree 0 in $H'[S]$, which implies that $\operatorname{sgn}_{H,i}(S) = \operatorname{sgn}_{H,j}(S) = 1$ since the same vertices have degree 1 in $H[S]$. Again we have that $\operatorname{sgn}_H(S) = f_{i,j}(\sigma')$.

Lastly, assume that $0 \notin \{\sigma'(i), \sigma'(j)\}$ and either $\sigma'(i) \neq 2$ or $\sigma'(j) \neq 2$. Assume without loss of generality that $\sigma'(i) = 1$ and $\sigma'(j) \in \{1, 2\}$, and let $v \in S \cap \operatorname{lab}_{H'}^{-1}(i)$ and $u \in S \cap \operatorname{lab}_{H'}^{-1}(j)$, where $\deg_{H'[S]}(v) = 1$. Since ψ is by definition an irredundant clique-width expression, it follows that $\{v, u\} \notin E(H')$. Consequently, the operation $\eta_{i,j}$ adds the edge $\{v, u\}$ in H, thus $\deg_{H[S]}(v) > \deg_{H'[S]}(v) = 1$. In that case $\operatorname{sgn}_{H,i}(S)$ is undefined, as is the case for $f_{i,j}(\sigma')$. ◁

Consequently, the number of partial matchings of H of size k and signature σ is exactly equal to the sum of the number of partial matchings of H' of size k and signature σ' over all $\sigma' \in f_{i,j}^{-1}(\sigma)$. The statement now follows by the hypothesis for $\mathrm{DP}_{H'}[\cdot, \cdot]$ and Equation (4.2). □

Relabeling, $H = \rho_{i\to j}(H')$. For each pair of distinct $i, j \in [\mathrm{cw}]$, we define a partial function $g_{i\to j} \colon \{0,1,2\}^{\mathrm{cw}} \to \{0,1,2\}^{\mathrm{cw}}$. In particular, for $\sigma' \in \{0,1,2\}^{\mathrm{cw}}$ we set $g_{i\to j}(\sigma') = \sigma$ such that $\sigma(w) = \sigma'(w)$ for all $w \in [\mathrm{cw}] \setminus \{i, j\}$, $\sigma(i) = 0$, and

$$\sigma(j) = \begin{cases} \max\{\sigma'(i), \sigma'(j)\} & \text{if } 0 \in \{\sigma'(i), \sigma'(j)\}, \\ 1 & \text{if } \sigma'(i) = \sigma'(j) = 1. \end{cases}$$

For a fixed size $k \in [0, n]$ and signature $\sigma \in \{0,1,2\}^{\mathrm{cw}}$ we set the value of $\mathrm{DP}_H[k, \sigma]$ to be

$$\mathrm{DP}_H[k, \sigma] = \sum_{\sigma' \in g_{i\to j}^{-1}(\sigma)} \mathrm{DP}_{H'}[k, \sigma']. \tag{4.3}$$

If $g_{i\to j}^{-1}(\sigma) = \varnothing$ we set $\mathrm{DP}_H[k, \sigma] = 0$. Notice that computing $g_{i\to j}^{-1}(\sigma)$ requires $\mathcal{O}(\mathrm{cw})$ time, thus by Equation (4.3) we can fill the table $\mathrm{DP}_H[\cdot, \cdot]$ in time $\mathcal{O}^\star(3^{\mathrm{cw}})$.

Lemma 4.13. *Let $H = \rho_{i\to j}(H')$. Assume that for all $k \in [0, n]$ and $\sigma \in \{0,1,2\}^{\mathrm{cw}}$, $\mathrm{DP}_{H'}[k, \sigma]$ is equal to the number of partial matchings S of H' such that $|S| = k$ and $\operatorname{sgn}_{H'}(S) = \sigma$. Then, $\mathrm{DP}_H[k, \sigma]$ is equal to the number of partial matchings S of H such that $|S| = k$ and $\operatorname{sgn}_H(S) = \sigma$.*

Proof. We show that the partial function $g_{i\to j}$ describes precisely the effect of the operation $\rho_{i\to j}$ on the signature of a partial matching S.

▷ Claim 4.14. Let $S \subseteq V(H')$ such that $\operatorname{sgn}_{H'}(S) = \sigma'$. Then, $\operatorname{sgn}_H(S) = \sigma$ if and only if $g_{i\to j}(\sigma') = \sigma$.

Proof of the claim. First, notice that the labels of all vertices with label different than i remain the same between H' and H, therefore it holds that $\mathrm{lab}_H(v) = \mathrm{lab}_{H'}(v)$ for all $v \in V(H') \setminus \mathrm{lab}_{H'}^{-1}(i)$. This implies that $S \cap \mathrm{lab}_H^{-1}(w) = S \cap \mathrm{lab}_{H'}^{-1}(w)$ for all $w \in [\mathrm{cw}] \setminus \{i, j\}$. Furthermore, notice that no new edge is added in H, consequently it holds that $\deg_{H[S]}(v) = \deg_{H'[S]}(v)$ for all $v \in V(H)$. This implies that $\mathrm{sgn}_{H,w}(S) = \mathrm{sgn}_{H',w}(S)$ for all $w \in [\mathrm{cw}] \setminus \{i, j\}$. Moreover, since H is obtained from H' by relabeling all vertices of label i to label j, it holds that $\mathrm{lab}_H^{-1}(j) = \mathrm{lab}_{H'}^{-1}(i) \cup \mathrm{lab}_{H'}^{-1}(j)$ and $\mathrm{lab}_H^{-1}(i) = \varnothing$, thus $\mathrm{sgn}_{H,i}(S) = 0$. In that case, $\mathrm{sgn}_H(S)$ and $f_{i,j}(\sigma')$ agree on all labels $w \in [\mathrm{cw}] \setminus \{j\}$. It remains to argue for label j.

Assume first that $0 \in \{\sigma'(i), \sigma'(j)\}$. In that case, it holds that either $S \cap \mathrm{lab}_{H'}^{-1}(i) = \varnothing$ or $S \cap \mathrm{lab}_{H'}^{-1}(j) = \varnothing$. Assume that $S \cap \mathrm{lab}_{H'}^{-1}(i) = \varnothing$, that is, $\mathrm{sgn}_{H',i}(S) = 0$, which implies that $S \cap \mathrm{lab}_H^{-1}(j) = S \cap \mathrm{lab}_{H'}^{-1}(j)$, thus $\mathrm{sgn}_{H,j}(S) = \mathrm{sgn}_{H',j}(S) = \max\{\mathrm{sgn}_{H',j}(S), \mathrm{sgn}_{H',i}(S)\}$. In a similar way, when $S \cap \mathrm{lab}_{H'}^{-1}(i) = \varnothing$ we get that $\mathrm{sgn}_{H',j}(S) = 0$ and $\mathrm{sgn}_{H,j}(S) = \mathrm{sgn}_{H',i}(S) = \max\{\mathrm{sgn}_{H',i}(S), \mathrm{sgn}_{H',j}(S)\}$. Thus, in this case we have that $\mathrm{sgn}_H(S) = f_{i,j}(\sigma')$.

Next, assume that $\sigma'(i) = \sigma'(j) = 1$. Then, it holds that for all $v_1 \in S \cap \mathrm{lab}_{H'}^{-1}(i)$ and $v_2 \in S \cap \mathrm{lab}_{H'}^{-1}(j)$ we have that $\deg_{H'[S]}(v_1) = \deg_{H'[S]}(v_2) = 1$. Since $\mathrm{lab}_H^{-1}(j) = \mathrm{lab}_{H'}^{-1}(i) \cup \mathrm{lab}_{H'}^{-1}(j)$ and for all $v \in V(H)$ it holds that $\deg_{H[S]}(v) = \deg_{H'[S]}(v)$, it follows that $\mathrm{sgn}_{H,j}(S) = 1$. Again we have that $\mathrm{sgn}_H(S) = f_{i,j}(\sigma')$.

Lastly, assume that $0 \notin \{\sigma'(i), \sigma'(j)\}$ and either $\sigma'(i) \neq 1$ or $\sigma'(j) \neq 1$. Assume without loss of generality that $\sigma'(i) = 2$ and $\sigma'(j) \in \{1, 2\}$. In that case, it holds that $S \cap \mathrm{lab}_{H'}^{-1}(i) = \{v\}$ with $\deg_{H'[S]}(v) = 0$ as well as $S \cap \mathrm{lab}_{H'}^{-1}(i) \neq \varnothing$. Consequently it holds that $S \cap \mathrm{lab}_H^{-1}(j)$ contains at least 2 vertices v_1, v_2, with $\deg_{H[S]}(v_1) = 0$, thus $\mathrm{sgn}_{H,j}(S)$ is undefined, as is the case for $f_{i,j}(\sigma')$. ◁

Consequently, the number of partial matchings of H of size k and signature σ is exactly equal to the sum of the number of partial matchings of H' of size k and signature σ' over all $\sigma' \in g_{i \to j}^{-1}(\sigma)$. The statement now follows by the hypothesis for $\mathrm{DP}_{H'}[\cdot, \cdot]$ and Equation (4.3). □

Disjoint union, $H = H_1 \oplus H_2$. For the Union nodes, an approach similar to the previous cases would yield a total running time of $\mathcal{O}^\star(9^{\mathrm{cw}})$ to populate the table $\mathrm{DP}_H[\cdot, \cdot]$, which can be further improved to $\mathcal{O}^\star(6^{\mathrm{cw}})$ by some slightly more careful analysis. In order to bring this down to $\mathcal{O}^\star(3^{\mathrm{cw}})$ we proceed in a different way by using the techniques introduced in [44, 95, 281].

We start with some definitions. Let $\tilde{\sigma} \in \{0, \tilde{1}, 2\}^{\mathrm{cw}}$ be a tuple with cw entries, each taking values over the set $\{0, \tilde{1}, 2\}$. We call any such tuple a *pseudosignature* and we say that the signature $\sigma \in \{0, 1, 2\}^{\mathrm{cw}}$ is *compatible* with the pseudosignature $\tilde{\sigma}$ if for all $w \in [\mathrm{cw}]$ it holds that (i) if $\tilde{\sigma}(w) = 0$ then $\sigma(w) = 0$, (ii) if $\tilde{\sigma}(w) = \tilde{1}$ then $\sigma(w) \in \{0, 1\}$, and (iii) if $\tilde{\sigma}(w) = 2$ then $\sigma(w) = 2$. Let for $H' \in \mathcal{H}$, $\widetilde{\mathrm{DP}}_{H'}[\cdot, \cdot]$ be the table which in entry $\widetilde{\mathrm{DP}}_{H'}[k, \tilde{\sigma}]$ contains the number of partial matchings in H' of size k whose signatures are compatible with $\tilde{\sigma}$. Our main idea is to first compute the tables $\widetilde{\mathrm{DP}}_{H_1}[\cdot, \cdot]$ and $\widetilde{\mathrm{DP}}_{H_2}[\cdot, \cdot]$, then from those compute the table $\widetilde{\mathrm{DP}}_H[\cdot, \cdot]$, and lastly use the latter to populate the table $\mathrm{DP}_H[\cdot, \cdot]$.

Step 1. Let $H_i \in \{H_1, H_2\}$. For $\lambda \in [0, \mathrm{cw}]$ we say that $\tilde{\sigma}_{\mathsf{m}} \in \{0, \tilde{1}, 2\}^{\lambda} \times \{0,1,2\}^{\mathrm{cw}-\lambda}$ is a *λ-mixed signature*. We say that the signature $\sigma \in \{0,1,2\}^{\mathrm{cw}}$ is *compatible* with the λ-mixed signature $\tilde{\sigma}_{\mathsf{m}}$ if the following hold:

- For all $w \in [\lambda]$ it holds that (i) if $\tilde{\sigma}_{\mathsf{m}}(w) = 0$ then $\sigma(w) = 0$, (ii) if $\tilde{\sigma}_{\mathsf{m}}(w) = \tilde{1}$ then $\sigma(w) \in \{0,1\}$, and (iii) if $\tilde{\sigma}_{\mathsf{m}}(w) = 2$ then $\sigma(w) = 2$.
- For all $w \in [\lambda+1, \mathrm{cw}]$, $\sigma(w) = \tilde{\sigma}_{\mathsf{m}}(w)$.

We proceed inductively. For $\lambda \in [0, \mathrm{cw}]$ let $\widetilde{\mathrm{DP}}^{\lambda}_{H_i}[\cdot,\cdot]$ denote a table taking values over $[0,n] \times \{0, \tilde{1}, 2\}^{\lambda} \times \{0,1,2\}^{\mathrm{cw}-\lambda}$, where intuitively $\widetilde{\mathrm{DP}}^{\lambda}_{H_i}[k, \tilde{\sigma}_{\mathsf{m}}]$ is equal to the number of partial matchings of H_i of size k whose signature is compatible with $\tilde{\sigma}_{\mathsf{m}}$. For $\lambda = 0$, this is precisely the table $\mathrm{DP}_{H_i}[\cdot,\cdot]$, while for $\lambda = \mathrm{cw}$ this will be the table $\widetilde{\mathrm{DP}}_{H_i}[\cdot,\cdot]$.

It suffices to explain how to compute $\widetilde{\mathrm{DP}}^{\lambda+1}_{H_i}[\cdot,\cdot]$ from the table $\widetilde{\mathrm{DP}}^{\lambda}_{H_i}[\cdot,\cdot]$ in time $\mathcal{O}^\star(3^{\mathrm{cw}})$, and repeat the whole procedure cw times. Let $k \in [0,n]$ and $\tilde{\sigma}_{\mathsf{m}} \in \{0, \tilde{1}, 2\}^{\lambda+1} \times \{0,1,2\}^{\mathrm{cw}-(\lambda+1)}$ be a $(\lambda+1)$-mixed signature. Consider two cases. If $\tilde{\sigma}_{\mathsf{m}}(\lambda+1) \in \{0,2\}$, then set $\widetilde{\mathrm{DP}}^{\lambda+1}_{H_i}[k, \tilde{\sigma}_{\mathsf{m}}] = \widetilde{\mathrm{DP}}^{\lambda}_{H_i}[k, \tilde{\sigma}_{\mathsf{m}}]$. Otherwise, it holds that $\tilde{\sigma}_{\mathsf{m}}(\lambda+1) = \tilde{1}$, and let $\tilde{\sigma}'_{\mathsf{m}}, \tilde{\sigma}''_{\mathsf{m}} \in \{0, \tilde{1}, 2\}^{\lambda} \times \{0,1,2\}^{\mathrm{cw}-\lambda}$ denote the λ-mixed signatures obtained from $\tilde{\sigma}_{\mathsf{m}}$ by substituting the $(\lambda+1)$-th entry with 1 and 0 respectively, that is, $\tilde{\sigma}'_{\mathsf{m}} = \tilde{\sigma}_{\mathsf{m}}[(\lambda+1) \mapsto 0]$ and $\tilde{\sigma}''_{\mathsf{m}} = \tilde{\sigma}_{\mathsf{m}}[(\lambda+1) \mapsto 1]$. In that case, set $\widetilde{\mathrm{DP}}^{\lambda+1}_{H_i}[k, \tilde{\sigma}_{\mathsf{m}}] = \widetilde{\mathrm{DP}}^{\lambda}_{H_i}[k, \tilde{\sigma}'_{\mathsf{m}}] + \widetilde{\mathrm{DP}}^{\lambda}_{H_i}[k, \tilde{\sigma}''_{\mathsf{m}}]$.

Step 2. Utilizing the tables $\widetilde{\mathrm{DP}}_{H_1}[\cdot,\cdot]$ and $\widetilde{\mathrm{DP}}_{H_2}[\cdot,\cdot]$ constructed in Step 1, along with making use of the FFT technique introduced by Cygan and Pilipczuk [95], we compute in time $\mathcal{O}^\star(3^{\mathrm{cw}})$ the table $\widetilde{\mathrm{DP}}_{H}[\cdot,\cdot]$, also indexed over $[0,n] \times \{0, \tilde{1}, 2\}^{\mathrm{cw}}$. Initially we set every entry of said table to 0.

In the following, fix $k \in [0,n]$ and a subset $A \subseteq [\mathrm{cw}]$. Let $\mathcal{S}_A \subseteq \{0, \tilde{1}, 2\}^{\mathrm{cw}}$ denote the set of pseudosignatures such that $\tilde{\sigma} \in \mathcal{S}_A$ iff for all $w \in [\mathrm{cw}]$, $\tilde{\sigma}(w) = \tilde{1} \iff w \in A$. We show how to compute all entries $\widetilde{\mathrm{DP}}_H[k, \tilde{\sigma}]$ with $\tilde{\sigma} \in \mathcal{S}_A$ in time $\mathcal{O}^\star(2^{\mathrm{cw}-|A|})$. Since $\sum_{A \subseteq [\mathrm{cw}]} 2^{\mathrm{cw}-|A|} = 3^{\mathrm{cw}}$ and k takes $n+1$ different values, this suffices for the stated time bound. Let $w_1, \ldots, w_{\mathrm{cw}-|A|}$ denote in increasing order the non $\tilde{1}$-valued coordinates of a pseudosignature in $\mathcal{S}_A$, that is, for all $j \in [\mathrm{cw}-|A|]$ and $\tilde{\sigma} \in \mathcal{S}_A$, we have that $\tilde{\sigma}(w_j) \neq \tilde{1}$.

We first fix $k_1, k_2 \in [0,k]$ such that $k_1 + k_2 = k$. Notice that there are $\mathcal{O}(n)$ such pairs of k_1, k_2. Intuitively, k_i represents the number of vertices of the partial matching of H_i that account towards the total size of the partial matching in H. Next we define a *binary encoding* $\mathrm{bin}_A \colon \mathcal{S}_A \to [0, 2^{(\mathrm{cw}-|A|)} - 1]$ of every pseudosignature in $\mathcal{S}_A$ such that for all $\tilde{\sigma} \in \mathcal{S}_A$ the j-th bit of the binary representation of $\mathrm{bin}_A(\tilde{\sigma})$, denoted by $\mathrm{bin}_{A,j}(\tilde{\sigma})$, is equal to

$$\mathrm{bin}_{A,j}(\tilde{\sigma}) = \begin{cases} 0 & \text{if } \tilde{\sigma}(w_j) = 0, \\ 1 & \text{if } \tilde{\sigma}(w_j) = 2. \end{cases}$$

For every entry $\widetilde{\mathrm{DP}}_{H_i}[k_i, \tilde{\sigma}]$ with $\tilde{\sigma} \in \mathcal{S}_A$ we introduce a monomial $\widetilde{\mathrm{DP}}_{H_i}[k_i, \tilde{\sigma}] \cdot x^{\mathrm{bin}_A(\tilde{\sigma})}$, and we set P_{A,H_i,k_i} to denote the polynomial comprised of all such monomials, that is,

$$P_{A,H_i,k_i} = \sum_{\tilde{\sigma} \in \mathcal{S}_A} \widetilde{\mathrm{DP}}_{H_i}[k_i, \tilde{\sigma}] \cdot x^{\mathrm{bin}_A(\tilde{\sigma})}.$$

Notice that the degree of P_{A,H_i,k_i} is at most $2^{\mathrm{cw}-|A|}$, while all coefficients (i.e., the number of partial solutions) are upper-bounded by 2^n. Let $P^{r_i}_{A,H_i,k_i}$ denote the restriction of P_{A,H_i,k_i} to monomials whose degree has *exactly* r_i out of the $\mathrm{cw}-|A|$ bits being 1. Fix $r_1, r_2 \in [0, \mathrm{cw}-|A|]$ and multiply polynomials $P^{r_1}_{A,H_1,k_1}$ and $P^{r_2}_{A,H_2,k_2}$ in $\mathcal{O}^\star(2^{\mathrm{cw}-|A|})$ time using FFT.[2] We iterate over all $\tilde{\sigma} \in \mathcal{S}_A$ having exactly $r_1 + r_2$ bits of $\mathrm{bin}_A(\tilde{\sigma})$ being 1 and check what coefficient, say s, stands in front of $x^{\mathrm{bin}_A(\tilde{\sigma})}$ in the resulting polynomial. We update the value of $\widetilde{\mathrm{DP}}_H[k,\tilde{\sigma}]$ by adding to its previous value $+s$. We repeat over all $r_1, r_2 \in [0, \mathrm{cw}-|A|]$, $k_1, k_2 \in [0,k]$, and $k \in [0,n]$, needing in total $\mathcal{O}^\star(2^{\mathrm{cw}-|A|})$ time. Finally, we fill the whole table $\widetilde{\mathrm{DP}}_H[\cdot,\cdot]$ in time $\mathcal{O}^\star(3^{\mathrm{cw}})$ by repeating over all $A \subseteq [\mathrm{cw}]$.

Step 3. Given $\widetilde{\mathrm{DP}}_H[\cdot,\cdot]$, we populate the values of $\mathrm{DP}_H[\cdot,\cdot]$ by repeating a procedure analogous to the one described in Step 1. The total running time is once again $\mathcal{O}^\star(3^{\mathrm{cw}})$.

Lemma 4.15. *Let $H = H_1 \oplus H_2$. Assume that for all $i \in \{1,2\}$, $k \in [0,n]$, and $\sigma \in \{0,1,2\}^{\mathrm{cw}}$, $\mathrm{DP}_{H_i}[k,\sigma]$ is equal to the number of partial matchings S of H_i such that $|S| = k$ and $\mathrm{sgn}_{H_i}(S) = \sigma$. Then, $\mathrm{DP}_H[k,\sigma]$ is equal to the number of partial matchings S of H such that $|S| = k$ and $\mathrm{sgn}_H(S) = \sigma$.*

Proof. It is easy to see that, for $H_i \in \{H_1, H_2\}$, for all $k \in [0,n]$ and $\tilde{\sigma} \in \{0,\tilde{1},2\}^{\mathrm{cw}}$, $\widetilde{\mathrm{DP}}_{H_i}[k,\tilde{\sigma}]$ is equal to the number of partial matchings S of H_i such that $|S| = k$ and $\mathrm{sgn}_{H_i}(S)$ is compatible with $\tilde{\sigma}$.

We proceed to show the correctness of Step 2 of the construction. Let S be a partial matching of H with $\sigma = \mathrm{sgn}_H(S)$. Moreover, let for $i \in \{1,2\}$, $S_i = S \cap V(H_i)$ with $\sigma_i = \mathrm{sgn}_{H_i}(S_i)$.

▷ Claim 4.16. For all $w \in [\mathrm{cw}]$ it holds that

- $\sigma(w) = 0 \iff \sigma_1(w) = \sigma_2(w) = 0$,
- $\sigma(w) = 1 \iff (\sigma_1(w), \sigma_2(w)) \in \{(0,1),(1,0),(1,1)\}$,
- $\sigma(w) = 2 \iff (\sigma_1(w), \sigma_2(w)) \in \{(0,2),(2,0)\}$.

Proof of the claim. Let $i \in \{1,2\}$. Notice that for all $v \in S_i$ it holds that $\mathrm{lab}_{H_i}(v) = \mathrm{lab}_H(v)$, thus $S \cap \mathrm{lab}_H^{-1}(w) = (S_1 \cap \mathrm{lab}_{H_1}^{-1}(w)) \cup (S_2 \cap \mathrm{lab}_{H_2}^{-1}(w))$. Furthermore, it holds that $\deg_{H_i[S_i]}(v) = \deg_{H[S]}(v)$.

For the first item, due to the previous paragraph we have that if $\sigma(w) = 0$, that is, $S \cap \mathrm{lab}_H^{-1}(w) = \varnothing$, then $S_1 \cap \mathrm{lab}_{H_1}^{-1}(w) = S_2 \cap \mathrm{lab}_{H_2}^{-1}(w) = \varnothing$, that is, $\sigma_1(w) = \sigma_2(w) = 0$. It can be easily shown that the converse is also true.

As for the second item, assume that $\sigma(w) = 1$, that is, $S \cap \mathrm{lab}_H^{-1}(w) \neq \varnothing$ and for all of its vertices v, it holds that $\deg_{H[S]}(v) = 1$. In that case, it follows that for all $v \in (S_1 \cap \mathrm{lab}_{H_1}^{-1}(w)) \cup (S_2 \cap \mathrm{lab}_{H_2}^{-1}(w))$, $\deg_{H_i[S_i]}(v) = 1$, with at most one $S_i \cap \mathrm{lab}_{H_i}^{-1}(w)$ being empty. Conversely, assuming that $(\sigma_1(w), \sigma_2(w)) \in \{(0,1),(1,0),(1,1)\}$, it follows that for every vertex v of the non-empty set $S \cap \mathrm{lab}_H^{-1}(w)$ it holds that $\deg_{H[S]}(v) = 1$.

[2] Recall that using FFT we can multiply two polynomials of degree n whose coefficients are upper-bounded by $2^{\mathcal{O}(n)}$ in $\mathcal{O}(n \log n)$ time [243].

Lastly, assume that $\tilde{\sigma}(w) = 2$, which implies that $S \cap \mathrm{lab}^{-1}(w) = \{v\}$ and $\deg_{H[S]}(v) = 0$. Assume that $v \in S_{i_1}$ and let S_{i_2} denote the other set among $\{S_1, S_2\}$. Then, it follows that $S_{i_1} \cap \mathrm{lab}^{-1}_{H_{i_1}}(w) = \{v\}$ with $\deg_{H_{i_1}[S_{i_1}]}(v) = 0$ and $S_{i_2} \cap \mathrm{lab}^{-1}_{H_{i_2}}(w) = \varnothing$, implying that $\sigma_{i_1}(w) = 2$ and $\sigma_{i_2}(w) = 0$. It can be easily shown that the converse is also true. $\lhd$

Recall that $\sigma \in \{0,1,2\}^{\mathrm{cw}}$ is compatible with $\tilde{\sigma} \in \{0,\tilde{1},2\}^{\mathrm{cw}}$ if for all $w \in [\mathrm{cw}]$ we have that (i) if $\tilde{\sigma}(w) = 0$, then $\sigma(w) = 0$, (ii) if $\tilde{\sigma}(w) = \tilde{1}$, then $\sigma(w) \in \{0,1\}$, and (iii) if $\tilde{\sigma}(w) = 2$, then $\sigma(w) = 2$. Fix $k \in [0,n]$ and $\tilde{\sigma} \in \{0,\tilde{1},2\}^{\mathrm{cw}}$, with $\tilde{\sigma} \in \mathcal{S}_A$ for some $A \subseteq [\mathrm{cw}]$. Claim 4.16 implies that the number of partial matchings of H of size k and signature compatible with $\tilde{\sigma}$ is equal to the number of partial matchings of H_1 of size k_1 and signature compatible with $\tilde{\sigma}_1$ times the number of partial matchings of H_2 of size k_2 and signature compatible with $\tilde{\sigma}_2$, summing over all $k_1 + k_2 = k$ and $\tilde{\sigma}_1, \tilde{\sigma}_2 \in \mathcal{S}_A$ such that for all $w \in [\mathrm{cw}]$ (i) if $\tilde{\sigma}(w) = 0$, then $\tilde{\sigma}_1(w) = \tilde{\sigma}_2(w) = 0$, and (ii) if $\tilde{\sigma}(w) = 2$, then $(\tilde{\sigma}_1(w), \tilde{\sigma}_2(w)) \in \{(0,2),(2,0)\}$. Notice that the latter condition imposed on the pseudosignatures $\tilde{\sigma}_1, \tilde{\sigma}_2$ is equivalent to demanding that the number r_1 of 1-bits of $\mathrm{bin}_A(\tilde{\sigma}_1)$ plus the number r_2 of 1-bits of $\mathrm{bin}_A(\tilde{\sigma}_2)$ is equal to the number of 1-bits of $\mathrm{bin}_A(\tilde{\sigma})$.

In that case, for fixed $r_1, r_2 \in [0, \mathrm{cw} - |A|]$, it holds that the coefficient of a monomial $x^{\mathrm{bin}_A(\tilde{\sigma})}$ of $P^{r_1}_{A,H_1,k_1} \cdot P^{r_2}_{A,H_2,k_2}$, where $\mathrm{bin}_A(\tilde{\sigma})$ has exactly $r_1 + r_2$ bits set to 1, is equal to the number of partial matchings of H of size $k = k_1 + k_2$ and signature compatible with $\tilde{\sigma} \in \mathcal{S}_A$ that occur due to the union of partial matchings of H_1 of size k_1 and signature compatible with $\tilde{\sigma}_1$ and partial matchings of H_2 of size k_2 and signature compatible with $\tilde{\sigma}_2$, where $\tilde{\sigma}_1, \tilde{\sigma}_2 \in \mathcal{S}_A$, $\mathrm{bin}_A(\tilde{\sigma}_i)$ has exactly r_i bits set to 1, and $\mathrm{bin}_A(\tilde{\sigma})$ has exactly $r_1 + r_2$ bits set to 1. Since we iterate over all $r_1, r_2 \in [0, \mathrm{cw} - |A|]$ and $k_1, k_2 \in [0,k]$, it holds that indeed $\widetilde{\mathrm{DP}}_H[k,\tilde{\sigma}]$ contains the number of partial matchings of H of size k and signature that is compatible with $\tilde{\sigma}$, for all $\tilde{\sigma} \in \mathcal{S}_A$.

As for Step 3, given all the previous discussion, it is easy to see that it indeed constructs the table $\mathrm{DP}_H[\cdot,\cdot]$ so that $\mathrm{DP}_H[k,\sigma]$ contains the number of partial matchings of H of size k and signature σ. This completes the proof. $\square$

Wrapping up. We argue about the correctness of the algorithm by a bottom-up induction on the clique-width cw-expression of G. We claim that for all $H \in \mathcal{H}$, $k \in [0,k]$, and $\sigma \in \{0,1,2\}^{\mathrm{cw}}$, $\mathrm{DP}_H[k,\sigma]$ is equal to the number of partial matchings in H of size k and signature σ. Let $H = i(v)$ be a singleton graph for some label $i \in [\mathrm{cw}]$. There are only two possible partial matchings in H, the first being $\{v\}$ and the second being $\varnothing$. Notice that then by Equation (4.1), the statement holds for singleton graphs. For the Join, Relabel, and Union nodes the statement holds due to Lemmas 4.11, 4.13, and 4.15. As for the algorithm's running time, notice that for any cw-labeled graph $H \in \mathcal{H}$ arising in ψ, the table $\mathrm{DP}_H[\cdot,\cdot]$ can be filled in time $\mathcal{O}^\star(3^{\mathrm{cw}})$. As the number of such graphs is $\mathcal{O}(|\psi|)$, this implies that our algorithm runs in time $\mathcal{O}^\star(3^{\mathrm{cw}})$. Finally, notice that for all $H \in \mathcal{H}$, the number of induced matchings of H of size ℓ is equal to the sum of $\mathrm{DP}_H[2\ell,\sigma]$ over all $\sigma \in \{0,1\}^{\mathrm{cw}}$. This concludes the proof of Theorem 4.9. $\square$

4.2. Acyclic Matching

In this section we consider the Acyclic Matching problem. Chaudhary and Zehavi [61] recently gave an $\mathcal{O}^\star(6^{\mathrm{tw}})$ algorithm for this problem, where tw denotes the treewidth of the input graph. We first notice that, with some slightly more careful analysis of its running time, we can show that the algorithm in fact runs in time $\mathcal{O}^\star(5^{\mathrm{tw}})$. Our main contribution however is to show that this is optimal under the pw-SETH, even for the more restricted parameterization by pathwidth.

Theorem 4.17. *Given a graph G along with a nice tree decomposition of G of width* tw*, there is an algorithm that computes the size of the maximum acyclic matching in G in time $\mathcal{O}^\star(5^{\mathrm{tw}})$. The algorithm is randomized, cannot give false positives, and may give false negatives with probability at most $1/3$.*

Proof. The algorithm is identical to the one of Chaudhary and Zehavi [61], with the only difference being that we analyze its running time slightly more carefully. In the following we only sketch the main ideas and argue about the claimed running time.

This is a standard DP algorithm over the nodes of the tree decomposition. In order to deal with the acyclicity constraint, the authors employ the Cut & Count technique [94], while fast subset convolution is used to speed up the computation in the Join nodes of the tree decomposition (see [92, Chapter 11]). Observe that the $\mathcal{O}^\star(6^{\mathrm{tw}})$ running time claimed in [61] is solely due to the Join nodes, as for any other type of node $\mathcal{O}^\star(5^{\mathrm{tw}})$ time suffices to populate all the entries of the corresponding table.

The key observation is that computing all entries of the table of a Join node can be done in time $\mathcal{O}^\star(5^{\mathrm{tw}})$. Following the same notation, let $\mathcal{B}_x$ denote a bag of a Join node x of the tree decomposition. We aim to populate the table $\mathcal{A}_x[a, b, c, d, w, s]$, where $a, c \in [0, n]$, $b \in [0, n-1]$, $w \in [0, 12n^2]$, and d and s are colorings such that $d\colon \mathcal{B}_x \to \{0, 1, 2\}$ and $s\colon \mathcal{B}_x \to \{0, l, r\}$ with $d(v) = 0 \iff s(v) = 0$ for all $v \in \mathcal{B}_x$; we call such colorings d and s *compatible*. Notice that the table $\mathcal{A}_x$ has $\mathcal{O}^\star(5^{\mathrm{tw}})$ entries.

The authors prove that for fixed values of a, b, c, w, s, and for all colorings d such that $d^{-1}(0) = R \subseteq \mathcal{B}_x$, one can compute the entries $\mathcal{A}_x[a, b, c, d, w, s]$ in time $2^{|\mathcal{B}_x \setminus R|} n^{\mathcal{O}(1)}$. Consequently, summing over all subsets $R \subseteq \mathcal{B}_x$ and noticing that $\sum_{R \subseteq \mathcal{B}_x} 2^{|\mathcal{B}_x \setminus R|} = 3^{|\mathcal{B}_x|} \leq 3^{\mathrm{tw}+1}$ implies that for fixed values of a, b, c, w, s, the total time to compute the entries $\mathcal{A}_x[a, b, c, d, w, s]$ for *all* colorings d is $\mathcal{O}^\star(3^{\mathrm{tw}})$. Since $a, c \in [0, n]$, $b \in [0, n-1]$, $w \in [0, 12n^2]$, and for every d there are at most 2^{tw} compatible colorings s, it follows that there are $\mathcal{O}^\star(2^{\mathrm{tw}})$ ways to fix our parameters, resulting in a total time of $\mathcal{O}^\star(6^{\mathrm{tw}})$ to compute all entries of the table $\mathcal{A}_x$. Notice however that for a coloring d where $d^{-1}(0) = R \subseteq \mathcal{B}_x$, it holds that there are at most $2^{|\mathcal{B}_x \setminus R|}$ compatible colorings s. Since $\sum_{R \subseteq \mathcal{B}_x} 2^{|\mathcal{B}_x \setminus R|} \cdot 2^{|\mathcal{B}_x \setminus R|} = 5^{|\mathcal{B}_x|} \leq 5^{\mathrm{tw}+1}$ it holds that for fixed values of a, b, c, w, the total time to compute the entries of $\mathcal{A}_x[a, b, c, d, w, s]$ for *all compatible* colorings d and s is $\mathcal{O}^\star(5^{\mathrm{tw}})$, and thus the total time to compute all entries of the table $\mathcal{A}_x$ is $\mathcal{O}^\star(5^{\mathrm{tw}})$. □

Next we prove the following lower bound for Acyclic Matching.

Theorem 4.18. *For any constant $\varepsilon > 0$, there is no $\mathcal{O}^\star((5-\varepsilon)^{\mathrm{pw}})$ algorithm deciding* Acyclic Matching, *where* pw *denotes the pathwidth of the input graph, unless the pw-SETH is false.*

Proof. We present a reduction from the 4-CSP-5 problem of Corollary 2.3 to Acyclic Matching. We are given a CSP instance ψ whose variables take values from [5], a partition of its variables into two sets V_1, V_2, a path decomposition $B_1, \ldots, B_t$ of ψ of width p such that each bag contains $\mathcal{O}(\log p)$ variables of V_2, and an injective function b mapping each constraint to a bag that contains its variables. We want to construct an Acyclic Matching instance (G, ℓ) with $\mathrm{pw}(G) = p + o(p)$, such that if ψ is satisfiable, then G has an acyclic matching of size ℓ, while if G has an acyclic matching of size ℓ, ψ admits a monotone satisfying multi-assignment which is consistent for V_2. We assume that the variables of ψ are numbered $X = \{x_1, \ldots, x_n\}$, the constraints are $c_1, \ldots, c_m$, and that the given path decomposition is nice. We construct G as follows.

XOR Gadget. Given two vertices v_1 and v_2, when we say that we connect them via a *XOR gadget* $\hat{O}(v_1, v_2)$ we first add the edge $\{v_1, v_2\}$, then introduce two vertices $p_1^{v_1,v_2}$ and $p_2^{v_1,v_2}$ called the *private vertices of the XOR gadget*, and lastly add edges so that v_1, v_2, and the private vertices of the gadget form a cycle; for an illustration see Figure 4.3. We say that $v_1, v_2, p_1^{v_1,v_2}, p_2^{v_1,v_2}$ are the *vertices of the XOR gadget* $\hat{O}(v_1, v_2)$, while the *edges of a XOR gadget* are the edges whose endpoints both are among its vertices.

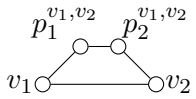


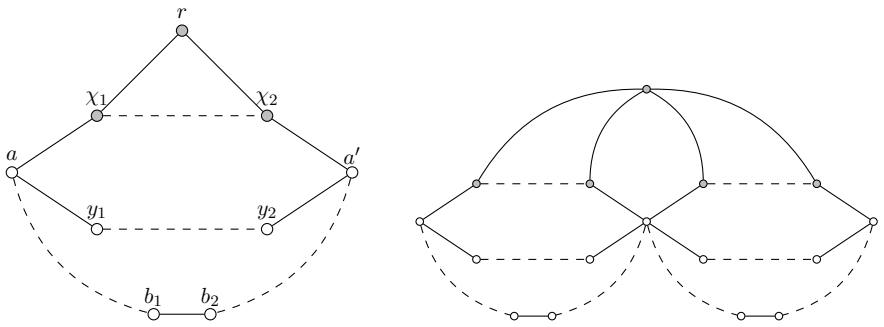
Figure 4.3.: XOR gadget $\hat{O}(v_1, v_2)$.

Root vertex. We start our construction by introducing a vertex r, which we refer to as the *root vertex*. Furthermore, we attach a leaf r' to r.

Block and Variable Gadgets. For every variable x_i and every bag with $x_i \in B_j$, where $j \in [j_1, j_2]$, construct a *block gadget* $\hat{B}_{i,j}$. Here we consider two cases, depending on whether $x_i \in V_1$ or $x_i \in V_2$.

If $x_i \in V_1$, then we construct the block gadget as depicted in Figure 4.4a. In order to do so, we introduce vertices $a, a', \chi_1, \chi_2, y_1, y_2, b_1, b_2$. Then, we attach leaves l_1 and l_2 to χ_1 and χ_2 respectively. Next, we add edges from a to χ_1 and y_1, from a' to χ_2 and y_2, from the root vertex r to χ_1 and χ_2, as well as the edge $\{b_1, b_2\}$. Finally, we add the XOR gadgets $\hat{O}(a, b_1)$, $\hat{O}(a', b_2)$, $\hat{O}(\chi_1, \chi_2)$, and $\hat{O}(y_1, y_2)$. Next, for $j \in [j_1, j_2 - 1]$, we serially connect the block gadgets $\hat{B}_{i,j}$ and $\hat{B}_{i,j+1}$ so that the vertex a' of $\hat{B}_{i,j}$ is the vertex a of $\hat{B}_{i,j+1}$, thus resulting in the "path" $\hat{P}_i$ consisting of such serially connected block gadgets, called a *variable gadget*. For an illustration see Figure 4.4b. Intuitively, for any optimal matching M, it will hold that either $a, a' \notin V_M$ or $a, a' \in V_M$. In the

latter case, it will hold that $|V_M \cap \{\chi_1, \chi_2\}| = |V_M \cap \{y_1, y_2\}| = 1$, resulting in 4 choices for a total of 5 possible choices per path gadget; we map each choice to an assignment where x_i receives a value in [5].



(a) Block gadget. (b) Serially connected block gadgets.

Figure 4.4.: Case where $x_i \in V_1$. Gray vertices have a leaf attached, dashed edges denote XOR gadgets.

As for the case where $x_i \in V_2$, then the block gadget $\hat{B}_{i,j}$ is constructed as follows. We introduce 6 vertices $u^{i,j}, u_1^{i,j}, u_2^{i,j}, u_3^{i,j}, u_4^{i,j}, u_5^{i,j}$, we add an edge from $u^{i,j}$ to all vertices in $\{u_k^{i,j} : k \in [5]\}$, and for all $1 \leq k_1 < k_2 \leq 5$ we add a XOR gadget $\hat{O}(u_{k_1}^{i,j}, u_{k_2}^{i,j})$. Next, for $j \in [j_1, j_2 - 1]$, for all *distinct* $k_1, k_2 \in [5]$, we add a XOR gadget $\hat{O}(u_{k_1}^{i,j}, u_{k_2}^{i,j+1})$, and we call the *variable gadget* $\hat{P}_i$ this sequence of serially connected block gadgets.

Constraint Gadget. This gadget is responsible for determining constraint satisfaction, based on the choices made in the rest of the graph. Let c be a constraint of ψ, where $b(c) = j$ and c involves variables $x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4} \in B_j$. Consider the set $\mathcal{S}_c$ of the at most 5^4 satisfying assignments of c. We construct the *constraint gadget* $\hat{C}_c$ as follows.

- For each $\sigma \in \mathcal{S}_c$ construct a vertex $v_{c,\sigma}$.
- For distinct $\sigma_1, \sigma_2 \in \mathcal{S}_c$, introduce a XOR gadget $\hat{O}(v_{c,\sigma_1}, v_{c,\sigma_2})$.
- Introduce a vertex v_c and add edges between v_c and all vertices $\{v_{c,\sigma} : \sigma \in \mathcal{S}_c\}$.
- For each $\sigma \in \mathcal{S}_c$ and $\alpha \in [4]$, consider the following cases:
  - if $x_{i_\alpha} \in V_1$ and $\sigma(x_{i_\alpha}) = 1$, then add a XOR gadget between $v_{c,\sigma}$ and vertices $\{b_1, b_2, \chi_2, y_2\}$ of $\hat{B}_{i,j}$,
  - if $x_{i_\alpha} \in V_1$ and $\sigma(x_{i_\alpha}) = 2$, then add a XOR gadget between $v_{c,\sigma}$ and vertices $\{b_1, b_2, \chi_1, y_2\}$ of $\hat{B}_{i,j}$,
  - if $x_{i_\alpha} \in V_1$ and $\sigma(x_{i_\alpha}) = 3$, then add a XOR gadget between $v_{c,\sigma}$ and vertices $\{a, a', y_1, y_2\}$ of $\hat{B}_{i,j}$,
  - if $x_{i_\alpha} \in V_1$ and $\sigma(x_{i_\alpha}) = 4$, then add a XOR gadget between $v_{c,\sigma}$ and vertices $\{b_1, b_2, \chi_2, y_1\}$ of $\hat{B}_{i,j}$,

- if $x_{i_\alpha} \in V_1$ and $\sigma(x_{i_\alpha}) = 5$, then add a XOR gadget between $v_{c,\sigma}$ and vertices $\{b_1, b_2, \chi_1, y_1\}$ of $\hat{B}_{i,j}$,
- if $x_{i_\alpha} \in V_2$, then add a XOR gadget between $v_{c,\sigma}$ and all vertices in $\{u_k^{i,j} : k \in [5] \setminus \{\sigma(x_{i_\alpha})\}\}$.

This completes the construction of G. We set L to be the number of XOR gadgets introduced in G, and we set $\ell = 1 + L + 2L_1 + L_2 + m$, where L_i denotes the number of block gadgets constructed due to variables belonging to V_i, and m is the number of constraints of ψ. In that case, (G, ℓ) is the constructed instance of Acyclic Matching.

Lemma 4.19. *If ψ is satisfiable, then G has an acyclic matching of size ℓ.*

Proof. Let $\sigma \colon X \to [5]$ be a satisfying assignment for ψ, that is, σ satisfies all constraints $c_1, \ldots, c_m$. We will construct a matching $M \subseteq E(G)$ of size $|M| \geq \ell$ such that $G[V_M]$ is a forest.

- First, add to M the edge $\{r, r'\}$.
- Then, for every XOR gadget $\hat{O}(v_1, v_2)$ in G, include in M the edge between its two private vertices $\{p_1^{v_1,v_2}, p_2^{v_1,v_2}\}$.
- Next, for every constraint gadget $\hat{C}_c$, since σ is a satisfying assignment, there exists some vertex v_{c,σ_c} with σ_c being the restriction of σ to the variables involved in c. Add in M the edge $\{v_c, v_{c,\sigma_c}\}$.
- For each $x_i \in V_2$ and $j \in [t]$ such that $x_i \in B_j$ we select into M the edge $\{u^{i,j}, u_k^{i,j}\}$, where $\sigma(x_i) = k$.
- Finally, for each $x_i \in V_2$ and $j \in [t]$ such that $x_i \in B_j$, we include in M the following edges from the block gadget $\hat{B}_{i,j}$:
  - if $\sigma(x_i) = 1$, the edges $\{y_1, a\}$ and $\{\chi_1, l_1\}$,
  - if $\sigma(x_i) = 2$, the edges $\{y_1, a\}$ and $\{\chi_2, l_2\}$,
  - if $\sigma(x_i) = 3$, the edges $\{b_1, b_2\}$ and $\{\chi_1, l_1\}$,
  - if $\sigma(x_i) = 4$, the edges $\{y_2, a'\}$ and $\{\chi_1, l_1\}$,
  - if $\sigma(x_i) = 5$, the edges $\{y_2, a'\}$ and $\{\chi_2, l_2\}$.

Consequently, M is a matching of size $1 + L + m + L_2 + 2L_1 = \ell$.

We claim that $G[V_M]$ is a forest. Consider any constraint gadget $\hat{C}_c$, and let v_{c,σ_c} such that $\{v_c, v_{c,\sigma_c}\} \in M$. Notice that since σ_c is the restriction of σ to its involved variables, one can easily verify that none of the vertices that v_{c,σ_c} has a XOR gadget with belongs to V_M. Consequently, the vertices of the constraint gadgets belonging to V_M induce acyclic components.

It remains to argue that the graph induced by r, r', and the vertices of the block gadgets in V_M is acyclic. Notice that this is indeed the case for the block gadgets corresponding to variables in V_2, thus in the following we focus on the block gadgets

corresponding to variables in V_1. Let $x_i \in V_1$ and $j \in [t]$ such that $x_i \in B_j$. First, notice that the graph induced by the vertices of any single block gadget $\hat{B}_{i,j}$ in V_M along with $\{r, r'\}$ is acyclic and vertices a and a' are disconnected. Furthermore, for $j_1, j_2 \in [t]$ and distinct $x_{i_1}, x_{i_2} \in V_1$, notice that the block gadgets $\hat{B}_{i_1,j_1}$ and $\hat{B}_{i_2,j_2}$ are disconnected in $G[V_M] - r$, thus if $G[V_M]$ contains a cycle, then this cycle involves only block gadgets corresponding to a single variable x_i. Finally, it holds that any two sequential block gadgets (as in Figure 4.4b) are connected in $G - r$ via their single common vertex which appears as a in the first and a' in the second gadget. Consequently, any cycle in $G[V_M]$ involving at least two block gadgets $\hat{B}_{i,j_1}$ and $\hat{B}_{i,j_2}$ with $j_1 < j_2$ must contain r, the vertex a' of $\hat{B}_{i,j_1}$, and the vertex a of $\hat{B}_{i,j_2}$. If $j_2 = j_1 + 1$, it is easy to verify that there exists only a single path in $G[V_M]$ that connects r and vertex a' of $\hat{B}_{i,j_1}$ (which coincides with the vertex a of $\hat{B}_{i,j_2}$). Otherwise, i.e., when $j_2 > j_1 + 1$, given that the vertices a and a' of any single block gadget are disconnected in $G[V_M] - r$, it follows that vertex a' of $\hat{B}_{i,j_1}$ and a of $\hat{B}_{i,j_2}$ are disconnected in $G[V_M] - r$, thus no cycle exists in $G[V_M]$. This completes the proof. □

Lemma 4.20. *If G has an acyclic matching of size ℓ, then ψ admits a monotone satisfying multi-assignment which is consistent for V_2.*

Proof. We start with the following claim.

▷ Claim 4.21. There exists an acyclic matching $M_{\max}$ of G of maximum size such that for any XOR gadget $\hat{O}(v_1, v_2)$ of G it holds that $|\{v_1, v_2\} \cap V_{M_{\max}}| \leq 1$ and $\{p_1^{v_1,v_2}, p^{v_1,v_2}\} \in M_{\max}$.

Proof of the claim. Let $M_{\max}$ be a maximum acyclic matching of G, and assume that G contains a XOR gadget $\hat{O}(v_1, v_2)$. We show that we can obtain an acyclic matching $M'_{\max}$ of G of the same size such that $|\{v_1, v_2\} \cap V_{M'_{\max}}| \leq 1$ and $\{p_1^{v_1,v_2}, p^{v_1,v_2}\} \in M'_{\max}$. To this end, we consider different cases depending on the vertices of the XOR gadget that belong to $V_{M_{\max}}$.

If $v_1, v_2 \notin V_{M_{\max}}$, then $\{p_1^{v_1,v_2}, p_2^{v_1,v_2}\} \in M_{\max}$ as $M_{\max}$ is of maximum size.

Now assume that $|\{v_1, v_2\} \cap V_{M_{\max}}| = 1$, and without loss of generality $v_1 \in V_{M_{\max}}$ (the other case is analogous). Let $e \in M_{\max}$ be the edge incident with v_1. If $e \neq \{v_1, p_1^{v_1,v_2}\}$, then it holds that $\{p_1^{v_1,v_2}, p_2^{v_1,v_2}\} \in M_{\max}$ as $M_{\max}$ is of maximum size. Otherwise it holds that $p_2^{v_1,v_2} \notin V_{M_{\max}}$ and the acyclic matching $M'_{\max} = (M_{\max} \setminus \{e\}) \cup \{\{p_1^{v_1,v_2}, p_2^{v_1,v_2}\}\}$ has the same size.

Lastly, assume that $v_1, v_2 \in V_{M_{\max}}$. Notice that in that case, either $p_1^{v_1,v_2} \notin V_{M_{\max}}$ or $p_2^{v_1,v_2} \notin V_{M_{\max}}$, as otherwise $G[V_{M_{\max}}]$ contains a cycle. If $\{v_1, v_2\} \in M_{\max}$, then $M'_{\max} = (M_{\max} \setminus \{\{v_1, v_2\}\}) \cup \{\{p_1^{v_1,v_2}, p_2^{v_1,v_2}\}\}$ is an acyclic matching of the same size. Otherwise, it holds that for either v_1 or v_2 there exists an edge $e \in M_{\max}$ incident with it that does not belong to the XOR gadget $\hat{O}(v_1, v_2)$. Assume without loss of generality that e is incident with v_1, and let $e' \in M_{\max}$ be the edge incident with v_2 belonging to $M_{\max}$. Then, the acyclic matching $M'_{\max} = (M_{\max} \setminus \{e'\}) \cup \{\{p_1^{v_1,v_2}, p_2^{v_1,v_2}\}\}$ is of the same size. ◁

Armed with the previous claim, we can now prove the existence of an acyclic matching in G with specific properties.

▷ Claim 4.22. There exists an acyclic matching M of maximum size of G such that, for any XOR gadget $\hat{O}(v_1, v_2)$ of G it holds that $|\{v_1, v_2\} \cap V_M| \leq 1$ and $\{p_1^{v_1,v_2}, p^{v_1,v_2}\} \in M$. Furthermore, it holds that

- $\{r, r'\} \in M$,
- for all constraint gadgets $\hat{C}_c$, there exists $\sigma \in \mathcal{S}_c$ such that $\{v_c, v_{c,\sigma_c}\} \in M$,
- for all block gadgets $\hat{B}_{i,j}$,
  - if $x_i \in V_1$ then $|\{\{\chi_1, l_1\}, \{\chi_2, l_2\}\} \cap M| = |\{\{y_1, a\}, \{y_2, a'\}, \{b_1, b_2\}\} \cap M| = 1$,
  - otherwise if $x_i \in V_2$ there exists $k \in [5]$ such that $\{u^{i,j}, u_k^{i,j}\} \in M$.

Proof of the claim. Let $M_{\max}$ be an acyclic matching of G of maximum size as in Claim 4.21, where $|M_{\max}| \geq \ell$ since by assumption G has an acyclic matching of size at least ℓ. Consider the following reduction rule.

Rule (†)**.** Let $\{u, v\} \in M_{\mathsf{in}}$, and let $l \neq v$ be a leaf attached to u in G, while v is not a leaf. Then, replace M_{in} with $M_{\mathsf{out}} = (M_{\mathsf{in}} \setminus \{\{u, v\}\}) \cup \{\{u, l\}\}$.

It is easy to see that in Rule (†) if M_{in} is an acyclic matching, then so is M_{out}, while the size of the two matchings is the same. Let M denote the acyclic matching obtained after exhaustively applying Rule (†) in $M_{\max}$; since all vertices of G have at most one attached leaf, such a matching is unique. Notice that $|M| \geq \ell$, while for any XOR gadget $\hat{O}(v_1, v_2)$ of G it holds that $|\{v_1, v_2\} \cap V_M| \leq 1$ and $\{p_1^{v_1,v_2}, p^{v_1,v_2}\} \in M$.

Notice that due to Rule (†), if $r \in V_M$, then $\{r, r'\} \in M$. Furthermore, for any edge in $M \setminus \{\{r, r'\}\}$ that is not due to a XOR gadget it holds that either both of its endpoints belong to the vertices of a block gadget, or they both belong to the vertices of a constraint gadget; that is due to the fact that any connected pair of vertices from both block and constraint gadgets or from different block gadgets is connected via a XOR gadget.

Notice that in the constraint gadget $\hat{C}_c$, all vertices $\{v_{c,\sigma} : \sigma \in \mathcal{S}_c\}$ are pairwise connected via XOR gadgets, thus at most one of them belongs to V_M. Along with the previous paragraph, it follows that either V_M contains no vertex from $\hat{C}_c$, or there exists $\sigma_c \in \mathcal{S}_c$ such that $\{v_c, v_{c,\sigma_c}\} \in M$.

Now consider the block gadget $\hat{B}_{i,j}$, where $j \in [t]$ such that $x_i \in B_j$. Assume first that $x_i \in V_1$, and notice that if $\chi_1 \in V_M$, then $\{\chi_1, l_1\} \in M$ and $\chi_2 \notin V_M$ (similarly for χ_2). Furthermore, one can easily verify that $|\{\{y_1, a\}, \{y_2, a'\}, \{b_1, b_2\}\} \cap M| \leq 1$. If on the other hand $x_i \in V_2$, then since vertices $\{u_k^{i,j} : k \in [5]\}$ are pairwise connected via XOR gadgets, it holds that either V_M contains no vertex of $\hat{B}_{i,j}$, or there exists $k \in [5]$ such that $\{u^{i,j}, u_k^{i,j}\} \in M$.

Recall that $|M| \geq \ell = 1 + L + 2L_1 + L_2 + m$, where L is the number of XOR gadgets in G, L_i is the number of block gadgets constructed due to variables belonging to V_i, and m is the number of constraints of ψ. Due to the discussion so far and summing over all constraint and block gadgets, M contains at most ℓ edges. Consequently, it follows

that the bounds obtained in the previous paragraphs are tight for all constraint and block gadgets, which implies that the inclusions are as described in the statement and $\{r, r'\} \in M$. ◁

Now let M be an acyclic matching of G as in Claim 4.22. Consider a multi-assignment $\sigma\colon X \times [t] \to [5]$ over all pairs $x_i \in X$ and $j \in [t]$ with $x_i \in B_j$. In particular, consider two cases. If $x_i \in V_2$ and $\{u^{i,j}, u^{i,j}_k\} \in M$, then $\sigma(x_i, j) = k \in [5]$. Otherwise (i.e., $x_i \in V_1$), if for the edges of $\hat{B}_{i,j}$ it holds that

- $\{y_1, a\}, \{\chi_1, l_1\} \in M$, then $\sigma(x_i, j) = 1$,
- $\{y_1, a\}, \{\chi_2, l_2\} \in M$, then $\sigma(x_i, j) = 2$,
- $\{b_1, b_2\} \in M$, then $\sigma(x_i, j) = 3$,
- $\{y_2, a'\}, \{\chi_1, l_1\} \in M$, then $\sigma(x_i, j) = 4$,
- $\{y_2, a'\}, \{\chi_2, l_2\} \in M$, then $\sigma(x_i, j) = 5$.

Notice that σ is well-defined due to Claim 4.22.

▷ Claim 4.23. It holds that σ is monotone-increasing, as well as consistent for all $x_i \in V_2$.

Proof of the claim. Regarding the consistency for $x_i \in V_2$, notice that it follows due to the XOR gadgets between vertices belonging to sequential block gadgets $\hat{B}_{i,j}$ and $\hat{B}_{i,j+1}$. In the following we argue about the monotonicity. Assume that $x_i \in V_1$ and $j \in [j_1, j_2 - 1]$, where B_{j_1} and B_{j_2} denote the first and last bag of the decomposition containing x_i respectively.

We first argue that if for $\hat{B}_{i,j}$ it holds that $\{y_2, a'\} \in M$, then for $\hat{B}_{i,j+1}$ it holds that $\{y_2, a'\} \in M$. Indeed, since vertices a' and a of the first and the latter block gadget coincide, it follows that for $\hat{B}_{i,j+1}$ we have $\{y_1, a\}, \{b_1, b_2\} \notin M$, and since $|\{\{y_1, a\}, \{y_2, a'\}, \{b_1, b_2\}\} \cap M| = 1$ for any block gadget, the statement follows.

Now we argue that if for $\hat{B}_{i,j}$ it holds that $\{y_2, a'\}, \{\chi_2, l_2\} \in M$, then for $\hat{B}_{i,j+1}$ it holds that $\{y_2, a'\}, \{\chi_2, l_2\} \in M$. In this case, due to the previous paragraph we have that in $\hat{B}_{i,j+1}$, $\{y_2, a'\} \in M$ as well as $a \in V_M$, implying that $\chi_1 \notin V_M$ (as otherwise $G[V_M]$ has a cycle), which in turn implies that $\{\chi_2, l_2\} \in M$. Consequently, it holds that if $\sigma(x_i, j) = 5$ then $\sigma(x_i, j + 1) = 5$, while in conjunction with the previous paragraph we get that if $\sigma(x_i, j) = 4$ then $\sigma(x_i, j + 1) \in \{4, 5\}$.

Next, it is easy to see that if in $\hat{B}_{i,j}$ we have that $\{b_1, b_2\} \in M$, then in $\hat{B}_{i,j+1}$ we have that $\{y_1, a\} \notin M$; indeed, in $\hat{B}_{i,j}$ since $b_2 \in V_M$ it holds that $a' \notin V_M$, however this implies that for $\hat{B}_{i,j+1}$ it holds that $a \notin V_M$. Consequently, if $\sigma(x_i, j) = 3$ then $\sigma(x_i, j + 1) \in \{3, 4, 5\}$.

Finally assume that in $\hat{B}_{i,j}$ we have that $\{y_1, a\}, \{\chi_2, l_2\} \in M$. In that case, it holds that in $\hat{B}_{i,j+1}$ we have $\chi_1 \notin V_M$ (otherwise $G[V_M]$ contains a cycle), implying that $\{\chi_2, l_2\} \in M$. Consequently, we have that if $\sigma(x_i, j) = 2$ then $\sigma(x_i, j + 1) \neq 1$. ◁

It remains to argue that σ is a satisfying multi-assignment for ψ. Consider a constraint c with $b(c) = j$, involving four variables from $V_{i_1}, V_{i_2}, V_{i_3}, V_{i_4}$. Let $\sigma_c \in \mathcal{S}_c$ such that

$\{v_c, v_{c,\sigma_c}\} \in M$. We claim that σ_c must be consistent with σ. Indeed, if σ_c assigns value $k \in [5]$ to $x_i \in V_2$, then it follows that $(\{u_k^{i,j} : k \in [5]\} \setminus \{u_{\sigma(x_i,j)}^{i,j}\}) \notin V_M$ due to the associated XOR gadgets, thus $\{u^{i,j}, u_{\sigma(x_i,j)}^{i,j}\} \in M$. On the other hand, if σ assigns value $k \in [5]$ to $x_i \in V_2$, then

- if $k = 1$, then in $\hat{B}_{i,j}$ we have that $b_1, b_2, \chi_2, y_2 \notin V_M$, implying that $\{y_1, a\}, \{\chi_1, l_1\} \in M$,
- if $k = 2$, then in $\hat{B}_{i,j}$ we have that $b_1, b_2, \chi_1, y_2 \notin V_M$, implying that $\{y_1, a\}, \{\chi_2, l_2\} \in M$,
- if $k = 3$, then in $\hat{B}_{i,j}$ we have that $a, a', y_1, y_2 \notin V_M$, implying that $\{b_1, b_2\} \in M$,
- if $k = 4$, then in $\hat{B}_{i,j}$ we have that $b_1, b_2, \chi_2, y_1 \notin V_M$, implying that $\{y_2, a'\}, \{\chi_1, l_1\} \in M$,
- if $k = 5$, then in $\hat{B}_{i,j}$ we have that $b_1, b_2, \chi_1, y_1 \notin V_M$, implying that $\{y_2, a'\}, \{\chi_1, l_1\} \in M$.

This completes the proof. □

Finally, to bound the pathwidth of the graph G, start with the decomposition of the primal graph of ψ and in each B_j replace each $x_i \in V_2 \cap B_j$ with all the vertices of $\hat{B}_{i,j}$. We further add in B_j all the vertices of $\hat{B}_{i,j+1}$ for $x_i \in V_2 \cap B_{j+1}$, as well as the vertices r and r'. For each constraint c, let $b(c) = j$ and add into B_j all the vertices of $\hat{C}_c$ and any private vertex of a XOR gadget involving a vertex of $\hat{C}_c$, for a total of at most $1 + 5^4 + 2\binom{5^4}{2} + 8 \cdot 5^4$ vertices. So far each bag contains $p + \mathcal{O}(\log p)$ vertices. To cover the remaining vertices, for each $j \in [t]$ we replace the bag B_j with a sequence of bags such that all of them contain the vertices we have added to B_j so far, the first bag contains the vertices a belonging to all $\hat{B}_{i,j}$ for all $x_i \in V_1 \cap B_j$ and the last bag contains the vertices a' belonging to all $\hat{B}_{i,j}$ for all $x_i \in V_1 \cap B_j$. We insert a sequence of $\mathcal{O}(p)$ bags between these two, at each step adding all vertices apart from a of a block gadget $\hat{B}_{i,j}$ and then removing all vertices of the same block gadget apart from a', for $x_i \in V_1 \cap B_j$. □

5. Maximum Node-Disjoint Paths

Publication note. An extended abstract of the results presented in this chapter appeared in the proceedings of the IPEC 2025 [216].

Chapter summary. This chapter investigates the Maximum Node-Disjoint Paths problem under various structural parameterizations. As our main contributions, we present an efficient FPT-AS parameterized by tree-depth, that in time $f(\mathrm{td}, \varepsilon)n^{\mathcal{O}(1)}$ computes a $(1-\varepsilon)$-approximate solution, while we show that obtaining an FPT-AS for the more general parameterization by pathwidth would falsify the PIH.

One of the most important problems of structural graph theory has arguably been Node-Disjoint Paths, where given a graph G and k pairs of its vertices (s_i, t_i) for $i = 1, \ldots, k$, called demands, the goal is to determine whether there exist k vertex-disjoint paths connecting s_i and t_i. This extensively studied problem [4, 60, 72, 188, 199, 225, 226, 266, 287] is one of the very first to be proven NP-complete (for k being part of the input) [182], and it has a central role in the field of structural graph theory as well as in parameterized complexity [231], as the breakthrough result by Robertson and Seymour [263] that it is fixed-parameter tractable (FPT) parameterized by k (i.e., admits an $f(k)n^{\mathcal{O}(1)}$ algorithm for some function f, where $n = |V(G)|$) is the culmination of their long and influential series of works on Graph Minors.

Here we concern ourselves with Maximum Node-Disjoint Paths (MaxNDP), the natural generalization of Node-Disjoint Paths where one asks whether at least $\ell \leq k$ demands can be routed by vertex-disjoint paths (we say that a demand is routed when there exists a path connecting its endpoints in the set of vertex-disjoint paths of the solution). Notice that one could alternatively phrase this as an optimization problem and ask for the maximum number of demands that can be routed. Even though MaxNDP has been intensely studied with respect to its approximability [76, 77, 78, 79, 196], our understanding regarding its tractability under the perspective of parameterized complexity is rather limited. Given the rich literature regarding Node-Disjoint Paths and the importance of its structural parameterizations (indeed, Scheffler's $\mathrm{tw}^{\mathcal{O}(\mathrm{tw})}n^{\mathcal{O}(1)}$ algorithm [266] is a key ingredient of the proof of [263]), the quest to study MaxNDP under the same point of view is strongly motivated, with the hope of extending some of these results to it. Alas, prior work by Ene, Mnich, Pilipczuk, and Risteski [112] shows that MaxNDP is already W[1]-hard when parameterized by the tree-depth of the input graph (in fact their proof implies hardness for the combined parameter vertex integrity[1] plus feedback vertex number). On the other hand, notice that MaxNDP is trivially FPT by k; one can simply reduce it to 2^k instances of Node-Disjoint Paths.

[1] A graph has vertex integrity at most k if there exists a set of at most p vertices such that their deletion results in a graph with connected components of size at most $k - p$.

A natural question arising from this observation is whether a parameterization by ℓ renders the problem tractable. In this spirit, Marx and Wollan [239] studied this setting and proved that the problem is W[1]-hard even when parameterized by the combined parameter ℓ plus the treewidth of the input graph; a closer look into their proof reveals that their result extends to graphs of bounded pathwidth plus feedback vertex number. This plethora of negative results fails to answer which parameterizations render the problem tractable, and whether a parameterization by ℓ plus some structural parameter (larger than or incomparable to treewidth) may lift it to FPT.

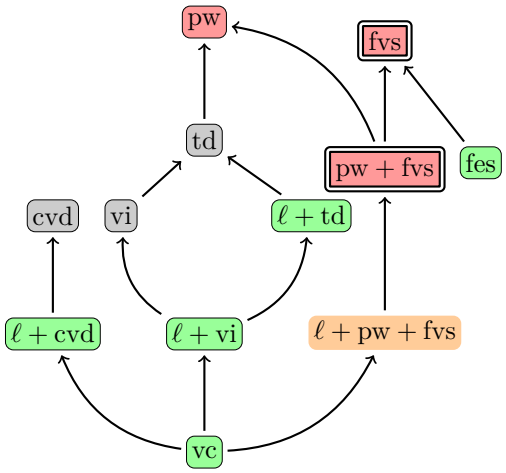


Figure 5.1.: Our results and hierarchy of the related graph parameters. For any graph, if the parameter at the tail of an arrow is a constant, that is also the case for the one at its head. Green and gray indicate that the problem is FPT (Theorems 5.2 and 5.3) and that it admits an FPT approximation scheme (Theorems 5.4 and 5.5) respectively. Yellow indicates W[1]-hardness, orange that there is no FPT approximation scheme (Theorem 5.6), and red XNLP-completeness (Theorem 5.8). Prior to this work, it was only known that the problem is W[1]-hard parameterized by vi+fvs [112] and by ℓ+pw+fvs [239].

Our contribution. In the present paper we thoroughly investigate the complexity of MaxNDP under different parameterizations, and determine exactly when it is rendered tractable, by additionally employing the use of approximation in the process (see Figure 5.1 for a synopsis of our results). We start by showing that the problem is FPT parameterized by the number of vertices of an optimal solution by developing a simple algorithm that makes use of the color-coding technique introduced by Alon, Yuster, and Zwick [7]. We then prove that, albeit simple, this algorithm is in fact sufficient to pinpoint exactly when the problem is fixed-parameter tractable: utilizing a variety of structural observations, we develop FPT algorithms for various parameterizations (most involving ℓ as a parameter) at the core of all of which lies the previously mentioned algorithm. Along the way we also develop an FPT algorithm for the parameterization by the feedback edge number. These

positive results, in conjunction with the hardness results of [112, 239], clearly showcase the transition of the problem from FPT to W[1]-hard for its various parameterizations (with the exception of cluster vertex deletion number, where it is unknown whether the problem is W[1]-hard).

Given the apparent hardness of the problem, we move on to consider it under the perspective of parameterized approximation (see [115] for a survey of the area). Here, we observe that utilizing the previously developed FPT algorithms when ℓ is also a parameter, can in fact lead to efficient FPT approximation schemes (FPT-ASes) in case of solely structural parameterizations of the problem, when parameterized by the cluster vertex deletion number, the vertex integrity, or the tree-depth of the input graph; in the latter two cases the problem is known to be W[1]-hard [112].

The FPT approximation schemes developed indicate that the W[1]-hardness can, in some cases, be overcome via the use of approximation. Given the relationship of our studied parameters as well as the FPT-AS for tree-depth, a natural question is whether an analogous approximation scheme exists for the parameterization by pathwidth as well. Notice that the W[1]-hardness by ℓ plus the pathwidth pw and the feedback vertex number fvs of the input graph already excludes the existence of an approximation scheme of running time $f(\mathrm{pw}, \mathrm{fvs}, \varepsilon)n^{\mathcal{O}(1)}$, yet one of time $f(\mathrm{pw}, \mathrm{fvs}, \varepsilon)n^{g(\varepsilon)}$ remains possible. Our next result is to exclude the existence of such a scheme under the *Parameterized Inapproximability Hypothesis* [228], which was recently proved to hold under the ETH [158], and thus precisely determine the parameter border where the problem transitions from "hard but approximable" to "inapproximable". By slightly modifying our reduction, we subsequently show that the problem is XNLP-complete when parameterized solely by the pathwidth of the input graph, where XNLP is a complexity class that has been recently brought forth by Bodlaender, Groenland, Nederlof, and Swennenhuis [41], and such a result implies W[t]-hardness for all integers $t \geq 1$.

Lastly, we proceed to a more fine-grained examination of the hardness of MaxNDP when parameterized by the tree-depth of the input graph. Standard dynamic programming techniques can be used to obtain an $n^{\mathcal{O}(\mathrm{tw})}$ algorithm, while previous work by Ene, Mnich, Pilipczuk, and Risteski [112] implies that the problem cannot be solved in time $n^{o(\sqrt{\mathrm{td}})}$ under the ETH for graphs of tree-depth td, thereby leaving hope for an $n^{o(\mathrm{td})}$ algorithm. We revisit said proof, and by employing the recursive structure introduced in Chapter 3 we bridge this gap and prove that MaxNDP cannot be solved in time $n^{o(\mathrm{td})}$ under the ETH, rendering the $n^{\mathcal{O}(\mathrm{tw})}$ algorithm optimal even for this much smaller class of graphs.

Related work. Even though MaxNDP has been well-studied under the scope of approximation algorithms, the 20 years old algorithm of ratio $\mathcal{O}(\sqrt{n})$ due to Kolliopoulos and Stein [196] remains the state of the art in general graphs. This has been improved in the case of grid [75] and planar graphs [76], resulting in approximation ratios $\tilde{\mathcal{O}}(n^{1/4})$ and $\tilde{\mathcal{O}}(n^{9/19})$ respectively, where standard $\tilde{\mathcal{O}}$ notation is used to hide polylogarithmic terms. For graphs of pathwidth pw, Ene, Mnich, Pilipczuk, and Risteski [112] have presented an algorithm of approximation ratio $\mathcal{O}(\mathrm{pw}^3)$. Regarding inapproximability results, after a series of works, Chuzhoy, Kim, and Nimavat [78, 79] have shown that the problem cannot

be approximated in polynomial time (i) within a factor of $2^{\mathcal{O}(\log^{1-\varepsilon} n)}$ for any constant ε, assuming NP $\not\subseteq$ DTIME($n^{\text{polylog}n}$), and (ii) within a factor of $n^{\mathcal{O}(1/(\log\log n)^2)}$, assuming that there exists some constant $\delta > 0$ such that NP $\not\subseteq$ DTIME(2^{n^δ}).

The problem has been also studied under a parameterized complexity perspective. As already noted, it is trivially FPT by k by a simple reduction to 2^k instances of Node-Disjoint Paths, which is well-known to be FPT by the number of demands. On the other hand, Marx and Wollan [239] have shown that it becomes W[1]-hard when parameterized by $\ell + \text{tw}$, with a closer look into their proof revealing that their result extends to the parameterization by $\ell + \text{pw} + \text{fvs}$. Regarding structural parameterizations, Ene, Mnich, Pilipczuk, and Risteski [112] have proved that the problem is W[1]-hard when parameterized by the tree-depth of the input graph; in fact, their proof extends to graphs of bounded vertex integrity plus feedback vertex number. Fleszar, Mnich, and Spoerhase [122] have proposed an algorithm of running time $(k + \text{fvs})^{\mathcal{O}(\text{fvs})} n^{\mathcal{O}(1)}$ as an alternative to the $2^k \text{fvs}^{\mathcal{O}(\text{fvs})} n^{\mathcal{O}(1)}$ algorithm obtained by reducing the instance to Node-Disjoint Paths and then using Scheffler's algorithm [266].

5.1. FPT Algorithms and Approximation Schemes

Here we present various tractability results for MaxNDP. We start by proving in Section 5.1.1 that the problem is FPT when parameterized by the number of vertices involved in an optimal solution; using this as well as some structural observations, we obtain FPT algorithms when parameterized by vc, $\ell + \text{cvd}$, $\ell + \text{vi}$, and $\ell + \text{td}$. We additionally develop an FPT algorithm for the parameterization by fes. Moving on, in Section 5.1.2 we obtain FPT approximation schemes for various structural parameterizations of the problem, for most of which it is known to be W[1]-hard, by making use of the previous FPT algorithms.

5.1.1. Exact Algorithms

We start with the following theorem.

Theorem 5.1. *Let $\mathcal{I} = (G, \mathcal{M}, \ell)$ be an instance of MaxNDP. Additionally, let τ be such that there exists a family $\mathcal{P}$ of vertex-disjoint paths each routing a demand of $\mathcal{M}$, where $|\mathcal{P}| = \min\{\ell, \text{OPT}(\mathcal{I})\}$ and $\sum_{P\in\mathcal{P}} |V(P)| \leq \tau$, with $|V(P)|$ denoting the number of vertices in path P. There is an algorithm that, given $\mathcal{I}$ and τ, decides $\mathcal{I}$ in time $2^{\mathcal{O}(\tau)} n^{\mathcal{O}(1)}$.*

Proof. Let $C = [\tau]$ be a set of τ colors. Randomly color the vertices of G with colors from C, and let A be the event where every one of the at most τ vertices of $\mathcal{P}$ receives a distinct color. Then, it follows that

$$\Pr[A] \geq \frac{\tau!}{\tau^\tau} > \frac{\left(\frac{\tau}{e}\right)^\tau}{\tau^\tau} = e^{-\tau},$$

therefore event A holds with probability at least $e^{-\tau}$.

Now, let $T[S]$ be equal to the maximum number of demands that can be routed by paths using only vertices of colors belonging to $S \subseteq C$, where each color is used in at most one path. Moreover, let $f(S) = 1$ if there exists at least one demand that can be routed using only vertices of colors belonging to S and 0 otherwise. Notice that f can be computed in polynomial time. Then, it holds that

$$T[S] = \max_{S' \subseteq S} \{f(S') + T[S \setminus S']\},$$

thus one can compute $T[C]$ in time $2^{\mathcal{O}(\tau)} n^{\mathcal{O}(1)}$ for a given coloring of the vertices of G.

By repeating this procedure $2^{\mathcal{O}(\tau)}$ times, with high probability there exists some iteration where event A holds, and the total running time is $2^{\mathcal{O}(\tau)} n^{\mathcal{O}(1)}$. By using standard techniques [92, Section 5.6], one can derandomize the described algorithm and obtain a deterministic one of the same running time. □

Using Theorem 5.1 we can obtain various parameterized algorithms, by bounding the number of vertices of an optimal solution using some simple observations.

Theorem 5.2. *Given an instance $\mathcal{I} = (G, \mathcal{M}, \ell)$ of MaxNDP, there exist algorithms that decide $\mathcal{I}$ in time*

- $2^{\mathcal{O}(\mathrm{vc})} n^{\mathcal{O}(1)}$,
- $2^{\mathcal{O}(\mathrm{cvd}+\ell)} n^{\mathcal{O}(1)}$,
- $2^{\mathcal{O}(\mathrm{vi}^2+\mathrm{vi}\cdot\ell)} n^{\mathcal{O}(1)}$,
- $2^{\mathcal{O}(2^{\mathrm{td}}\cdot\ell)} n^{\mathcal{O}(1)}$,

where vc*,* cvd*,* vi*, and* td *denote the vertex cover, cluster vertex deletion number, vertex integrity, and tree-depth of G respectively.*

Proof. We prove the statement by providing bounds on the number of vertices involved in an optimal solution, denoted by τ, and then using the algorithm of Theorem 5.1. Fix an optimal solution $\mathcal{P}$, comprised of paths P_i with endpoints s_i and t_i, for $i \in [r]$, where $0 \leq r \leq \ell$ and $(s_i, t_i) \in \mathcal{M}$. We will denote the deletion set in each case by $S \subseteq V(G)$. Note that there is no need to actually compute said deletion set.

In the case of the vertex cover, notice that at least one endpoint of every edge in $P_i \in \mathcal{P}$ belongs to S, while each vertex is involved in at most 2 edges. In that case, it follows that $\tau \leq 3\mathrm{vc}$.

We next consider the case of cluster vertex deletion number. For every path P_i, either both s_i and t_i belong to the same clique of $G - S$ or not. In the first case, there exists an optimal solution that considers the path on vertex set $\{s_i, t_i\}$ instead. In the latter, every such path involves at least one vertex of S, for a total of at most cvd such paths. Moreover, if such a path involves more than 2 vertices of the same clique of $G - S$, say $u_1, u_2, \ldots, u_q$, indexed by their order of appearance in the path, there exists an optimal solution which is obtained by taking the edge between u_1 and u_q instead. Since every vertex of S in such a path might be neighbors with vertices belonging to at most 2

different cliques of $G - S$, it follows that after "short-cutting" all such paths, at most $\text{cvd} + 4\text{cvd} = 5\text{cvd}$ vertices are used. In total, it follows that $\tau \leq 5\text{cvd} + 2\ell$.

We then consider the case of vertex integrity. For every path P_i, either $V(P_i) \cap S = \varnothing$ or $V(P_i) \cap S \neq \varnothing$. In the first case, it follows that such a path contains at most vi vertices. In the latter, P_i has at least one vertex of S, for a total of at most vi such paths. Moreover, since every vertex of S in P_i might be neighbors with vertices belonging to at most 2 different connected components of $G - S$, at most $\text{vi} + \text{vi}(2\text{vi}) = 2\text{vi}^2 + \text{vi}$ vertices are used. In total, it follows that $\tau \leq 2\text{vi}^2 + \text{vi} + \text{vi} \cdot \ell$.

For tree-depth, it holds that any path in G has length at most 2^{td}, therefore $\tau \leq 2^{\text{td}} \cdot \ell$. □

Given the W[1]-hardness of MaxNDP when parameterized by the feedback vertex number implied by previous works [112, 239], we move on to consider the parameterization by the feedback edge number, and show that it renders the problem tractable.

Theorem 5.3. *Given an instance $\mathcal{I} = (G, \mathcal{M})$ of MaxNDP, there exists an algorithm that computes $\text{OPT}(\mathcal{I})$ in time $3^{\text{fes}} n^{\mathcal{O}(1)}$, where* fes *denotes the feedback edge number of G.*

Proof. The algorithm will perform branching and reduce the instance to a collection of cycles, in which case the problem is polynomial-time solvable. We first present some reduction rules where Rules 2 and 3 are only applied after exhaustively applying Rule 1.

Rule 1. Let $\mathcal{I} = (G, \mathcal{M})$ be an instance of MaxNDP, and $u \in V(G)$ such that $(u, u) \in \mathcal{M}$. Then, replace $\mathcal{I}$ with $\mathcal{I}' = (G - u, \mathcal{M}')$, where $\mathcal{M}' \subseteq \mathcal{M}$ contains the demands of $\mathcal{M}$ whose endpoints both differ from u. It holds that $\text{OPT}(\mathcal{I}) = \text{OPT}(\mathcal{I}') + 1$.

Rule 2. Let $\mathcal{I} = (G, \mathcal{M})$ be an instance of MaxNDP, and $u \in V(G)$ such that $\deg_G(u) = 0$. Then, replace $\mathcal{I}$ with $\mathcal{I}' = (G - u, \mathcal{M}')$, where $\mathcal{M}' \subseteq \mathcal{M}$ contains the demands of $\mathcal{M}$ whose endpoints both differ from u. It holds that $\text{OPT}(\mathcal{I}) = \text{OPT}(\mathcal{I}')$.

Rule 3. Let $\mathcal{I} = (G, \mathcal{M})$ be an instance of MaxNDP, and $u \in V(G)$ such that $\deg_G(u) = 1$, where v denotes its single neighbor, i.e., $N_G(u) = \{v\}$. Then, replace $\mathcal{I}$ with $\mathcal{I}' = (G - u, \mathcal{M}')$, where $\mathcal{M}'$ is obtained by replacing any demand (u, w) in $\mathcal{M}$ with (v, w). It holds that $\text{OPT}(\mathcal{I}) = \text{OPT}(\mathcal{I}')$.

It is easy to see that applying these rules in their respective order is safe, and let $\mathcal{I} = (G, \mathcal{M})$ denote the instance obtained after exhaustively doing so. Assume without loss of generality that G is connected, otherwise solve each connected component independently. Notice that all vertices of G have degree at least 2. Recall that for G' subgraph of G it holds that $\text{fes}(G') \leq \text{fes}(G)$, and since G is a subgraph of the initial graph $\text{fes}(G) \leq \text{fes}$ follows. Moreover, for a connected graph G, it holds that $\text{fes}(G) = |E(G)| - |V(G)| + 1$.

Let $\mathcal{P}$ be a maximum-cardinality set of vertex-disjoint paths routing demands of $\mathcal{M}$ in G. Let $v \in V(G)$ such that $\deg_G(v) \geq 3$, where $e_1, e_2, e_3 \in E(G)$ denote three edges incident with v. Notice that at least one among those edges, say e_1, does not take part in

$\mathcal{P}$, that is, $e_1 \notin E(P)$ for all paths $P \in \mathcal{P}$. Perform branching on all 3 cases and delete the corresponding edge from the graph of the produced instance; we claim that this reduces the feedback edge number of all connected components of the resulting graph. To see this, consider two cases: for $j \in [3]$, either $G - e_j$ is connected or not; notice that in the latter case we can solve for each connected component independently. If $G - e_j$ is connected, then it holds that $\text{fes}(G - e_j) = \text{fes}(G) - 1$. Otherwise, $G - e_j$ is comprised of two connected components C_1 and C_2, in which case $\text{fes}(C_1) + \text{fes}(C_2) = (|E(C_1)| + |E(C_2)|) - (|V(C_1)| + |V(C_2)|) + 2 = (|E(G)| - 1) - |V(G)| + 2 = \text{fes}(G)$. We next prove that $\text{fes}(C_i) > 0$ for $i \in [2]$. Assume that this is not the case, and let without loss of generality $\text{fes}(C_1) = 0$, in which case C_1 is a tree. Moreover, let w denote the vertex that is incident with e_j and belongs to C_1. Since no reduction rule can be further applied in G, it follows that $\deg_{G[C_1]}(u) > 2$ for all vertices $u \neq w$ belonging to C_1. In that case, $G[C_1]$ has at most one leaf, consequently it is a singleton. Then however, $\deg_G(w) = 1$ and the reduction rules could have been further applied in G, a contradiction. Notice that this additionally implies that if $\text{fes}(G) = 1$, then $G - e_j$ is connected. From the previous discussion, it follows that

$$T(z) \leq 3 \max_{\substack{z_1, z_2 > 0 \\ z_1 + z_2 = z}} \{T(z_1) + T(z_2)\}, \tag{5.1}$$

where $T(z)$ denotes the number of connected components resulting from our algorithm for $z \in [\text{fes}]$ being the feedback edge number of the connected input graph, with each component having maximum degree 2.

We first argue that when the graph has maximum degree 2, then the problem is polynomial-time solvable. Let $\mathcal{I}^\star = (G^\star, \mathcal{M}^\star)$ denote one such instance, where every vertex of the connected graph $G^\star$ is of degree exactly 2, as otherwise the reduction rules can still be applied. Consequently, $G^\star$ is a cycle, thus it suffices to guess the endpoints of one routed demand and delete the involved vertices (since $G^\star$ is a cycle there are only two different such paths), thus reducing $G^\star$ into a path where the reduction rules can be applied exhaustively. Therefore, one can compute $\text{OPT}(\mathcal{I}^\star)$ in polynomial time.

Lastly, we prove by induction that $T(z) \leq 3(3^z - 9/5)$ for all $z \in [\text{fes}]$. For the base case, $T(1) \leq 3$ holds. Assume that the statement holds for all $z' \in [z-1]$. Then, by Equation (5.1) it holds that $T(z) \leq 9(3^{z_1} + 3^{z_2} - 18/5) \leq 9(3^{z_1+z_2-1} + 3^1 - 18/5) = 9(3^{z-1} - 3/5) = 3(3^z - 9/5)$, where the first inequality is due to the induction hypothesis whereas the second due to the function 3^x being convex. □

5.1.2. Approximation Schemes

Using the FPT algorithms of Theorem 5.2, we develop FPT approximation schemes for MaxNDP when parameterized solely by structural parameters.

Theorem 5.4. *Given an instance $\mathcal{I} = (G, \mathcal{M})$ of MaxNDP, one can $(1-\varepsilon)$-approximate $\text{OPT}(\mathcal{I})$ in time $2^{\mathcal{O}(\text{cvd}/\varepsilon)} n^{\mathcal{O}(1)}$ and $2^{\mathcal{O}(\text{vi}^2/\varepsilon)} n^{\mathcal{O}(1)}$, where* cvd *and* vi *denote the cluster vertex deletion number and vertex integrity of G respectively.*

Proof. Let $S \subseteq V(G)$ denote the deletion set, which can be computed in time $2^{\mathcal{O}(\text{cvd})}n^{\mathcal{O}(1)}$ for cvd [277] and $2^{\mathcal{O}(\text{vi}\log\text{vi})}n^{\mathcal{O}(1)}$ for vi [104]. Notice that $\text{OPT}(\mathcal{I}[G-S]) \geq \text{OPT}(\mathcal{I}) - |S|$, since every vertex of S can be used to route at most one demand. Consider the case where $\text{OPT}(\mathcal{I}) - |S| \geq (1-\varepsilon)\,\text{OPT}(\mathcal{I}) \iff \text{OPT}(\mathcal{I}) \geq |S|/\varepsilon$. Then, $\text{OPT}(\mathcal{I}[G-S]) \geq (1-\varepsilon)\,\text{OPT}(\mathcal{I})$. Alternatively, it holds that $\text{OPT}(\mathcal{I}) < |S|/\varepsilon$ and the algorithm of Theorem 5.2 can be used, with $\ell = \mathcal{O}(|S|/\varepsilon)$.

It remains to compute $\text{OPT}(\mathcal{I}[G-S])$. In the case of the cluster vertex deletion set, $G-S$ is a collection of cliques. In that case, one can compute $\text{OPT}(\mathcal{I}[G-S])$ in polynomial time by reducing to an instance of Maximum Matching, on the same vertex set and on edge set equal to the pairs of $\mathcal{M}$ where both endpoints belong to the same clique. In the case of vertex integrity, it holds that $\text{OPT}(\mathcal{I}[G-S])$ can be computed in FPT time, since it is comprised of $\mathcal{O}(n)$ instances of size at most vi, each of which is solvable in time $2^{\mathcal{O}(\text{vi})}n^{\mathcal{O}(1)}$ due to Theorem 5.1. □

Theorem 5.5. *Given an instance $\mathcal{I} = (G, \mathcal{M})$ of MaxNDP, one can $(1-\varepsilon)$-approximate $\text{OPT}(\mathcal{I})$ in time $2^{\mathcal{O}(2^{\text{td}}/\varepsilon)}n^{\mathcal{O}(1)}$, where* td *denotes the tree-depth of G.*

Proof. Since G has tree-depth td, one can compute an *elimination forest* of G in time $2^{\mathcal{O}(\text{td}^2)}n^{\mathcal{O}(1)}$ due to [247, 260]. This is a rooted forest on the same vertex set where every pair of vertices adjacent in G adheres to the ancestor/descendant relation. Assume that G is connected, otherwise solve for each connected component. Let r denote the root of the elimination tree, in which case it follows that every connected component of $G-r$ has tree-depth at most $\text{td}-1$. Notice that it holds that $\text{OPT}(\mathcal{I}[G-r]) \geq \text{OPT}(\mathcal{I}) - 1$, since r can be used to route at most one demand. Let $\varepsilon' < \varepsilon$ and consider the case where

$$\text{OPT}(\mathcal{I}) - 1 \geq \frac{1-\varepsilon}{1-\varepsilon'} \cdot \text{OPT}(\mathcal{I}) \iff \text{OPT}(\mathcal{I}) \geq \frac{1-\varepsilon'}{\varepsilon-\varepsilon'},$$

thus setting $\varepsilon' = \varepsilon/2$ implies that $\text{OPT}(\mathcal{I}) \geq \frac{2-\varepsilon}{\varepsilon}$. In this case, it holds that an $(1-\varepsilon')$-approximation of $\text{OPT}(\mathcal{I}[G-r])$ is an $(1-\varepsilon)$-approximation of $\text{OPT}(\mathcal{I})$. On the other hand, if $\text{OPT}(\mathcal{I}) < \frac{2-\varepsilon}{\varepsilon} = \mathcal{O}(1/\varepsilon)$, we can use the algorithm of Theorem 5.2, running in time $2^{\mathcal{O}(2^{\text{td}}/\varepsilon)}n^{\mathcal{O}(1)}$.

Consequently, one can recursively argue about the existence of an approximation scheme, since in the case where a graph has tree-depth equal to 1, the instance is polynomial-time solvable. We proceed with bounding the scheme's running time. Let $T(\text{td}, \varepsilon)$ denote the running time for graph G with tree-depth td and error ε, while n' denotes the number of connected components of $G-r$. Notice that it holds that

$$T(\text{td}, \varepsilon) \leq \max\Big\{2^{\mathcal{O}(2^{\text{td}}/\varepsilon)}n^{\mathcal{O}(1)}, n' \cdot T(\text{td}-1, \varepsilon/2)\Big\} \leq 2^{\mathcal{O}(2^{\text{td}}/\varepsilon)}n^{\mathcal{O}(1)} + n' \cdot T(\text{td}-1, \varepsilon/2),$$

while the number of the nodes of the same height in the recursion tree is at most n, since each node corresponds to a connected component. Consequently, it holds that

$$T(\text{td}, \varepsilon) \leq 2^{\mathcal{O}(2^{\text{td}}/\varepsilon)}n^{\mathcal{O}(1)} + n \cdot \sum_{i=1}^{\text{td}} 2^{\mathcal{O}(2^{\text{td}-i} \cdot \frac{1}{\varepsilon/2^i})}n^{\mathcal{O}(1)} = 2^{\mathcal{O}(2^{\text{td}}/\varepsilon)}n^{\mathcal{O}(1)}.$$ □

5.2. Inapproximability

Given the FPT approximation scheme of Theorem 5.5, a natural question arising is whether such an approximation scheme exists for the parameterization by pathwidth as well. Due to the W[1]-hardness of MaxNDP parameterized by $\ell + \mathrm{pw} + \mathrm{fvs}$ [239], there can be no $(1-\varepsilon)$-approximation scheme running in time $f(\mathrm{pw}, \mathrm{fvs}, \varepsilon)n^{\mathcal{O}(1)}$, yet one of running time $f(\mathrm{pw}, \mathrm{fvs}, \varepsilon)n^{g(\varepsilon)}$ might be possible.[2] In this section we answer this question in the negative and prove that there exists some constant $0 < c' < 1$ such that MaxNDP cannot be approximated within a factor of c' in time $f(\mathrm{pw}, \mathrm{fvs})n^{\mathcal{O}(1)}$, for any function f, unless the Parameterized Inapproximability Hypothesis [228] fails.

Theorem 5.6. *Assuming the PIH, there exists a constant $c' > 0$ such that MaxNDP does not admit a c'-approximation algorithm of running time $f(\mathrm{pw}, \mathrm{fvs})n^{\mathcal{O}(1)}$.*

Proof. Let $\mathcal{I} = (G, k)$ be an instance of Multicolored Densest k-Subgraph, and recall that we assume that G is given to us partitioned into k independent sets $V_1, \ldots, V_k$, where $V_i = \{v_1^i, \ldots, v_n^i\}$. Moreover, let $E^{i_1,i_2} \subseteq E(G)$ denote the edges of G with one endpoint in V_{i_1} and the other in V_{i_2}. For every color class $i \in [k]$, let $s_i = |\{j \in [k] : E^{i,j} \neq \varnothing\}|$, and assume without loss of generality that $1 \leq s_i \leq 3$. Set $\sigma = \sum_{i \in [k]} s_i/2$, and notice that $\frac{k}{2} \leq \sigma \leq \frac{3k}{2}$. Let $\mathrm{OPT}(\mathcal{I}) \leq \sigma$ denote the optimal value of instance $\mathcal{I}$, i.e., the maximum number of edges among the induced subgraphs of G that contain one vertex per color class V_i.

We will construct in polynomial time an instance $\mathcal{J} = (H, \mathcal{M})$ of MaxNDP, where $\mathrm{pw}(H) = \mathcal{O}(k)$, $\mathrm{fvs}(H) = \mathcal{O}(k)$, and $|V(H)| = n^{\mathcal{O}(1)}$, while $\mathcal{M} \subseteq \binom{V(H)}{2}$ is a set of demands. Moreover, it will hold that $\mathrm{OPT}(\mathcal{J}) \leq \ell$, where $\mathrm{OPT}(\mathcal{J})$ denotes the optimal value of instance $\mathcal{J}$, and $\ell = 2k + \sigma$. We will present a reduction such that (i) if $\mathrm{OPT}(\mathcal{I}) = \sigma$, then $\mathrm{OPT}(\mathcal{J}) = \ell$, and (ii) if $\mathrm{OPT}(\mathcal{I}) < c \cdot \sigma$, then $\mathrm{OPT}(\mathcal{J}) < c' \cdot \ell$, for constants c and c' where $0 < c < 1$ and $c' = \frac{9+c}{10}$.

Choice Gadget. For an independent set V_i, we construct the *choice gadget* $\hat{C}_i$ in the following way. First, for $p \in [n]$, we construct paths on vertex sets $\hat{P}_p^i = \{w_q^{i,p} : q \in [3]\}$, as well as paths $\hat{R}_p^i$ on 3 unnamed vertices each. Then, we introduce vertices v_p^i and u_p^i, for $p \in [n+1]$. Next, we add edges $\{u_{n+1}^i, v_1^i\}$ and $\{v_{n+1}^i, u_1^i\}$. Lastly, for $p \in [n]$, we add edges $\{v_p^i, w_1^{i,p}\}$ and $\{w_3^{i,p}, v_{p+1}^i\}$, as well as an edge from u_p^i to one endpoint of $\hat{R}_p^i$, and an edge from u_{p+1}^i to the other endpoint of $\hat{R}_p^i$. See Figure 5.2 for an illustration. Subsequently, add to $\mathcal{M}$ all pairs (v_p^i, u_{p+1}^i) and (v_p^i, u_{p-1}^i) for $p \in [2, n]$, as well as the pairs (v_1^i, u_2^i) and (v_{n+1}^i, u_n^i). Let $\mathcal{M}_c \subseteq \mathcal{M}$ denote all such pairs added to $\mathcal{M}$ in this step of the construction. Intuitively, we will consider a one-to-one mapping between the vertex v_p^i of V_i being chosen and the vertices of $\hat{P}_p^i$ not being used to route any of the demands in $\mathcal{M}_c$.

[2] Assuming there existed such an algorithm of running time $f(\mathrm{pw}, \mathrm{fvs}, \varepsilon)n^{\mathcal{O}(1)}$, setting $\varepsilon < 1/\mathrm{OPT}(\mathcal{I})$ results in obtaining a solution of value at least $(1-\varepsilon)\,\mathrm{OPT}(\mathcal{I}) > \mathrm{OPT}(\mathcal{I}) - 1$, i.e., optimal, in time $f(\mathrm{pw}, \mathrm{fvs}, \mathrm{OPT}(\mathcal{I}))n^{\mathcal{O}(1)}$.

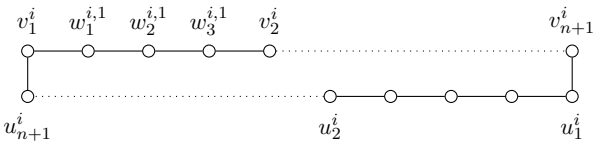


Figure 5.2.: Choice gadget $\hat{C}_i$.

Adjacency vertices. Let $E^{i,j} \neq \varnothing$. Introduce an *adjacency vertex* $e_{i,j}$, and add edges $\{e_{i,j}, w_x^{i,p}\}$ and $\{e_{i,j}, w_y^{j,p}\}$ where $p \in [n]$, and $x, y \in [3]$ such that no other adjacency vertex is adjacent to them (since $s_i \leq 3$ for all $i \in [k]$, there always exist such x and y). If $e = \{v_p^i, v_q^j\} \in E^{i,j}$, then add the pair $(w_x^{i,p}, w_y^{j,q})$ in $\mathcal{M}$. Notice that all adjacency vertices have disjoint neighborhoods, and let $F \subseteq V(H)$ be the set of all adjacency vertices, where $|F| = \sigma$.

This concludes the construction of the instance $\mathcal{J}$. Notice that $H - F$ is a collection of k choice gadgets, each of which is a cycle. Consequently, the graph obtained from $H - F$ by deleting a vertex per choice gadget is a collection of paths, thus both $\mathrm{pw}(H)$ and $\mathrm{fvs}(H)$ are at most $\mathcal{O}(k)$.

We first prove that if $\mathrm{OPT}(\mathcal{I}) = \sigma$, then $\mathrm{OPT}(\mathcal{J}) = \ell$. Consider a function $h\colon [k] \to [n]$ such that $G[\mathcal{V}]$ has σ edges, where $\mathcal{V} = \{v_{h(i)}^i \in V_i : i \in [k]\} \subseteq V(G)$. We construct a family of ℓ vertex-disjoint paths as follows. First, for every $i \in [k]$, we route two demands in $\hat{C}_i$. In particular, we route the demands $(v_{h(i)}^i, u_{h(i)+1}^i)$ as well as $(v_{h(i)+1}^i, u_{h(i)}^i)$, using the vertices of the shortest path of $\hat{C}_i$ in each case. Note that in this step we have created $2k$ vertex-disjoint paths connecting terminal pairs belonging to $\mathcal{M}_c$, and in every gadget $\hat{C}_i$ the only unused vertices are those of $\hat{P}_{h(i)}^i$ and $\hat{R}_{h(i)}^i$. Then, consider the adjacency vertex $e_{i,j}$, where $i, j \in [k]$. Route the demand $(w_x^{i,h(i)}, w_y^{j,h(j)})$ via $e_{i,j}$, since both endpoints of the demand are its neighbors and have not been used in any path so far, for some $x, y \in [3]$. Such a demand indeed exists in $\mathcal{M}$, since $\{v_{h(i)}^i, v_{h(j)}^j\} \in E^{i,j}$; if that were not the case then $G[\mathcal{V}]$ has less than σ edges. This procedure results in σ additional demands being routed. Notice that since the neighborhoods of all adjacency vertices are disjoint, the ℓ resulting paths are indeed vertex-disjoint.

It remains to prove that if $\mathrm{OPT}(\mathcal{I}) < c \cdot \sigma$, then $\mathrm{OPT}(\mathcal{J}) < c' \cdot \ell$ (or its contrapositive, as we will do in actuality). We start with Claim 5.7.

▷ Claim 5.7. At most 2 demands can be routed in a choice gadget $\hat{C}_i$ in $H - F$, in which case the only unused vertices are those of $\hat{P}_p^i$ and $\hat{R}_p^i$, for some $p \in [n]$. Additionally, it holds that $\mathrm{OPT}(\mathcal{J}) \leq \ell$.

Proof of the claim. Let $N = 3n + (n+1)$ and notice that $\hat{C}_i$ is a C_{2N}, thus there are exactly 2 simple paths connecting any pair of its vertices. Moreover, the number of vertices in any simple path between v_p^i and u_p^i, including said endpoints, is $N + 1$. Consequently, to route any demand with both endpoints in $\hat{C}_i$, either $N + 1 - 4$ or $N + 1 + 4$ vertices are used. In case more than 2 demands are routed, at least $3(N + 1 - 4) = 3N - 9$ vertices are used, which is a contradiction since $\hat{C}_i$ consists of $2N$ vertices.

Assume that exactly 2 demands are routed. Moreover, assume that a demand is routed using $N+1+4$ vertices. Then, both routed demands use at least $(N+1+4)+(N+1-4) > 2N$ vertices, which is a contradiction. Therefore, both demands that are routed in $\hat{C}_i$ use the shortest path connecting their endpoints.

Let (v^i_p, u^i_q) and $(v^i_{p'}, u^i_{q'})$ denote the demands routed, where $p < p'$ and $p \in [n]$. It holds that $q = p + 1$, since otherwise $q = p - 1$, and $v^i_{p'}$ belongs to the shortest path connecting (v^i_p, u^i_q), a contradiction. Symmetrically, it follows that $q' = p' - 1$. Lastly, it holds that $q > q'$, since otherwise $u^i_{q'}$ belongs to the shortest path connecting (v^i_p, u^i_q). Consequently, it follows that $q > q' \iff p + 2 > p'$, which implies that $p < p' < p + 2$, i.e., $p' = p + 1$. In that case, the only unused vertices are those of $\hat{P}^i_p$ and $\hat{R}^i_p$.

Notice that $H - F$ is a collection of k choice gadgets. Since in each such gadget at most 2 demands can be routed, it follows that $\mathrm{OPT}(\mathcal{J}) \leq 2k + |F| = 2k + \sigma = \ell$. $\lhd$

We now move on to prove that if $\mathrm{OPT}(\mathcal{J}) \geq c' \cdot \ell$, then $\mathrm{OPT}(\mathcal{I}) \geq c \cdot \sigma$. Let $\mathcal{P}$ denote a collection of $\mathrm{OPT}(\mathcal{J})$ vertex-disjoint paths of H routing demands of $\mathcal{M}$, where $\mathrm{OPT}(\mathcal{J}) \geq c' \cdot \ell$. Additionally, let $\mathcal{C}^{\mathcal{P}}_j$ contain the choice gadgets $\hat{C}_i$ such that there exist *exactly* j paths in $\mathcal{P}$ that route demands in the graph induced by $\hat{C}_i$, for $j \in [0, 2]$. Notice that $(\mathcal{C}^{\mathcal{P}}_0, \mathcal{C}^{\mathcal{P}}_1, \mathcal{C}^{\mathcal{P}}_2)$ defines a partition of the choice gadgets due to Claim 5.7. Moreover, let $\mathcal{P}^F \subseteq \mathcal{P}$ be comprised of the paths of $\mathcal{P}$ that contain vertices of F. We will say that a path $P \in \mathcal{P}^F$ *intersects* $\hat{C}_i$ if P contains a vertex of $\hat{C}_i$. We define the *loss* of a solution $\mathcal{P}$ to be $L_{\mathcal{P}} = \ell - |\mathcal{P}|$. Notice that due to Claim 5.7 it holds that $L_{\mathcal{P}} = 2|\mathcal{C}^{\mathcal{P}}_0| + |\mathcal{C}^{\mathcal{P}}_1| + (|F| - |\mathcal{P}^F|)$.

It holds that $\mathrm{OPT}(\mathcal{J}) = |\mathcal{P}| = \ell - L_{\mathcal{P}} \geq c' \cdot \ell = \ell - (1 - c') \cdot \ell$, thus it follows that $L_{\mathcal{P}} \leq (1 - c') \cdot \ell$. Construct a new solution $\mathcal{P}'$ in the following way: for every $\hat{C}_i \in \mathcal{C}^{\mathcal{P}}_0 \cup \mathcal{C}^{\mathcal{P}}_1$, route two demands of $\mathcal{M}_c$ such that there exist two vertex-disjoint paths using only vertices of $\hat{C}_i$. Afterwards, remove any paths of $\mathcal{P}^F$ that are not vertex-disjoint with those. There are at most 3 vertices of F adjacent to vertices of $\hat{C}_i$, and by choosing the routed demands of $\hat{C}_i$ in such a way that at least one of those vertices of $\hat{C}_i$ remains unused, it follows that at most 2 paths of $\mathcal{P}^F$ intersect $\hat{C}_i$. Thus it follows that $L_{\mathcal{P}'} \leq (|F| - |\mathcal{P}^F|) + 2(|\mathcal{C}^{\mathcal{P}}_0| + |\mathcal{C}^{\mathcal{P}}_1|) \leq 2L_{\mathcal{P}}$, therefore $|\mathcal{P}'| = \ell - L_{\mathcal{P}'} \geq \ell - 2L_{\mathcal{P}}$. Furthermore, notice that since for all $i \in [k]$ it holds that $\hat{C}_i \in \mathcal{C}^{\mathcal{P}'}_2$, due to Claim 5.7 it follows that only the vertices of $\hat{P}^i_{h(i)}$ and $\hat{R}^i_{h(i)}$ remain unused by the paths of $\mathcal{P}'$ that route demands in $\mathcal{M}_c$, for some function $h\colon [k] \to [n]$. Consequently, for any routed demand $(w^{i,p}_x, w^{j,q}_y) \in \mathcal{M} \setminus \mathcal{M}_c$, it holds that $p = h(i)$ and $q = h(j)$.

Let $\mathcal{V} = \{v^i_{h(i)} : i \in [k]\}$, and notice that $|\mathcal{V} \cap V_i| = 1$ for all $i \in [k]$. Let $A = |E(G[\mathcal{V}])|$ denote the number of edges present in the subgraph induced by $\mathcal{V}$. We will prove that $A \geq c \cdot \sigma$, in which case it follows that $\mathrm{OPT}(\mathcal{I}) \geq c \cdot \sigma$. Notice that $A \geq \ell - 2L_{\mathcal{P}} - 2k = \sigma - 2L_{\mathcal{P}}$, since this is the number of routed demands in $\mathcal{M} \setminus \mathcal{M}_c$ by $\mathcal{P}'$, while $(w^{i,h(i)}_x, w^{j,h(j)}_y) \in \mathcal{M}$ implies that $\{v^i_{h(i)}, v^j_{h(j)}\} \in E^{i,j}$.

It suffices to prove that $\sigma - 2L_{\mathcal{P}} \geq c \cdot \sigma$. Since $\sigma - 2L_{\mathcal{P}} \geq \sigma - 2(1 - c') \cdot \ell$, we will prove $\sigma - 2(1 - c') \cdot \ell \geq c \cdot \sigma$ instead, which is equivalent to $c' \geq 1 + \frac{\sigma}{2\ell}(c - 1)$. Since $c - 1 < 0$ and $\frac{\sigma}{2\ell} = \frac{1}{2} \cdot \frac{\sigma}{2k+\sigma} \geq \frac{1}{2} \cdot \frac{k/2}{2k+k/2} = \frac{1}{10}$, the statement holds for $c' = 1 + \frac{1}{10}(c-1) = \frac{9+c}{10}$. $\square$

5.3. XNLP-completeness

The W[1]-hardness results of [112, 239] already imply that MaxNDP is W[1]-hard parameterized by the pathwidth of the input graph. Here we examine in more detail the parameterization solely by pathwidth, and prove that in this case the problem is in fact XNLP-complete. This complexity class was recently brought forth by Bodlaender, Groenland, Nederlof, and Swennenhuis [41] and consists of the parameterized problems such that an instance (x, k), where x can be encoded with n bits and k denotes the parameter, can be solved non-deterministically in time $f(k)n^{\mathcal{O}(1)}$ and space $f(k)\log n$, for some computable function f.

Such a completeness result in fact implies that MaxNDP parameterized by pathwidth is W[t]-hard for all $t \in \mathbb{N}$. To prove said result, we reduce from the XNLP-complete Chained Multicolored Clique, and use a construction quite similar to the one of Theorem 5.6.

Theorem 5.8. *MaxNDP parameterized by the pathwidth of the input graph is XNLP-complete.*

Proof. We first argue that MaxNDP parameterized by the pathwidth of the input graph belongs to XNLP. Let $\mathcal{I} = (G, \mathcal{M}, \ell)$ be an instance of MaxNDP, where $n = |\mathcal{I}|$ denotes the number of bits needed to encode $\mathcal{I}$, and assume that we are also given a path decomposition of G of width pw. We describe a non-deterministic algorithm for $\mathcal{I}$ that uses $\mathcal{O}(\text{pw}\log n)$ bits of memory. Fix an optimal solution, and observe that for each bag, at most pw paths of the solution intersect the bag. The algorithm guesses and stores for each bag both endpoints of the paths of the solution intersecting it ($2\text{pw}\log n$) bits. Moreover, the algorithm guesses and stores for each vertex of the bag an integer from $[0, \ell]$, where 0 indicates that this vertex is not used in the solution, otherwise said vertex is part of the i-th satisfied demand path. For vertices for which we have not stored 0, we also remember their degree, i.e., how many of their neighbors that belong to the same path are present in the bags of the decomposition up to this point (this is an integer in $\{0, 1, 2\}$). In total, $\mathcal{O}(\text{pw}\log n)$ bits are used for this information. It is easy to guess and update these values when introducing a vertex. When Forgetting, the vertex must either have degree 2, or alternatively degree 1 and be an endpoint of its path. If both endpoints of a path have been Forgotten, we remove this from the list of active paths. Keep a counter of how many correct satisfied paths we have seen, using $\mathcal{O}(\log n)$ additional bits, and check in the end that it is at least ℓ.

In order to prove the XNLP-hardness, we present a *parameterized logspace reduction* [41] from Chained Multicolored Clique, which is known to be XNLP-complete parameterized by k [41], and is formally defined as follows:

Chained Multicolored Clique

Instance: Graph $G = (V, E)$, a partition of V into sets $V_1, \ldots, V_r$, as well as a coloring function $f\colon V \to [k]$, where for every $\{u, v\} \in E$, if $u \in V_{i_1}$ and $v \in V_{i_2}$, then $|i_1 - i_2| \leq 1$.

Goal: Determine whether there exists $W \subseteq V$ such that for every $i \in [r-1]$, $W \cap (V_i \cup V_{i+1})$ is a clique, and for all $i \in [r]$ and $j \in [k]$, there is a vertex $w \in W \cap V_i$ with $f(w) = j$.

On a high level, Chained Multicolored Clique asks whether there exists a clique with $2k$ vertices in $V_i \cup V_{i+1}$ for each $i \in [r-1]$, containing a vertex of color j both in V_i and in V_{i+1}, for all colors $j \in [k]$. Importantly, the same vertices in V_i are chosen in the clique for both $V_{i-1} \cup V_i$ and $V_i \cup V_{i+1}$. We call such a set a *chained multicolored clique*. Furthermore, one can assume without loss of generality that $|\{v \in V_i : f(v) = j\}| = n$, for every $i \in [r]$ and $j \in [k]$, since one can arbitrarily add a sufficient number of vertices of degree 0.

Let $(G, f, \{V_i : i \in [r]\})$ be an instance of Chained Multicolored Clique, and let $V_{i,j} = \{v_1^{i,j}, \ldots, v_n^{i,j}\}$ denote the set of vertices belonging to V_i that are of color j. We will construct in polynomial time an equivalent instance $(H, \mathcal{M}, \ell)$ of MaxNDP, where H is a graph of pathwidth $\mathrm{pw}(H) = \mathcal{O}(k^2)$ and size $|V(H)| = \mathcal{O}(|V(G)| \cdot k)$, $\mathcal{M} \subseteq \binom{V(H)}{2}$ is a set of demands, and $\ell = 2rk + r\binom{k}{2} + (r-1)k^2$, such that G has a chained multicolored clique if and only if at least ℓ demands of $\mathcal{M}$ can be routed.

The construction will be very similar to the one of Theorem 5.6, albeit with a few differences. In particular, we will once again employ the use of the choice gadgets introduced there and we refer the reader to Figure 5.2 for an illustration.

Choice Gadget. For every set $V_{i,j}$, where $i \in [r]$ and $j \in [k]$, we construct the *choice gadget* $\hat{C}_{i,j}$ in the following way. First, for $p \in [n]$, we construct paths on vertex sets $\hat{P}_p^{i,j} = \{w_q^{i,j,p} : q \in [3k]\}$, as well as paths $\hat{R}_p^{i,j}$ on $3k$ unnamed vertices each. Then, we introduce vertices $v_p^{i,j}$ and $u_p^{i,j}$, for $p \in [n+1]$. Next, we add edges $\{u_{n+1}^{i,j}, v_1^{i,j}\}$ and $\{v_{n+1}^{i,j}, u_1^{i,j}\}$. Lastly, for $p \in [n]$, we add edges $\{v_p^{i,j}, w_1^{i,j,p}\}$ and $\{w_{3k}^{i,j,p}, v_{p+1}^{i,j}\}$, as well as an edge from $u_p^{i,j}$ to one endpoint of $\hat{R}_p^{i,j}$, and an edge from $u_{p+1}^{i,j}$ to the other endpoint of $\hat{R}_p^{i,j}$. Subsequently, add to $\mathcal{M}$ all pairs $(v_p^{i,j}, u_{p+1}^{i,j})$ as well as $(v_p^{i,j}, u_{p-1}^{i,j})$, for $p \in [2, n]$, as well as the pairs $(v_1^{i,j}, u_2^{i,j})$ and $(v_{n+1}^{i,j}, u_n^{i,j})$. Let $\mathcal{M}_c \subseteq \mathcal{M}$ denote all such pairs added to $\mathcal{M}$ in this step of the construction. Intuitively, we will consider a one-to-one mapping between the vertex $v_p^{i,j}$ of $V_{i,j}$ belonging to a supposed chained multicolored clique of G and the vertices of $\hat{P}_p^{i,j}$ not being used to route any of the demands in $\mathcal{M}_c$.

Adjacency vertices. Next, we introduce some *adjacency vertices* into the graph. For every $i \in [r]$ and $\{j_1, j_2\} \in \binom{[k]}{2}$, introduce vertex e_{j_1,j_2}^i and add edges $\{e_{j_1,j_2}^i, w_{j_2}^{i,j_1,p}\}$ and $\{e_{j_1,j_2}^i, w_{j_1}^{i,j_2,p}\}$ for all $p \in [n]$. Additionally, if $\{v_p^{i,j_1}, v_q^{i,j_2}\} \in E(G)$, then add the pair $(w_{j_2}^{i,j_1,p}, w_{j_1}^{i,j_2,q})$ in $\mathcal{M}$. Next, for every $i \in [r-1]$ and $j, j' \in [k]$, introduce vertex

$e^{i,i+1}_{j,j'}$ and add edges $\{e^{i,i+1}_{j,j'}, w^{i,j,p}_{2k+j'}\}$ and $\{e^{i,i+1}_{j,j'}, w^{i+1,j',p}_{k+j}\}$ for all $p \in [n]$. Additionally, if $\{v^{i,j}_p, v^{i+1,j'}_q\} \in E(G)$, then add the pair $(w^{i,j,p}_{2k+j'}, w^{i+1,j',q}_{k+j})$ in $\mathcal{M}$. Notice that all adjacency vertices have disjoint neighborhoods.

A *set gadget* $\hat{S}_i$, where $i \in [r]$, is composed of the choice gadgets $\hat{C}_{i,j}$ for all $j \in [k]$ as well as all the adjacency vertices $e^i_{j,j'}$, where $\{j, j'\} \in \binom{[k]}{2}$.

This concludes the construction of H. For an illustration, see Figure 5.3. It holds that $|V(H)| = rk(6nk + 2n + 2) + r\binom{k}{2} + (r-1)k^2 = \mathcal{O}(rnk^2)$, while $|V(G)| = rnk$, therefore $|V(H)| = \mathcal{O}(|V(G)| \cdot k)$ follows. It remains to argue about the space usage of the algorithm that constructs the instance $(H, \mathcal{M}, \ell)$. Notice that in order to uniquely identify the vertices of H, it suffices to keep track of 5 different indices: one identifies the kind of vertex (u, v, w, $e^i_{j_1,j_2}$, $e^{i,i+1}_{j,j'}$, or a vertex of some path $\hat{R}$ in a choice gadget), and the rest are used to distinguish among vertices of the same kind. In that case, the number of all such encodings is $(nrk)^{\mathcal{O}(1)}$. Moreover, given one such encoding, we can determine the other encodings (i.e., vertices) with which the first either has an edge, or takes part in a demand, since it suffices to only alter some of these indices by some constant, thus the space usage is $\mathcal{O}(\log(nkr))$.

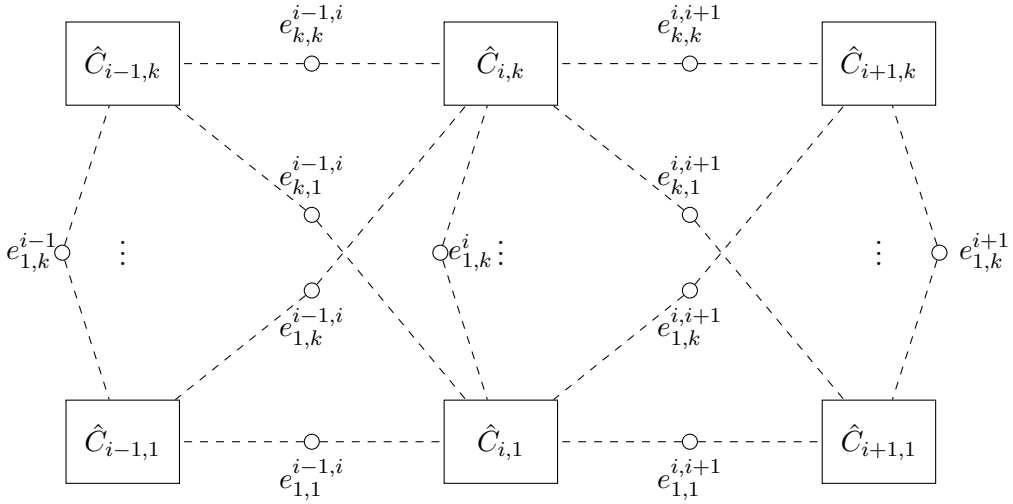


Figure 5.3.: Part of graph H regarding set gadgets $\hat{S}_{i-1}$, $\hat{S}_i$, and $\hat{S}_{i+1}$. The vertices denote the middle vertex of the corresponding adjacency path, while dashed lines indicate that the respective adjacency path has neighbors in the corresponding choice gadget.

Lemma 5.9. *It holds that* $\mathrm{pw}(H) = \mathcal{O}(k^2)$.

Proof. In the following, let j_1, j_2, j, and j' such that $\{j_1, j_2\} \in \binom{[k]}{2}$ and $j, j' \in [k]$. Start with $i = 1$ and consider a path decomposition that consists of bags containing all vertices $e^{i-1,i}_{j,j'}$, $e^i_{j_1,j_2}$, and $e^{i,i+1}_{j,j'}$, and then going through all choice gadgets of level i one by one, before moving on to level $i + 1$ and so on, until level r. □

Lemma 5.10. *If G contains a chained multicolored clique, then $(H, \mathcal{M}, \ell)$ is a Yes instance of* MaxNDP.

Proof. Let $\mathcal{W} \subseteq V(G)$ be a chained multicolored clique of G, consisting of vertices $\{v^{i,j}_{s(i,j)} : i \in [r], j \in [k]\}$. We construct a family of ℓ vertex-disjoint paths as follows.

First, for every $i \in [r]$ and $j \in [k]$, we route two demands in $\hat{C}_{i,j}$. In particular, we route the demands $(v^{i,j}_{s(i,j)}, u^{i,j}_{s(i,j)+1})$ as well as $(v^{i,j}_{s(i,j)+1}, u^{i,j}_{s(i,j)})$, using the vertices of the shortest path of $\hat{C}_{i,j}$ in each case. Note that in this step we have created $2rk$ vertex-disjoint paths connecting terminal pairs belonging to $\mathcal{M}_c$, and in every gadget $\hat{C}_{i,j}$ the only unused vertices are those of $\hat{P}^{i,j}_{s(i,j)}$ and $\hat{R}^{i,j}_{s(i,j)}$.

Then, for $i \in [r]$ and $\{j_1, j_2\} \in \binom{[k]}{2}$, consider the adjacency vertex $e^{i}_{j_1,j_2}$. Route the demand $(w^{i,j_1,s(i,j_1)}_{j_2}, w^{i,j_2,s(i,j_2)}_{j_1})$ via said adjacency vertex, since both endpoints of the demand are its neighbors and have not been used in any path so far. This procedure results in $r\binom{k}{2}$ additional demands being routed.

Lastly, for $i \in [r-1]$ and $j, j' \in [k]$, consider the adjacency vertex $e^{i,i+1}_{j,j'}$. Route the demand $(w^{i,j,s(i,j)}_{2k+j'}, w^{i+1,j',s(i+1,j')}_{k+j})$ via said adjacency vertex, since both endpoints of the demand are its neighbors and have not been used in any path so far. Such a demand indeed exists in $\mathcal{M}$, since $\{v^{i,j}_{s(i,j)}, v^{i+1,j'}_{s(i+1,j')}\} \in E(G)$. This procedure results in $(r-1)k^2$ additional demands being routed.

Notice that since the neighborhoods of all adjacency vertices are disjoint, the ℓ resulting paths are indeed vertex disjoint. □

Lemma 5.11. *If $(H, \mathcal{M}, \ell)$ is a Yes instance of* MaxNDP*, then G contains a chained multicolored clique.*

Proof. Let $\mathcal{P} = \{P_1, \ldots, P_\ell\}$ be a set of ℓ vertex-disjoint paths connecting terminal pairs of $\mathcal{M}$ in H. Assume without loss of generality that said paths are simple, as well as that the only edges among vertices of a path appear between its consecutive vertices. Moreover, define $\mathcal{P}_c = \{P_i \in \mathcal{P} : P_i \text{ routes a demand in } \mathcal{M}_c\}$. Set S to be the set consisting of all the adjacency vertices in H, where $|S| = r\binom{k}{2} + (r-1)k^2$.

Notice that at most $|S|$ paths of $\mathcal{P}$ contain vertices of S, consequently at least $\ell - |S| = 2rk$ paths route demands using only vertices of $H - S$. Moreover, $H - S$ is a collection of rk disconnected choice gadgets. We start with Claim 5.12, whose proof is omitted since it is analogous to the one of Claim 5.7.

▷ Claim 5.12. At most 2 demands can be routed in a choice gadget $\hat{C}_{i,j}$ in $H - S$, in which case the only unused vertices are those of $\hat{P}^{i,j}_p$ and $\hat{R}^{i,j}_p$, for some $p \in [n]$.

Consequently, due to Claim 5.12, it follows that exactly $|S|$ paths of $\mathcal{P}$ contain vertices of S. Moreover, in each choice gadget $\hat{C}_{i,j}$, exactly 2 demands are routed, for a total of $2rk$ demands of $\mathcal{M}_c$, leaving unused only the vertices of $\hat{P}^{i,j}_p$ and $\hat{R}^{i,j}_p$ for some $p \in [n]$. Next we prove that any path not routing a demand of $\mathcal{M}_c$ contains exactly 3 vertices.

▷ Claim 5.13. If $P \in \mathcal{P} \setminus \mathcal{P}_c$, then P is of size 3.

Proof of the claim. Since $P \notin \mathcal{P}_c$, it contains a single vertex of S, denoted by s. Let $\hat{C}_{i_1,j_1}$ and $\hat{C}_{i_2,j_2}$ denote the 2 choice gadgets s has neighbors in. Set $S' = S \setminus \{s\}$. Then, it holds that P routes one demand of $\mathcal{M} \setminus \mathcal{M}_c$, using the vertices of the graph $H - S'$. In that case, one endpoint of such a demand can only be a vertex of $\hat{C}_{i_1,j_1}$ and the other of $\hat{C}_{i_2,j_2}$, and due to the construction of $(H, \mathcal{M}, \ell)$, it holds that for every such demand, both its endpoints are neighbors of s. ◁

Let $\mathcal{W} \subseteq V(G)$ be a set of cardinality rk, containing vertex $v^{i,j}_{s(i,j)} \in V_i$ if, for choice gadget $\hat{C}_{i,j}$, it holds that the vertices of $\hat{P}^{i,j}_{s(i,j)}$ are not used to route demands of $\mathcal{M}_c$. Notice that $|\{w \in W \cap V_i : f(w) = j\}| = 1$, for all $i \in [r]$ and $j \in [k]$. We will prove that $\mathcal{W}$ is a chained multicolored clique of G.

Let i_1, i_2, j_1 and j_2 such that (i) either $i_1 = i_2$ and $\{j_1, j_2\} \in \binom{[k]}{2}$, (ii) or $i_2 = i_1 + 1$ and $j_1, j_2 \in [k]$. Let $v^{i_1,j_1}_{s(i_1,j_1)}, v^{i_2,j_2}_{s(i_2,j_2)}$ belong to $\mathcal{W}$. Let $P \in \mathcal{P}$ denote the path containing vertex e, where $e = e^{i_1}_{j_1,j_2}$ in case (i) and $e = e^{i_1,i_2}_{j_1,j_2}$ otherwise. Due to Claim 5.13, it holds that P is comprised of e, as well as two of its neighbors, one in $\hat{C}_{i_1,j_1}$ and one in $\hat{C}_{i_2,j_2}$. Since the only neighbors of e that are not used by paths in $\mathcal{P}_c$ are $w^{i_1,j_1,s(i_1,j_1)}_{c_1+j_2}$ and $w^{i_2,j_2,s(i_2,j_2)}_{c_2+j_1}$, we infer that $(w^{i_1,j_1,s(i_1,j_1)}_{c_1+j_2}, w^{i_2,j_2,s(i_2,j_2)}_{c_2+j_1}) \in \mathcal{M}$, where $c_1 = c_2 = 0$ in case (i) and $c_1 = 2k$, $c_2 = k$ otherwise. Consequently, $\{v^{i_1,j_1}_{s(i_1,j_1)}, v^{i_2,j_2}_{s(i_2,j_2)}\} \in E(G)$. Since this holds for any two such vertices belonging to $\mathcal{W}$, it follows that G has a chained multicolored clique. □

Therefore, in polynomial time and with logarithmic space, we can construct a graph H, where $\mathrm{pw} = \mathcal{O}(k^2)$ due to Lemma 5.9, as well as a set of pairs $\mathcal{M}$, such that, due to Lemmas 5.10 and 5.11, deciding whether at least ℓ pairs of $\mathcal{M}$ can be routed is equivalent to deciding whether G has a chained multicolored clique. □

5.4. Refining Hardness for Bounded Tree-depth Graphs

In this section, we refine the hardness result of Ene, Mnich, Pilipczuk, and Risteski [112] for bounded tree-depth graphs, by employing a recursive structure introduced in [215]. The reduction of [112] starts from an instance (G, k) of k-Multicolored Clique, and produces an equivalent instance of MaxNDP on a graph of tree-depth, vertex integrity, and feedback vertex number $\mathcal{O}(k^2)$, implying a $n^{o(\sqrt{\mathrm{td}})}$ lower bound under the ETH. We refine their approach, resulting in a reduction that keeps tree-depth linear in k, thereby improving the lower bound to $n^{o(\mathrm{td})}$ under the ETH. As a consequence of our result, it follows that the standard $n^{\mathcal{O}(\mathrm{tw})}$ algorithm for the problem is optimal, even if one considers the class of graphs of bounded tree-depth.

In order to achieve this result, we combine ideas from both [112] and Chapter 3. On a high level, the reduction of [112] consists of k choice gadgets, each of which is used to encode the vertex which is chosen to take part in a supposed clique per color class. Afterwards, it suffices to add $\binom{k}{2}$ vertices in order to verify the existence of edges among all the chosen vertices of the color classes. The deletion of those $\binom{k}{2}$ vertices then gives the

bounds for the tree-depth of the graph. In order to avoid this quadratic dependence, we make use of a recursive structure introduced in Section 3.3 meant to verify the existence of edges between the chosen vertices, while keeping the tree-depth of the resulting graph linear in k.

Theorem 5.14. *For any computable function f, if there exists an algorithm that solves* MaxNDP *in time $f(\mathrm{td})n^{o(\mathrm{td})}$, where* td *denotes the tree-depth of the input graph, then the ETH is false.*

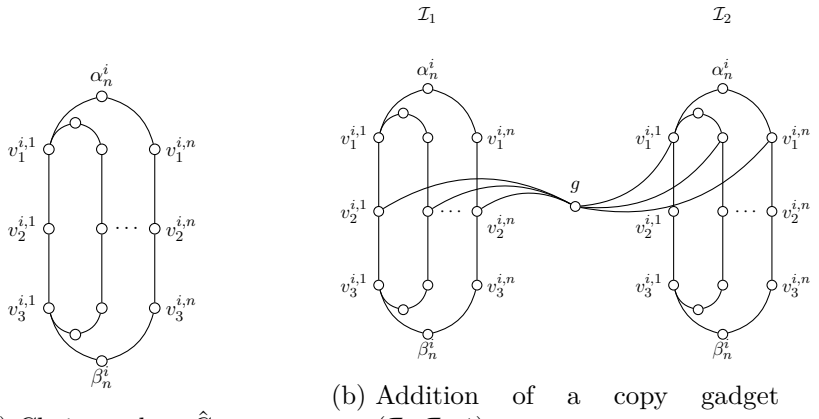
Proof. Let (G, k) be an instance of k-Multicolored Clique, such that every vertex of G has a self loop, i.e., $\{v, v\} \in E(G)$, for all $v \in V(G)$. Recall that we assume that G is given to us partitioned into k independent sets $V_1, \ldots, V_k$, where $V_i = \{v_1^i, \ldots, v_n^i\}$. Assume without loss of generality that $k = 2^z$, for some $z \in \mathbb{N}$ (one can do so by adding dummy independent sets connected to all the other vertices of the graph). Moreover, let $E^{i_1,i_2} \subseteq E(G)$ denote the edges of G with one endpoint in V_{i_1} and the other in V_{i_2}. We will construct in polynomial time an equivalent instance $(H, \mathcal{M}, \ell)$ of MaxNDP, where H is a graph of tree-depth $\mathrm{td}(H) = \mathcal{O}(k)$, feedback vertex number $\mathrm{fvs}(H) = \mathcal{O}(k^2)$, and size $|V(H)| = n^{\mathcal{O}(1)}$, $\mathcal{M} \subseteq \binom{V(H)}{2}$ is a set of demands, and ℓ is an integer, such that G has a k-clique if and only if at least ℓ demands of $\mathcal{M}$ can be routed in H.

Choice Gadget. For an independent set V_i, we construct the *choice gadget* $\hat{C}_i$ as depicted in Figure 5.4a. We first construct paths on vertex sets $\hat{P}_j^i = \{v_1^{i,j}, v_2^{i,j}, v_3^{i,j}\}$, where $j \in [n]$. Afterwards, for every $j \in [2, n]$, we introduce vertices α_j^i and β_j^i, connecting them with $v_1^{i,1}, v_1^{i,j}$ and $v_3^{i,1}, v_3^{i,j}$ respectively. Moreover, we add the pair (α_j^i, β_j^i) to $\mathcal{M}$. Intuitively, we will consider a one-to-one mapping between the vertex v_j^i of V_i belonging to a supposed k-clique of G and the vertices of $\hat{P}_j^i$ not being used to route any of the demands added in this step.

Copy Gadget. Given two instances $\mathcal{I}_1$, $\mathcal{I}_2$ of a choice gadget $\hat{C}_i$, when we say that we add a *copy gadget* $(\mathcal{I}_1, \mathcal{I}_2, t)$, where $t \in \{1, 2\}$, we introduce a *copy vertex* g, connect it with all vertices $v_{t+1}^{i,j}$ of $\mathcal{I}_1$ and all vertices $v_1^{i,j}$ of $\mathcal{I}_2$ for $j \in [n]$ and add all the pairs $(v_{t+1}^{i,j}, v_1^{i,j})$ to $\mathcal{M}$.

Adjacency Gadget. Let $i_1, i_2, i_1', i_2' \in [k]$. For $i_1 \leq i_2$ and $i_1' \leq i_2'$, we define the *adjacency gadget* $\hat{A}(i_1, i_2, i_1', i_2')$ as follows:

- Consider first the case when $i_1 = i_2 = i$ and $i_1' = i_2' = i'$. Let the adjacency gadget contain instances of the choice gadgets $\hat{C}_i$ and $\hat{C}_{i'}$, as well as a *validation vertex* $m_{i,i'}$. Add edges between $m_{i,i'}$ and all vertices $v_2^{i,j}$ and $v_2^{i',j}$ for $j \in [n]$. If $e = \{v_j^i, v_{j'}^{i'}\} \in E^{i,i'}$, then add the pair $(v_2^{i,j}, v_2^{i',j'})$ to $\mathcal{M}$.
- Now consider the case when $i_1 < i_2$ and $i_1' < i_2'$. Then, let $\hat{A}(i_1, i_2, i_1', i_2')$ contain instances of choice gadgets $\hat{C}_i$ and $\hat{C}_{i'}$, where $i \in [i_1, i_2]$ and $i' \in [i_1', i_2']$, which we will refer to as the *original choice gadgets* of $\hat{A}(i_1, i_2, i_1', i_2')$, as well as the adjacency gadgets



(a) Choice gadget $\hat{C}_i$.

(b) Addition of a copy gadget $(\mathcal{I}_1, \mathcal{I}_2, 1)$.

Figure 5.4.: Choice gadgets and how to copy them.

- $\hat{A}\Big(i_1, \Big\lfloor \frac{i_1+i_2}{2} \Big\rfloor, i'_1, \Big\lfloor \frac{i'_1+i'_2}{2} \Big\rfloor\Big)$,
- $\hat{A}\Big(i_1, \Big\lfloor \frac{i_1+i_2}{2} \Big\rfloor, \Big\lceil \frac{i'_1+i'_2}{2} \Big\rceil, i'_2\Big)$,
- $\hat{A}\Big(\Big\lceil \frac{i_1+i_2}{2} \Big\rceil, i_2, i'_1, \Big\lfloor \frac{i'_1+i'_2}{2} \Big\rfloor\Big)$,
- $\hat{A}\Big(\Big\lceil \frac{i_1+i_2}{2} \Big\rceil, i_2, \Big\lceil \frac{i'_1+i'_2}{2} \Big\rceil, i'_2\Big)$.

Lastly, let $\mathcal{I}$ denote the original choice gadget $\hat{C}_p$, where $p \in [i_1, i_2] \cup [i'_1, i'_2]$. Notice that there are exactly two instances of choice gadget $\hat{C}_p$ appearing as original choice gadgets in the adjacency gadgets just introduced, say instances $\mathcal{I}_1$ and $\mathcal{I}_2$. Add copy gadgets $(\mathcal{I}, \mathcal{I}_1, 1)$ and $(\mathcal{I}, \mathcal{I}_2, 2)$.

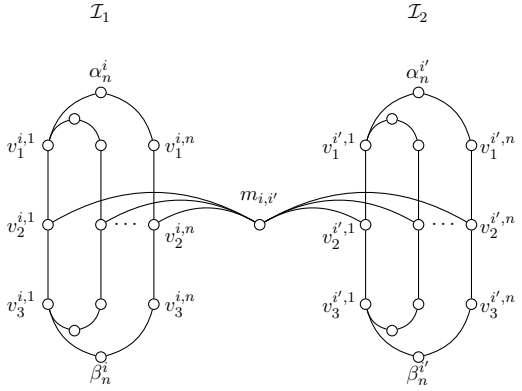


Figure 5.5.: Adjacency gadget $\hat{A}(i, i, i', i')$.

Let graph H be the adjacency gadget $\hat{A}(1, k, 1, k)$ and set $\ell = \gamma \cdot (n-1) + \delta$, where $\gamma = 2k(2k-1)$ and $\delta = 5k^2 - 4k$. Notice that it holds that $|V(H)| = (n \cdot k)^{\mathcal{O}(1)}$ as well as that all copy and validation vertices have disjoint neighborhoods. Additionally, let $\mathcal{M}_{\alpha\beta}$ denote the set of pairs of $\mathcal{M}$ of the form (α_j^i, β_j^i). This concludes the construction of the instance $(H, \mathcal{M}, \ell)$.

Lemma 5.15. *H has the following properties:*

- *γ is equal to the number of instances of choice gadgets present in H,*
- *δ is equal to the number of copy and validation vertices present in H.*

Proof. First we will prove that for every adjacency gadget $\hat{A}(i_1, i_2, i'_1, i'_2)$ appearing in H, it holds that $i_2 - i_1 = i'_2 - i'_1 = 2^c - 1$, for some $c \in \mathbb{N}$. The statement holds for $\hat{A}(1, k, 1, k)$, as well as when $i_2 - i_1 = i'_2 - i'_1 = 0$. Suppose that it holds for some $\hat{A}(i_1, i_2, i'_1, i'_2)$, i.e., $i_2 - i_1 = i'_2 - i'_1 = 2^c - 1 > 0$, for some $c \in \mathbb{N}$. Then, it follows that $\left\lfloor \frac{i_1+i_2}{2} \right\rfloor - i_1 = \lfloor i_2 - 2^{c-1} + 0.5 \rfloor - i_1 = i_2 - i_1 - 2^{c-1} = 2^{c-1} - 1$. Moreover, it follows that $i_2 - \left\lceil \frac{i_1+i_2}{2} \right\rceil = i_2 - \lceil i_1 + 2^{c-1} - 0.5 \rceil = i_2 - (i_1 + 2^{c-1}) = 2^{c-1} - 1$. Therefore, the stated property holds.

In that case, for some $\hat{A}(i_1, i_2, i'_1, i'_2)$, in every step of the recursion, intervals $[i_1, i_2]$ and $[i'_1, i'_2]$ are partitioned in the middle, and an adjacency gadget is considered for each of the four combinations. In that case, starting from $\hat{A}(1, k, 1, k)$, there is a single way to produce every adjacency gadget $\hat{A}(i_1, i_1, i_2, i_2)$, where $i_1, i_2 \in [k]$.

For the first statement, notice that the number of instances of choice gadgets is given by the recursive formula $T_1(k) = 2k + 4T_1(k/2)$, where $T_1(1) = 2$. In that case, it follows that

$$T_1(k) = \sum_{i=0}^{\log k} \left(4^i \cdot 2 \cdot \frac{k}{2^i} \right) = 2k \sum_{i=0}^{\log k} 2^i = 2k(2k-1) = \gamma.$$

For the second statement, notice that the number of copy plus the number of validation vertices is given by the recursive formula $T_2(k) = 4k + 4T_2(k/2)$, where $T_2(1) = 1$. Consequently,

$$\begin{aligned} T_2(k) &= \sum_{i=0}^{\log k - 1} \left(4^i \cdot 4 \cdot \frac{k}{2^i} \right) + 4^{\log k} = k^2 + 4k \sum_{i=0}^{\log k - 1} 2^i \\ &= k^2 + 4k(k-1) = 5k^2 - 4k = \delta, \end{aligned}$$

and the statement follows. □

Lemma 5.16. *It holds that* $\mathrm{td}(H) = \mathcal{O}(k)$ *and* $\mathrm{fvs}(H) = \mathcal{O}(k^2)$.

Proof. Let $T(\kappa)$ denote the tree-depth of $\hat{A}(i_1, i_2, i'_1, i'_2)$ in the case when $i_2 - i_1 = i'_2 - i'_1 = \kappa$. First, notice that, for $i_1, i_2 \in [k]$, the tree-depth of $\hat{A}(i_1, i_1, i_2, i_2)$ is less than 8. To see this, observe that by removing vertices $m_{i_1,i_2}, v_1^{i_1,1}, v_3^{i_1,1}, v_1^{i_2,1}$ and $v_3^{i_2,1}$, all remaining connected components are paths of at most 5 vertices. Consequently, $T(1) \leq 8$.

Now, consider the adjacency gadget $\hat{A}(i_1, i_2, i'_1, i'_2)$, where $i_2 - i_1 = i'_2 - i'_1 = \kappa$. This is comprised of adjacency gadgets

- $\hat{A}\left(i_1, \left\lfloor \frac{i_1+i_2}{2} \right\rfloor, i'_1, \left\lfloor \frac{i'_1+i'_2}{2} \right\rfloor\right)$,
- $\hat{A}\left(i_1, \left\lfloor \frac{i_1+i_2}{2} \right\rfloor, \left\lceil \frac{i'_1+i'_2}{2} \right\rceil, i'_2\right)$,
- $\hat{A}\left(\left\lceil \frac{i_1+i_2}{2} \right\rceil, i_2, i'_1, \left\lfloor \frac{i'_1+i'_2}{2} \right\rfloor\right)$,
- $\hat{A}\left(\left\lceil \frac{i_1+i_2}{2} \right\rceil, i_2, \left\lceil \frac{i'_1+i'_2}{2} \right\rceil, i'_2\right)$,

as well as of exactly 2κ original choice gadgets, each of which is connected with two copy gadgets to other instances of choice gadgets present in the adjacency gadgets. By removing all copy vertices g (see Figure 5.4b), all the original choice gadgets as well as the adjacency gadgets are disconnected. Therefore, it holds that $T(\kappa) \leq 4\kappa + T(\kappa/2)$, thus, it follows that

$$T(k) \leq 8 + 4 \sum_{i=0}^{\log k - 1} \frac{k}{2^i} = \mathcal{O}(k).$$

For the feedback vertex number, let $S \subseteq V(H)$ consist of all the copy and validation vertices. Due to Lemma 5.15, it holds that $|S| = \delta = \mathcal{O}(k^2)$, while $H - S$ is a collection of γ disconnected choice gadgets. Let $S' = S \cup \{v_1^{i,1}, v_3^{i,1} : \hat{C}_i \text{ in } H\}$, and notice that $H - S'$ is a collection of paths of length at most 5, while $|S'| = \mathcal{O}(k^2)$, thus the statement follows. □

Lemma 5.17. *If G contains a k-clique, then $(H, \mathcal{M}, \ell)$ is a Yes instance of MaxNDP.*

Proof. Consider $s\colon [k] \to [n]$ such that $\mathcal{V} = \{v_{s(i)}^i : i \in [k]\} \subseteq V(G)$ is a k-clique of G. We construct a family of ℓ vertex-disjoint paths as follows.

First, for every instance of $\hat{C}_i$, where $i \in [k]$, and every $j \in [2, n]$, we route a path from α_j^i to β_j^i through the path $\hat{P}_j^i$ if $j \neq s(i)$; if $j = s(i)$, then we use the path $\hat{P}_1^i$ instead. Note that in this step we have created $\gamma \cdot (n-1)$ vertex-disjoint paths connecting terminal pairs, and in every gadget $\hat{C}_i$ the only unused vertices are vertices on the path $\hat{P}_{s(i)}^i$.

Then, consider the adjacency gadget $\hat{A}(i, i, i', i')$, where $i, i' \in [k]$. For every such adjacency gadget, we take the 3-vertex path from $v_2^{i,s(i)}$ to $v_2^{i',s(i')}$ through $m_{i,i'}$; note that the assumption that $\{v_{s(i)}^i, v_{s(i')}^{i'}\} \in E(G)$ ensures that $(v_2^{i,s(i)}, v_2^{i',s(i')}) \in \mathcal{M}$.

Lastly, let $(\mathcal{I}_1, \mathcal{I}_2, t)$ be a copy gadget, where $\mathcal{I}_1$, $\mathcal{I}_2$ are instances of $\hat{C}_i$ and $t \in \{1, 2\}$. For every such copy gadget, we take the 3-vertex path from $v_{t+1}^{i,s(i)}$ of $\mathcal{I}_1$ to $v_1^{i,s(i)}$ of $\mathcal{I}_2$ through the copy vertex g connecting them.

Since the neighborhoods of all the copy and validation vertices are disjoint, this is a valid routing, and it follows that exactly $\gamma \cdot (n-1) + \delta = \ell$ demands are routed. □

Lemma 5.18. *If $(H, \mathcal{M}, \ell)$ is a Yes instance of MaxNDP, then G contains a k-clique.*

Proof. Let $\mathcal{P} = \{P_1, \ldots, P_\ell\}$ be a set of ℓ vertex-disjoint paths connecting terminal pairs of $\mathcal{M}$ in H. Assume without loss of generality that said paths are simple, as well as that the only edges among vertices of the path appear between consecutive vertices. Moreover, let $\mathcal{P}_{\alpha\beta} \subseteq \mathcal{P}$ contain all the paths of $\mathcal{P}$ that route demands of $\mathcal{M}_{\alpha\beta}$. Set S to be the set consisting of all the copy and validation vertices in H, where $|S| = \delta$ due to Lemma 5.15.

Notice that at most δ paths of $\mathcal{P}$ contain vertices of S, consequently at least $\ell - \delta = \gamma \cdot (n-1)$ paths route demands using only vertices of $H - S$. Moreover, $H - S$ is a collection of γ disconnected choice gadgets, each of which can be used to route at most $n-1$ demands. Consequently, it follows that exactly δ paths of $\mathcal{P}$ contain vertices of S, while in each choice gadget, exactly $n-1$ demands are routed using vertices of $H - S$,

thus all the demands of $\mathcal{M}_{\alpha\beta}$ are routed. Let P_i, for $i \in [\delta]$, denote the paths of $\mathcal{P} \setminus \mathcal{P}_{\alpha\beta}$, each of which contains exactly one vertex of S.

Notice that by routing all $n-1$ demands (α_j^i, β_j^i) in a choice gadget $\hat{C}_i$, where $j \in [2, n]$, only the vertices of $\hat{P}_j^i$ remain unused, for some $j \in [n]$. To see why this property holds, notice that the shortest path connecting the terminals of such a demand is of length 5, including said terminals. Therefore, at most 3 vertices of a choice gadget may remain unused. However, if said vertices belonged to more than a single path $\hat{P}_j^i$, then at most $n-2$ demands can be routed, which is a contradiction. Since every such demand can be routed via either $\hat{P}_j^i$ or $\hat{P}_1^i$, there exist $n-1$ paths of $\mathcal{P}_{\alpha\beta}$ that route all demands associated with $\hat{C}_i$ and does not use the vertices of $\hat{P}_j^i$, for some $j \in [n]$.

Next, we show that every P_i, for $i \in [\delta]$, involves exactly 3 vertices, with one being a vertex of S and the other two being its neighbors. Fix one such i, and let s denote the copy or validation vertex of S appearing in P_i, while $\mathcal{I}_1$ and $\mathcal{I}_2$ denote the choice gadgets that have vertices adjacent to s. Set $S' = S \setminus \{s\}$. Then, it holds that P_i routes one demand of $\mathcal{M} \setminus \mathcal{M}_{\alpha\beta}$, using the vertices of the graph $H - S'$. In that case, one endpoint of such a demand can only be a vertex of $\mathcal{I}_1$ and the other of $\mathcal{I}_2$, and due to the construction of $(H, \mathcal{M}, \ell)$, it holds that for every such demand, both its endpoints are neighbors of s.

Next we show that, for $i \in [k]$, for every instance of $\hat{C}_i$ it holds that the only vertices not appearing in paths of $\mathcal{P}_{\alpha\beta}$ are those of $\hat{P}_{s(i)}^{i,j}$, for some function $s\colon [k] \to [n]$. Assume there exists a copy gadget $(\mathcal{I}_1, \mathcal{I}_2, t)$, where $\mathcal{I}_1$ and $\mathcal{I}_2$ are instances of $\hat{C}_i$, $t \in \{1, 2\}$ and s is the corresponding copy vertex. Moreover, let $P \in \mathcal{P}$ such that $s \in P$. Then, there exists $s(i) \in [n]$ such that the vertices of $\hat{P}_{s(i)}^i$ are not used to route demands in $\mathcal{M}_{\alpha\beta}$, for both $\mathcal{I}_1$ and $\mathcal{I}_2$, as otherwise no demand of $\mathcal{M} \setminus \mathcal{M}_{\alpha\beta}$ can be routed in the graph induced by $\mathcal{I}_1$, $\mathcal{I}_2$ and s. Furthermore, notice that for every $s_1, s_2 \in S$, $N(s_1) \neq N(s_2)$, while for every instance $\mathcal{I}' \neq \mathcal{I}$ of $\hat{C}_i$, there is a sequence of copy gadgets $(\mathcal{I}, \mathcal{I}_1, t_1), \ldots, (\mathcal{I}_p, \mathcal{I}', t_{p+1})$, where $\mathcal{I}$ denotes the original choice gadget $\hat{C}_i$ in $\hat{A}(1, k, 1, k)$.

Let $\mathcal{V} = \{v_{s(i)}^i : i \in [k]\} \subseteq V(G)$, where $|\mathcal{V} \cap V_i| = 1$, for all $i \in [k]$. We will prove that $\mathcal{V}$ is a clique. Let $v_{s(i_1)}^{i_1}, v_{s(i_2)}^{i_2} \in \mathcal{V}$. Consider the adjacency gadget $\hat{A}(i_1, i_1, i_2, i_2)$ and let $P_i \in \mathcal{P}$ be the path of $\mathcal{P}$ that contains m_{i_1,i_2}. It holds that P_i is comprised of m_{i_1,i_2}, as well as two of its neighbors, one in $\mathcal{I}_1$ of $\hat{C}_{i_1}$ and one in $\mathcal{I}_2$ of $\hat{C}_{i_2}$. Since the only neighbors of m_{i_1,i_2} that are not used by paths in $\mathcal{P}_{\alpha\beta}$ are $v_2^{i_1,s(i_1)}$ and $v_2^{i_2,s(i_2)}$, we infer that $(v_2^{i_1,s(i_1)}, v_2^{i_2,s(i_2)}) \in \mathcal{M}$ and, consequently, $\{v_{s(i_1)}^{i_1}, v_{s(i_2)}^{i_2}\} \in E^{i_1,i_2}$. Since this holds for any two vertices belonging to $\mathcal{V}$, it follows that G has a k-clique. □

Therefore, in polynomial time, we can construct a graph H, of tree-depth $\mathrm{td} = \mathcal{O}(k)$ and $\mathrm{fvs} = \mathcal{O}(k^2)$ due to Lemma 5.16, as well as a set of pairs $\mathcal{M}$, such that, due to Lemmas 5.17 and 5.18, deciding whether at least ℓ pairs of $\mathcal{M}$ can be routed is equivalent to deciding whether G has a k-clique. □

6. Vertex Integrity and Component Order Connectivity

Publication note. An extended abstract of the results presented in this chapter appeared in the proceedings of the MFCS 2024 [164].

Chapter summary. This chapter investigates Vertex Integrity and Component Order Connectivity through the lens of structural parameterization. Our main result is an improved W[1]-hardness result for much more restrictive parameters. Towards this result, we consider a restricted variant of the well-studied Unary Bin Packing problem which also remains W[1]-hard parameterized by the number of bins. Using this, along with pseudopolynomial-time algorithms for Subset Sum, we are able to significantly strengthen previous W[1]-hardness results. Apart from this, we present several other complementary (positive and negative) results that shed light on the structural parameterized complexity of both problems.

The *vertex integrity* of a graph is a vulnerability measure indicating how easy it is to break down the graph into small pieces. More precisely, the vertex integrity $\mathrm{vi}(G)$ of a graph G is defined as $\mathrm{vi}(G) = \min_{S \subseteq V(G)}\{|S| + \max_{D \in \mathtt{cc}(G-S)} |D|\}$, that is, to calculate the vertex integrity of a graph we must find a separator that minimizes the size of the separator itself plus the size of the largest remaining connected component. Intuitively, a graph has low vertex integrity not only when it contains a small separator, but more strongly when it contains a small separator such that its removal leaves a collection of small connected components.

Vertex integrity was first introduced more than thirty years ago by Barefoot et al. [17], but has recently received particular attention from the parameterized complexity community since it can be considered as a very natural structural parameter: when a graph has vertex integrity k, large classes of NP-hard problems admit FPT algorithms with running times of the form $f(k)n^{\mathcal{O}(1)}$ [206]. Note that vertex integrity has a clear relationship with other, well-known structural parameters (see also Figure 6.1): it is more restrictive than tree-depth, pathwidth, and treewidth (all these parameters are upper-bounded by vertex integrity) but more general than vertex cover (a graph of vertex cover k has vertex integrity at most $k + 1$). "Price of generality" questions, where one seeks to discover for a given problem the most general parameter for which an FPT algorithm is possible, are a central topic in structural parameterized complexity, and vertex integrity therefore plays a role as a natural stepping stone in the hierarchy of standard parameters [28, 141, 142, 148, 150, 213].

The investigation of the parameterized complexity aspects of vertex integrity is, therefore, an active field of research, but it is important to remember that a prerequisite for any such parameter to be useful is that it should be tractable to calculate the parameter

itself (before we try to use it to solve other problems). Since, unsurprisingly, computing the vertex integrity exactly is NP-complete [80], in this paper we focus on this problem from the point of view of parameterized complexity. We consider both the unweighted, as well as a natural weighted variant of the problem. Formally, we want to solve the following:

Unweighted (Weighted) Vertex Integrity

Instance: A graph G (with binary vertex weights $w\colon V(G) \to \mathbb{Z}^+$), an integer k.

Goal: Determine whether $\mathrm{vi}(G) \leq k$ ($\mathrm{wvi}(G) \leq k$).

The point of view we adopt is that of structural parameterized complexity, where vertex integrity is the target problem we are trying to solve, and not necessarily the parameter. Instead, we parameterize by standard structural width measures, such as variations of treewidth. The questions we would like to address are of several forms:

1. For which structural parameters is it FPT to compute the vertex integrity?
2. For which such parameters is it possible to obtain an FPT algorithm with single-exponential complexity?
3. For which parameters can the weighted version of the problem be handled as well as the unweighted version?

To put these questions in context, we recall some facts from the state of the art. When the parameter k is the vertex integrity itself, Fellows and Stueckle show an $\mathcal{O}(k^{3k}n)$-time algorithm for Unweighted Vertex Integrity [121], and later Drange et al. proposed an $\mathcal{O}(k^{k+1}n)$-time algorithm even for Weighted Vertex Integrity [104], so this problem is FPT. More recently, Gima et al. [149] took up the study of vertex integrity in the same structurally parameterized spirit as the one we adopt here and presented numerous results which already give some answers to the questions we posed above. In particular, for the first question they showed that Unweighted Vertex Integrity is W[1]-hard by pathwidth (and hence by treewidth); for the second question they showed that the problem admits a single-exponential algorithm for parameter modular-width; and for the third question they showed that the problem is (weakly) NP-hard on sub-divided stars, which rules out FPT algorithms for most structural parameters.

Our results. Although the results of [149] are rather comprehensive, they leave open several important questions about the complexity of vertex integrity. In this paper we resolve the questions explicitly left open by [149] and go on to present several other results that further clarify the picture for vertex integrity. In particular, our results are as follows (see also Figure 6.1):

The first question we tackle is an explicit open problem from [149]: is Unweighted Vertex Integrity FPT parameterized by tree-depth? This is a very natural question, because tree-depth is the most well-known parameter that sits between pathwidth,

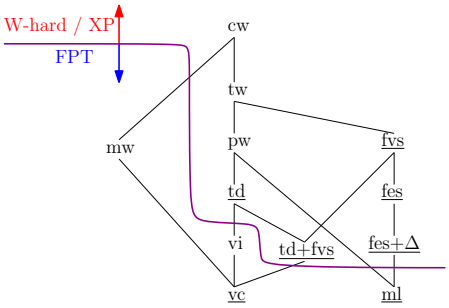


Figure 6.1.: The parameterized complexity of Unweighted Vertex Integrity, with the underlined parameters indicating our results. A connection between two parameters implies that the one above generalizes the one below; that is, the one above is upper-bounded by a function of the one below. All of our FPT algorithms have single-exponential parametric dependence, while the ones for vc and mw extend to the weighted case as well.

where the problem is W[1]-hard by [149], and vertex integrity itself, where the problem is FPT. We resolve this question via a reduction from Bounded Degree Vertex Deletion, showing that Unweighted Vertex Integrity is W[1]-hard for tree-depth (Theorem 6.6).

A second question left open by [149] is the complexity of Unweighted Vertex Integrity for parameter feedback vertex set. Taking a closer look at our reduction from Bounded Degree Vertex Deletion, which is known to be W[1]-hard for this parameter, we observe that it also settles this question, showing that Unweighted Vertex Integrity is also hard. However, in this case we are motivated to dig a little deeper and consider a parameter, feedback edge set, which is a natural restriction of feedback vertex set and typically makes most problems FPT. Our second result is to show that Unweighted Vertex Integrity is in fact W[1]-hard even when parameterized by feedback edge set and the maximum degree of the input graph (Theorem 6.16). We achieve this via a reduction from Unary Bin Packing parameterized by the number of bins, which is W[1]-hard [179]. An aspect of our reduction which may be of independent interest is that we use a variant of Unary Bin Packing where we are given a choice of only two possible bins per item (we observe that the reduction of [179] applies to this variant).

We complement these mostly negative results with a fixed-parameter tractability result for a more restrictive parameter: we show that Unweighted Vertex Integrity is FPT by max-leaf number (Theorem 6.21) indeed by a single-exponential FPT algorithm. Note that when a graph has bounded max-leaf number, then it has bounded degree and bounded feedback edge set number, therefore this parameterization is a special case of the one considered in Theorem 6.16. Hence, this positive result closely complements the problem's hardness in the more general case.

Moving on, we consider the parameterization by modular-width, and take a second look at the $2^{\mathcal{O}(\mathrm{mw})}n^{\mathcal{O}(1)}$ algorithm provided by [149], which is able to handle the weighted

case of the problem, but only for polynomially-bounded weights. Resolving another open problem posed by [149], we show how to extend their algorithm to handle the general case of weights encoded in binary (Theorem 6.24).

Next, we ask the question of whether a single-exponential FPT algorithm is possible for parameters other than max-leaf and modular-width. We answer this affirmatively for vertex cover, even in the weighted case (Theorem 6.28), obtaining a faster and simpler algorithm for the unweighted case (Theorem 6.25).

Finally, we consider the question of whether approximation can be used in order to overcome the W[1]-hardness of Unweighted Vertex Integrity for the parameterization by treewidth. To this end, by making use of a rounding technique introduced by Lampis [207], in Theorem 6.30 we obtain an *efficient FPT approximation scheme* that for any $\varepsilon > 0$ returns a $(1+\varepsilon)$-approximate solution in time $(\mathrm{tw}/\varepsilon)^{\mathcal{O}(\mathrm{tw})} n^{\mathcal{O}(1)}$.

Related work. The concept of vertex integrity is natural enough that it has appeared in many slight variations under different names in the literature. We mention in particular, the fracture number [108], which is the minimum k such that it is possible to delete k vertices from a graph so that all remaining components have size at most k, and the starwidth [278], which is the minimum width of a tree decomposition that is a star. Both of these are easily seen to be at most a constant factor away from vertex integrity. Similarly, the safe set number [23, 136] seeks a separator such that every component of the separator is only connected to smaller components. These concepts are so natural that sometimes they are used as parameters without an explicit name, for example [36] uses the parameter "size of a deletion set to a collection of components of bounded size". As observed by [149], despite these similarities, sometimes computing these parameters can have different complexity, especially when weights are allowed. Another closely related computational problem, that we also use, is the Component Order Connectivity problem [104], where we are given explicit distinct bounds on the size of the separator sought and the allowed size of the remaining components.

6.1. Preliminaries

Given a weight function $w\colon V(G) \to \mathbb{Z}^+$, $w(S)$ denotes the sum of the weights of the vertices of S, that is, $w(S) = \sum_{s\in S} w(s)$.

For a vertex-weighted graph G with $w\colon V(G) \to \mathbb{Z}^+$, we define its *weighted vertex integrity*, denoted by $\mathrm{wvi}(G)$, as

$$\mathrm{wvi}(G) = \min_{S\subseteq V(G)} \left\{ w(S) + \max_{D\in \mathtt{cc}(G-S)} w(D) \right\},$$

where $\mathtt{cc}(G-S)$ is the set of connected components of $G-S$. A set S such that $w(S) + \max_{D\in\mathtt{cc}(G-S)} w(D) \leq k$ is called a $\mathrm{wvi}(k)$-set. The *vertex integrity* of a graph G, denoted by $\mathrm{vi}(G)$, is defined in an analogous way, by setting $w(v) = 1$ for all $v \in V(G)$. In that case, $S \subseteq V(G)$ is a $\mathrm{vi}(k)$-set if $|S| + \max_{D\in\mathtt{cc}(G-S)} |D| \leq k$.

A vertex $v \in S$ is called *redundant* if at most one connected component of $G - S$ contains neighbors of v. A set $S \subseteq V(G)$ is *irredundant* if S contains no redundant vertex. Notice that it suffices to only search for irredundant wvi(k)-sets when solving Vertex Integrity, since if $v \in S$ is redundant and S is a wvi(k)-set, that is the case for set $S \setminus \{v\}$ as well.

Theorem 6.1 ([104, 149])**.** *A graph with a* wvi(k)*-set has an irredundant* wvi(k)*-set.*

Component Order Connectivity. Apart from Vertex Integrity, we will also consider another closely related problem, and we will make use of the following known reduction.

Component Order Connectivity

Instance: A graph G as well as integers ℓ and p.
Goal: Determine whether there exists $S \subseteq V(G)$ such that $|S| \leq p$ and all components of $G - S$ have size at most ℓ.

Theorem 6.2 ([149, Lemma 4.3])**.** *There is a polynomial-time reduction from Component Order Connectivity to Unweighted Vertex Integrity that only attaches leaves to the vertices and introduces new stars.*

6.2. Tree-depth

In this section we show that both Component Order Connectivity and Unweighted Vertex Integrity are W[1]-hard parameterized by tree-depth plus feedback vertex set number, with the latter result resolving a question of [149]. Furthermore, we show that neither problem can be solved in time $f(\mathrm{td})n^{o(\mathrm{td})}$ under the ETH; it is easy to see that both problems admit standard DP algorithms of running time $n^{\mathcal{O}(\mathrm{tw})}$, where tw denotes the treewidth of the input graph, rendering them optimal modulo ETH, even for the parameterization by tree-depth.

Theorem 6.3. *Component Order Connectivity is W[1]-hard parameterized by* $\mathrm{td} + \mathrm{fvs}$*. Moreover, it cannot be solved in time* $f(\mathrm{td})n^{o(\mathrm{td})}$ *under the ETH.*

Proof. We give a parameterized reduction from Bounded Degree Vertex Deletion, which is W[1]-hard by tree-depth plus feedback vertex set number [145] and cannot be solved in time $f(\mathrm{td})n^{o(\mathrm{td})}$ under the ETH (cf. Theorem 3.32). In Bounded Degree Vertex Deletion we are given a graph G and two integers k and d, and we are asked to determine whether there exists $S \subseteq V(G)$ of size $|S| \leq k$ such that the maximum degree of $G - S$ is at most d.

Let (G, k, d) be an instance of Bounded Degree Vertex Deletion, where $n = |V(G)|$ and $m = |E(G)|$. In the following we construct an equivalent instance (G', ℓ, p) of Component Order Connectivity, where $\ell = d + 1$ and $p = k + m$. We construct G' from G as follows: We subdivide every edge $e = \{u, v\} \in E(G)$ three times, thus replacing it with a path on vertices u, u_v, y_e, v_u, v, where $T_e = \{u_v, y_e, v_u\}$. Next, we

attach $d-1$ leaves to y_e (see Figure 6.2). This concludes the construction of G'. Notice that the subdivision of the edges three times and the attachment of pendant vertices does not change the feedback vertex set number, while the tree-depth is only increased by an additive constant. Thus, it holds that $\mathrm{fvs}(G') = \mathrm{fvs}(G)$ and $\mathrm{td}(G') = \mathrm{td}(G) + \mathcal{O}(1)$.

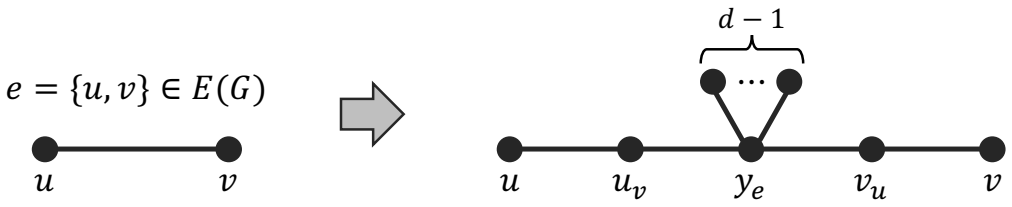


Figure 6.2.: Edge gadget for edge $e = \{u, v\} \in E(G)$.

▷ Claim 6.4. If (G, k, d) is a yes-instance of Bounded Degree Vertex Deletion, then (G', ℓ, p) is a yes-instance of Component Order Connectivity.

Proof of the claim. Let $S \subseteq V(G)$ of size at most k such that the maximum degree of $G - S$ is at most d. We will construct a set $S' \subseteq V(G')$ such that $|S'| \leq p$ and every connected component of $G' - S'$ has size at most ℓ. Initially set $S' = S$. Then, add one vertex to S' per edge $e = \{u, v\} \in E(G)$ as follows. If $u, v \in S$ or $u, v \notin S$, we add y_e to S'. Otherwise, if $u \in S$ and $v \notin S$, we add v_u to S'; symmetrically, if $u \notin S$ and $v \in S$, we add u_v instead. Notice that $|S'| = |S| + m \leq k + m = p$, therefore it suffices to show that the size of each connected component of $G' - S'$ is at most $\ell = d + 1$.

Consider a connected component D of $G' - S'$. Assume that D does not contain any vertices of $V \setminus S$. If D is a leaf it holds that $|D| \leq d+1$. Alternatively, D is a subgraph of the graph induced by u_v (or v_u), y_e, and its attached leaves, for some $e = \{u, v\} \in E(G)$, in which case $|D| \leq d + 1$. Now assume that D contains $u \in V \setminus S$. Notice that u is the only vertex of $V \setminus S$ present in D, since $S' \cap T_e \neq \varnothing$ for all $e \in E(G)$. Moreover, let $N(u) \setminus S = \{u_i : i \in [q]\}$ denote its neighbors in $G - S$, where $q \leq d$ since the maximum degree of $G - S$ is at most d. In that case, it follows that D consists of u, as well as the vertices u_{u_i} for all $i \in [q]$. Consequently, $|D| = q + 1 \leq d + 1$. ◁

▷ Claim 6.5. If (G', ℓ, p) is a yes-instance of Component Order Connectivity, then (G, k, d) is a yes-instance of Bounded Degree Vertex Deletion.

Proof of the claim. Let $S' \subseteq V(G')$ such that $|S'| \leq p = k + m$ and $|D| \leq \ell = d + 1$, for all connected components $D \in \mathtt{cc}(G' - S')$. Assume that S' does not contain any leaves; if it does, substitute them with their single neighbor. Moreover, $S' \cap T_e \neq \varnothing$ for all $e \in E(G)$, since otherwise $G' - S'$ has a component of size at least $d + 2 > \ell$, which is a contradiction. Assume without loss of generality that $|S' \cap T_e| = 1$, for all $e = \{u, v\} \in E(G)$; if that is not the case, there is always a vertex of $\{u_v, v_u\}$, say u_v, such that $u_v \in S'$ and $S' \cap \{y_e, v_u\} \neq \varnothing$, in which case one may consider the deletion set $(S' \cup \{u\}) \setminus \{u_v\}$ instead (the argument is symmetric in case $v_u \in S'$).

Let $S = S' \cap V$, where $|S| \leq k$. We will prove that $G - S$ has maximum degree at most d. Let D_u denote the connected component of $G' - S'$ that contains $u \in V \setminus S'$; in fact this is the only vertex of $V \setminus S'$ present in D_u, since $S' \cap T_e \neq \varnothing$ for all $e \in E(G)$. Notice that for all $e = \{u, v\} \in E(G)$ where $u, v \notin S'$, it holds that $y_e \in S'$: if that were not the case, then either D_u or D_v contains at least $d + 2 > \ell$ vertices, due to $\{u, u_v, y_e\}$ or $\{v, v_u, y_e\}$ and the leaves of y_e respectively. For $u \in V \setminus S$, let $N(u) \setminus S = \{u_i : i \in [q]\}$, for some integer q, denote its neighbors in $G - S$, where $e_i = \{u, u_i\} \in E(G)$ for $i \in [q]$. It suffices to show that $q \leq d$. Assume that this is not the case, i.e., $q > d$. Then, since $S' \cap T_{e_i} = \{y_{e_i}\}$ for $i \in [q]$, it follows that D_u contains vertices u and u_{u_i}, therefore $|D_u| \geq q + 1 > d + 1 = \ell$, which is a contradiction. Consequently, $|N(u) \setminus S| \leq d$ for all $u \in V \setminus S$, i.e., $G - S$ has maximum degree d. ◁

This completes the proof. □

Theorem 6.6. *Unweighted Vertex Integrity is W[1]-hard parameterized by* td+fvs. *Moreover, it cannot be solved in time* $f(\mathrm{td})n^{o(\mathrm{td})}$ *under the ETH.*

Proof. The statement follows by Theorems 6.2 and 6.3, as the reduction of Theorem 6.2 increases the tree-depth by 1 and leaves the feedback vertex set number unchanged. □

6.3. Feedback Edge Set plus Maximum Degree

In this section we prove that Component Order Connectivity is W[1]-hard parameterized by $p + \mathrm{fes} + \Delta + \mathrm{pw}$, while Vertex Integrity is W[1]-hard parameterized by $\mathrm{fes} + \Delta + \mathrm{pw}$. Since our reduction is significantly more involved than the one of Section 6.2, we proceed in several steps. We start from an instance of Unary Bin Packing where the parameter is the number of bins and consider a variant where we are also supplied in the input, for each item, a choice of two possible bins to place it. We first observe that the reduction of [179] shows that this variant is also W[1]-hard. We then reduce this to a *semi-weighted* version of Component Order Connectivity, where placing a vertex in the separator always costs 1, but vertices have weights which they contribute to their components if they are not part of the separator. Subsequently, we show how to remove the weights and the prescription on the separator size to obtain hardness for Component Order Connectivity and Vertex Integrity.

6.3.1. Preliminary Tools

Unary Bin Packing. Given a multiset $A = \{a_1, \ldots, a_n\}$ of integers in unary (i.e., $a_i = \mathcal{O}(n^c)$ for some constant c), as well as $k \in \mathbb{Z}^+$, Unary Bin Packing asks whether we can partition A into k multisets $\mathcal{A}_1, \ldots, \mathcal{A}_k$, such that $\Sigma(\mathcal{A}_i) = \Sigma(A)/k$, for all $i \in [k]$. This problem is well-known to be W[1]-hard parameterized by the number of bins k [179]. We formally define a restricted version where every item is allowed to choose between *exactly two* bins, and by delving deeper into the proof of [179] we observe that an analogous hardness result follows.

Restricted Unary Bin Packing

Instance: A multiset $A = \{a_1, \ldots, a_n\}$ of integers in unary, $k \in \mathbb{Z}^+$, as well as a function $f\colon A \to \binom{[k]}{2}$.

Goal: Determine whether there is a partition of A into multisets $\mathcal{A}_1, \ldots, \mathcal{A}_k$, such that for all $i \in [k]$ it holds that (i) $\Sigma(\mathcal{A}_i) = \Sigma(A)/k$, and (ii) $\forall a \in \mathcal{A}_i$, $i \in f(a)$.

Theorem 6.7. *Restricted Unary Bin Packing is W[1]-hard parameterized by the number of bins.*

Proof. The W[1]-hardness of Unary Bin Packing parameterized by the number of bins is shown via an intermediate problem, called 10-Unary Vector Bin Packing [179]. In said problem, we are given n items $\mathcal{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$, where every item is a 10-dimensional vector belonging to $\mathbb{N}^{10}$ encoded in unary. Additionally, we are given k "bin" vectors $\mathcal{B} = \{\mathbf{B}_1, \ldots, \mathbf{B}_k\}$, and the question is whether we can partition the items into k sets $J_1, \ldots, J_k$ such that $\sum_{\mathbf{s} \in J_i} \mathbf{s} \leq \mathbf{B}_i$, for all $i \in [k]$.

This problem is known to be W[1]-hard parameterized by the number of bins, via an fpt-reduction from Subgraph Isomorphism parameterized by the number of edges of the sought subgraph. Fix $1 \leq i < j \leq k$ and notice that for the intended solution of said reduction ([179, Lemma 10]) it holds that:

- every item $\mathbf{s}_{i,j}(e)$ is either placed in bin $\mathbf{P}_{i,j}$ or bin $\mathbf{R}$,
- every item $\mathbf{t}_{i,i}(v)$ is either placed in bin $\mathbf{Q}_i$ or bin $\mathbf{R}$,
- every item $\mathbf{t}_{i,j}(v)$ and $\mathbf{t}_{j,i}(v)$ is either placed in bin $\mathbf{P}_{i,j}$ or bin $\mathbf{Q}_i$.

Consequently, the hardness result holds in the case when every item is allowed to be placed in one amongst two specified bins, which we call the 10-Unary Restricted Vector Bin Packing problem, where we are additionally given a function f mapping items to their pair of allowed bins.

As a second step, the authors reduce 10-Unary Vector Bin Packing to Unary Bin Packing ([179, Lemma 6]). Assume that $(\mathcal{S}, \mathcal{B}, f)$ denotes the initial instance of 10-Unary Restricted Vector Bin Packing, where $f(\mathbf{s}) \in \binom{\mathcal{B}}{2}$ denotes the bins an item $\mathbf{s} \in \mathcal{S}$ may be placed into. To this end, the authors introduce a bin B_i per bin $\mathbf{B}_i$, an item s_i per item $\mathbf{s}_i$, as well as k additional items $t_1, \ldots, t_k$, which are used to encode the capacity of bins $\mathbf{B}_i$ and are sufficiently large to guarantee that no two of them are placed in the same bin. It suffices to slightly modify said reduction, in order to show hardness for Restricted Unary Bin Packing. In particular, we introduce an additional copy s'_i per item s_i, an additional copy t'_i per item t_i, as well as an additional copy B'_i per bin B_i. Next, if $f(\mathbf{s}) = \{\mathbf{B}_i, \mathbf{B}_j\}$, we set $f'(s) = \{B_i, B_j\}$ and $f'(s') = \{B'_i, B'_j\}$. Moreover, we set $f'(t_i) = f'(t'_i) = \{B_i, B'_i\}$. Since all items t_i and t'_i are placed in distinct bins, the correctness follows as in the proof of [179]. □

Semi-weighted problems. In this section we study semi-weighted versions of Component Order Connectivity and Vertex Integrity, which we first formally define. Then, we prove that the first can be reduced to the latter, while retaining the size of the minimum feedback edge set and only increasing the maximum degree and the pathwidth by at most 1.

Semi-Weighted Component Order Connectivity

Instance: A vertex-weighted graph $G = (V, E, w)$, as well as integers $\ell, p \in \mathbb{Z}^+$.
Goal: Determine whether there exists $S \subseteq V$ of size $|S| \leq p$ such that $w(D) \leq \ell$ for all $D \in \mathtt{cc}(G - S)$.

Semi-Weighted Vertex Integrity

Instance: A vertex-weighted graph $G = (V, E, w)$, as well as an integer $\ell \in \mathbb{Z}^+$.
Goal: Determine whether there exists $S \subseteq V$ such that $|S| + w(D) \leq \ell$ for all $D \in \mathtt{cc}(G - S)$.

Theorem 6.8. *Semi-Weighted Component Order Connectivity parameterized by* $\mathrm{fes} + \Delta + \mathrm{pw}$ *is fpt-reducible to Semi-Weighted Vertex Integrity parameterized by* $\mathrm{fes} + \Delta + \mathrm{pw}$.

Proof. We will closely follow the proof of [149, Lemma 4.3]. Let (G, ℓ, p) be an instance of Semi-Weighted Component Order Connectivity, where w denotes the weight function of G. Assume without loss of generality that for every vertex $v \in V(G)$ it holds that $w(v) \leq \ell$, since otherwise it necessarily belongs to the deletion set. We will construct an equivalent instance (G', k) of Semi-Weighted Vertex Integrity, where $k = \ell p + \ell + p$ and w' denotes the weight function of G'. Construct G' in the following way: Make a copy of G, where $w'(v) = w(v)$, for all $v \in V(G)$. Then, attach to every vertex v a leaf l_v, and set $w'(l_v) = p \cdot w(v)$. Finally, introduce an independent set $I = \{v_i : i \in [k+1]\}$, with every vertex of which having weight $w'(v_i) = k - p = \ell p + \ell$. This concludes the construction of the instance. Notice that we constructed G' from G by only adding one leaf per vertex and vertices of degree 0, therefore $\mathrm{fes}(G') = \mathrm{fes}(G)$, $\Delta(G') \leq \Delta(G) + 1$, and $\mathrm{pw}(G') \leq \mathrm{pw}(G) + 1$.

For the forward direction, assume there exists $S \subseteq V(G)$ such that $|S| \leq p$ and $w(D) \leq \ell$, for all connected components D of $G - S$. It suffices to prove that $w(D') \leq k - p = \ell p + \ell$, for all connected components D' of $G' - S$. If $D' \cap V(G) = \varnothing$, then D' contains (i) either a single vertex l_v of weight $w'(l_v) = p \cdot w(v) \leq p\ell$, (ii) or a single vertex v_i of weight $w'(v_i) = \ell p + \ell$, and the statement holds. Alternatively, $D' \cap V(G)$ induces a connected component D of $G - S$, therefore $w(D') \leq w(D) + p \cdot w(D) = \ell + \ell p$.

For the converse direction, assume there exists $S' \subseteq V(G')$ such that $|S'| + w'(D') \leq k$, for all connected components D' of $G' - S'$. Assume without loss of generality that S' does not contain any vertex of degree 1; if that is the case, substitute it with its single neighbor

and this set remains a valid solution. Notice that $|I| = k+1 > |S'|$, and let $v_i \in I$ belong to $G' - S'$. In that case, it follows that $\max_{D' \in \mathtt{cc}(G'-S')} w'(D') \geq w'(v_i) = k - p$, where $\mathtt{cc}(G' - S')$ denotes the connected components of $G' - S'$. Consequently, it follows that $|S'| \leq p$. Let $S = S' \cap V(G)$ and D be an arbitrary connected component of $G - S$. Since $|S| \leq |S'| \leq p$, it suffices to prove that $w(D) \leq \ell$. Let D' be the connected component of $G' - S'$ such that $D \subseteq D'$. Since S' does not contain vertices of degree 1, it holds that for each vertex of D, D' contains the corresponding leaf attached to it, and thus, $w(D') = w(D) + p \cdot w(D) = (p+1)w(D)$. Since $w(D') \leq k = \ell p + \ell + p < (p+1)(\ell+1)$, it follows that $w(D) < \ell + 1$. □

6.3.2. Hardness Result

Using the results of Section 6.3.1, we proceed to proving the main theorem of this section. To this end, we present a reduction from Restricted Unary Bin Packing to an instance (G, ℓ, p) of Semi-Weighted Component Order Connectivity such that $p + \mathrm{fes}(G) + \Delta(G) + \mathrm{pw}(G) \leq f(k)$ for some function f, where k denotes the number of bins of the Restricted Unary Bin Packing instance.

We first provide a sketch of our reduction. For every bin of the Restricted Unary Bin Packing instance, we introduce a clique of $\mathcal{O}(k)$ heavy vertices and then connect any pair of such cliques via two paths. The weights are set in such a way that an optimal solution will only delete vertices from said paths. In order to construct a path for a pair of bins, we compute the set of all subset sums of the items that can be placed in these two bins, and introduce a vertex of medium weight per such subset sum. Moreover, every such vertex corresponding to subset sum s is preceded by exactly s vertices of weight 1. An optimal solution will cut the path in such a way that the number of vertices of weight 1 will be partitioned between the two bins, encoding the subset sum of the elements that are placed on each bin. The second path that we introduce has balancing purposes, allowing us to exactly count the number of vertices of medium weight that every connected component will end up with.

Theorem 6.9. *Semi-Weighted Component Order Connectivity is W[1]-hard parameterized by* $p + \mathrm{fes} + \Delta + \mathrm{pw}$.

Proof. Let (A, k, f) be an instance of Restricted Unary Bin Packing, where $A = \{a_1, \ldots, a_n\}$ denotes the multiset of items given in unary, k is the number of bins, and $f \colon A \to \binom{[k]}{2}$ dictates the pair of bins an item may be placed into. In the following, consider $B = \Sigma(A)/k$, as well as $M = kB + 1$ and $L = 8k^2BM$. Notice that $M > kB$, while $L/(4k) \in \mathbb{N}$ and $L/(4k) > (k-1) \cdot 2BM + B$. We will reduce (A, k, f) to an equivalent instance (G, ℓ, p) of Semi-Weighted Component Order Connectivity, where w denotes the weight function of G, $\ell = L + (k-1) \cdot 2BM + B$ denotes the maximum component weight, and $p = 2\binom{k}{2}$ denotes the size of the deletion set.

For every $i \in [k]$, we introduce a clique on vertex set $\hat{C}_i$, which is comprised of $4k$ vertices, each of weight $L/(4k)$. Fix i and j such that $1 \leq i < j \leq k$, and let $H_{i,j} = \{a \in A : f(a) = \{i, j\}\}$ denote the multiset of of all items of A which can be placed either on bin i or bin j, where $\Sigma(H_{i,j}) \leq 2B$. Let $\mathcal{S}(H_{i,j}) = \{\Sigma(H) : H \subseteq H_{i,j}\}$

denote the set[1] of all subset sums of $H_{i,j}$, and notice that since every element of $H_{i,j}$ is in unary, $\mathcal{S}(H_{i,j})$ can be computed in polynomial time using e.g. Bellman's classical DP algorithm [22].

Next, we will construct two paths connecting the vertices of $\hat{C}_i$ and $\hat{C}_j$. First, introduce vertex set $\hat{U}_{i,j} = \{v_q^{i,j} : q \in [0, 4B - |\mathcal{S}(H_{i,j})| + 1]\}$, where $w(v) = M$ for all $v \in \hat{U}_{i,j}$. Add edges $(v_q^{i,j}, v_{q+1}^{i,j})$ for all $q \in [0, 4B - |\mathcal{S}(H_{i,j})|]$, as well as $(v_1, v_0^{i,j})$ and $(v_{4B-|\mathcal{S}(H_{i,j})|+1}^{i,j}, v_2)$, for all $v_1 \in \hat{C}_i$ and $v_2 \in \hat{C}_j$. Next, we introduce the vertex set $\hat{D}_{i,j} = \{s_q^{i,j} : q \in [\Sigma(H_{i,j})]\} \cup \{\sigma_q^{i,j} : q \in \mathcal{S}(H_{i,j})\}$, with the s-vertices being of weight 1 and the σ-vertices of weight M. Then, add the following edges:

- $(v_1, \sigma_0^{i,j})$ and $(\sigma_{\Sigma(H_{i,j})}^{i,j}, v_2)$, for all $v_1 \in \hat{C}_i$ and $v_2 \in \hat{C}_j$,
- for $q \in \mathcal{S}(H_{i,j})$, add the edges $(s_q^{i,j}, \sigma_q^{i,j})$ if $q \neq 0$ and $(\sigma_q^{i,j}, s_{q+1}^{i,j})$ if $q \neq \Sigma(H_{i,j})$, and
- for $q \in [\Sigma(H_{i,j})] \setminus \mathcal{S}(H_{i,j})$, add the edge $(s_q^{i,j}, s_{q+1}^{i,j})$.

This concludes the construction of G. See Figure 6.3 for an illustration.

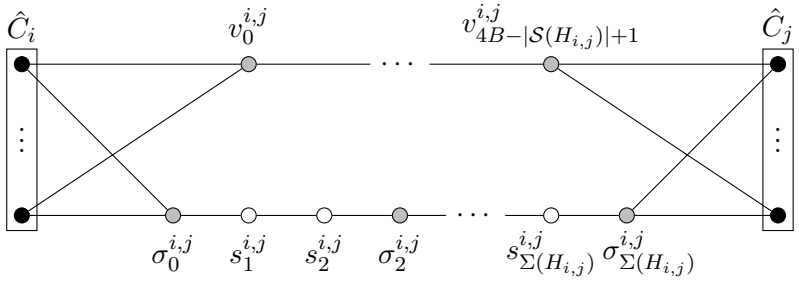


Figure 6.3.: Rectangles denote cliques of size $4k$. Here we assume that $1 \leq i < j \leq k$, $1 \notin \mathcal{S}(H_{i,j})$, and $2 \in \mathcal{S}(H_{i,j})$. The white, gray, or black color indicates weight of 1, M, or $L/(4k)$ respectively.

Lemma 6.10. *If (A, k, f) is a Yes-instance of* Restricted Unary Bin Packing*, then (G, ℓ, p) is a Yes-instance of* Semi-Weighted Component Order Connectivity*.*

Proof. Let $(\mathcal{A}_1, \ldots, \mathcal{A}_k)$ be a partition of A such that for all $i \in [k]$ it holds that (i) $\Sigma(\mathcal{A}_i) = B$, and (ii) $\forall a \in \mathcal{A}_i$, $i \in f(a)$. Fix $1 \leq i < j \leq k$, and let $\mathcal{H}_{i,j} \subseteq H_{i,j}$ such that $\mathcal{A}_i \cap H_{i,j} = \mathcal{H}_{i,j}$, while $\mathcal{A}_j \cap H_{i,j} = H_{i,j} \setminus \mathcal{H}_{i,j}$. Note that for all $i \in [k]$, it holds that

$$\sum_{j \in [i-1]} (\Sigma(H_{j,i}) - \Sigma(\mathcal{H}_{j,i})) + \sum_{j \in [i+1,k]} \Sigma(\mathcal{H}_{i,j}) = B. \tag{6.1}$$

Additionally, let $r_{i,j} = |\mathcal{S}(H_{i,j}) \cap [0, \Sigma(\mathcal{H}_{i,j}) - 1]|$ denote the number of subset sums of $H_{i,j}$ of sum at most $\Sigma(\mathcal{H}_{i,j}) - 1$. Set $S = \{\sigma_{\Sigma(\mathcal{H}_{i,j})}^{i,j}, v_{2B-r_{i,j}}^{i,j} : 1 \leq i < j \leq k\}$. It holds

[1]We stress that $\mathcal{S}(H_{i,j})$ is a set and not a multiset.

that $|S| = 2\binom{k}{2} = p$. In the following, we will prove that every connected component of $G - S$ has weight at most ℓ.

Notice that the vertices of $\hat{C}_i$ and $\hat{C}_j$ are in different connected components of $G - S$, and let $\hat{\mathcal{C}}_i$ denote the connected component of $G - S$ that contains the vertices of $\hat{C}_i$, for all $i \in [k]$. Furthermore, there is no other connected component in $G - S$.

Fix $1 \leq i < j \leq k$. Notice that $S \cap \hat{D}_{i,j} = \{\sigma^{i,j}_{\Sigma(\mathcal{H}_{i,j})}\}$. Consequently, it holds that $\hat{\mathcal{C}}_i$ contains $\Sigma(\mathcal{H}_{i,j})$ vertices of weight 1 belonging to $\hat{D}_{i,j}$, as well as $r_{i,j} + 2B - r_{i,j} = 2B$ vertices of weight M from $\hat{D}_{i,j} \cup \hat{U}_{i,j}$. As for $\hat{\mathcal{C}}_j$, it contains $\Sigma(H_{i,j}) - \Sigma(\mathcal{H}_{i,j})$ vertices of weight 1 belonging to $\hat{D}_{i,j}$, as well as $|\mathcal{S}(H_{i,j})|-1-r_{i,j}+4B-|\mathcal{S}(H_{i,j})|+1-(2B-r_{i,j}) = 2B$ vertices of weight M from $\hat{D}_{i,j} \cup \hat{U}_{i,j}$. For any fixed $i \in [k]$, it follows that

$$\begin{aligned} w(\hat{\mathcal{C}}_i) &= L + \sum_{j \in [i-1]} (\Sigma(H_{i,j}) - \Sigma(\mathcal{H}_{i,j}) + 2BM) + \sum_{j \in [i+1,k]} (\Sigma(\mathcal{H}_{i,j}) + 2BM) \\ &= L + (k-1) \cdot 2BM + B \\ &= \ell, \end{aligned}$$

where the second equality is due to Equation (6.1). This concludes the proof. □

Lemma 6.11. *If (G, ℓ, p) is a Yes-instance of* Semi-Weighted Component Order Connectivity*, then (A, k, f) is a Yes-instance of* Restricted Unary Bin Packing*.*

Proof. Let $S_0 \subseteq V(G)$ such that $|S_0| \leq p = 2\binom{k}{2}$, and for every connected component of $G - S_0$ it holds that the sum of the weights of its vertices is at most $\ell = L+(k-1)\cdot 2BM+B$. The following claim shows that there exists $S \subseteq V(G)$ such that $S \cap \hat{C}_i = \varnothing$, for all $i \in [k]$; in fact S contains exactly 1 vertex per path between cliques.

▷ Claim 6.12. There exists a set $S \subseteq V(G)$ such that $|S| \leq p$, and for every connected component of $G - S$ it holds that the sum of the weights of its vertices is at most ℓ. Additionally, for all $1 \leq i < j \leq k$, it holds that $|S \cap \hat{U}_{i,j}| = |S \cap \hat{D}_{i,j}| = 1$.

Proof of the claim. To prove the statement we will first introduce the following reduction rule.

Rule (†)**.** Let $Z \subseteq V(G)$ be a deletion set such that $|Z \cap \hat{C}_i| \geq 2(k-1)$ for some $i \in [k]$, while every connected component of $G - Z$ has weight at most ℓ. Then, replace Z with $Z' = (Z \setminus \hat{C}_i) \cup N(\hat{C}_i)$, where $N(\hat{C}_i) = \bigcup_{v \in \hat{C}_i} N(v) \setminus \hat{C}_i$.

It is easy to see that $|Z'| \leq |Z|$, since $|N(\hat{C}_i)| = 2(k-1)$. Moreover, it holds that every connected component of $G - Z'$ has weight at most ℓ. To see this, it suffices to consider the connected component that contains the vertices of $\hat{C}_i \setminus Z$. Let $\hat{\mathcal{C}}_i$ denote the set of vertices of the connected component of $G-Z$ where $\hat{C}_i \setminus Z \subseteq \hat{\mathcal{C}}_i$, and $\hat{\mathcal{C}}'_i = \hat{\mathcal{C}}_i \setminus (\hat{C}_i \cup N(\hat{C}_i))$. It holds that $w(\hat{\mathcal{C}}'_i) \leq w(\hat{\mathcal{C}}_i) \leq \ell$, as well as $w(\hat{C}_i) = L \leq \ell$, while in $G - Z'$ the connected component $\hat{\mathcal{C}}_i$ is split into the connected component $\hat{C}_i$ as well as a partition of $\hat{\mathcal{C}}'_i$.

Starting from S_0, let $S \subseteq V(G)$ be the set obtained after applying Rule (†) exhaustively. Note that this process will finish in at most k steps. Observe that since $L/(4k) >$

$(k-1)\cdot 2BM + B$, it follows that $\ell < L/(4k)\cdot(4k+1)$, thus at most $4k$ vertices of weight $L/(4k)$ may be in the same connected component of $G - S$.

Assume there exist $1 \leq i < j \leq k$ such that either $S \cap \hat{U}_{i,j} = \varnothing$ or $S \cap \hat{D}_{i,j} = \varnothing$. In that case, the vertices of $(\hat{C}_i \cup \hat{C}_j) \setminus S$ are in the same connected component of $G - S$, and since every such vertex is of weight $L/(4k)$, it follows that $|S \cap (\hat{C}_i \cup \hat{C}_j)| \geq 4k$. Assume without loss of generality that $0 \leq |S \cap \hat{C}_i| \leq |S \cap \hat{C}_j| \leq 4k$, thus $|S \cap \hat{C}_j| \geq 2k$ follows, which contradicts the exhaustive application of Rule (†). Consequently, for all $1 \leq i < j \leq k$ it holds that $S \cap \hat{D}_{i,j} \neq \varnothing$ as well as $S \cap \hat{U}_{i,j} \neq \varnothing$. Since $|S| \leq 2\binom{k}{2}$, it follows that $|S \cap \hat{D}_{i,j}| = |S \cap \hat{U}_{i,j}| = 1$. ◁

Let $\hat{\mathcal{C}}_i$ denote the connected component of $G - S$ containing the vertices of $\hat{C}_i$. Notice that $\mathsf{cc}(G - S) = \{\hat{\mathcal{C}}_i : i \in [k]\}$.

▷ Claim 6.13. It holds that all vertices of S are of weight M. Furthermore, for all $i \in [k]$ it holds that $w(\hat{\mathcal{C}}_i) = \ell$.

Proof of the claim. For $1 \leq i < j \leq k$, let $d_{i,j} \in S \cap \hat{D}_{i,j}$ denote the single vertex of $\hat{D}_{i,j}$ that belongs to S. Notice that the single vertex of $S \cap \hat{U}_{i,j}$ is by definition of weight M, while $w(d_{i,j}) \in \{1, M\}$.

For the first statement, it suffices to prove that for all $1 \leq i < j \leq k$, $w(d_{i,j}) \geq M$. Notice that $G - S$ has exactly k connected components $\hat{\mathcal{C}}_1, \ldots, \hat{\mathcal{C}}_k$, whose weight sums up to

$$\sum_{i \in [k]} w(\hat{\mathcal{C}}_i) = w(G) - \sum_{1 \leq i < j \leq k} w(d_{i,j}) - \binom{k}{2} \cdot M, \tag{6.2}$$

where the last term is due to the vertices in $S \cap \hat{U}_{i,j}$ which are of weight M, therefore

$$w(G) - \sum_{1 \leq i < j \leq k} w(d_{i,j}) - \binom{k}{2} \cdot M \leq k \cdot \ell. \tag{6.3}$$

In order to compute $w(G)$, notice that

- $w(\hat{C}_i) = L$, for all $i \in [k]$,
- $w(\hat{U}_{i,j}) = (4B - |\mathcal{S}(H_{i,j})| + 2) \cdot M$, for all $1 \leq i < j \leq k$,
- $w(\hat{D}_{i,j}) = \Sigma(H_{i,j}) + |\mathcal{S}(H_{i,j})| \cdot M$, for all $1 \leq i < j \leq k$,

and adding up the last two items gives $w(\hat{U}_{i,j} \cup \hat{D}_{i,j}) = \Sigma(H_{i,j}) + (4B+2) \cdot M$. Consequently, it follows that

$$\begin{aligned} w(G) &= kL + kB + \binom{k}{2} \cdot (4B + 2) \cdot M \\ &= k(L + B + (k-1) \cdot 2BM) + \binom{k}{2} \cdot 2M \\ &= k\ell + \binom{k}{2} \cdot 2M, \end{aligned}$$

which due to Equation (6.3) gives

$$k\ell + \binom{k}{2} \cdot 2M - \sum_{1 \leq i < j \leq k} w(d_{i,j}) - \binom{k}{2} \cdot M \leq k \cdot \ell,$$

thus

$$\binom{k}{2} \cdot M \leq \sum_{1 \leq i < j \leq k} w(d_{i,j}),$$

and since $w(d_{i,j}) \leq M$, it follows that $w(d_{i,j}) = M$, for all $1 \leq i < j \leq k$.

As for the weight of the components $\hat{\mathcal{C}}_i$, due to Equation (6.2) it follows that $w(\hat{\mathcal{C}}_i) = \ell$ for all $i \in [k]$. ◁

Due to Claim 6.13, it follows that $S \cap \hat{D}_{i,j} = \{\sigma_q^{i,j}\}$, for some $q \in \mathcal{S}(H_{i,j})$. Since $w(\hat{\mathcal{C}}_i) = \ell$, while $kB < M$, it follows that $\hat{\mathcal{C}}_i$ contains exactly $(k-1) \cdot 2B$ vertices of weight M, as well as exactly B vertices of weight 1. Let $(\mathcal{A}_1, \ldots, \mathcal{A}_k)$ be a partition of A defined in the following way: for all $1 \leq i < j \leq k$, if $\sigma_q^{i,j} \in S$, then $\mathcal{A}_i \cap H_{i,j} = \mathcal{H}_{i,j}$ and $\mathcal{A}_j \cap H_{i,j} = H_{i,j} \setminus \mathcal{H}_{i,j}$, where $\mathcal{H}_{i,j} \subseteq H_{i,j}$ such that $\Sigma(\mathcal{H}_{i,j}) = q$. Notice that $\Sigma(\mathcal{A}_i)$ is equal to the number of vertices of weight 1 in $\hat{\mathcal{C}}_i$, therefore $\Sigma(\mathcal{A}_i) = B$ follows. □

Lemma 6.14. *It holds that* $\mathrm{fes}(G) = \mathcal{O}(k^3)$, $\Delta(G) = \mathcal{O}(k)$, *and* $\mathrm{pw}(G) = \mathcal{O}(k^2)$.

Proof. Let $F \subseteq E(G)$ contain all edges between vertices of $\hat{C}_i$, for all $i \in [k]$, as well as all edges which are adjacent to the endpoints of the paths $\hat{U}_{i,j}$ and $\hat{D}_{i,j}$. Notice that

$$|F| = k \cdot \binom{4k}{2} + 4\binom{k}{2} \cdot 4k = \mathcal{O}(k^3),$$

while the graph remaining after the deletion of the edges in F is a forest.

For the maximum degree, notice that $|N(v)| = 4k - 1 + 2(k-1) = \mathcal{O}(k)$, for all $v \in \bigcup_{i \in [k]} \hat{C}_i$. As for the vertices of the paths, they are all of degree 2, apart from the endpoints which are neighbors with all the vertices of a single clique, therefore of degree $\mathcal{O}(k)$.

For the pathwidth, notice that the graph obtained from G by deleting all vertices of $\bigcup_{i \in [k]} \hat{C}_i$ is a union of paths. □

Theorem 6.9 now follows due to Lemmas 6.10, 6.11, and 6.14. □

We are now ready to prove the main theorems of this section.

Theorem 6.15. *Component Order Connectivity is W[1]-hard parameterized by* $p + \mathrm{fes} + \Delta + \mathrm{pw}$.

Proof. The statement follows due to Theorem 6.9 and the fact that Semi-Weighted Component Order Connectivity can be reduced to Component Order Connectivity by attaching to every vertex v of weight $w(v)$ a path on $w(v) - 1$ vertices (notice that we can assume that $w(v) \leq \ell$, as otherwise v belongs to the deletion set). This transformation does not increase fes, while Δ and pw may increase by at most a constant. The reduction leaves p unchanged. □

Theorem 6.16. *Unweighted Vertex Integrity is W[1]-hard parameterized by* $\text{fes} + \Delta + \text{pw}$.

Proof. By Theorems 6.8 and 6.9, it follows that Semi-Weighted Vertex Integrity is W[1]-hard parameterized by $\text{fes} + \Delta + \text{pw}$. Moreover, Semi-Weighted Vertex Integrity can be reduced to Vertex Integrity by attaching to every vertex v of weight $w(v)$ a path on $w(v) - 1$ vertices (notice that we can assume that $w(v) \leq \ell$, as otherwise v belongs to the deletion set). This transformation does not increase fes, while Δ and pw may increase by at most a constant. □

6.4. Max-Leaf Number

In this section, we consider the parameterization by max-leaf number. For a connected graph G we denote by $\text{ml}(G)$ the maximum number of leaves of any spanning tree of G. This is a well-studied but very restricted parameter [118, 119, 205]. In particular, it is known that if a graph G has $\text{ml}(G) \leq k$, then in fact G is a subdivision of a graph on $\mathcal{O}(k)$ vertices [190]. We are motivated to study this parameter because in a sense it lies close to the intractability boundary established in Section 6.3. Observe that if a graph is a sub-division of a graph on k vertices, then it has maximum degree at most k and feedback edge set at most k^2; however, graphs of small feedback edge set and small degree do not necessarily have small max-leaf number (consider a long path where we attach a leaf to each vertex). Interestingly, the graphs we construct in Section 6.3.2 *do* have small max-leaf number, if we consider semi-weighted instances. However, adding the necessary simple gadgets in order to simulate weights increases the max-leaf number of the graphs of our reduction. It is thus a natural question whether this is necessary. In this section, we show that indeed this is inevitable, as Vertex Integrity and Component Order Connectivity are FPT parameterized by ml.

We start with a high-level overview of our approach. As mentioned, we will rely on the result of Kleitman and West [190] who showed that if a graph $G = (V, E)$ has $\text{ml}(G) \leq k$, then there exists a set X of size $|X| = \mathcal{O}(k)$ such that all vertices of $V \setminus X$ have degree at most 2. Our main tool is Lemma 6.19 which allows us to "rotate" solutions: whenever we have a cycle in our graph, we can, roughly speaking, exchange every vertex of S in the cycle with the next vertex, until we reach a point where our solution removes strictly more vertices of X. We therefore guess the largest intersection of an optimal separator with X, and can now assume that in every remaining cycle, the separator S is not using any vertices. This allows us to simplify the graph in a way that removes all cycles and reduces the case to a tree, which is polynomial-time solvable.

Let us now give more details. We first recall the result of [190].

Lemma 6.17. *In any graph G, the set X of vertices of degree at least 3 has size at most* $|X| \leq 12\text{ml}(G) + 32$.

Proof. The statement is trivially true if the maximum degree of the graph is less than 3, so in the following assume that this is not the case. Kleitman and West actually proved that in a graph G with n vertices and minimum degree 3 or more, $\text{ml}(G) \geq n/4$.

Given a graph G, let X_G and Y_G denote the set of vertices of degree at least 3 and at most 2 respectively, i.e., $V(G) = X_G \cup Y_G$ where $X_G = \{v \in V(G) : \deg_G(v) \geq 3\}$ and $Y_G = \{v \in V(G) : \deg_G(v) \leq 2\}$. Start from an arbitrary (connected) graph G, and greedily contract any edge in G, as long as the number of vertices of degree at least 3 is not reduced. Call the resulting graph G', where $|X_{G'}| \geq |X_G|$. Notice that the contraction of an edge does not increase the max-leaf number, thus $\mathrm{ml}(G') \leq \mathrm{ml}(G)$. In case $Y_{G'} = \varnothing$ the statement immediately follows by the result of Kleitman and West, so in the following assume that $Y_{G'} \neq \varnothing$. Moreover, for all $v \in Y_{G'}$, it holds that $N_{G'}(v) \cap X_{G'} \neq \varnothing$; if that were not the case, then contracting an edge incident to v does not decrease the degree of any vertex in $X_{G'}$, which is a contradiction since in G' any such edge has already been contracted.

▷ Claim 6.18. It holds that $|Y_{G'}| \leq \mathrm{ml}(G')$.

Proof of the claim. It suffices to prove that there exists a spanning tree of G' where every vertex of $Y_{G'}$ is a leaf. Let $v \in Y_{G'}$, and notice that if $\deg_{G'}(v) = 1$, then v is a leaf in any spanning tree of G'. In the following, assume that $N_{G'}(v) = \{u_1, u_2\}$, where $N_{G'}(v) \cap X_{G'} \neq \varnothing$. Moreover, we claim that $\{u_1, u_2\} \in E(G')$. Assume otherwise, and notice that in that case, the contraction of any edge incident on v does not change the degree of any other vertex, which is a contradiction.

First consider the case where $u_1, u_2 \in X_{G'}$. Given a spanning tree of G' that contains both edges $\{v, u_1\}$ and $\{v, u_2\}$, one can obtain another spanning tree that contains edges $\{v, u_1\}$ and $\{u_1, u_2\}$ instead.

For the remaining case, assume without loss of generality that $u_1 \in Y_{G'}$ and $u_2 \in X_{G'}$. Then, given a spanning tree of G' that contains the edges $\{v, u_1\}$ and either $\{v, u_2\}$ or $\{u_1, u_2\}$, one can obtain another spanning tree that contains edges $\{v, u_2\}$ and $\{u_1, u_2\}$ instead.

(a) Case 1. (b) Case 2.

Figure 6.4.: Black vertices belong to $X_{G'}$.

Consequently, there exists a spanning tree of G' where every vertex of $Y_{G'}$ is a leaf, thus $|Y_{G'}| \leq \mathrm{ml}(G')$ follows. ◁

Lastly, we apply the result of Kleitman and West. If $|Y_{G'}| \geq 3$, add at most $|Y_{G'}| \leq \mathrm{ml}(G')$ edges connecting vertices of $Y_{G'}$, so that the resulting graph G'' has minimum degree at least 3. Each such addition increases the max-leaf number by at most 2, therefore it holds that $\mathrm{ml}(G'') \leq \mathrm{ml}(G') + 2\mathrm{ml}(G')$, while $|X_{G''}| = |X_{G'} \cup Y_{G'}| > |X_{G'}|$, and since $|X_{G''}| \leq 4\mathrm{ml}(G'')$ it follows that $|X_G| \leq 12\mathrm{ml}(G)$. It remains to consider the case where $1 \leq |Y_{G'}| \leq 2$. Then, it holds that $|X_{G'}| \geq 3$, since otherwise it follows that either $X_{G'} = \varnothing$ or $Y_{G'} = \varnothing$. Consequently, add at most 2 edges per vertex of $Y_{G'}$,

so that the resulting graph G'' has minimum degree 3, therefore $\mathrm{ml}(G'') \leq \mathrm{ml}(G') + 8$, while $|X_{G''}| = |X_{G'} \cup Y_{G'}| > |X_{G'}|$, and since $|X_{G''}| \leq 4\mathrm{ml}(G'')$ it follows that $|X_G| \leq 4\mathrm{ml}(G) + 32$. □

Our main lemma is now the following:

Lemma 6.19. *Let $G = (V, E)$ be a graph and X be a set of vertices such that all vertices of $V \setminus X$ have degree at most 2 in G. For all positive integers ℓ, p, if there exists a separator S of size at most p such that all components of $G - S$ have size at most ℓ, then there exists such a separator S that also satisfies the following property: for every cycle C of G with $C \cap X \neq \varnothing$ we either have $C \cap S = \varnothing$ or $C \cap X \cap S \neq \varnothing$.*

Proof. We will show that if we have a separator S that does not satisfy the property, then we can obtain an equally good separator that contains strictly more elements of X. Applying this argument exhaustively will yield the lemma.

Fix some separator S of size at most p such that all components of $G - S$ have size at most ℓ. Suppose that we have a cycle C in G of length t, say $C = v_0, v_1, \ldots, v_{t-1}, v_0$ such that C contains some vertices of X and some vertices of S, but no vertex that belongs to both X and S. Suppose that $C \cap S = \{v_{i_0}, v_{i_1}, \ldots, v_{i_{r-1}}\}$, where $r = |C \cap S|$ and $i_0 < i_1 < \ldots < i_{r-1}$. We claim that the set $S' = (S \setminus C) \cup \{v_{i_0+1}, v_{i_1+1}, \ldots, v_{i_{r-1}+1}\}$ (where additions are modulo t) is a separator of the same size as S which also leaves components of size at most ℓ in $G - S'$. Informally, what we are doing is replacing every vertex of $C \cap S$ with the next vertex in the cycle C. It is clear that $|S| = |S'|$. Furthermore, because $C \cap S \cap X = \varnothing$, all vertices of $C \cap S$ have degree 2 in G and one of their neighbors is in S'. Let D_j be the component of $G - S$ that is adjacent to v_{i_j} and $v_{i_{j-1}}$ (where subtraction is done modulo r). We observe that $G - S'$ has the component $D_j \cup \{v_{i_j}\} \setminus \{v_{i_{j-1}+1}\}$, which has the same size as D_j. Here we are using the fact that v_{i_j} has degree 2 and its neighbor v_{i_j+1} is in S'. Hence, this procedure does not increase the size of any component (in fact, if $C \cap S' \cap X \neq \varnothing$, the size of some components, which previously contained vertices of $C \cap S' \cap X$, is reduced, since such components are divided in $G - S'$). If the new separator S' has $C \cap S' \cap X \neq \varnothing$, we have increased the intersection of the separator with X. If not, we repeat this process. Since at each step the intersection between the separator and X cannot decrease, we eventually obtain a separator that satisfies the condition of the lemma. □

We are now ready to state the main results of this section.

Theorem 6.20. *Component Order Connectivity can be solved in time $2^{\mathcal{O}(\mathrm{ml})} n^{\mathcal{O}(1)}$.*

Proof. We are given a graph G and let X be the set of vertices of degree at least 3. By Lemma 6.17, $|X| \leq 12\mathrm{ml}(G) + 32$. Let S be a separator of minimum size such that $G - S$ has only components of size at most ℓ. Among all such separators, select an S such that $S \cap X$ is maximized. Fix this separator S for the analysis of the rest of the algorithm.

Our algorithm as a first step guesses $S \cap X$, removes these vertices from the graph and sets $p := p - |S \cap X|$. In the remainder, we focus on the case where this guess was correct. To simplify the rest of the presentation we assume that some vertices of the

graph may be marked as undeletable, meaning that we are only looking for separators that do not contain such vertices. Initially, the vertices of $X \setminus S$ are undeletable.

We now perform a series of polynomial-time simplification steps. First, if the current graph contains a connected component of size at most ℓ, we remove this component. This is clearly correct. Second, if the graph contains a component where all vertices have degree at most 2, we compute in polynomial time an optimal deletion set for this component, update p appropriately, and remove this component.

We are now in a situation where every cycle C in our current graph must contain a vertex of X. By Lemma 6.19 any such cycle C must have $C \cap S = \varnothing$. We now contract all the vertices of C into a single vertex and attach to this vertex $|C| - 1$ leaves. We mark this new vertex as undeletable, that is, we will only look for separators S that do not contain it. We also place this vertex in X. It is not hard to see that this transformation is correct, since if the optimal solution does not place any vertex of C into S, we can replace C with a single undeletable vertex which contributes $|C|$ to the size of its component.

Performing the above exhaustively results in a graph that contains no cycles. The problem can now easily be solved optimally via dynamic programming in polynomial time. □

Theorem 6.21. *Unweighted Vertex Integrity can be solved in time* $2^{\mathcal{O}(\mathrm{ml})}n^{\mathcal{O}(1)}$.

Proof. In order to prove the statement, we go over all possible values of the size of the deletion set ℓ and solve Component Order Connectivity for each such value. The minimum value of $\ell + p$ over all solutions is the value of the optimal solution for Unweighted Vertex Integrity. Since ℓ can take at most n values, while by Theorem 6.20 each such instance can be solved in time $2^{\mathcal{O}(\mathrm{ml})}n^{\mathcal{O}(1)}$, the statement follows. □

6.5. Modular-width

In this section we revisit an algorithm of [149] establishing that Weighted Vertex Integrity can be solved in time $2^{\mathcal{O}(\mathrm{mw})}n^{\mathcal{O}(1)}$, on graphs of modular-width mw, but only if weights are polynomially bounded in n (or equivalently, if weights are given in unary). It was left as an explicit open problem in [149] whether this algorithm can be extended to the case where weights are given in binary and can therefore be exponential in n. We resolve this problem positively, by showing how the algorithm of [149] can be modified to work also in this case, without a large increase in its complexity.

The high-level idea of the algorithm of [149] is to perform dynamic programming to solve the related Weighted Component Order Connectivity problem. In this problem we are given a target component weight ℓ and a deletion budget p and are asked if it is possible to delete from the graph a set of vertices with total weight at most p so that the maximum weight of any remaining component is at most ℓ. Using this algorithm as a black box, we can then solve Weighted Vertex Integrity by iterating over all possible values of ℓ, between 1 and the target vertex integrity. If vertex weights are polynomially bounded, this requires a polynomial number of iterations, giving the

algorithm of [149]. However, if weights are given in binary, the target vertex integrity could be exponential in n, so in general, it does not appear possible to guess the weight of the heaviest component in an optimal solution.

Our contribution to the algorithm of [149] is to observe that for graphs of modular-width mw the weight of the heaviest component may take at most $2^{\mathcal{O}(\mathrm{mw})}n$ distinct possible values. Hence, for this parameter, guessing the weight of the heaviest component in an optimal solution can be done in FPT time. We can therefore plug in this result to the algorithm of [149] to obtain an algorithm for Weighted Vertex Integrity with binary weights running in time $2^{\mathcal{O}(\mathrm{mw})}n^{\mathcal{O}(1)}$.

Our observation is based on the following lemma.

Lemma 6.22. *Let $G = (V, E)$ be an instance of Weighted Vertex Integrity. There exists an optimal solution using a separator S such that for all connected components D of $G - S$ and modules M of G we have one of the following: (i) $M \cap D = \varnothing$, (ii) $M \subseteq D$, or (iii) $D \subseteq M$.*

Proof. Let S be a separator giving an optimal solution, that is a solution minimizing the sum of the weight of S and the weight of the heaviest component of $G - S$. Our strategy is to show that if S does not satisfy the condition of the lemma, then S contains some vertex z which in the terminology of [149] is redundant, that is, z has neighbors in at most one connected component of $G - S$. As observed in [149], this implies that $S \setminus \{z\}$ is a separator that gives a solution that is at least as good as that given by S, so we can remove z from S. Repeating this exhaustively will produce a separator S that satisfies the conditions of the lemma.

Suppose then that S is a separator such that for some component D of $G - S$ and some module M we have $M \cap D \neq \varnothing$, $D \setminus M \neq \varnothing$, and $M \setminus D \neq \varnothing$. Let $x \in D \cap M$ and $y \in D \setminus M$ such that x, y are adjacent (such x, y must exist, since D is connected). Since M is a module and $y \notin M$, we have that y is adjacent to all of M. Hence, $M \setminus S \subseteq D$ because all vertices of $M \setminus S$ are adjacent to $y \in D$. It follows that each vertex $z \in M \setminus D$ must belong to the separator S. We claim that z is redundant, that is, z only has neighbors in D and in no other connected component of $G - S$. This is not hard to see, since each neighbor of z is either in M (so is adjacent to $y \in D$) or is adjacent to all of M (hence also to x). □

We also recall the algorithmic result of [149].

Theorem 6.23 ([149])**.** *There exists an algorithm that takes as input a vertex-weighted graph $G = (V, E)$ and an integer ℓ and computes the minimum integer p such that there exists a separator S of G of weight at most p such that each component of $G - S$ has weight at most ℓ. The algorithm runs in time $2^{\mathcal{O}(\mathrm{mw})}n^{\mathcal{O}(1)}$, where n is the size of the input.*

Putting Lemma 6.22 and Theorem 6.23 together we obtain the main result of this section.

Theorem 6.24. *There exists an algorithm that solves* Weighted Vertex Integrity *in time* $2^{\mathcal{O}(\mathrm{mw})}n^{\mathcal{O}(1)}$*, where* mw *is the modular-width of the input graph,* n *is the size of the input, and weights are allowed to be written in binary.*

Proof. Fix some optimal separator S satisfying the conditions of Lemma 6.22. Our strategy is to use these conditions to guess the weight of the heaviest component of $G-S$. If this weight is ℓ, then we can simply call the algorithm of Theorem 6.23. More precisely, we will show that by considering $\mathcal{O}(2^{\mathrm{mw}}n)$ distinct values, we are guaranteed to find ℓ. Hence, by executing the algorithm of Theorem 6.23 $\mathcal{O}(2^{\mathrm{mw}}n)$ times and picking the best solution we find the optimal solution.

Consider the modular decomposition tree associated with G, which can be computed in polynomial time [241] and is recursively constructed as follows: if $G=(V,E)$ has at most k vertices, then G has a root labeled V with k children, each corresponding to a vertex of V; while if $|V|>k$, then V can be partitioned into $k'\leq k$ modules $V_1,\ldots,V_{k'}$, so we construct a root labeled V and give it as children the roots of the trees constructed for $G[V_i]$, for $i\in[k']$. Each node of the constructed tree is labeled with a module of G.

Let D be the heaviest component of $G-S$. Find the node of the modular decomposition that is as far from the root as possible and satisfies that the corresponding module M has $D\subseteq M$. (Such a node exists, since $D\subseteq V$.) The module M can be decomposed into $k'\leq k$ modules $M_1,\ldots,M_{k'}$. For each such module we have either $M_i\cap D=\varnothing$ or $M_i\subseteq D$, by Lemma 6.22 and because if we had $D\subseteq M_i$, this would contradict our choice of M. It must therefore be the case that for some $I\subseteq[k']$ we have $D=\bigcup_{i\in I}M_i$.

We now observe that we have $\mathcal{O}(n)$ choices of M (since the modular decomposition has $\mathcal{O}(n)$ nodes), and 2^{mw} choices for I. Hence, there are $\mathcal{O}(2^{\mathrm{mw}}n)$ possible values of the weight of the heaviest component ℓ. □

6.6. Vertex Cover Number

In this section, we design single-exponential algorithms for Vertex Integrity parameterized by vertex cover number. We suppose that a minimum vertex cover C of size vc is given since it can be computed in time $\mathcal{O}(1.2738^{\mathrm{vc}}+\mathrm{vc}n)$ [64]. We start by presenting an algorithm for Unweighted Vertex Integrity, before moving on to the weighted version of the problem.

Theorem 6.25. *Unweighted Vertex Integrity can be solved in time* $5^{\mathrm{vc}}n^{\mathcal{O}(1)}$.

Proof. Let C be a vertex cover of G of size $|C|=\mathrm{vc}$. Without loss of generality, we assume that $k<\mathrm{vc}+1$, since otherwise C is a separator that realizes the vertex integrity at most k. For the analysis, fix an optimal separator S, which by Theorem 6.1 can be assumed to be irredundant. We first guess $S\cap C$, delete its vertices, and decrease k by $|S\cap C|$. In the remainder, assume that $S\cap C=\varnothing$. Let $I=V\setminus C$ be the independent set of the graph. We have to decide for each vertex of I whether to place it in S or not.

We keep track of the vertices of I which have been marked as belonging to S, denoted by I_S, those that have been marked as not belonging to S, denoted by $I_{\bar{S}}$, and those which are undecided, denoted by I_U (initially $I=I_U$). We also keep track of the connected

components of the graph induced by $C \cup I_{\bar{S}}$. Now, as long as there exists an undecided vertex $v \in I_U$ such that all its neighbors in C are in the same connected component, we mark that v is not in S and move it to $I_{\bar{S}}$ (because v would be redundant). Consider then an undecided $v \in I_U$ that has neighbors in C which are in distinct components of the graph induced by $C \cup I_{\bar{S}}$. We consider two cases: $v \in S$ and $v \notin S$. In the first case, we remove v from the graph and decrease k by 1. In the latter, we observe that the number of connected components made up of vertices of $C \cup I_{\bar{S}}$ has decreased. We continue this branching procedure until $k < 0$ (in which case we reject); or all vertices have been decided (in which case we evaluate the solution). The algorithm is correct, assuming that our guess of $S \cap C$ was correct, because for all vertices of I we either know that we have made the correct choice (because S is irredundant) or we consider both possible choices.

For the running time, we define a potential function as the sum of k plus the number of connected components induced by $C \cup I_{\bar{S}}$. Then for both branches, the potential function decreases by at least 1. Moreover, the value of the potential function is bounded by $k - |C \cap S| + |C \setminus S| \leq 2\mathrm{vc} - 2|C \cap S|$. Note that the branching algorithm is applied to the graph after deleting $C \cap S$. Therefore, the running time is bounded by

$$\begin{aligned}
\sum_{S \cap C \subseteq C} 2^{2\mathrm{vc}-2|C\cap S|} n^{\mathcal{O}(1)} &= \sum_{S \subseteq C} 4^{\mathrm{vc}-|C\cap S|} n^{\mathcal{O}(1)} \\
&= \sum_{S \subseteq C} 4^{\mathrm{vc}-|C\cap S|} 1^{|C\cap S|} n^{\mathcal{O}(1)} \\
&= \sum_{i=0}^{\mathrm{vc}} \binom{\mathrm{vc}}{i} 4^{\mathrm{vc}-i} 1^i n^{\mathcal{O}(1)} \\
&= 5^{\mathrm{vc}} n^{\mathcal{O}(1)},
\end{aligned}$$

and the statement follows. □

We now move on to the weighted case of the problem. It is clear that Weighted Vertex Integrity is FPT parameterized by vertex cover, due to Theorem 6.24 and the relation between modular-width and vertex cover. However, this gives a double-exponential dependence on vc, as $\mathrm{mw} \leq 2^{\mathrm{vc}} + \mathrm{vc}$ and there are some graphs for which this is essentially tight. We would like to obtain an algorithm that is as efficient as that of Theorem 6.25. The algorithm of Theorem 6.25, however, cannot be applied to the weighted case because the case of the branching where we place a vertex of the independent set in the separator is not guaranteed to make much progress (the vertex could have very small weight compared to our budget).

Before we proceed, it is worth thinking a bit about how this can be avoided. One way to obtain a faster FPT algorithm would be, rather than guessing only the intersection of the optimal separator S with the vertex cover C, to also guess how the vertices of $C \setminus S$ are partitioned into connected components in the optimal solution. This would immediately imply the decision for all vertices of the independent set: vertices with neighbors in two components must clearly belong to S, while the others cannot belong to

S if S is irredundant. This algorithm would give a complexity of $\mathrm{vc}^{\mathcal{O}(\mathrm{vc})}n^{\mathcal{O}(1)}$, however, because the number of partitions of C is slightly super-exponential.

Let us sketch the high-level idea of how we handle this. Our first step is, similarly to Theorem 6.24, to calculate the weight $w_{\max}$ of the most expensive connected component of the optimal solution. For this, there are at most $2^{\mathrm{vc}} + n$ possibilities, because this component is either a single vertex, or it has a non-empty intersection with C. However, if we fix its intersection with C, then this fixes its intersection with the independent set: the component must contain (by irredundancy) exactly those vertices of the independent set all of whose neighbors in C are contained in the component. Having fixed a value of $w_{\max}$ we simply seek the best separator so that all components have weight at most $w_{\max}$. The reason we perform this guessing step is that this version of this problem is easier to decompose: if we have a disconnected graph, we simply calculate the best separator in each part and take the sum (this is not as clear for the initial version of Vertex Integrity).

Suppose then that we have fixed $w_{\max}$, how do we find the best partition of C into connected components? We apply a win/win argument: if the optimal partition has a connected component that contains many (say, more than $\mathrm{vc}/10$) vertices of C, we simply guess the intersection of the component with C and complete it with vertices from the independent set, as previously, while placing vertices with neighbors inside and outside the component in the separator. If the weight of the component is at most $w_{\max}$, we recurse in the remaining instance, which has vertex cover at most $9\mathrm{vc}/10$. The complexity of this procedure works out as $T(\mathrm{vc}) \leq 2^{\mathrm{vc}} \cdot T(9\mathrm{vc}/10) = 2^{\mathcal{O}(\mathrm{vc})}$.

What if the optimal partition of C only has components with few vertices of C? In that case we observe that we do not need to compute the full partition (which would take time $\mathrm{vc}^{\mathrm{vc}}$), but it suffices to guess a good bipartition of C into two sets, of roughly the same size (say, both sets have size at least $2\mathrm{vc}/5$), such that the two sets are a coarsening of the optimal partition. In other words, we compute two subsets of C, of roughly equal size, such that the intersection of each connected component with C is contained in one of the two sets. This is always possible in this case, because no connected component has a very large intersection with C. Now, all vertices of I which have neighbors on both sides of the bipartition of C must be placed in the separator. But once we do this, we have disconnected the instance into two independent instances, each of vertex cover at most $3\mathrm{vc}/5$. The complexity of this procedure again works out as $T(\mathrm{vc}) \leq 2^{\mathrm{vc}} \cdot 2 \cdot T(3\mathrm{vc}/5) = 2^{\mathcal{O}(\mathrm{vc})}$.

Let us now proceed to the technical details. To solve Weighted Vertex Integrity, we first define the annotated and optimization version of the problem.

Annotated Weighted Vertex Integrity with Vertex Cover

Instance: A vertex-weighted graph $G = (V, E, w)$, a vertex cover C of G, an integer $w_{\max}$.

Goal: Find a minimum weight irredundant wvi-set $S \subseteq V \setminus C$ such that $w(D) \leq w_{\max}$ for all $D \in \mathtt{cc}(G - S)$. If there is no such S, report NO.

Then we give an algorithm that solves Annotated Weighted Vertex Integrity with Vertex Cover.

Theorem 6.26. *Annotated Weighted Vertex Integrity with Vertex Cover can be solved in time* $2^{\mathcal{O}(|C|)}n^{\mathcal{O}(1)}$.

Proof. Let $\mathcal{I} = (G, C, w_{\max})$ be an instance of Annotated Weighted Vertex Integrity with Vertex Cover, where $\mathrm{OPT}(\mathcal{I})$ denotes the weight of the optimal solution S, provided such a set S exists. Additionally, let $k = |C|$ and $I = V \setminus C$.

Assume that $N(v) = \varnothing$ for some $v \in I$. We claim that if $w(v) > w_{\max}$, then there is no irredundant wvi-set $S \subseteq V \setminus C$ such that $w(D) \leq w_{\max}$ for all $D \in \mathtt{cc}(G - S)$ and we report NO. Assume there existed such a set S. Then, since $w(v) > w_{\max}$, it follows that $v \in S$. However, since $N(v) = \varnothing$, it follows that no component of $G - S$ contains neighbors of v, thus v is redundant, which is a contradiction. If on the other hand it holds that $w(v) \leq w_{\max}$, then reduce $\mathcal{I}$ to $\mathcal{I}' = (G - v, C, w_{\max})$, where $\mathrm{OPT}(\mathcal{I}) = \mathrm{OPT}(\mathcal{I}')$. The correctness of the reduction rule is easy to see. In the following, assume $N(v) \neq \varnothing$, for all $v \in I$. We perform branching on whether there exists a component D in $G - S$ such that $|D \cap C| \geq k/10$ or not.

In the first case, we guess $D \cap C$. Since S is irredundant, it holds that $D \setminus C = \{v \in I : N(v) \subseteq D \cap C\}$. Moreover, let $I_R \subseteq I$ be the set of vertices with neighbors in both $D \cap C$ and $C \setminus D$. If D is not connected or $w(D) > w_{\max}$, we immediately reject. Otherwise, notice that all vertices $v \in I_R$ must belong to S, thus we obtain a smaller instance $\mathcal{I}' = (G - (D \cup I_R), C \setminus D, w_{\max})$, where $|C \setminus D| \leq 9k/10$ and $\mathrm{OPT}(\mathcal{I}) = w(I_R) + \mathrm{OPT}(\mathcal{I}')$.

In the second case, i.e., when for all components $D \in \mathtt{cc}(G - S)$ it holds that $|D \cap C| < k/10$, we first show the following claim.

▷ Claim 6.27. There exists a set $A \subseteq C$ of size $k/2 \leq |A| \leq 3k/5$ such that every component D of $G - S$ satisfies either $D \cap C \subseteq A$ or $D \cap C \subseteq C \setminus A$.

Proof of the claim. Let $\mathcal{D} = \{D_1, \ldots, D_p\}$ be the set of connected components of $G - S$. Note that $\bigcup_{D \in \mathcal{D}}(D \cap C) = C$ since $S \cap C = \varnothing$. By the assumption, $|D \cap C| < k/10$ holds for all $D \in \mathcal{D}$. Set $A = \bigcup_{i=1}^{j}(D_i \cap C)$ where j is the maximum index such that $|A \setminus D_j| < k/2$. By the definition of A, $|A| \geq k/2$ holds. Moreover, since $|D_j| < k/10$, $|A| \leq k/2 + k/10 = 3k/5$. Thus, the claim holds. ◁

Next, we guess $A \subseteq C$ by considering all possible subsets of C whose sizes are between $k/2$ and $3k/5$. Since every component D of $G - S$ satisfies either $D \cap C \subseteq A$ or $D \cap C \subseteq C \setminus A$, all vertices of $I_R \subseteq I$ must belong to S, where $v \in I_R$ if it has neighbors in both A and $C \setminus A$. In that case, we obtain two separate instances $\mathcal{I}_1 = (G[A \cup I_A], A, w_{\max})$ and $\mathcal{I}_2 = (G[(C \setminus A) \cup I_{C \setminus A}], C \setminus A, w_{\max})$, where $I_Z = \{v \in I : N(v) \subseteq Z\}$ for $Z \in \{A, C \setminus A\}$, and $\mathrm{OPT}(\mathcal{I}) = w(I_R) + \mathrm{OPT}(\mathcal{I}_1) + \mathrm{OPT}(\mathcal{I}_2)$.

Consequently, for both cases we obtain smaller instances and hence, we only have to recursively compute the annotated problems. Finally, we analyze the running time of our algorithm. Let $T(k)$ denote the running time when $|C| = k$. Then it holds that $T(k) \leq 2^k n^{\mathcal{O}(1)} \cdot T(9k/10) + 2^k n^{\mathcal{O}(1)} \cdot 2T(3k/5) + n^{\mathcal{O}(1)}$, thus $T(k) = 2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ follows. □

Theorem 6.28. *Weighted Vertex Integrity can be solved in time* $2^{\mathcal{O}(\mathrm{vc})}n^{\mathcal{O}(1)}$.

Proof. Let C be a vertex cover of G of size $|C| = \mathrm{vc}$ and $I = V \setminus C$. For the analysis, fix an optimal separator S, which by Theorem 6.1 can be assumed to be irredundant. We first guess $S \cap C$, and set $C' = C \setminus S$. In the following, let for $v \in I$, $N(v)$ denote the neighborhood of v in G, while $N'(v) = N(v) \cap C'$. Let $D_{\max}$ denote a connected component of $G - S$ of maximum weight $w(D_{\max}) = w_{\max}$, and set $D_{\max} \cap C' = A \subseteq C'$. We argue that given $S \cap C$, there are at most $2^{|C'|} + n \leq 2^{\mathrm{vc}} + n$ cases regarding $D_{\max}$, where $n = |V|$. If $A = \varnothing$, then $D_{\max} = \{v\}$ for some vertex $v \in I$ with $N(v) \subseteq S \cap C$, for a total of at most $|I| \leq n$ choices. Otherwise, it holds that $A \neq \varnothing$, and due to the irredundancy of S it follows that $D_{\max} \setminus A = \{v \in I : \varnothing \neq N'(v) \subseteq A\}$, while any vertex of $I_R = \{v \in I : N'(v) \cap A, N'(v) \cap (C' \setminus A) \neq \varnothing\}$ must belong to S.

Consequently, it suffices to guess $S \cap C$ and $D_{\max}$, assure that $D_{\max}$ is indeed connected as intended, and then compute the expression

$$\mathrm{OPT}(\mathcal{I}) + w(S \cap C) + w(I_R) + w_{\max},$$

where $\mathcal{I} = (G - (D_{\max} \cup I_R \cup (S \cap C)), C', w_{\max})$ is an instance of Annotated Weighted Vertex Integrity with Vertex Cover and $\mathrm{OPT}(\mathcal{I})$ the weight of its optimal solution. Due to Theorem 6.26, $\mathrm{OPT}(\mathcal{I})$ can be computed in time $2^{\mathcal{O}(\mathrm{vc})}n^{\mathcal{O}(1)}$, therefore the total running time is $2^{\mathrm{vc}} \cdot (2^{\mathrm{vc}} + n) \cdot n^{\mathcal{O}(1)} \cdot 2^{\mathcal{O}(\mathrm{vc})}n^{\mathcal{O}(1)} = 2^{\mathcal{O}(\mathrm{vc})}n^{\mathcal{O}(1)}$. □

6.7. FPT Approximation Scheme

In this section we show that Component Order Connectivity and Unweighted Vertex Integrity admit FPT approximation schemes parameterized by treewidth. More precisely, we show the following.

Theorem 6.29. *There is an algorithm which, for all* $\varepsilon > 0$*, when given as input a graph* G *of treewidth* tw *and an integer* $p \in \mathbb{N}$*, returns in time* $(\mathrm{tw}/\varepsilon)^{\mathcal{O}(\mathrm{tw})}n^{\mathcal{O}(1)}$ *a set* $S \subseteq V(G)$ *of size at most* p *such that the maximum size of a connected component of* $G - S$ *is at most* $(1 + \varepsilon)$ *times the minimum possible value among all sets* $S' \subseteq V(G)$ *of size at most* p*.*

Proof sketch. The algorithm is a simple adaptation of the algorithm of Theorem 7.17. In particular, we use the same signature and pseudo-signature definitions, however on the additional dimension we keep track of the maximum size of a component of the partial solution instead. □

Notice that by iterating over all possible values of p we can also obtain an FPT approximation scheme for the minimization version of Weighted Vertex Integrity.

Theorem 6.30. *There is an algorithm which, for all* $\varepsilon > 0$*, when given as input a graph* G *of treewidth* tw*, returns in time* $(\mathrm{tw}/\varepsilon)^{\mathcal{O}(\mathrm{tw})}n^{\mathcal{O}(1)}$ *a set* $S \subseteq V(G)$ *such that* $|S| + \max_{C \in \mathtt{cc}(G-S)} |C|$ *is at most* $(1 + \varepsilon) \cdot \mathrm{vi}(G)$*.*

Proof. The algorithm simply iterates over all possible values of p and for each value it runs the algorithm of Theorem 6.29. It then returns the best solution found. Assume there exists $S^* \subseteq V(G)$ such that $|S^*| + \max_{C \in \mathtt{cc}(G-S^*)} |C| = \mathrm{vi}(G)$, and let $p^* = |S^*|$. Then, the algorithm of Theorem 6.29 run with parameter p^* returns a solution S such that $\max_{C \in \mathtt{cc}(G-S)} |C| \leq (1+\varepsilon) \cdot \max_{C \in \mathtt{cc}(G-S^*)} |C|$. Hence,

$$|S| + \max_{C \in \mathtt{cc}(G-S)} |C| \leq p^* + (1+\varepsilon) \cdot \max_{C \in \mathtt{cc}(G-S^*)} |C| \leq (1+\varepsilon) \cdot \mathrm{vi}(G).$$

This concludes the proof. □

7. Critical Node Cut

Chapter summary. This chapter investigates the structural parameterized complexity of Critical Node Cut, a well-studied generalization of the classical Vertex Cover problem. Our main result is a significantly improved W[1]-hardness result for the problem. Along the way, we also derive new exact and approximate parameterized algorithms, as well as incompressibility results for the parameterization by vertex cover number.

Let G be a simple, undirected, and unweighted graph, and let k and x be two non-negative integers. The Critical Node Cut problem (CNC for short) asks whether we can delete at most k vertices from G so that the remaining graph contains at most x connected vertex pairs. This problem arises in various applications, such as budgeted immunization for contagion control, analyzing brain connectivity, and uncovering structures in social graphs (see [13, 169] and references therein).

Importantly, CNC generalizes the well-known Vertex Cover problem, which corresponds to the special case $x = 0$. Given its NP-hardness, we study CNC through the lens of parameterized complexity. Previous work has shown that the problem is fixed-parameter tractable (FPT) when parameterized by the vertex cover number of the input graph [268], but W[1]-hard when parameterized by the natural parameter k or by the treewidth tw [169].[1] As a matter of fact, Agrawal, Lokshtanov, and Mouawad have shown that this hardness persists even when parameterizing by the combined parameter $k + \mathrm{tw}$ [5].

Our Contribution. Our first result, which constitutes our main technical contribution, is to significantly improve over the aforementioned hardness result of Agrawal, Lokshtanov, and Mouawad [5]. In particular, in Theorem 7.2 we prove that CNC remains W[1]-hard even when parameterized by the combined parameter $k + \mathrm{fes} + \Delta + \mathrm{pw}$, where fes, Δ, and pw denote the feedback edge set number, the maximum degree, and the pathwidth of the input graph respectively. Notice that this also implies that the problem is W[1]-hard when parameterized by $k + \mathrm{ctw}$, where ctw denotes the cutwidth of the input graph, as for any graph it holds that $\mathrm{ctw} \leq \mathrm{pw} \cdot \Delta$. In fact, this intractability result is tight in the sense that tree-depth together with maximum degree trivially yields FPT, since this parameterization bounds the graph size.[2] To obtain our result, we adapt the reduction for the related Vertex Integrity problem presented in Section 6.3.

Moving on, we obtain several results concerning the tractability of the problem under different structural parameterizations. We identify three structural parameters that render CNC fixed-parameter tractable, thereby contrasting the hardness result above:

[1] More strongly, the reduction in [169] implies W[1]-hardness with respect to tree-depth.

[2] A graph of tree-depth td and maximum degree Δ has at most Δ^{td} vertices.

max-leaf number (Theorem 7.8), vertex integrity (Theorem 7.11), and modular-width (Theorem 7.12). Our techniques towards obtaining those results include reductions to acyclic instances [164], N-fold Integer Programming [200], and dynamic programming on suitable decompositions. We also consider the parameterization by clique-width cw, for which the problem is known to be W[1]-hard, and obtain an algorithm with running time $n^{\mathcal{O}(2^{\mathrm{cw}})}$ (Theorem 7.13) thus placing it in XP.

We then revisit the parameterization by treewidth and identify the bottleneck in standard dynamic programming as the need to track large numerical values. To address this, we apply a rounding technique introduced by Lampis [207], obtaining an FPT approximation scheme. Specifically, in Theorem 7.17 we give an algorithm that, for any $\varepsilon > 0$, computes in $(\mathrm{tw}/\varepsilon)^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$ time a deletion set of size at most k such that the number of connected pairs in the resulting graph is at most $(1+\varepsilon)$ times the minimum possible.

Finally, we investigate the compressibility of CNC. We prove that, unless standard complexity assumptions fail, the problem does not admit a polynomial kernel when parameterized by the vertex cover number of the input graph (Theorem 7.22). This negative result highlights that even very restrictive structural parameterizations do not yield efficient preprocessing algorithms (under standard assumptions).

Related Work. CNC is a well-studied problem; see [204, Section 5.2] and the references therein. Approximation algorithms have been proposed [283], while Hermelin, Kaspi, Komusiewicz, and Navon [169] initiated the problem's study under parameterized complexity, presenting a plethora of both positive and negative results with respect to natural and structural parameterizations. Among others, they prove a (tight) $n^{o(k)}$ lower bound under the ETH, W[1]-hardness by tree-depth, and that the problem is FPT parameterized by $x + \mathrm{tw}$. Polynomial-time algorithms are known for trees [272],[3] and more generally, the problem admits a DP algorithm of running time $n^{\mathcal{O}(\mathrm{tw})}$ [3]. As previously mentioned, CNC is known to be W[1]-hard parameterized by $k+\mathrm{tw}$ [5]. On the other hand, it admits FPT algorithms with single-exponential parametric dependence when parameterized by vertex cover number [268] or neighborhood diversity [267].

7.1. Preliminaries

Given a graph G, $\mathtt{cc}(G)$ denotes the set of its connected components, while $\mathrm{cp}(G)$ denotes the number of pairs of connected vertices in G, that is, $\mathrm{cp}(G) = \sum_{C \in \mathtt{cc}(G)} \binom{|V(C)|}{2}$.

We define the *height of a node of a tree decomposition* as the distance from the root of the tree decomposition to said node. In that case, the *height of a tree decomposition* is defined as the maximum distance from the root to any of its nodes.

[3] That is the case even for weighted variants of the problem.

Critical Node Cut

Instance: A graph $G = (V, E)$ as well as integers k and x.
Goal: Determine whether there exists a set $S \subseteq V$ with $|S| \leq k$ such that in $G - S$ there are at most x pairs of connected vertices, that is, whether $\mathrm{cp}(G - S) \leq x$.

Lemma 7.1. *Let $k \in \mathbb{N}$. Assume we are given n functions $f_i \colon [0, k] \to \mathbb{N}$, where $i \in [n]$, and define $f \colon [0, k] \to \mathbb{N}$ such that for all $k' \in [0, k]$,*

$$f(k') = \min_{\substack{k_i \in [0, k'] \\ k_1 + \ldots + k_n = k'}} \sum_{i \in [n]} f_i(k_i).$$

One can compute the function f in time $\mathcal{O}(nk^2)$.

Proof. We will perform dynamic programming. Consider the table $T[a][b]$, where $a \in [n]$ and $b \in [0, k]$. On a high-level, $T[a][b]$ contains the minimum sum over the first a functions by partitioning a total budget of b over them, that is,

$$T[a][b] = \min_{\substack{k_i \in [0, b] \\ k_1 + \ldots + k_a = b}} \sum_{i \in [a]} f_i(k_i). \tag{7.1}$$

In that case it suffices to compute the values of the table for $a = n$.

We set $T[a][0] = \sum_{i \in [a]} f_i(0)$ for all $a \in [n]$, and $T[1][b] = f_1(b)$ for all $b \in [0, k]$. Then, for $a \in [2, n]$ and $b \in [k]$, let

$$T[a][b] = \min_{b' \in [0, b]} \left\{ T[a-1][b - b'] + f_a(b') \right\}.$$

Notice that the DP table has a total of $n(k+1)$ cells, each of which requires $\mathcal{O}(k)$ time to be filled, assuming that the evaluation of a function f_i requires $\mathcal{O}(1)$ time. Consequently, the total running time is $\mathcal{O}(nk^2)$. As for the correctness of Equation (7.1), one can easily argue about it by induction. □

7.2. W[1]-hardness

In this section we prove that Critical Node Cut is W[1]-hard when parameterized by the combined parameter $k + \mathrm{fes} + \Delta + \mathrm{pw}$, where fes, Δ, and pw denote the feedback edge number, the maximum degree, and the pathwidth of the input graph, respectively. This significantly improves over the previous result of Agrawal, Lokshtanov, and Mouawad [5] who showed that the problem is W[1]-hard parameterized by k plus the treewidth of the input graph. Our reduction follows along the lines of the one presented in Section 6.3. We reduce from the Restricted Unary Bin Packing problem, which we formally define below and is known to be W[1]-hard parameterized by the number of bins k (cf. Chapter 6).

Restricted Unary Bin Packing

Instance: A multiset $A = \{a_1, \ldots, a_n\}$ of integers in unary, $k \in \mathbb{N}$, as well as a function $f\colon A \to \binom{[k]}{2}$.

Goal: Determine whether there is a partition of A into multisets $\mathcal{A}_1, \ldots, \mathcal{A}_k$, such that for all $i \in [k]$ it holds that (i) $\Sigma(\mathcal{A}_i) = \Sigma(A)/k$, and (ii) $\forall a \in \mathcal{A}_i$, $i \in f(a)$.

We first present a sketch of our reduction. The *weight* of a vertex refers to the length of a path attached to it; this is later formalized in the construction. Similarly to the construction of Section 6.3, for every bin of the Restricted Unary Bin Packing problem instance we introduce a clique of $\mathcal{O}(k^2)$ heavy vertices, and then connect any pair of such cliques via two paths. The weights are set in such a way that an optimal solution will delete only vertices from the connection paths. In order to construct a path for a pair of bins, we compute the set of all subset sums of the items that can be placed in these two bins, and introduce a vertex of medium weight per such subset sum. This step can be completed in polynomial time, as all items are in unary. Moreover, every such vertex corresponding to subset sum s is preceded by exactly s vertices of weight 1. An optimal solution will cut the path so that the number of vertices of weight 1 will be partitioned between the two bins, encoding the subset sum of the elements placed in each bin. The second path that we introduce has balancing purposes, allowing us to exactly count the number of vertices of medium weight that every connected component will end up with. In the end, the intended solution deletes only vertices of medium weight, resulting in components of only two possible sizes.

Theorem 7.2. *Critical Node Cut is W[1]-hard parameterized by* $k + \mathrm{fes} + \Delta + \mathrm{pw}$.

Proof. Let (A, k, f) be an instance of Restricted Unary Bin Packing, where $A = \{a_1, \ldots, a_n\}$ denotes the multiset of items given in unary, k is the number of bins, and $f\colon A \to \binom{[k]}{2}$ dictates the pair of bins an item may be placed into. In the following let $B = \Sigma(A)/k$, $c = 6k^2$, $M = kB + 1$, $L = 36k^4BM$, and $T = L + (k-1) \cdot 2BM + B$. Notice that $M > kB$ and $L/c = 6k^2BM$. We will reduce (A, k, f) to an equivalent instance (G, k', x) of Critical Node Cut, where $k' = 2\binom{k}{2}$ and $x = k \cdot \binom{T}{2} + 2\binom{k}{2} \cdot \binom{M-1}{2}$ denote the size of the deletion set and the bound on the number of pairs of connected vertices, respectively.

We proceed to describe the construction of graph G. For ease of presentation, when we say that a vertex $v \in V(G)$ has weight $w \geq 1$, it means that we introduce a path on $w - 1$ vertices and connect one of its endpoints to v.

For every $i \in [k]$, we introduce a clique on vertex set $\hat{C}_i$, which is comprised of c vertices, each of weight L/c. Fix i and j such that $1 \leq i < j \leq k$, and let $H_{i,j} = \{a \in A : f(a) = \{i, j\}\}$ denote the multiset of all items of A which can be placed either on bin i or bin j, where $\Sigma(H_{i,j}) \leq 2B$. Let $\mathcal{S}(H_{i,j}) = \{\Sigma(H) : H \subseteq H_{i,j}\}$ denote the set[4] of all subset sums of $H_{i,j}$, and notice that since every element of $H_{i,j}$ is encoded in

[4]We stress that $\mathcal{S}(H_{i,j})$ is a set and not a multiset.

unary, $\mathcal{S}(H_{i,j})$ can be computed in polynomial time using, e.g., Bellman's classical DP algorithm [22]. We next construct two paths connecting the vertices of $\hat{C}_i$ and $\hat{C}_j$. First, we introduce the vertex set $\hat{U}_{i,j} = \{v_q^{i,j} : q \in [0, 4B - |\mathcal{S}(H_{i,j})| + 1]\}$, all the vertices of which are of weight M. Add edges $(v_q^{i,j}, v_{q+1}^{i,j})$ for all $q \in [0, 4B - |\mathcal{S}(H_{i,j})|]$, as well as $(v_1, v_0^{i,j})$ and $(v_{4B-|\mathcal{S}(H_{i,j})|+1}^{i,j}, v_2)$, for all $v_1 \in \hat{C}_i$ and $v_2 \in \hat{C}_j$. Next, we introduce the vertex set $\hat{D}_{i,j} = \{s_q^{i,j} : q \in [\Sigma(H_{i,j})]\} \cup \{\sigma_q^{i,j} : q \in \mathcal{S}(H_{i,j})\}$, with the s-vertices being of weight 1 and the σ-vertices of weight M. Then, add the following edges:

- $(v_1, \sigma_0^{i,j})$ and $(\sigma_{\Sigma(H_{i,j})}^{i,j}, v_2)$, for all $v_1 \in \hat{C}_i$ and $v_2 \in \hat{C}_j$,
- for $q \in \mathcal{S}(H_{i,j})$, add the edges $(s_q^{i,j}, \sigma_q^{i,j})$ if $q \neq 0$ and $(\sigma_q^{i,j}, s_{q+1}^{i,j})$ if $q \neq \Sigma(H_{i,j})$, and
- for $q \in [\Sigma(H_{i,j})] \setminus \mathcal{S}(H_{i,j})$, add the edge $(s_q^{i,j}, s_{q+1}^{i,j})$.

This concludes the construction of G. See Figure 7.1 for an illustration. Notice that the number of vertices of G excluding those belonging to the cliques $\hat{C}_1, \ldots, \hat{C}_k$ or to a path attached to them is $(4B+2)M \cdot \binom{k}{2} + kB < 6k^2BM = L/c$.

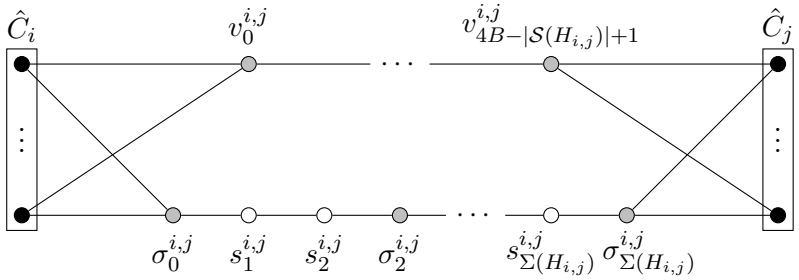


Figure 7.1.: Rectangles denote cliques of size c. Here we assume that $1 \leq i < j \leq k$, $1 \notin \mathcal{S}(H_{i,j})$, and $2 \in \mathcal{S}(H_{i,j})$. The gray and black color indicates weight of M and L/c, respectively.

Lemma 7.3. *If (A, k, f) is a Yes-instance of* Restricted Unary Bin Packing*, then (G, k', x) is a Yes-instance of* Critical Node Cut*.*

Proof. Let $(\mathcal{A}_1, \ldots, \mathcal{A}_k)$ be a partition of A such that for all $i \in [k]$ it holds that (i) $\Sigma(\mathcal{A}_i) = B$, and (ii) $\forall a \in \mathcal{A}_i$, $i \in f(a)$. Fix $1 \leq i < j \leq k$, and let $\mathcal{H}_{i,j} \subseteq H_{i,j}$ such that $\mathcal{A}_i \cap H_{i,j} = \mathcal{H}_{i,j}$, while $\mathcal{A}_j \cap H_{i,j} = H_{i,j} \setminus \mathcal{H}_{i,j}$. Note that for all $i \in [k]$, it holds that

$$\sum_{j \in [i-1]} (\Sigma(H_{j,i}) - \Sigma(\mathcal{H}_{j,i})) + \sum_{j \in [i+1,k]} \Sigma(\mathcal{H}_{i,j}) = B. \tag{7.2}$$

Additionally, let $r_{i,j} = |\mathcal{S}(H_{i,j}) \cap [0, \Sigma(\mathcal{H}_{i,j}) - 1]|$ denote the number of subset sums of $H_{i,j}$ of sum at most $\Sigma(\mathcal{H}_{i,j}) - 1$.

We are now ready to construct the cut S. Set $S = \{\sigma_{\Sigma(\mathcal{H}_{i,j})}^{i,j}, v_{2B-r_{i,j}}^{i,j} : 1 \leq i < j \leq k\}$. It holds that $|S| = 2\binom{k}{2} = k'$. Additionally, it holds that $G - S$ has exactly $k + 2\binom{k}{2}$

connected components: every clique $\hat{C}_i$ belongs to a distinct component, while all vertices of S are of weight M, thus resulting in $|S|$ additional components of size $M-1$. To prove the statement, it suffices to show that the connected components of $G-S$ apart from the latter $|S|$ ones have size at most T.

Let for all $i \in [k]$, $\mathcal{C}_i$ denote the connected component of $G-S$ that contains the vertices of $\hat{C}_i$. Fix $1 \leq i < j \leq k$. Since $S \cap (\hat{D}_{i,j} \cup \hat{U}_{i,j}) = \{\sigma^{i,j}_{\Sigma(\mathcal{H}_{i,j})}, v^{i,j}_{2B-r_{i,j}}\}$, it holds that $\mathcal{C}_i$ contains $\Sigma(\mathcal{H}_{i,j})$ vertices of weight 1 belonging to $\hat{D}_{i,j}$, as well as $r_{i,j} + (2B - r_{i,j}) = 2B$ vertices of weight M from $\hat{D}_{i,j} \cup \hat{U}_{i,j}$. As for $\mathcal{C}_j$, it contains $\Sigma(H_{i,j}) - \Sigma(\mathcal{H}_{i,j})$ vertices of weight 1 belonging to $\hat{D}_{i,j}$, as well as $(|\mathcal{S}(H_{i,j})| - 1 - r_{i,j}) + (4B - |\mathcal{S}(H_{i,j})| + 2 - 1 - (2B - r_{i,j})) = 2B$ vertices of weight M from $\hat{D}_{i,j} \cup \hat{U}_{i,j}$. For any fixed $i \in [k]$, it follows that

$$\begin{aligned}
|V(\mathcal{C}_i)| &= L + \sum_{j \in [i-1]} \left(\Sigma(H_{i,j}) - \Sigma(\mathcal{H}_{i,j}) + 2BM\right) + \sum_{j \in [i+1,k]} \left(\Sigma(\mathcal{H}_{i,j}) + 2BM\right) \\
&= L + (k-1) \cdot 2BM + B \\
&= T,
\end{aligned}$$

where the second equality is due to Equation (7.2). This concludes the proof. □

Lemma 7.4. *If (G, k', x) is a Yes-instance of* Critical Node Cut*, then (A, k, f) is a Yes-instance of* Restricted Unary Bin Packing*.*

Proof. Let $S \subseteq V(G)$ with $|S| \leq k' = 2\binom{k}{2}$ such that the number of pairs of connected vertices in $G - S$ is minimized and is at most x. Notice that $|S| = k'$, as otherwise there always exists a vertex whose deletion reduces said number. For all $i \in [k]$ it holds that $|\hat{C}_i| > k'$, thus the vertices of $\hat{C}_i \setminus S$ belong to the same connected component of $G - S$.

▷ Claim 7.5. For any $i, j \in [k]$ with $1 \leq i < j \leq k$ it holds that $|S \cap \hat{U}_{i,j}| \leq 1$ and $|S \cap \hat{D}_{i,j}| \leq 1$.

Proof of the claim. Let $i, j \in [k]$ with $1 \leq i < j \leq k$ such that $|S \cap \hat{Z}_{i,j}| \geq 2$, where $\hat{Z}_{i,j} \in \{\hat{U}_{i,j}, \hat{D}_{i,j}\}$, and let v denote the rightmost vertex of $\hat{Z}_{i,j}$ that belongs to S, that is, no vertex of $\hat{Z}_{i,j}$ between v and the vertices of $\hat{C}_j$ belongs to S. Let $\mathcal{C}_j$ denote the connected component of $G - S$ that contains the vertices of $\hat{C}_j$. It holds that $|V(\mathcal{C}_j)| \geq 2L/3$, since $c \geq 3k'$. Let $v' \in \hat{C}_j \setminus S$ and consider the deletion set $S' = (S \setminus \{v\}) \cup \{v'\}$, where $|S'| = |S|$.

We argue that $\mathrm{cp}(G - S') < \mathrm{cp}(G - S)$, thus contradicting the optimality of S. Indeed, starting from $G - S$ it holds that not deleting v increases the number of connected pairs by at most $4BM \cdot |V(\mathcal{C}_j)| + 4BM \cdot M = (2L/3k^2c) \cdot (|V(\mathcal{C}_j)| + M)$, while subsequently

deleting v' decreases the number of pairs by at least $(|V(\mathcal{C}_j)| - L/c) \cdot (L/c)$. It holds that

$$\begin{aligned}
(|V(\mathcal{C}_j)| - L/c) \cdot (L/c) > (2L/3k^2c) \cdot (|V(\mathcal{C}_j)| + M) &\iff \\
|V(\mathcal{C}_j)| - L/c > (2/3k^2) \cdot (|V(\mathcal{C}_j)| + M) &\iff \\
|V(\mathcal{C}_j)| \left(1 - \frac{2}{3k^2}\right) - L/c > \frac{2}{3}k^2 M &\impliedby \\
|V(\mathcal{C}_j)| \left(1 - \frac{2}{3k^2}\right) - L/c > L/c &\impliedby \\
\frac{2L}{3} \cdot \left(1 - \frac{2}{3k^2}\right) > \frac{2L}{6k^2} &\iff \\
\left(1 - \frac{2}{3k^2}\right) > \frac{1}{2k^2} &\iff \\
1 \geq \frac{5}{6k^2}, &
\end{aligned}$$

which holds for all $k \geq 1$ and where we used the fact that $\frac{2}{3}k^2 M < \frac{L}{c}$ as well as that $|V(\mathcal{C}_j)| \geq 2L/3$. ◁

We next argue that every connected component of $G - S$ contains vertices belonging to at most one clique $\hat{C}_i$.

▷ Claim 7.6. For any connected component $\mathcal{C}$ of $G - S$ it holds that

$$|\{i \in [k] : V(\mathcal{C}) \cap \hat{C}_i \neq \varnothing\}| \leq 1\,.$$

Proof of the claim. We first fix some notation used throughout the proof. Let $\Pi \subseteq 2^{[k]}$ be the partition of $[k]$ such that $p \in \Pi$ if and only if there exists a connected component $\mathcal{C}_p \in \mathtt{cc}(G - S)$ with $p = \{i \in [k] : V(\mathcal{C}_p) \cap \hat{C}_i \neq \varnothing\}$. Notice that for $p \neq \varnothing$, such a component is unique. For $p \in \Pi$ we further define the set $\hat{R}(p)$ which is composed of the vertices $\bigcup_{i \in p} \hat{C}_i \cup \bigcup_{i,j \in p} (\hat{U}_{i,j} \cup \hat{D}_{i,j})$ plus the vertices belonging to the paths attached to the former vertices. Let $g \colon \mathtt{cc}(G - S) \to [0, k]$ such that $g(\mathcal{C}) = |\{i \in [k] : V(\mathcal{C}) \cap \hat{C}_i \neq \varnothing\}|$ for all $\mathcal{C} \in \mathtt{cc}(G - S)$. Consider a component $\mathcal{C}_{p_{\max}} \in \mathtt{cc}(G - S)$ that maximizes g, where $g(\mathcal{C}_{p_{\max}}) = |p_{\max}| \geq 1$. It suffices to prove that $|p_{\max}| < 2$. Towards a contradiction assume that $|p_{\max}| \geq 2$.

Let $\mathcal{C}_{p'}$ with $g(\mathcal{C}_{p'}) = |p'| < |p_{\max}|$. We argue that $S \cap \hat{R}(p') = \varnothing$. To see this, notice that the deletion of such a vertex results in at most

$$\frac{L}{c} \cdot \left(|p'| \cdot L + \frac{L}{c}\right)$$

disconnected pairs, where we use the fact that the number of vertices of G excluding those belonging to the cliques $\hat{C}_1, \ldots, \hat{C}_k$ or to a path attached to them is less than L/c. On the other hand, $|V(\mathcal{C}_{p_{\max}})| \geq |p_{\max}| \cdot L - k' \cdot (L/c) \geq (|p_{\max}| - 1) \cdot L + (2L)/3$, thus the deletion of a vertex $v \in \hat{C}_i \setminus S$ with $i \in p_{\max}$ results in at least

$$\frac{L}{c} \cdot \left((|p_{\max}| - 1) \cdot L + \frac{2L}{3}\right) > \frac{L}{c} \cdot \left(|p'| \cdot L + \frac{L}{c}\right)$$

disconnected pairs.

The previous paragraph, along with Claim 7.5 and the fact that $|S| = 2\binom{k}{2}$, imply that there exists $p \in \Pi$ with $|p| = |p_{\max}|$ such that $|S \cap \hat{R}(p)| \geq 2\binom{|p|}{2}$. Let $S' = (S \setminus \hat{R}(p)) \cup \{v_0^{i,j}, \sigma_0^{i,j} : i, j \in p\}$, where $|S'| \leq |S|$. We argue that $\text{cp}(G - S') < \text{cp}(G - S)$. Let x_1 (resp., x_2) denote the number of pairs of connected vertices in $G - S$ (resp., $G - S'$) such that at least one vertex belongs to $\hat{R}(p)$ (notice that the rest of the pairs remain the same in the two graphs). It holds that

$$\begin{aligned} 2 \cdot x_1 &> 2 \cdot \binom{(|p| - 1) \cdot L + 2L/3}{2} \\ &\geq ((|p| - 1)L + 2L/3) \cdot ((|p| - 1)L + L/3) \\ &= L^2(|p|^2 - |p| + 2/9), \end{aligned}$$

where the first inequality is due to the component of $G - S$ that contains the vertices of $\{\hat{C}_i : i \in p\} \setminus S$. On the other hand, in $G - S'$ the vertices of each $\hat{C}_i$ for $i \in p$ are in distinct connected components, thus it holds that

$$\begin{aligned} 2 \cdot x_2 &< 2|p| \cdot \binom{L + L/c}{2} \\ &\leq |p|(L + L/c)^2 \\ &= L^2(|p| + |p| \cdot \frac{2c + 1}{c^2}) \\ &\leq L^2(|p| + 2/9), \end{aligned}$$

where the first inequality is due to the fact that the number of vertices of G apart from those of $\bigcup_{i \in [k]} \hat{C}_i$ or attached to them is at most L/c. The last inequality is due to the fact that $|p| \cdot \frac{2c+1}{c^2} \leq k \cdot \frac{12k^2+1}{36k^4}$, which for $k \geq 2$ is upper-bounded by $2/9$.

Finally, we have that

$$\begin{gathered} x_1 > x_2 \impliedby \\ L^2(|p|^2 - |p| + 2/9) \geq L^2(|p| + 2/9) \iff \\ |p|^2 \geq 2|p|, \end{gathered}$$

which holds as by assumption $|p| \geq 2$. This contradicts the optimality of set S. $\lhd$

By Claim 7.6 and the fact that $k' = 2\binom{k}{2}$ it follows that S contains exactly one vertex per path between cliques. In that case, since the vertices of those paths are of weight 1 or M, regarding the connected components of $G - S$ it holds that there are exactly k of size at least L, as well as $a \in [\binom{k}{2}, 2\binom{k}{2}]$ of size exactly $M - 1$, where $a \geq \binom{k}{2}$ since half the paths between cliques contain only vertices of weight M. Further notice that the number of pairs of connected vertices is minimized when each component of size at least

L contains $\frac{|V(G)|-a\cdot M}{k}$ vertices,[5] while for all $a \leq 2\binom{k}{2}$ it holds that

$$k\cdot\binom{(|V(G)|-a\cdot M)/k}{2}+a\cdot\binom{M-1}{2} > k\cdot\binom{(|V(G)|-(a+1)\cdot M)/k}{2}+(a+1)\cdot\binom{M-1}{2}.$$

Consequently, the minimum number of pairs of connected vertices is obtained exactly when there are $2\binom{k}{2}$ components in $G-S$ of size $M-1$, and the rest of the vertices are partitioned equally among the remaining k components. Notice that

$$\begin{aligned}|V(G)| &= k\cdot L + (4B+2)M\cdot\binom{k}{2} + kB\\ &= k\cdot(L+2BM(k-1)+B) + 2M\cdot\binom{k}{2}\\ &= k\cdot T + 2\binom{k}{2}\cdot M,\end{aligned}$$

thus since $x = k\cdot\binom{T}{2} + 2\binom{k}{2}\cdot\binom{M-1}{2}$ it follows that this is indeed the case for $G-S$. Consequently, for all $1\leq i<j\leq k$, there exists $\sigma_q^{i,j}\in S$ with $q\in\mathcal{S}(H_{i,j})$.

Let $\mathcal{C}_i$ denote the connected component of $G-S$ containing the vertices of $\hat{C}_i$. Since $\mathcal{C}_i$ contains T vertices, while $kB<M$, it follows that $\hat{\mathcal{C}}_i$ contains exactly B vertices of weight 1. Let $(\mathcal{A}_1,\ldots,\mathcal{A}_k)$ be a partition of A defined in the following way: for all $1\leq i<j\leq k$, if $\sigma_q^{i,j}\in S$, then $\mathcal{A}_i\cap H_{i,j}=\mathcal{H}_{i,j}$ and $\mathcal{A}_j\cap H_{i,j}=H_{i,j}\setminus\mathcal{H}_{i,j}$, where $\mathcal{H}_{i,j}\subseteq H_{i,j}$ such that $\Sigma(\mathcal{H}_{i,j})=q$. Notice that $\Sigma(\mathcal{A}_i)$ is equal to the number of vertices of weight 1 in $\mathcal{C}_i$, therefore $\Sigma(\mathcal{A}_i)=B$ follows. □

Lemma 7.7. *It holds that* $\mathrm{fes}(G)=\mathcal{O}(k^5)$, $\Delta(G)=\mathcal{O}(k^2)$, *and* $\mathrm{pw}(G)=\mathcal{O}(k^3)$.

Proof. For the feedback edge number, let $F\subseteq E(G)$ contain all edges between vertices of $\hat{C}_i$, for all $i\in[k]$, as well as all edges between such vertices and endpoints of paths $\hat{U}_{i,j}$ and $\hat{D}_{i,j}$. Notice that

$$|F| = k\cdot\binom{c}{2} + 4\binom{k}{2}\cdot c = \mathcal{O}(k^5),$$

while the graph remaining after the deletion of the edges in F is a caterpillar forest. For the pathwidth bound notice that $G-\bigcup_{i=1}^{k}\hat{C}_i$ is a caterpillar forest, thus $\mathrm{pw}(G)=\mathcal{O}(k^3)$. Lastly, for the maximum degree notice that for all $v\in\bigcup_{i\in[k]}\hat{C}_i$, $|N(v)|=c+2(k-1)=\mathcal{O}(k^2)$. As for the vertices of $\hat{D}_{i,j}\cup\hat{U}_{i,j}$, they are of degree at most 3, apart from the endpoints which are neighbors with all the vertices of a single clique, therefore of degree $\mathcal{O}(k^2)$. Any remaining vertex is of degree at most 2. □

Due to Lemmas 7.3, 7.4, and 7.7, Theorem 7.2 follows. □

[5] To see this notice that $\binom{n-\varepsilon}{2}+\binom{n+\varepsilon}{2} > 2\cdot\binom{n}{2}$ for all $\varepsilon>0$.

7.3. Algorithms

In this section we identify several structural parameters that render Critical Node Cut fixed-parameter tractable, namely max-leaf number, vertex integrity, and modular-width. Furthermore, we show that the problem parameterized by clique-width belongs to XP.

7.3.1. Max-Leaf Number

Here we present a single-exponential algorithm for Critical Node Cut parameterized by the max-leaf number of the input graph. This is a well-studied but very restricted parameter [118, 119, 205]. Note that this nicely complements the W[1]-hardness of the problem parameterized by fes + Δ which follows by Theorem 7.2; graphs of max-leaf number at most k are known to be a subdivision of a graph of at most $\mathcal{O}(k)$ vertices [190], thus their feedback edge set number and their maximum degree are bounded by a function of k. As a matter of fact, the reduction of Theorem 7.2 produces a graph that, on a first look, appears to have bounded max-leaf number. The size of this parameter increases due to attaching long paths on the weighted vertices, and Theorem 7.8 implies that this increase is unavoidable, as the problem is FPT under this parameterization.

Our algorithm closely follows the one presented in Chapter 6 for Vertex Integrity parameterized by max-leaf number. In particular, our first step is to contract any cycles, reducing the instance to a weighted forest on which some of the vertices are marked as undeletable. Then we apply a simple DP algorithm for trees in order to compute, for every connected component of the instance, the minimum number of pairs of connected vertices remaining after at most b vertex deletions from said component, for all $b \in [0, k]$. Having computed this for every component, applying Lemma 7.1 then allows us to determine the number of deletions per component such that the total number of connected pairs is minimized.

Theorem 7.8. *Given an instance $\mathcal{I} = (G, k, x)$ of Critical Node Cut, there is an algorithm that decides $\mathcal{I}$ in time $2^{\mathcal{O}(\mathrm{ml})}n^{\mathcal{O}(1)}$, where* ml *denotes the max-leaf number of G.*

Proof. Let $X = \{v \in V : \deg_G(v) \geq 3\}$ denote the set of vertices of degree at least 3 in G. By a well-known result by Kleitman and West [190] it holds that $|X| = \mathcal{O}(\mathrm{ml})$. We make use of Lemma 6.19, which we restate for the sake of completeness.

Lemma 7.9 (Lemma 6.19)**.** *Let $G = (V, E)$ be a graph and X be a set of vertices such that all vertices of $V \setminus X$ have degree at most 2 in G. For all positive integers ℓ, p, if there exists a separator S of size at most p such that all components of $G - S$ have size at most ℓ, then there exists such a separator S that also satisfies the following property: for every cycle C of G with $C \cap X \neq \varnothing$ we either have $C \cap S = \varnothing$ or $C \cap X \cap S \neq \varnothing$.*

The proof of Lemma 6.19 uses an exchange argument such that for every component either its size remains the same or it is split into multiple components. In that case, exactly the same proof can be used to show the following.

Lemma 7.10. *Let $G = (V, E)$ be a graph and X be the set of its vertices of degree at least 3 in G. For all positive integers k, x, if there exists a separator S of size at most k such that the number of pairs of connected vertices in $G - S$ is at most x, then there exists such a separator S that also satisfies the following property: for every cycle C of G with $C \cap X \neq \varnothing$ we either have $C \cap S = \varnothing$ or $C \cap X \cap S \neq \varnothing$.*

Now, fix an optimal solution $S \subseteq V$ of size $|S| \leq k$ that maximizes $|S \cap X|$. We first guess its intersection with X and delete these vertices, while marking the vertices of $X \setminus S$ undeletable. For every component $\mathcal{C}_i \in \mathtt{cc}(G - (S \cap X))$, we compute a function $f_i \colon [0, k - |S \cap X|] \to \left[\binom{|V(\mathcal{C}_i)|}{2}\right]$, such that for all $j \in [0, k - |S \cap X|]$, $f_i(j)$ is equal to the minimum number of pairs of connected vertices of $\mathcal{C}_i$ after the deletion of exactly j of its vertices. Notice that if $V(\mathcal{C}_i) \cap X = \varnothing$, then $\mathcal{C}_i$ is either a path or a cycle and it is straightforward to compute f_i.

Assume that $V(\mathcal{C}_i) \cap X \neq \varnothing$. If $\mathcal{C}_i$ does not contain any cycles, then it is a tree. Otherwise, let C denote a cycle appearing in $\mathcal{C}_i$, in which case it holds that $C \cap X \neq \varnothing$, and by Lemma 7.10 it follows that $S \cap C = \varnothing$. In that case, we contract C to a single vertex v of weight equal to the sum of weights of the vertices of the cycle, that is, $\mathrm{w}(v) = \sum_{c \in C} \mathrm{w}(c)$, mark it as undeletable and add it to X. We repeat this exhaustively, resulting in a component which is a weighted tree. In that case one can easily adapt the DP algorithm for trees by Di Summa, Grosso, and Locatelli [272] to account for the non-deletability and the weights and compute f_i in polynomial time.

Finally, having computed for every component $\mathcal{C}_i \in \mathtt{cc}(G - (S \cap X))$ the function f_i, it suffices to use Lemma 7.1 in order to compute a $k_i \in [0, k - |S \cap X|]$ for every $\mathcal{C}_i$ so that $\sum k_i \leq k$ and $\sum f_i(k_i)$ is minimized, and verify whether $\sum f_i(k_i) \leq x$. Regarding the running time of our algorithm, guessing $S \cap X$ incurs an overhead of $2^{\mathcal{O}(\mathrm{ml})}$, while the rest of the steps require polynomial time. □

7.3.2. Vertex Integrity

The *vertex integrity* of a graph $G = (V, E)$, denoted $\mathrm{vi}(G)$, is the minimum integer k such that there is a vertex set $U \subseteq V$ with $|U| + \max_{C \in \mathtt{cc}(G-S)} |V(C)| \leq k$, where $\mathtt{cc}(G - S)$ denotes the set of connected components of $G - S$. Here we show that the parameterization by vertex integrity renders Critical Node Cut fixed-parameter tractable. Given the fact that the vertex integrity of a graph is upper-bounded by its vertex cover number, this improves over the known FPT algorithm parameterized by vertex cover number [268]. On the other hand, the vertex integrity of a graph is lower-bounded by its tree-depth, under which parameterization the problem remains W[1]-hard [169], so our result is in that sense tight.

Theorem 7.11. *Given an instance $\mathcal{I} = (G, k, x)$ of Critical Node Cut, there is an algorithm that decides $\mathcal{I}$ in time $\mathrm{vi}^{\mathcal{O}(\mathrm{vi}^2)} n^{\mathcal{O}(1)}$, where vi denotes the vertex integrity of G.*

Proof. Let $U \subseteq V(G)$ such that for all connected components $C \in \mathtt{cc}(G - S)$ it holds that $|U| + |V(C)| \leq \mathrm{vi}(G)$; such a set can be computed in time $\mathrm{vi}^{\mathcal{O}(\mathrm{vi})} n^{\mathcal{O}(1)}$ [104]. Furthermore,

let $\mathtt{cc}(G-U) = \{C^1, \ldots, C^m\}$ denote the connected components of $G-U$, where $m \in [0, n]$; by assumption, it holds that $|V(C^j)| \leq \mathrm{vi}$ for all $j \in [m]$.

On a high level, our algorithm consists of two separate steps. In Step 1, we guess the intersection of U with an optimal deletion set, as well as the way the non-deleted vertices of U are partitioned in connected components in the final graph. All remaining vertices of the optimal deletion set belong to components of $G-U$. Next, it suffices to encode the choices in the rest of the graph as an m-fold IP (see [200]) of a small number of global constraints and small coefficients in the right-hand side, and find a solution that indeed minimizes the number of pairs of connected vertices in the final graph, while respecting the guesses of Step 1.

Fix a deletion set $S^* \subseteq V(G)$ such that $|S^*| \leq k$ and the number of pairs of connected vertices in $G - S^*$ is minimum. We start by guessing the intersection of said deletion set with the modulator U, that is, let $S = U \cap S^*$. We further guess how $U \setminus S$ is partitioned into connected components in $G - S^*$, and let $V_1, \ldots, V_\ell$ denote this partition, where $\ell \leq \mathrm{vi}$. Notice that for $i \in [\ell]$, $G[V_i]$ is not necessarily connected, however it holds that if $uv \in E$ and $u, v \in U \setminus S$, then u and v must belong to the same connected component of $G - S^*$. Any guess $S, V_1, \ldots, V_\ell$ that adheres to the latter constraint, that is, no connected component of $G[U \setminus S]$ intersects more than one blocks of the partition, will be called *valid*.

For each valid guess, we will create an m-fold IP that computes the deletions from the sets $V(C^j)$, $j \in [m]$ that minimize the number of pairs of connected vertices from the remaining graph, under the assumption that the guesses so far are correct. From all the solutions we will return the one that minimizes the number of pairs of connected vertices. Prior to presenting the m-fold IP, notice that since we have guessed that S will be deleted, the number of deletions that we further do is at most $k - |S|$. We start by creating ℓ constants $n_i = |V_i|$, for $i \in [\ell]$.

Local constraints. Now we will consider each connected component C^j, $j \in [m]$ separately. Fix a $j \in [m]$. We say that the subset $X \subseteq V(C^j)$ is *invalid* if there exist $s, t \in U \setminus S$ such that (i) $s \in V_{i_1}$, $t \in V_{i_2}$, and $i_1 \neq i_2$, and (ii) s and t belong to the same connected component of $G[(U \setminus S) \cup (V(C^j) \setminus X)]$. We say that a subset of $V(C^j)$ is *valid* if it is not invalid. Notice that $\varnothing$ might be a valid subset of $V(C^j)$.

For any given C^j, $j \in [m]$, let $S^j_1, \ldots, S^j_\mu$, where $\mu \leq 2^{|V(C^j)|} \leq 2^{\mathrm{vi}}$, be an enumeration of all valid subsets of $V(C^j)$. Notice that the intersection of an optimal solution with $V(C^j)$ is exactly one of the sets $S^j_1, \ldots, S^j_\mu$. In order to force this, we will create a variable $I^j_q \in \{0, 1\}$ for each S^j_q that will indicate whether we select to delete the set S^j_q or not. We will also add the following constraint that forces the IP to select at most one of these sets:

$$\sum_{q \in [\mu]} I^j_q = 1.$$

Next, for each $q \in [\mu]$ and $i \in [\ell]$, we compute the constant $c^{j,q}_i$ which is equal to the number of vertices of $V(C^j) \setminus S^j_q$ that are connected to vertices of V_i in $G[V_i \cup (V(C^j) \setminus S^j_q)]$.

This indicates that $c_i^{j,q}$ vertices of $V(C^j)$ will be connected to V_i in the final graph if we delete S_q^j.

We further compute for each $q \in [\mu]$ the constant $p^{j,q}$ which is equal to the number of pairs of connected vertices of $V(C^j)$ that will appear in the final graph and which are not connected with any set V_i, $i \in [\ell]$. This can be done in time $\mathcal{O}(\mathrm{vi}^2)$ as the graph $G[(U \setminus S) \cup (V(C^j) \setminus X)]$ has at most vi vertices. Notice that $p^{j,q} \leq \binom{\mathrm{vi}}{2}$.

Notice that the constraints I_q^j, $q \in [\mu]$, along with the constants $c_i^{j,q}$, $q \in [\mu]$, $i \in [\ell]$, allow us to compute the number of vertices of C^j connected with each V_i in the final graph, without knowing which set S_q^j has been deleted. In particular, we know that the number of vertices that belong in C^j and will be connected with V_i, for any $i \in [\ell]$, is exactly

$$x_i^j = \sum_{q \in [\mu]} I_q^j c_i^{j,q}.$$

Similarly, we can compute the number of pairs of connected vertices in C^j after the deletions that are not connected with any set V_i, $i \in [\ell]$, which is

$$p^j = \sum_{q \in [\mu]} I_q^j p^{j,q}.$$

Finally, we introduce for each $q \in [\mu]$ a constant $d_q^j = |S_q^j|$ that accounts for the number of deletions made in C^j when the intersection of the deletion set with S^j is exactly S_q^j, that is, we set $d_q^j = |S_q^j|$. Consequently, regardless of the set we delete, the number of deletions in C^j is exactly

$$d^j = \sum_{q \in [\mu]} I_q^j d_q^j.$$

We remark that, for each $j \in [m]$, we have $\mathcal{O}(2^{\mathrm{vi}})$ variables and $\mathcal{O}(1)$ constraints. Also, all the coefficients are bounded by $\binom{\mathrm{vi}}{2}$. Furthermore, those can be considered as j sets of variables and constraints as, for the moment, there is no constraint that includes variables related to distinct connected components C^{j_1} and C^{j_2}, where $j_1 \neq j_2$.

Global constraints. In this part, we deal with the global variables and constraints. First we introduce the following constraint so that we force the number of deletions to be bounded:

$$\sum_{j \in [m]} d^j = \sum_{j \in [m]} \sum_{q \in [\mu]} I_q^j d_q^j = k - |S|.$$

Notice that $k - |S|$ is not a coefficient but a right-hand side constant, i.e., it is not multiplied by any variable.

Next, we create one variable x_i for each $i \in [\ell]$ which represents the number of vertices that appear in the connected components each V_i belongs to and we set

$$x_i = n_i + \sum_{j \in [m]} x_i^j = n_i + \sum_{j \in [m]} \sum_{q \in [\mu]} I_q^j c_i^{j,q}.$$

Lastly, we create a variable to account for the number of pairs of connected vertices which are not connected to any V_i, $i \in [\ell]$. Those are exactly:

$$p = \sum_{j \in [m]} p^j = \sum_{j \in [m]} \sum_{q \in [\mu]} I_q^j p^{j,q}.$$

Notice that the number of global constraints is $\mathcal{O}(\mathrm{vi})$ and the size of any coefficient is $\mathcal{O}(\mathrm{vi}^2)$.

It remains to define the objective function. In particular, we want to minimize the following:

$$p + \sum_{i \in [\ell]} \binom{x_i}{2}.$$

The above formulation is an m-fold IP with a separable convex objective function. Indeed, if we ignore the $\mathcal{O}(\mathrm{vi})$ global constraints, then we can partition the variables into m (disjoint) sets of size $2^{\mathcal{O}(\mathrm{vi})}$ and the constraints into m (disjoint) sets of size at most $\mathcal{O}(1)$ such that, the variables of the i-th variable set appear only in the i-th set of constraints. Additionally, all the coefficients are at most vi^2. Therefore, we can conclude that the m-fold IP runs in $\mathrm{vi}^{\mathcal{O}(\mathrm{vi}^2)} n^{\mathcal{O}(1)}$ time [200]. Therefore, it remains to prove that this returns a solution of the Critical Node Cut problem.

We argue that the IP computes a value that represent a feasible set of deletions. Notice that the IP always computes the number of connected pairs based on the assumption that the considered partition $V_1, \ldots, V_\ell$ of $U \setminus S$ satisfies the following two properties after the deletions:

1. for each $i \in [\ell]$, all vertices of V_i belong to the same connected component,
2. there is no connected component that intersects with more than one V_i, $i \in [\ell]$.

Although we have guaranteed the second condition (by considering only valid subsets of each C^j), we have no guarantee about the first. In order to see that the algorithm will indeed return the minimum number of connected pairs, it suffices to observe the following. Given the sets $S^1_{q_i}, \ldots, S^m_{q_m}$ we will delete from each component $C^1, \ldots, C^m$,

- if both conditions hold in $G - \left(S \cup \bigcup_{j=1}^m S^j_{q_j}\right)$, then the optimization function computes the actual number of connected pairs,
- if only the second condition holds in $G - \left(S \cup \bigcup_{j=1}^m S^j_{q_j}\right)$, then the optimization function computes a number of connected pairs greater than or equal to the number of connected pairs.

Therefore, when we have made the correct guesses, the algorithm returns the optimal solution while, in all other cases, the algorithm returns a value greater than or equal to a feasible solution (which cannot be smaller than the optimal). Finally, we need to check whether the number of connected pairs we computed is at most x.

Running time. It remains to argue about the running time of our algorithm. We first compute the set U in time $\mathrm{vi}^{\mathcal{O}(\mathrm{vi})}n^{\mathcal{O}(1)}$ [104]. Then we are guessing which vertices of U will be included in an optimal deletion set, as well as the way the non-deleted vertices of U are partitioned in connected components in the final graph. These guesses result in an $2^{\mathrm{vi}}\mathrm{vi}^{\mathrm{vi}}$ overhead. For each guess, we create an m-fold IP.

Note that each m-fold IP requires $(a{\cdot}r{\cdot}s)^{(r^2\cdot s+r\cdot s^2)}(m{\cdot}t{\cdot}L)^{\mathcal{O}(1)}$ time [110, Corollary 97].[6] In this expression, a denotes the largest coefficient, r denotes the number of global constraints, while s and t denote the number of constraints and the number of variables respectively, in each block of local constraints. Finally, L denotes the length of the binary encoding of the numerical input data of the m-fold IP. In our case we have that $a \leq \mathrm{vi}^2$, $r \leq \mathrm{vi}$, $s = \mathcal{O}(1)$, and $t \leq 2^{\mathrm{vi}}$. Furthermore, $m \leq n$ and $L = \Theta\big((r + m \cdot s) \cdot m \cdot t \log n\big)$. Therefore, each m-fold IP runs in $\mathrm{vi}^{\mathcal{O}(\mathrm{vi}^2)}n^{\mathcal{O}(1)}$ time. In total, the algorithm runs in $\mathrm{vi}^{\mathcal{O}(\mathrm{vi}^2)}n^{\mathcal{O}(1)}$ time. □

7.3.3. Modular-width

All our previous results are concerned with *sparse* classes of graphs. Here we prove that Critical Node Cut is fixed-parameter tractable parameterized by the modular-width of the input graph, thus improving over the analogous result for neighborhood diversity [267].

Theorem 7.12. *Given an instance $\mathcal{I} = (G, k, x)$ of Critical Node Cut, there is an algorithm that decides $\mathcal{I}$ in time $2^{\mathrm{mw}}n^{\mathcal{O}(1)}$, where* mw *denotes the modular-width of G.*

Proof. We first recall some necessary definitions. A *modular decomposition* of a graph G is a rooted tree T_G where any node b with c children is also associated with a graph H_b consisting of c vertices $v_1, \ldots, v_c$. Additionally, any node b of T_G represents a graph $G_b = (V_b, E_b)$ defined recursively:

- a leaf node represents an isolated vertex,
- a non-leaf node b with c children $b_1, \ldots, b_c$ represents the graph $G_b = (V_b, E_b)$ where $V_b = \bigcup_{i\in[c]} V_{b_i}$ and $E_b = \{u_iu_j : u_i \in V_{b_i},\ u_j \in V_{b_j} \text{ and } v_iv_j \in E(H_b)\} \cup \bigcup_{i\in[c]} E_{b_i}$.

Finally, the graph G_r, where r is the root of T_G, is isomorphic to G.

The *width* of a modular decomposition T_G is equal to the maximum number of children of any node in T_G. The *modular-width* of a graph G, denoted mw, is the minimum width among the modular decompositions of G. Notice that, by definition, T_G has exactly n leaves. Additionally, we can assume that there is no node in T_G with exactly one child; indeed, such a node would represent the same graph as its child node. Therefore, since T_G is a tree, it has at most $2n - 1$ nodes.

Let T_G be a modular decomposition of G computed in polynomial time [241]. We will perform dynamic programming over the nodes of T_G. Let b be a node of T_G. For the graph G_b, we will compute a function $f_b\colon [0, k] \to \mathbb{N}$ such that for $k' \in [0, |V_b| - 1]$,

[6] This result is a direct consequence of the results published in [171].

$f_b(k')$ is equal to the minimum number of pairs of connected vertices in G_b after k' vertex deletions. Notice that for any graph G there is no reason to delete more than $|V(G)| - 1$ vertices, thus we set $f_b(k') = +\infty$ for all $k' \in [|V_b|, k]$. This will also be useful later when we will need to guarantee that we are not deleting the whole set V_b. We explain how to compute recursively the function f_b.

If b is a leaf node of T_G, then the graph G_b is isomorphic to an isolated vertex, in which case G_b has no pairs of connected vertices. Consequently, we set $f_b(0) = 0$ and $f_b(k') = +\infty$ for all $k' \in [k]$.

Alternatively, it holds that b is a non-leaf node of T_G, and let $b_1, \ldots, b_\ell$, with $\ell \leq \mathrm{mw}$, denote the nodes of T_G that are children of b. Recall that b is related to a graph $H_b = (V_b, E_b)$ with $V(H_b) = \{v_1, \ldots, v_\ell\}$ and that in G_b, for all vertices $u_i \in V_{b_i}$ and $u_j \in V_{b_j}$ it holds that $u_i u_j \in E(G_b)$ if and only if $v_i v_j \in E(H_b)$.

To deal with this case we adopt an exhaustive approach. Fix $S \subseteq [\ell]$ and let $U = \bigcup_{i \in S} V_{b_i}$. We will compute the function $f_b^S \colon [0, |V_b| - 1] \to \mathbb{N}$ that, for $k' \in [0, |V_b| - 1]$, returns the minimum number of pairs of connected vertices remaining in G_b after k' vertex deletions under the assumptions that:

- all vertices of U are deleted, and
- no set V_{b_i}, $i \notin S$ is completely deleted.

Intuitively, set S indicates whether a set of vertices V_{b_i} is completely deleted or not. We set $f_b^S(k') = +\infty$ for all $k' < |U|$ as we cannot delete U with less than $|U|$ deletions.

We now consider values $k' \geq |U|$. Prior to explaining how to compute $f_b^S(k')$ we need the following definitions. Let $S' \subseteq [\ell] \setminus S$ such that, for each $i \in S'$ and $v \in V_{b_i}$ we have that $N_{G_b - U}(v) \subseteq V_{b_i}$. Notice that, by the definition of S', any vertex $v \in V_{b_i}$, $i \in S'$, does not share any edge with any vertex in $V_b - (U \cup V_{b_i})$. Let $U' = \bigcup_{i \in S'} V_{b_i}$. We also need to consider the connected components of $G_b - (U \cup U')$. Let $C_1, \ldots, C_m$, $m \leq \ell$, denote the vertex sets of the connected components of $G_b - (U \cup U')$. Due to the removal of U and U', we know that for any set $C \in \{C_1, \ldots, C_m\}$ there exist at least two integers $i, j \in [\ell] \setminus (S \cup S')$ such that $C \cap V_{b_i} \neq \varnothing$ and $C \cap V_{b_j} \neq \varnothing$.

Now, we are ready to explain how we compute the value $f_b^S(k')$, for $k' \geq |U|$. We know that $|U|$ of the deletions need to be used in order to delete the vertex set U. Assume that we know how the deletions are distributed between the modules V_{b_i}, $i \in S'$ and C_j, $j \in [m]$. In particular let k_i be the number of deletions from each module V_{b_i}, $i \in S$ and k'_j be the number of deletions from each connected component C_j, $j \in [m]$. Finally, notice that $k' = |U| + \sum_{i \in S'} k_i + \sum_{j \in [m]} k'_j$.

In order to compute $f_b^S(k')$, we first deal with each graph $G[C_j]$, $j \in [m]$, separately. Let $f_j \colon [k] \to \mathbb{N}$ be a function such that, for any $k'_j \in [k]$, $f_j(k'_j)$ equals the minimum number of pairs of connected vertices remaining in $G[C_j]$ after k'_j vertex deletions under the assumption that, for any child b_i, $i \in [\ell]$ of b such that $V_{b_i} \cap C_j \neq \varnothing$, V_{b_i} is not completely deleted. Notice that this assumption comes from the fact that we have previously guessed which modules will be completely deleted (set S, with the union of the vertex sets of the deleted modules denoted by U). Notice that, C_j is a connected

component of $G_b - (U \cup U')$ so it does not include vertices from any module which will be completely deleted.

Since for any $j \in [m]$ we have assumed that there is always at least one vertex of each module that remains in C_j after all deletions and $G[C_j]$ is connected, we have that after the deletions this component remains connected. Let μ_j be the number of modules that intersect C_j. If $k'_j > |C_j| - \mu_j$ it follows that we cannot respect the assumption that any module that intersects C_j will not be completely deleted, therefore we set $f_j(k'_j) = +\infty$ for all $k'_j > |C_j| - \mu_j$. For $k'_j \in [0, |C_j| - \mu_j]$, we set $f_j(k'_j) = \binom{|C_j|-k'_j}{2}$; indeed, by our assumption, C_j must remain connected after the deletions if $k'_j \in [0, |C_j| - \mu_j]$, thus $\binom{|C_j|-k'_j}{2}$ is the exact number of connected pairs remaining in C_j after k'_j deletions. Furthermore, if $f_j(k'_j) \neq +\infty$, there exists a deletion set $D \subseteq C_j$ such that $|D| = k'_j$ and $G[C_j \setminus D]$ has exactly $f_j(k'_j)$ pairs of connected vertices.

Now, let us consider pairs of connected vertices in $G_b - U$ that include vertices of V_{b_i}, for some $i \in S'$. Notice that by the definition of S', both vertices of such pairs belong to V_{b_i}. Since there are k_i deletions taking place in V_{b_i}, we can conclude that the minimum number of such pairs of connected vertices after k_i deletions is $f_{b_i}(k_i)$.

Therefore, we can conclude that

$$f_b^S(k) = \min_{k_i \geq 0, i \in S' \text{ and } k'_j \geq 0, j \in [m]} \left\{ \sum_{i \in S'} f_{b_i}(k_i) + \sum_{j \in [m]} f_j(k'_j) : k = |U| + \sum_{i \in S'} k_i + \sum_{j \in [m]} k'_j \right\}.$$

Notice that, as far as the value is not infinite, we can guarantee that there exists a solution that agrees with the restrictions implied by S.

To compute the function f_b assume that we have computed the functions f_b^S, for all $S \subseteq [\ell]$. Then we simply set $f_b(k') = \min_{S \subseteq [\ell]} f_b^S(k')$. Since we made all guesses $S \subseteq [\ell]$, we know that we have included the guess that agrees with the optimal solution. Let S^* be this guess. Notice that $f_b^{S^*}(k')$ is exactly equal to the minimum number of pairs of connected vertices in $G[V_b]$ after k' deletions. Also, for any other S', if $f_b^{S'}(k') \neq +\infty$, we have that $f_b^{S'}(k')$ equals the number of pairs that we can achieve if the deletions agree with S'. Since this is indeed a solution we have that $f_b^{S'}(k') \geq f_b^{S^*}(k')$. Therefore, we always return the minimum number of pairs.

Running time. We now argue about the running time of our algorithm. Let b denote a non-leaf node of T_G. We argue that we can compute f_b in time $2^{\mathrm{mw}} n^{\mathcal{O}(1)}$. Notice that after fixing the set S, we can compute S', the sets $C_1, \ldots C_m$, and the functions f_i, for all $i \in [m]$, in polynomial time. Also, all f_{b_i}, $i \in S$, have been recursively computed in the children nodes of b. Finally, by Lemma 7.1, we know that given the functions f_{b_i}, $i \in S'$ and f_j, $j \in [m]$, we can compute f_b^S in polynomial time. Notice that there are at most 2^{mw} functions f_b^S, since $S \subseteq [\ell] \subseteq [\mathrm{mw}]$. Since each function is computed in polynomial time, in total we need $2^{\mathrm{mw}} n^{\mathcal{O}(1)}$ time to compute them all. Furthermore, since $f_b(k') = \min_{S \subseteq [\ell]} f_b^S(k')$, we additionally need $2^{\mathrm{mw}} n$ time. Finally, observe that, after we compute the function f_r of the root node r, we can compute the minimum number of connected pairs after k deletions by checking the value $f_r(k)$. Since for every node of

the modular decomposition we need $2^{\mathrm{mw}}n^{\mathcal{O}(1)}$ time in the worst case, and the modular decomposition has at most $2n-1$ nodes, the total running time is $2^{\mathrm{mw}}n^{\mathcal{O}(1)}$. □

7.3.4. Clique-width

Theorem 7.2 implies that Critical Node Cut is W[1]-hard parameterized by clique-width. Here we show that for this parameterization, the problem belongs to XP. To obtain this result, we proceed by presenting a standard dynamic programming algorithm. On a high-level, as we go over the clique-width expression of the input graph, we consider a partition of the connected components of our graph based on the labels appearing in their vertices. For every such partitioning, we keep track of *both* the total size as well as the total number of connected pairs of vertices for the connected components belonging to the partitioning. As we prove, this amount of information is sufficient in order to solve Critical Node Cut.

Theorem 7.13. *There is an algorithm that, given a graph G along with an irredundant clique-width expression ψ of G of width* cw*, determines for all $k \in [0, n]$ the number of subsets $S \subseteq V(G)$ with $|S| = k$ that minimize the number of connected pairs of vertices of $G - S$, in time $n^{\mathcal{O}(2^{\mathrm{cw}})}$.*

Proof. Before describing our algorithm we start with some definitions and notations. Let $\mathcal{H}$ denote the set of all cw-labeled graphs generated by subexpressions of ψ. Furthermore, let $\mathcal{L} = 2^{[\mathrm{cw}]} \setminus \varnothing$ denote the set of all non-empty subsets of [cw]. We say that $\sigma = (\alpha, \beta)$ is a *signature*, where σ is a tuple consisting of functions $\alpha, \beta \colon \mathcal{L} \to \mathbb{N}$. Moreover, let Σ denote the set of all possible signatures, where $|\Sigma| = n^{\mathcal{O}(2^{\mathrm{cw}})}$. We say that the *label set* of a connected component of a labeled graph is the set of labels appearing in the component's vertices. For $H \in \mathcal{H}$ and $L \in \mathcal{L}$, let $\mathcal{C}_H^L \subseteq \mathtt{cc}(H)$ denote the set of connected components of H with label set L. Lastly, for $H \in \mathcal{H}$, we define the function $\mathrm{sgn}_H \colon 2^{V(H)} \to \Sigma$, such that for $S \subseteq V(H)$, the *H-signature of S* is the tuple $\mathrm{sgn}_H(S) = \sigma = (\alpha, \beta) \in \Sigma$, where for all $L \in \mathcal{L}$ it holds that

- $\alpha(L)$ is equal to the sum of the sizes of the connected components of $\mathcal{C}_{H-S}^L$,
- $\beta(L)$ is equal to the number of pairs of connected vertices belonging to connected components of $\mathcal{C}_{H-S}^L$.

Our algorithm proceeds by dynamic programming along ψ in a bottom-up fashion. In particular, for every $H \in \mathcal{H}$ it stores a table $\mathrm{DP}_H[\cdot, \cdot]$ where, as we show, for $k \in [0, n]$ and $\sigma \in \Sigma$, $\mathrm{DP}_H[k, \sigma]$ is equal to the number of subsets of $V(H)$ of size k and H-signature σ. We now proceed to describe how to populate the DP tables, as well as establish this invariant by induction.

Singleton $H = i(v)$. For $H = i(v)$, notice that $\mathrm{lab}_H^{-1}(i) = \{v\}$ and $\mathrm{lab}_H^{-1}(w) = \varnothing$ for all $w \in [\mathrm{cw}] \setminus \{i\}$. Consequently, for $k \in [0, n]$ and signature $\sigma = (\alpha, \beta) \in \Sigma$ we set

$$\mathrm{DP}_H[k, \sigma] = \begin{cases} 1 & \text{if } k = 0,\ \alpha \equiv \alpha_0[\{i\} \mapsto 1], \text{ and } \beta \equiv \beta_0, \\ 1 & \text{if } k = 1,\ \alpha \equiv \alpha_0, \text{ and } \beta \equiv \beta_0, \\ 0 & \text{otherwise,} \end{cases} \tag{7.3}$$

where for all $L \in \mathcal{L}$, $\alpha_0(L) = \beta_0(L) = 0$. By Equation (7.3) we can fill the table $\mathrm{DP}_H[\cdot, \cdot]$ in time $n^{\mathcal{O}(2^{\mathrm{cw}})}$. Furthermore, notice that $2^{V(H)} = \{\varnothing, \{v\}\}$ and one can easily verify that the invariant holds.

Disjoint union, $H = H_1 \oplus H_2$. We define a function $h\colon \Sigma \times \Sigma \to \Sigma$ such that, for $\sigma_1 = (\alpha_1, \beta_1)$ and $\sigma_2 = (\alpha_2, \beta_2)$, $h(\sigma_1, \sigma_2) = \sigma = (\alpha, \beta)$ where for all $L \in \mathcal{L}$ it holds that

- $\alpha(L) = \alpha_1(L) + \alpha_2(L)$,
- $\beta(L) = \beta_1(L) + \beta_2(L)$.

For a fixed size $k \in [0, n]$ and signature $\sigma \in \Sigma$, we set the value of $\mathrm{DP}_H[k, \sigma]$ to be

$$\mathrm{DP}_H[k, \sigma] = \sum_{k_1 + k_2 = k} \sum_{(\sigma_1, \sigma_2) \in h^{-1}(\sigma)} \mathrm{DP}_{H_1}[k_1, \sigma_1] \cdot \mathrm{DP}_{H_2}[k_2, \sigma_2]. \tag{7.4}$$

If $h^{-1}(\sigma) = \varnothing$ we set $\mathrm{DP}_H[k, \sigma] = 0$. Notice that computing $h^{-1}(\sigma)$ requires $n^{\mathcal{O}(2^{\mathrm{cw}})}$ time, thus by Equation (7.4) we can fill the table $\mathrm{DP}_H[\cdot, \cdot]$ in time $n^{\mathcal{O}(2^{\mathrm{cw}})}$.

▷ Claim 7.14. Let $H = H_1 \oplus H_2$. Assume that for $H_i \in \{H_1, H_2\}$, $k \in [0, n]$, and $\sigma \in \Sigma$, $\mathrm{DP}_{H_i}[k, \sigma]$ is equal to the number of subsets of $V(H_i)$ of size k and H_i-signature σ. Then it holds that $\mathrm{DP}_H[k, \sigma]$ is equal to the number of subsets of $V(H)$ of size k and H-signature σ.

Proof of the claim. Observe that $V(H) = V(H_1) \cup V(H_2)$ and $E(H) = E(H_1) \cup E(H_2)$. Thus, for every $S \subseteq V(H)$, it holds that $S = S_1 \cup S_2$ where $S_i = S \cap V(H_i)$ for $i \in \{1, 2\}$. Furthermore, notice that the connected components of $H - S$ are exactly the connected components of $H_1 - S_1$ and $H_2 - S_2$, each with the same label set. Consequently, for every $L \in \mathcal{L}$ it holds that

- $\alpha(L) = \alpha_1(L) + \alpha_2(L)$,
- $\beta(L) = \beta_1(L) + \beta_2(L)$,

where $\sigma = (\alpha, \beta)$ is the H-signature of S, and $\sigma_i = (\alpha_i, \beta_i)$ is the H_i-signature of S_i for $i \in \{1, 2\}$. This implies that $\sigma = h(\sigma_1, \sigma_2)$. The claim follows by Equation (7.4) and by the fact that there are $\mathrm{DP}_{H_1}[k_1, \sigma_1]$ ways to choose a subset of size k_1 with H_1-signature σ_1, and $\mathrm{DP}_{H_2}[k_2, \sigma_2]$ ways to choose a subset of size k_2 with H_2-signature σ_2, for every pair $(\sigma_1, \sigma_2) \in h^{-1}(\sigma)$ and every pair (k_1, k_2) such that $k = k_1 + k_2$. ◁

Relabeling, $H = \rho_{i\to j}(H')$. Observe that a connected component with label set $L \in \mathcal{L}$ in H has label set either L, $L\cup\{i\}$, or $L\cup\{i\}\setminus\{j\}$ in H'. For each pair of distinct $i, j \in [\mathrm{cw}]$ we define a function $f_{i\to j}\colon \Sigma \to \Sigma$ such that, for $\sigma' = (\alpha', \beta')$, $f_{i\to j}(\sigma') = \sigma = (\alpha, \beta)$ where for all $L \in \mathcal{L}$ it holds that

- if $i \in L$, then $\alpha(L) = \beta(L) = 0$,
- if $i, j \notin L$, then $\alpha(L) = \alpha'(L)$ and $\beta(L) = \beta'(L)$,
- if $i \notin L$ and $j \in L$, then it holds that
  - $\alpha(L) = \alpha'(L) + \alpha'(L \cup \{i\}) + \alpha'(L \cup \{i\} \setminus \{j\})$,
  - $\beta(L) = \beta'(L) + \beta'(L \cup \{i\}) + \beta'(L \cup \{i\} \setminus \{j\})$.

For a fixed size $k \in [0, n]$ and signature $\sigma \in \Sigma$, we set the value of $\mathrm{DP}_H[k, \sigma]$ to be

$$\mathrm{DP}_H[k, \sigma] = \sum_{\sigma' \in f^{-1}_{i\to j}(\sigma)} \mathrm{DP}_{H'}[k, \sigma']. \tag{7.5}$$

If $f^{-1}_{i\to j}(\sigma) = \varnothing$ we set $\mathrm{DP}_H[k, \sigma] = 0$. Notice that computing $f^{-1}_{i\to j}(\sigma)$ requires $n^{\mathcal{O}(2^{\mathrm{cw}})}$ time, thus by Equation (7.5) we can fill the table $\mathrm{DP}_H[\cdot, \cdot]$ in time $n^{\mathcal{O}(2^{\mathrm{cw}})}$.

▷ Claim 7.15. Let $H = \rho_{i\to j}(H')$. Assume that for $k \in [0, n]$ and $\sigma \in \Sigma$, $\mathrm{DP}_{H'}[k, \sigma]$ is equal to the number of subsets of $V(H')$ of size k and H'-signature σ. Then it holds that $\mathrm{DP}_H[k, \sigma]$ is equal to the number of subsets of $V(H)$ of size k and H-signature σ.

Proof of the claim. Observe that $V(H) = V(H')$ and $E(H) = E(H')$. Furthermore, notice that for all $S \subseteq V(H)$, the connected components of $H - S$ are exactly the connected components of $H' - S$, albeit with potentially different label sets. Let $S \subseteq V(H)$ such that $\mathrm{sgn}_H(S) = \sigma = (\alpha, \beta)$ and $\mathrm{sgn}_{H'}(S) = \sigma' = (\alpha', \beta')$. We argue that $f_{i\to j}(\sigma') = \sigma$. Given that each subset S has a unique H'-signature, Equation (7.5) and the fact that there are $\mathrm{DP}_{H'}[k, \sigma']$ ways to choose a subset of size k with H'-signature σ', this implies the claim.

Let $L \in \mathcal{L}$. If $i \in L$, then since $\mathrm{lab}^{-1}_H(i) = \varnothing$, it holds that no connected component of $H - S$ has label set L, thus $\alpha(L) = \beta(L) = 0$. If $i, j \notin L$, then the connected components of $H - S$ with label set L are exactly the connected components of $H' - S$ with label set L, thus $\alpha(L) = \alpha'(L)$ and $\beta(L) = \beta'(L)$. Lastly, if $i \notin L$ and $j \in L$, then the connected components of $H - S$ with label set L are exactly the connected components of $H' - S$ with label set L, $L\cup\{i\}$, and $L\cup\{i\}\setminus\{j\}$. Thus, $\alpha(L) = \alpha'(L) + \alpha'(L\cup\{i\}) + \alpha'(L\cup\{i\}\setminus\{j\})$ and $\beta(L) = \beta'(L) + \beta'(L\cup\{i\}) + \beta'(L\cup\{j\}\setminus\{j\})$. It follows that $f_{i\to j}(\sigma') = \sigma$, and this concludes the proof. ◁

Joining labels with edges, $H = \eta_{i,j}(H')$. For each pair of distinct $i, j \in [\mathrm{cw}]$ we define a function $g_{i,j}\colon \Sigma \to \Sigma$ as follows. Let for $\sigma' = (\alpha', \beta')$, $g_{i,j}(\sigma') = \sigma = (\alpha, \beta)$. We consider two cases. First, assume that either $\sum_{L\in\mathcal{L}_i} \alpha'(L) = 0$ or $\sum_{L\in\mathcal{L}_j} \alpha'(L) = 0$, where for $z \in \{i, j\}$, $\mathcal{L}_z = \{L \in \mathcal{L} : z \in L\}$ denotes the label sets containing label z. In that

case, we set $\sigma = \sigma'$. Otherwise, let $\mathcal{L}_{i,j}^{\sigma'} = \{L \in \mathcal{L} : \{i,j\} \cap L \neq \varnothing \text{ and } \alpha'(L) \neq 0\}$ and $L_{i,j}^{\sigma'} = \bigcup_{L \in \mathcal{L}_{i,j}^{\sigma'}} L$. For all $L \in \mathcal{L}$ we set the values of $\alpha(L)$ and $\beta(L)$ as follows:

- if $i,j \notin L$, then $\alpha(L) = \alpha'(L)$ and $\beta(L) = \beta'(L)$,
- if $\{i,j\} \cap L \neq \varnothing$ and $L \neq L_{i,j}^{\sigma'}$, then $\alpha(L) = \beta(L) = 0$,
- if $L = L_{i,j}^{\sigma'}$, then $\alpha(L) = \sum_{L \in \mathcal{L}_{i,j}^{\sigma'}} \alpha'(L)$ and $\beta(L) = \binom{\alpha(L)}{2}$.

For a fixed size $k \in [0,n]$ and signature $\sigma \in \Sigma$, we set the value of $\mathrm{DP}_H[k,\sigma]$ to be

$$\mathrm{DP}_H[k,\sigma] = \sum_{\sigma' \in g_{i,j}^{-1}(\sigma)} \mathrm{DP}_{H'}[k,\sigma']. \tag{7.6}$$

If $g_{i,j}^{-1}(\sigma) = \varnothing$ we set $\mathrm{DP}_H[k,\sigma] = 0$. Notice that computing $g_{i,j}^{-1}(\sigma)$ requires $n^{\mathcal{O}(2^{\mathrm{cw}})}$ time, thus by Equation (7.6) we can fill the table $\mathrm{DP}_H[\cdot,\cdot]$ in time $n^{\mathcal{O}(2^{\mathrm{cw}})}$.

▷ Claim 7.16. Let $H = \eta_{i,j}(H')$. Assume that for $k \in [0,n]$ and $\sigma \in \Sigma$, $\mathrm{DP}_{H'}[k,\sigma]$ is equal to the number of subsets of $V(H')$ of size k and H'-signature σ. Then it holds that $\mathrm{DP}_H[k,\sigma]$ is equal to the number of subsets of $V(H)$ of size k and H-signature σ.

Proof of the claim. Observe that $V(H) = V(H')$, each vertex has the same label in both H and H', and $E(H) = E(H') \cup \{uv : u \in \mathrm{lab}_{H'}^{-1}(i), v \in \mathrm{lab}_{H'}^{-1}(j)\}$. Let $S \subseteq V(H)$ such that $\mathrm{sgn}_H(S) = \sigma = (\alpha,\beta)$ and $\mathrm{sgn}_{H'}(S) = \sigma' = (\alpha',\beta')$. We argue that $g_{i,j}(\sigma') = \sigma$. Given that each subset S has a unique H'-signature, Equation (7.6) and the fact that there are $\mathrm{DP}_{H'}[k,\sigma']$ ways to choose a subset of size k with H'-signature σ', this implies the claim. Let $L \in \mathcal{L}$. We will consider multiple cases.

First, assume that either $\sum_{L \in \mathcal{L}_i} \alpha'(L) = 0$ or $\sum_{L \in \mathcal{L}_j} \alpha'(L) = 0$, where for $z \in \{i,j\}$, $\mathcal{L}_z = \{L \in \mathcal{L} : z \in L\}$ denotes the label sets containing label z. In that case, it follows that either $\mathrm{lab}_{H'}^{-1}(i) \subseteq S$ or $\mathrm{lab}_{H'}^{-1}(j) \subseteq S$. Consequently, no new edges are added in $H - S$ compared to $H' - S$, thus, for all $L \in \mathcal{L}$, it holds that $\alpha(L) = \alpha'(L)$ and $\beta(L) = \beta'(L)$, which implies that $\sigma = \sigma'$.

Alternatively it holds that $\sum_{L \in \mathcal{L}_i} \alpha'(L) \neq 0$ and $\sum_{L \in \mathcal{L}_j} \alpha'(L) \neq 0$. If $i,j \notin L$, then the connected components of $H - S$ with label set L are exactly the connected components of $H' - S$ with label set L, thus $\alpha(L) = \alpha'(L)$ and $\beta(L) = \beta'(L)$ follows. To handle the remaining cases, we let $\mathcal{L}_{i,j}^{\sigma'} = \{L \in \mathcal{L} : \{i,j\} \cap L \neq \varnothing \text{ and } \alpha'(L) \neq 0\}$ and $L_{i,j}^{\sigma'} = \bigcup_{L \in \mathcal{L}_{i,j}^{\sigma'}} L$. Since there are vertices u and v in $H' - S$ with labels i and j, respectively, it follows that in $H - S$, all vertices of label i and all vertices of label j belong to the same connected component. Furthermore, the label set of that connected component is exactly $L_{i,j}^{\sigma'}$, as it contains all labels of connected components of $H' - S$ with label set in $\mathcal{L}_{i,j}^{\sigma'}$. As for its size, it holds that $\alpha(L_{i,j}^{\sigma'}) = \sum_{L \in \mathcal{L}_{i,j}^{\sigma'}} \alpha'(L)$, from which we can infer the number of pairs of connected vertices $\beta(L_{i,j}^{\sigma'}) = \binom{\alpha(L_{i,j}^{\sigma'})}{2}$. For any other label set $L \in \mathcal{L}$ with $\{i,j\} \cap L \neq \varnothing$ and $L \neq L_{i,j}^{\sigma'}$, it holds that $\alpha(L) = \beta(L) = 0$ as either (i) $\alpha'(L) = \beta'(L) = 0$, or (ii) the connected components of $\mathcal{C}_{H'-S}^{L}$ are all connected in

$H - S$ and are part of the single connected component of label set $L_{i,j}^{\sigma'}$ in $H - S$. It follows that $g_{i,j}(\sigma') = \sigma$, and this concludes the proof. ◁

Correctness follows by induction and Claims 7.14 to 7.16. As for the running time, notice that for every $H \in \mathcal{H}$ the table $\mathrm{DP}_H[\cdot,\cdot]$ is filled in time $n^{\mathcal{O}(2^{\mathrm{cw}})}$. Since $|\mathcal{H}| = n^{\mathcal{O}(1)}$, the overall running time of our algorithm is $n^{\mathcal{O}(2^{\mathrm{cw}})}$. Finally, notice that for any $S \subseteq V(G)$ of size $k \in [0, n]$ and G-signature $\sigma = (\alpha, \beta)$, it holds that $\mathrm{cp}(G - S) = \sum_{L \in \mathcal{L}} \beta(L)$. Consequently, by iterating over all signatures $\sigma \in \Sigma$, we can determine the minimum value of $\mathrm{cp}(G - S)$ over all subsets $S \subseteq V(G)$ of size k, as well as the number of such subsets achieving this minimum. This concludes the proof. □

7.4. FPT Approximation Scheme

Given the fact that, as evidenced by Theorem 7.2, Critical Node Cut remains W[1]-hard even under severe structural parameterizations, in this section we aim to bypass this computational hardness by adding approximation into the mix. In particular, we design an efficient FPT-AS for the parameterization by treewidth by modifying the standard $n^{\mathcal{O}(\mathrm{tw})}$ DP algorithm [3] and making use of a technique introduced by Lampis [207].

Theorem 7.17. *There is an algorithm which, for all $\varepsilon > 0$, when given as input a graph G of treewidth* tw *returns in time $(\mathrm{tw}/\varepsilon)^{\mathcal{O}(\mathrm{tw})} n^{\mathcal{O}(1)}$ a set $S \subseteq V(G)$ of size at most k such that $\mathrm{cp}(G - S) \leq (1 + \varepsilon) \cdot \mathrm{cp}(G - S')$ for all $S' \subseteq V(G)$ of size $|S'| \leq k$, for all $k \leq n$.*

Proof. We first describe the main idea behind our algorithm. On a high level, we aim to develop a DP which, while traversing the tree decomposition, keeps track of the sizes of any *active* components (those whose vertices intersect the bag), while for the rest of the components (i.e., the *inactive* ones) there exists a variable on which we account for their number of pairs of connected vertices. To this end, assuming that the exact size of an active component is c, our DP stores a value $\hat{c} \leq (1 + \varepsilon') \cdot c$ where ε' is such that $\binom{\hat{c}}{2} \leq (1 + \varepsilon) \cdot \binom{c}{2}$. Notice that for this to hold, $c = 1$ implies that $\hat{c} = 1$.

Lemma 7.18. *Let $0 < \varepsilon < 1$ and $c \geq 2$. For $\varepsilon' = \varepsilon/4$ it holds that if $\hat{c} \leq (1 + \varepsilon') \cdot c$, then $\binom{\hat{c}}{2} \leq (1 + \varepsilon) \cdot \binom{c}{2}$.*

Proof. It suffices to show that

$$\begin{aligned} \frac{(1+\varepsilon') \cdot c \cdot ((1+\varepsilon') \cdot c - 1)}{2} \leq \frac{(1+\varepsilon) \cdot c \cdot (c-1)}{2} &\iff \\ (1+\varepsilon')^2 \cdot c - (1+\varepsilon') \leq (1+\varepsilon) \cdot c - (1+\varepsilon) &\iff \\ c \cdot (\varepsilon - 2\varepsilon' - (\varepsilon')^2) \geq \varepsilon - \varepsilon'. \end{aligned}$$

Let $\varepsilon' = \varepsilon/x$ for some $x > 1$. In that case multiplying everything with $x^2 > 0$ in the previous inequality gives

$$\begin{aligned} c \cdot (x^2 \cdot \varepsilon - 2x \cdot \varepsilon - \varepsilon^2) \geq x^2 \cdot \varepsilon - x \cdot \varepsilon &\iff \\ c \cdot (x^2 - 2x - \varepsilon) \geq x^2 - x. \end{aligned}$$

Since $\varepsilon < 1$, it holds that $x^2 - 2x - \varepsilon > x^2 - 2x - 1$, thus, since $c \geq 0$, it suffices to show that $c \cdot (x^2 - 2x - 1) \geq x^2 - x$ where $x^2 - 2x - 1 > 0$ for all $x \geq 3$. In that case, assuming that $x \geq 3$, it suffices to show that

$$c \geq \frac{x^2 - x}{x^2 - 2x - 1}.$$

Since $c \geq 2$, it suffices to have that $\frac{x^2-x}{x^2-2x-1} \leq 2$, which is true for all $x \geq 4$. □

Consequently, it suffices to present a dynamic program which correctly stores the size of any singleton active component, while for the rest of active components it allows for $(1+\varepsilon')$-approximate values on their sizes. In that case, the number of pairs of connected vertices accounted for every connected component is a $(1+\varepsilon)$-approximation, and since we sum over those, the final value has a $(1+\varepsilon)$-approximation ratio as well. In the rest of the proof we present a DP that does exactly as required.

Definitions and Notation. We assume familiarity with the definition and usual notation for treewidth (see e.g. [92, Section 7]). Let $\mathcal{T}$ denote the nice tree decomposition and X_t denote the bag of node t, while h_t denotes its height. We denote by $\mathcal{T}_t$ the subtree of $\mathcal{T}$ rooted at t, and by $G[\mathcal{T}_t]$ the subgraph due to the subtree $\mathcal{T}_t$ of the tree decomposition. For a node t, we denote by $X_t^{\downarrow}$ the union of the bags of the nodes of $\mathcal{T}_t$. A *partition* P of a set L is a collection of non-empty subsets of L that are pairwise non-intersecting and such that $\bigcup_{p \in P} p = L$; each set in P is called a *block* of P. The set of partitions of a finite set L is denoted by $\Pi(L)$, and $(\Pi(L), \sqsubseteq)$ forms a lattice where $P \sqsubseteq Q$ if for each block p of P there is a block q of Q with $p \subseteq q$. The join operation of this lattice is denoted by $\sqcup$. For example, we have $\{\{1,2\},\{3,4\},\{5\}\} \sqcup \{\{1\},\{2,3\},\{4\},\{5\}\} = \{\{1,2,3,4\},\{5\}\}$. Let $|P|$ denote the number of blocks of a partition P. For $P \in \Pi(L)$ and $X \subseteq L$, let $P_{\downarrow X} \in \Pi(X)$ be the partition $\{p \cap X : p \in P\} \setminus \{\varnothing\}$, and for a set Y, let $P_{\uparrow Y} \in \Pi(L \cup Y)$ be the partition $P \cup \left(\bigcup_{y \in Y \setminus L} \{\{y\}\}\right)$.

We are now ready to describe our algorithm. First we compute a nice tree decomposition of G of width at most $2\mathrm{tw}+1$ and at most $\mathcal{O}(n\mathrm{tw})$ bags, which can be computed in time $2^{\mathcal{O}(\mathrm{tw})} \cdot n$ [197]. Next we apply a result of Bodlaender and Hagerup [42, Lemma 2.2] to said nice tree decomposition, which allows us to edit it into a tree decomposition of height $\mathcal{O}(\log n)$, while the width remains $\mathcal{O}(\mathrm{tw})$; we further edit it so that it is *nice*, in which case the height becomes $h = \mathcal{O}(\mathrm{tw} \log n)$. Let $\delta = \varepsilon'/(2h)$ and define $B = \{0\} \cup \{(1+\delta)^j : j \in \mathbb{N}_0 \text{ and } (1+\delta)^j \leq (1+\varepsilon') \cdot n\}$. Informally, B is the set of rounded values that are used to approximate the sizes of the active components.

Our algorithm proceeds by dynamic programming along the nodes of $\mathcal{T}$ in a bottom-up fashion. For every node t we define the *pseudo-signature space* $\hat{\Sigma}_t$ which contains tuples $(Z, k, P, \hat{c})$ where $Z \subseteq X_t$, $k \in [0, n]$, $P \in \Pi(Z)$ defines a partition of the vertices of Z, and $\hat{c}\colon P \to B$ is a function from the blocks of P to B. We say that a set $S \subseteq X_t^{\downarrow}$ has *pseudo-signature* (in node t) $\hat{\sigma}_t = (Z, k, P, \hat{c}) \in \hat{\Sigma}_t$ if and only if

1. $Z = X_t \setminus S$,

2. $|S \setminus X_t| = k$,

3. P describes the partition of the vertices of Z into connected components of $G[\mathcal{T}_t] - S$,

4. for all $p \in P$, $\hat{c}(p)$ is the minimum element of B that upper-bounds the size of the component of $G[\mathcal{T}_t] - S$ that contains the vertices of p, without accounting for the vertices in X_t.

In an analogous way, we define the *signature space* Σ_t and the *signature* $\sigma_t = (Z, k, P, c)$ of a set $S \subseteq X_t^{\downarrow}$ (in node t), with the only difference being that $c\colon P \to [0, n]$ stores the *exact* size of the active components, without accounting for the vertices in X_t. We further define for all $t \in V(\mathcal{T})$ the functions $\mathrm{sgn}_t\colon 2^{X_t^{\downarrow}} \to \Sigma_t$ and $\hat{\mathrm{sgn}}_t\colon 2^{X_t^{\downarrow}} \to \hat{\Sigma}_t$ that map a set $S \subseteq X_t^{\downarrow}$ to its signature and pseudo-signature in node t respectively. For a node t, we say that σ_t and $\hat{\sigma}_t$ are *compatible* when they only differ in the last entry of the tuple, that is, the only difference is between c and $\hat{c}$. We say that a pair (σ_t, d) is an *exact solution* if there exists a set $S \subseteq X_t^{\downarrow}$, which we say that *realizes* the exact solution, such that $\mathrm{sgn}_t(S) = \sigma_t$, and the number of pairs of connected vertices in the *inactive* components of $G[\mathcal{T}_t] - S$ is equal to d. Analogously, we say that a pair $(\hat{\sigma}_t, \hat{d})$ is an *approximate solution* if there exists a set $S \subseteq X_t^{\downarrow}$, which we say that *realizes* it, such that $\hat{\mathrm{sgn}}_t(S) = \hat{\sigma}_t$, and the number of pairs of connected vertices in the *inactive* components of $G[\mathcal{T}_t] - S$ is equal to d, with $d \leq \hat{d} \leq (1+\varepsilon) \cdot d$.

For every node $t \in V(\mathcal{T})$ the algorithm stores a table $\mathrm{DP}_t[\cdot, \cdot]$ where, for $\hat{\sigma}_t \in \hat{\Sigma}_t$ and $\hat{d} \in [0, (1+\varepsilon) \cdot \binom{n}{2}]$, $\mathrm{DP}_t[\hat{\sigma}_t, \hat{d}]$ is a boolean variable indicating whether there exists an approximate solution $(\hat{\sigma}_t, \hat{d})$. Recall that the height h_t of a node t of the decomposition is the largest distance from the node to a leaf rooted in its sub-tree. Leaves are at height 0 and the root is at height h. In order to show that the algorithm indeed produces a $(1+\varepsilon)$-approximate solution, we want to maintain the following invariant:

- For each node t, if there exists an exact solution $(\sigma_t = (Z, k, P, c), d)$, then there exists a pseudo-signature $\hat{\sigma}_t = (Z, k, P, \hat{c}) \in \hat{\Sigma}_t$ compatible with σ_t such that for all $p \in P$, $c(p) \leq \hat{c}(p) \leq (1+\delta)^{h_t} \cdot c(p)$ and a value $\hat{d}$ with $d \leq \hat{d} \leq (1+\varepsilon) \cdot d$, such that $\mathrm{DP}_t[\hat{\sigma}_t, \hat{d}] = \mathsf{true}$.

- For each node t, if $\mathrm{DP}_t[\hat{\sigma}_t, \hat{d}] = \mathsf{true}$, where $\hat{\sigma}_t = (Z, k, P, \hat{c})$, then there exists an exact solution $(\sigma_t = (Z, k, P, c), d)$ where σ_t is compatible with $\hat{\sigma}_t$, $d \leq \hat{d}$, and for all $p \in P$, $c(p) \leq \hat{c}(p)$.

On a high-level, our approximation algorithm slightly overestimates both the size of the active components (by a factor of $(1+\delta)^{h_t}$) and the number of pairs of connected vertices in inactive components of the graph (by a factor of $(1+\varepsilon)$): on the one hand, if the approximate algorithm computes some solution, an exact solution that is at least that good exists; on the other hand, if an exact solution exists, the algorithm will compute one that satisfies the approximation guarantees. We now proceed to describe how to populate the DP tables, as well as establish this invariant by induction.

Leaf node. If t is a leaf node, then $X_t = \varnothing$ and $\hat{\Sigma}_t = \{(\varnothing, 0, \{\{\varnothing\}\}, [\{\varnothing\} \mapsto 0])\}$. In that case, it follows that

$$\mathrm{DP}_t[\hat{\sigma}_t, \hat{d}] = \begin{cases} \mathsf{true} & \text{if } \hat{\sigma}_t \in \hat{\Sigma}_t \text{ and } \hat{d} = 0, \\ \mathsf{false} & \text{otherwise.} \end{cases}$$

Notice that the invariant trivially holds for Leaf nodes.

Introduce node. Suppose t is an introduce node with child t' such that $X_t = X_{t'} \cup \{v\}$ for some $v \notin X_{t'}$. Let $\hat{\sigma}_t = (Z, k, P, \hat{c}) \in \hat{\Sigma}_t$ and consider two cases. If $v \notin Z$, then we set $\mathrm{DP}_t[\hat{\sigma}_t, \hat{d}] = \mathrm{DP}_{t'}[\hat{\sigma}_t, \hat{d}]$ for all $\hat{d} \in [0, (1+\varepsilon) \cdot \binom{n}{2}]$. Otherwise it holds that $v \in Z$ and we set

$$\mathrm{DP}_t[\hat{\sigma}_t, \hat{d}] = \bigvee_{\hat{\sigma}_{t'} \in \mathcal{S}^I_{\hat{\sigma}_t}} \mathrm{DP}_{t'}[\hat{\sigma}_{t'}, \hat{d}],$$

where $\mathcal{S}^I_{\hat{\sigma}_t} \subseteq \hat{\Sigma}_{t'}$ such that $\hat{\sigma}_{t'} = (Z', k', P', \hat{c}') \in \mathcal{S}^I_{\hat{\sigma}_t}$ if

1. $Z = Z' \cup \{v\}$,
2. $k = k'$,
3. P' is a partition of Z' such that merging all its blocks that contain vertices incident with v and adding v to the latter results to the partition P of Z, that is, formally we have $P = P'_{\uparrow\{v\}} \sqcup \{\{v\} \cup \{u \in Z' : u \in N_G(v)\}\}_{\uparrow Z}$,
4. $\hat{c}'$ is a cost function on the blocks of P' such that, for $\mathcal{P}' = \{p' \in P' : p' \cap N_G(v) \neq \varnothing\}$,[7] we have that (i) $\hat{c}(p) = \hat{c}'(p)$ for all $p \in P' \setminus \mathcal{P}'$, and (ii) $\hat{c}(p_v)$ is equal to the smallest value in B that upper-bounds $\sum_{p' \in \mathcal{P}'} \hat{c}'(p')$, where $p_v \in P$ such that $v \in p_v$.

Lemma 7.19. *Let t be an introduce node, with t' denoting its child node. Assume that the invariant holds for node t'. Then it also holds for node t.*

Proof. Suppose t is an introduce node with child t' such that $X_t = X_{t'} \cup \{v\}$ for some $v \notin X_{t'}$ and $h_t = h_{t'} + 1$. By induction, for an exact solution $(\sigma_{t'} = (Z', k', P', c'), d')$ we have calculated an approximate solution $(\hat{\sigma}_{t'} = (Z', k', P', \hat{c}'), \hat{d}')$ with $\hat{c}'(p') \leq (1+\delta)^{h_{t'}} \cdot c'(p')$ for all $p' \in P'$ and $\hat{d}' \leq (1+\varepsilon) \cdot d'$.

Let $(\sigma_t = (Z, k, P, c), d)$ denote an exact solution for node t, and $S \subseteq X_t^{\downarrow}$ a set that realizes it. If $v \notin Z$, it follows that $(\sigma_{t'}, d)$ is an exact solution for node t', where $\sigma_{t'} = \sigma_t = \mathrm{sgn}_{t'}(S)$, and by induction it easily follows that $(\hat{\sigma}_{t'}, \hat{d})$ is an approximate solution for node t that respects the approximation guarantees. Alternatively, it holds that $v \in Z$, in which case it follows that there exists an exact solution $(\sigma_{t'}, d)$ in node t' realized by $S \setminus \{v\}$. By induction, we have calculated the approximate solution $(\hat{\sigma}_{t'}, \hat{d})$, and let $\hat{\sigma}_t = (Z, k, P, \hat{c}) \in \hat{\Sigma}_t$ such that $\hat{\sigma}_{t'} \in \mathcal{S}^I_{\hat{\sigma}_t}$. We argue that $(\hat{\sigma}_t, \hat{d})$ is an approximate

[7] That is, $\mathcal{P}'$ contains all the blocks of P' that contain vertices incident with v.

solution that respects the guarantees. To see this, it suffices to argue about the size of the active component that contains v. Let $p_v \in P$ with $v \in p_v$, and let $p'_1, \ldots, p'_w \in P'$ denote the blocks of P' that contain vertices belonging to $N_G(v)$. It holds that

$$\begin{aligned}\hat{c}(p_v) &= (1+\delta)^{\lceil \log_{(1+\delta)}(\hat{c'}(p'_1)+\ldots+\hat{c'}(p'_w))\rceil} \\ &\leq (1+\delta)\cdot(\hat{c'}(p'_1)+\ldots+\hat{c'}(p'_w)) \\ &\leq (1+\delta)\cdot(1+\delta)^{h_{t'}}\cdot(c'(p'_1)+\ldots+c'(p'_w)) \\ &= (1+\delta)^{h_t}\cdot c(p_v),\end{aligned}$$

where $\log_{(1+\delta)}(\cdot)$ denotes the logarithm base $(1+\delta)$, and the second inequality is due to the induction hypothesis.

As for the second direction of the invariant, it is straightforward to argue about its validity by induction. □

Forget node. Suppose t is a forget node with child t' such that $X_t = X_{t'} \setminus \{v\}$ for some $v \in X_{t'}$. For all $\hat{\sigma}_t = (Z, k, P, \hat{c}) \in \hat{\Sigma}_t$ and $\hat{d} \in [0, (1+\varepsilon)\cdot\binom{n}{2}]$ we set

$$\mathrm{DP}_t[\hat{\sigma}_t, \hat{d}] = \bigvee_{\hat{\sigma}_{t'} \in \mathcal{S}^F_{\hat{\sigma}_t}} \mathrm{DP}_{t'}[\hat{\sigma}_{t'}, \hat{d} - \hat{d'}(\hat{\sigma}_{t'})],$$

where $\mathcal{S}^F_{\hat{\sigma}_t} \subseteq \hat{\Sigma}_{t'}$ such that $\hat{\sigma}_{t'} = (Z', k', P', \hat{c'}) \in \mathcal{S}^F_{\hat{\sigma}_t}$ if

1. $Z = Z' \setminus \{v\}$,
2. if $v \in Z'$ then $k = k'$, otherwise $k = k' + 1$,
3. $P = P'_{\downarrow Z}$,
4. for all $p' \in P'$ such that $v \notin p'$, we have $\hat{c}(p') = \hat{c'}(p')$,
5. if $\{v\} \in P'$ then $\hat{d'}(\hat{\sigma}_{t'}) = \binom{\hat{c'}(\{v\})+1}{2}$, otherwise $\hat{d'}(\hat{\sigma}_{t'}) = 0$ and, if $v \in Z'$, let $p'_v \in P'$ with $v \in p'_v$, in which case let $\hat{c}(p'_v \setminus \{v\})$ be equal to the minimum element of B that upper-bounds $\hat{c'}(p'_v) + 1$.

Intuitively, if v is the last vertex of an active component we accumulate the number of pairs of connected vertices in that component to $\hat{d}$, otherwise we increase the approximate size of said active component.

Lemma 7.20. *Let t be a forget node, with t' denoting its child node. Assume that the invariant holds for node t'. Then it also holds for node t.*

Proof. Suppose t is a forget node with child t' such that $X_t = X_{t'} \setminus \{v\}$ for some $v \in X_{t'}$ and $h_t = h_{t'} + 1$. By induction, for an exact solution $(\sigma_{t'} = (Z', k', P', c'), d')$ we have calculated an approximate solution $(\hat{\sigma}_{t'} = (Z', k', P', \hat{c'}), \hat{d'})$ with $\hat{c'}(p') \leq (1+\delta)^{h_{t'}}\cdot c'(p')$ for all $p' \in P'$ and $\hat{d'} \leq (1+\varepsilon)\cdot d'$.

Let $(\sigma_t = (Z, k, P, c), d)$ denote an exact solution for node t, and $S \subseteq X_t^{\downarrow}$ a set that realizes it. We consider different cases.

First, if $v \notin S$, it follows that $(\sigma_{t'} = (Z, k-1, P, c), d)$ is an exact solution for node t', and by induction it easily follows that $(\hat{\sigma}_t = (Z, k, P, \hat{c}), \hat{d})$ is an approximate solution for node t that respects the approximation guarantees.

Alternatively, it holds that $v \in S$, in which case it follows that there exists an exact solution $(\sigma'_t = (Z', k', P', c'), d')$ in node t' realized by S. By induction, we have calculated the approximate solution $(\hat{\sigma}_{t'} = (Z', k', P', \hat{c'}), \hat{d'})$ and let $\hat{\sigma}_t = (Z, k, P, \hat{c}) \in \hat{\Sigma}_t$ such that $\hat{\sigma}_{t'} \in \mathcal{S}^F_{\hat{\sigma}_t}$. Let $p'_v \in P'$ with $v \in p'_v$.

Consider the case where $\{v\} \notin P'$, that is, v is not the last vertex of an active component. We argue that the approximate solution $(\hat{\sigma}_t, \hat{d'})$ satisfies the approximation guarantees. It suffices to argue about the approximation guarantee on the size of the active component $p'_v \setminus \{v\}$; indeed, it holds that

$$\begin{aligned}
\hat{c}(p'_v \setminus \{v\}) &= (1+\delta)^{\lceil \log_{(1+\delta)}(\hat{c'}(p'_v)+1)\rceil} \\
&\leq (1+\delta) \cdot (\hat{c'}(p'_v) + 1) \\
&\leq (1+\delta) \cdot (1+\delta)^{h_{t'}} \cdot (c'(p'_v) + 1) \\
&= (1+\delta)^{h_t} \cdot c(p'_v \setminus \{v\}),
\end{aligned}$$

where $\log_{(1+\delta)}(\cdot)$ denotes the logarithm base $(1+\delta)$, and the second inequality is due to the induction hypothesis.

Lastly, consider the case where $\{v\} \in P'$, that is, v is the last vertex of an active component. We argue that the approximate solution $(\hat{\sigma}_t, \hat{d'} + \hat{d'}(\hat{\sigma}_{t'}))$ satisfies the approximation guarantees, where $\hat{d'}(\hat{\sigma}_{t'}) = \binom{\hat{c'}(\{v\})+1}{2}$. In suffices to argue about the approximation guarantee on the number of pairs of connected vertices in the inactive components. Let C denote the active component of $G[\mathcal{T}_{t'}] - S$ that contains v. Notice that if $|V(C)| = 1$, then $C = \{v\}$ and $c'(\{v\}) = \hat{c'}(\{v\}) = 0$, thus $\hat{d'}(\hat{\sigma}_{t'}) = 0$. Alternatively, it holds that

$$\begin{aligned}
\hat{c'}(p_v) + 1 &\leq (1+\delta)^{h_{t'}} \cdot c'(p_v) + 1 \\
&\leq (1+\delta)^{h_{t'}} \cdot (c'(p_v) + 1) \\
&\leq (1+\varepsilon') \cdot (c'(p_v) + 1) \\
&= (1+\varepsilon') \cdot |V(C)|,
\end{aligned}$$

where in the second inequality we use the fact that we have set δ to a value such that $(1+\delta)^h \leq e^{\delta h} = e^{\varepsilon'/2} \leq (1+\varepsilon')$ for small enough ε'.[8] It follows that the induction hypothesis, along with Lemma 7.18 in the second case, implies that indeed $\hat{d'} + \hat{d'}(\hat{\sigma}_{t'})$ is a $(1+\varepsilon)$-approximation of d.

As for the second direction of the invariant, it is straightforward to argue about its validity by induction. □

[8] It suffices to assume without loss of generality $\varepsilon' < 1/4$.

Join node. Finally, suppose that t is a join node with children t_1, t_2 such that $X_t = X_{t_1} = X_{t_2}$. For all $\hat{\sigma}_t = (Z, k, P, \hat{c}) \in \hat{\Sigma}_t$ and $\hat{d} \in [0, (1+\varepsilon) \cdot \binom{n}{2}]$ we set

$$\mathrm{DP}_t[\hat{\sigma}_t, \hat{d}] = \bigvee_{\substack{\hat{d}_1 + \hat{d}_2 = \hat{d}, \\ (\hat{\sigma}_{t_1}, \hat{\sigma}_{t_2}) \in \mathcal{R}^U_{\hat{\sigma}_t}}} (\mathrm{DP}_{t_1}[\hat{\sigma}_{t_1}, \hat{d}_1] \wedge \mathrm{DP}_{t_2}[\hat{\sigma}_{t_2}, \hat{d}_2]),$$

where $\mathcal{R}^U_{\hat{\sigma}_t} \subseteq \hat{\Sigma}_{t_1} \times \hat{\Sigma}_{t_2}$ such that $(\hat{\sigma}_{t_1}, \hat{\sigma}_{t_2}) = ((Z_1, k_1, P_1, \hat{c}_1), (Z_2, k_2, P_2, \hat{c}_2)) \in \mathcal{R}^U_{\hat{\sigma}_t}$ if

1. $Z = Z_1 = Z_2$,
2. $k = k_1 + k_2$,
3. $P = P_1 \sqcup P_2$,
4. for every $p \in P$, $\hat{c}(p)$ is equal to the minimum element of B that upper-bounds

$$\sum_{p_1 \subseteq p \wedge p_1 \in P_1} \hat{c}_1(p_1) + \sum_{p_2 \subseteq p \wedge p_2 \in P_2} \hat{c}_2(p_2).$$

Lemma 7.21. *Let t be a join node, with t_1 and t_2 denoting its children. Assume that the invariant holds for nodes t_1 and t_2. Then it also holds for node t.*

Proof. Suppose that t is a join node with children t_1, t_2 such that $X_t = X_{t_1} = X_{t_2}$ and $h_{t_1}, h_{t_2} \leq h_t - 1$.

Let $(\sigma_t = (Z, k, P, c), d)$ denote an exact solution for node t and $S \subseteq X_t^{\downarrow}$ a set that realizes it. Let for $j \in \{1, 2\}$, $\mathrm{sgn}_{t_j}(S) = \sigma_j = (Z, k_j, P_j, c_j)$, and (σ_j, d_j) denote the exact solution at node t_j that is realized by $S \cap X_{t_j}^{\downarrow}$. By induction, we have calculated, for $j \in \{1, 2\}$, approximate solutions $(\hat{\sigma}_{t_1} = (Z, k_j, P_j, \hat{c}_j), \hat{d}_j)$ with $\hat{c}_j(p) \leq (1+\delta)^{h_{t_j}} \cdot c_j(p)$ for all $p \in P_j$ and $\hat{d}_j \leq (1+\varepsilon) \cdot d_j$. Consider the approximate solution $(\hat{\sigma}_t, \hat{d}_t)$ where $(\hat{\sigma}_1, \hat{\sigma}_2) \in \mathcal{R}^U_{\hat{\sigma}_t}$ and $\hat{d} = \hat{d}_1 + \hat{d}_2$. We argue that it respects the approximation guarantees. Indeed, it holds that $\hat{d}_1 + \hat{d}_2 \leq (1+\varepsilon) \cdot (d_1 + d_2) = (1+\varepsilon) \cdot d$. Moreover, for $p \in P$ it holds that

$$\begin{aligned}
\hat{c}(p) &= (1+\delta)^{\left\lceil \log_{(1+\delta)} \left(\sum_{p_1 \subseteq p \wedge p_1 \in P_1} \hat{c}_1(p_1) + \sum_{p_2 \subseteq p \wedge p_2 \in P_2} \hat{c}_2(p_2) \right) \right\rceil} \\
&\leq (1+\delta) \cdot \Big(\sum_{p_1 \subseteq p \wedge p_1 \in P_1} \hat{c}_1(p_1) + \sum_{p_2 \subseteq p \wedge p_2 \in P_2} \hat{c}_2(p_2) \Big) \\
&\leq (1+\delta) \cdot (1+\delta)^{h_t - 1} \cdot \Big(\sum_{p_1 \subseteq p \wedge p_1 \in P_1} c_1(p_1) + \sum_{p_2 \subseteq p \wedge p_2 \in P_2} c_2(p_2) \Big) \\
&= (1+\delta)^{h_t} \cdot c(p),
\end{aligned}$$

where $\log_{(1+\delta)}(\cdot)$ denotes the logarithm base $(1+\delta)$, and the second inequality is due to the induction hypothesis.

As for the second direction of the invariant, it is straightforward to argue about its validity by induction. □

Wrapping up. The correctness of the algorithm follows by Lemmas 7.19 to 7.21. Using back-tracking we can compute a set that realizes a given approximate solution with polynomial overhead. It remains to argue about the running time. Notice that $|B| = \mathcal{O}(\frac{\log n}{\log(1+\delta)}) = \mathcal{O}(\frac{\log n}{\delta}) = \mathcal{O}(\frac{\mathrm{tw}\log^2 n}{\varepsilon'})$, where we used the fact that $e^{\delta/2} \leq 1+\delta$ for all $\delta < 1/2$. It follows that the running time of the presented algorithm is $2^{\mathcal{O}(\mathrm{tw})}\mathrm{tw}^{\mathcal{O}(\mathrm{tw})}|B|^{\mathcal{O}(\mathrm{tw})}n^{\mathcal{O}(1)} = (\log n/\varepsilon)^{\mathcal{O}(\mathrm{tw})}\mathrm{tw}^{\mathcal{O}(\mathrm{tw})}n^{\mathcal{O}(1)}$. To achieve the running time bound of $(\mathrm{tw}/\varepsilon)^{\mathcal{O}(\mathrm{tw})}n^{\mathcal{O}(1)}$ we use a well-known win/win argument: if $\mathrm{tw} \leq \sqrt{\log n}$ then $(\log n)^{\mathcal{O}(\mathrm{tw})} = (\log n)^{\mathcal{O}(\sqrt{\log n})} = n^{o(1)}$; alternatively, $\log n \leq \mathrm{tw}^2$ thus $(\log n)^{\mathcal{O}(\mathrm{tw})} = \mathrm{tw}^{\mathcal{O}(\mathrm{tw})}$. □

7.5. Kernelization

Given that, as we have shown in Theorem 7.11, Critical Node Cut is FPT parameterized by the vertex integrity of the input graph, a natural question arising is whether one can develop a polynomial kernel under this parameterization. In this section we prove that this cannot be the case under standard assumptions, even for the much more restricted parameterization by vertex cover number.

Theorem 7.22. *Critical Node Cut does not admit a polynomial kernel when parameterized by the vertex cover number of the input graph, unless* NP ⊆ coNP/poly.

Proof. We present a polynomial parameter transformation reducing from k-Multicolored Clique. In the latter, we are given a graph $G = (V, E)$ and a partition of V into k independent sets (also called color classes) $V_1, \ldots, V_k$, each of size n, and we are asked to determine whether G contains a k-clique. It is known that k-Multicolored Clique parameterized by $k \log n$ does not admit a polynomial kernel unless NP ⊆ coNP/poly [170].

Construction. Consider an instance (G, k) of k-Multicolored Clique, where $V_i = \{v_j^i : j \in [n]\}$ for all $i \in [k]$. For each color class V_i, we assign to each of its vertices a unique (among the vertices of V_i) bit-string of length $\log n$. For all $w \in [\log n]$, let $b_w \colon V \to \{0,1\}$ denote the w-th bit in the bit-string of $v_j^i \in V_i$. Construct the graph G' as follows.

- We first introduce a clique on vertex set $K = \{g_z^{i,w} : i \in [k],\ w \in [\log n],\ z \in \{0,1\}\}$, which we refer to as *core vertices*. We say that a core vertex $g_z^{i,w}$ *corresponds* to color class i, with K_i being composed of all core vertices corresponding to color class i, for $i \in [k]$. Furthermore, we say that a vertex $v \in V_i$ of G is *encoded* by the vertices $\{g_{b_w(v)}^{i,w} : w \in [\log n]\}$.
- Introduce a clique on the vertex set $C = \{c_i : i \in [k \log n + 1]\}$, and add edges such that every vertex of C is adjacent to all the core vertices.
- For every edge $e = \{u_1, u_2\} \in E(G)$, where $u_1 \in V_{i_1}$ and $u_2 \in V_{i_2}$, introduce an *adjacency vertex* h_e which is incident with all vertices encoding its endpoints, that is, to vertices $g_{b_w(u_1)}^{i_1,w}$ and $g_{b_w(u_2)}^{i_2,w}$ for all $w \in [\log n]$. Let H_{i_1,i_2} denote the set of all adjacency vertices due to edges in G between vertices in V_{i_1} and V_{i_2}.

- For all $\{i_1, i_2\} \in \binom{[k]}{2}$ (that is, for all pairs of different color classes), and for all $w_1, w_2 \in [\log n]$ and $z_1, z_2 \in \{0, 1\}$, introduce an independent set of size $A = |E|+1$, each vertex of which is incident with $g^{i_1,w_1}_{z_1}$ and $g^{i_2,w_2}_{z_2}$. We refer to the vertices added in this step of the construction as *dummy vertices*.

This concludes the construction of the graph G'. Notice that $G' - (K \cup C)$ is an independent set, consequently the vertex cover number of G' is at most $3k \log n + 1$. We will show that (G', k', x) is an equivalent instance of Critical Node Cut, where $k' = k \log n$ and $x = \binom{|V(G')|-k'-\binom{k}{2}-A\binom{k}{2}\log^2 n}{2}$.

For the forward direction, consider a function $s\colon [k] \to [n]$ such that $\mathcal{V} = \{v^i_{s(i)} : i \in [k]\}$ is a k-clique in G. In that case, let $S = \{g^{i,w}_{b_w(v^i_{s(i)})} : i \in [k],\, w \in [\log n]\}$ be a set of size k'. We will prove that $G' - S$ has at most x pairs of connected vertices. First, notice that for every pair of core vertices belonging to S that correspond to different color classes, their removal results in an independent set of dummy vertices of size A in $G' - S$. Since there are $\binom{k}{2}\log^2 n$ such pairs, $G' - S$ contains $A\binom{k}{2}\log^2 n$ isolated dummy vertices. Furthermore, we argue that the adjacency vertex of any edge in $G[\mathcal{V}]$ is isolated in $G' - S$. To see this, consider the adjacency vertex $h_e \in V(G')$ where $e = \{v^{i_1}_{s(i_1)}, v^{i_2}_{s(i_2)}\} \in E(G)$. It holds that $N_{G'}(h_e) = \{g^{i_1,w}_{b_w(v^{i_1}_{s(i_1)})}, g^{i_2,w}_{b_w(v^{i_2}_{s(i_2)})} : w \in [\log n]\} \subseteq S$, thus h_e is indeed an isolated vertex in $G' - S$. Since $\mathcal{V}$ induces a k-clique, there are $\binom{k}{2}$ such isolated adjacency vertices. Finally, S itself is of size k'. Consequently, the number of pairs of connected vertices in $G' - S$ is at most the number of pairs of non-isolated vertices in the graph, which is at most $\binom{|V(G')|-A\binom{k}{2}\log^2 n-\binom{k}{2}-k'}{2} = x$.

For the opposite direction, let $S \subseteq V(G')$ be of size at most k' such that $G' - S$ has at most x pairs of connected vertices. Notice that $|C| > k'$, consequently there exists a vertex $c \in C \setminus S$. Since $N_G(c) \supseteq K$ and C is a clique, it follows that all vertices of $(K \cup C) \setminus S$ are in the same connected component of $G' - S$; let this component be denoted by $D_{\mathsf{big}} \in \mathsf{cc}(G' - S)$. Furthermore, since for any vertex $v \in V(G') \setminus (K \cup C)$ it holds that $N_G(v) \subseteq K$, any such vertex either belongs to D_{big} or is isolated in $G' - S$.

▷ Claim 7.23. We have $|S \cap K_i| = \log n$, for all $i \in [k]$.

Proof of the claim. To prove the claim we argue about the number of isolated dummy vertices in $G' - S$. First, notice that $(n - \varepsilon)(n + \varepsilon) = n^2 - \varepsilon^2 < n^2$ for all $\varepsilon > 0$. Consequently, the number of isolated dummy vertices in $G' - S$ is maximized when $|S \cap K_i| = \log n$ for all $i \in [k]$, in which case the number of isolated dummy vertices is $A\binom{k}{2}\log^2 n$.

Assume that the claim is false. In that case, the number of isolated dummy vertices in $G' - S$ is at most $A\Big(\binom{k}{2}\log^2 n - 1\Big)$; any other dummy vertex in $G' - S$ must belong to D_{big}, which also contains all vertices of $(K \cup C) \setminus S$. Consequently, the size of $D_{\mathsf{big}} \in \mathsf{cc}(G' - S)$

is at least $|V(G')| - |E| - k' - A\Big(\binom{k}{2}\log^2 n - 1\Big)$. In that case

$$|V(G')| - |E| - k' - A\left(\binom{k}{2}\log^2 n - 1\right) > |V(G')| - A\binom{k}{2}\log^2 n - \binom{k}{2} - k' \iff$$
$$-|E| + A > -\binom{k}{2} \impliedby$$
$$A = |E| + 1,$$

thereby yielding a contradiction as $G' - S$ has more than x pairs of connected vertices. $\lhd$

Recall that $G' - S$ is composed of a connected component D_{big} containing all non-isolated vertices, as well as some isolated vertices. By Claim 7.23 it holds that $|S| = k'$, thus for $G' - S$ to have at most x pairs of connected vertices the number of its isolated vertices must be at least $\binom{k}{2} + A\binom{k}{2}\log^2 n$. Due to Claim 7.23 it follows that it has exactly $A\binom{k}{2}\log^2 n$ isolated dummy vertices, while none of the remaining vertices of $K \cup C$ can be isolated. Consequently, the deletion of S isolates at least $\binom{k}{2}$ adjacency vertices.

We argue that no two isolated adjacency vertices h_{e_1}, h_{e_2} belong to the same set H_{i_1,i_2}. Assume that this is the case, and let $h_{e_1}, h_{e_2} \in H_{i_1,i_2}$. For $u \in \{h_{e_1}, h_{e_2}\}$ it holds that $|N_{G'}(u) \cap K_{i_1}| = |N_{G'}(u) \cap K_{i_2}| = \log n$, while $N_{G'}(h_{e_1}) \neq N_{G'}(h_{e_2})$. Due to Claim 7.23, this leads to a contradiction. Consequently, there is a exactly one isolated adjacency vertex in $G' - S$ belonging to H_{i_1,i_2} for all $\{i_1, i_2\} \in \binom{[k]}{2}$.

Now consider one such isolated adjacency vertex $h_e \in H_{i_1,i_2}$. Notice that $|N_{G'}(h_e) \cap \{g_0^{p,w}, g_1^{p,w}\}| = 1$ for all $w \in [\log n]$ and $p \in \{i_1, i_2\}$. Since $S \supseteq N_{G'}(h_e)$, and this holds for all isolated adjacency vertices, due to Claim 7.23 it follows that $|S \cap \{g_0^{i,w}, g_1^{i,w}\}| = 1$ for all $w \in [\log n]$ and $i \in [k]$.

Let $s\colon [k] \to [n]$ such that $v^i_{s(i)}$ is encoded by the vertices $S \cap K_i$, for all $i \in [k]$. We claim that $\mathcal{V} = \{v^i_{s(i)} : i \in [k]\}$ induces a k-clique in G. Consider $\{i_1, i_2\} \in \binom{[k]}{2}$. Notice that there exists an isolated adjacency vertex h_e in $G' - S$ belonging to H_{i_1,i_2}; the neighborhood of h_e is exactly the vertices encoding its two endpoints in G, thus $\{v^{i_1}_{s(i_1)}, v^{i_2}_{s(i_2)}\} \in E(G)$. $\square$

8. Conclusion

In this thesis we examined the parameterized complexity of several NP-hard graph problems through the lens of structural parameters, and we employed approximation when exact tractability was out of reach. Guided by the technique-first perspective of Sections 1.2 and 1.3, we contributed several reusable tools and used them to obtain tight upper and lower bounds, FPT approximation schemes, and clear borders between tractable and intractable regimes. Here we summarize what each chapter achieves and outline concrete directions for future work.

Chapter 3. We studied Bounded Degree Vertex Deletion and Defective Coloring and precisely determined their complexity under some of the most commonly used structural parameters. We gave ETH-tight lower bounds for td via a recursive copy-gadget construction and used d-detecting families to obtain tight lower bounds for vertex cover, while also settling the status of $\mathrm{tw} + \Delta$. *Future work.* Can we obtain an ETH-based $n^{o(\mathrm{fvs})}$ lower bound for Bounded Degree Vertex Deletion and for Defective Coloring when $\chi_d = 2$?

Chapter 4. For Induced Matching and Acyclic Matching we proved tight pw-SETH lower bounds for the pw parameterization (equivalence even for Induced Matching), thereby rendering the known tw-based algorithms optimal in their base. We also gave a tight $\mathcal{O}^\star(3^{\mathrm{cw}})$ algorithm for Induced Matching under cw. *Future work.* Develop a (pw-)SETH-optimal algorithm for Acyclic Matching parameterized by cw. Despite recent progress on connectivity problems parameterized by cw [48, 47, 168], the case of Feedback Vertex Set, which is closely related to Acyclic Matching, remains open. Another direction is an optimal, SETH-tight algorithm for Acyclic Matching parameterized by cutwidth; such results are known for many problems [46, 156, 178, 238, 279], including Feedback Vertex Set [45].

Chapter 5. We analyzed MaxNDP, giving a near-complete picture under standard parameterizations and complementing it with parameterized approximation. We showed an efficient FPT-AS on bounded td via a simple win/win + color-coding + assembly pipeline, and we proved inapproximability on pw under PIH, thereby identifying a crisp parameter border. *Future work.* Although Theorem 5.6 rules out an approximation scheme on pw, a constant-factor approximation running in time $f(\mathrm{tw})\, n^{\mathcal{O}(1)}$ may still be possible. Among our FPT algorithms (cf. Section 5.1), all but Theorem 5.3 rely on bounding the number of vertices in an optimal solution, leading to double-exponential running times in some cases; the optimality of these dependences is open. It is also unknown whether the problem is FPT when parameterized by cvd or ctw.

Chapter 6. We presented several new results for Vertex Integrity and related component-bounded cut problems. *Future work.* Can the slightly super-exponential $k^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ algorithm (with k being the vertex integrity) be improved to single-exponential? Although we obtained such an improvement for vertex cover, we conjecture the answer is negative for vertex integrity itself. On the approximation side, it is natural to seek FPT approximations in W[1]-hard regimes, and a constant-factor or even $(1+\varepsilon)$-approximation algorithm running in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ would be an interesting avenue for research.

Chapter 7. We investigated Critical Node Cut under structural parameterizations, proving stronger W[1]-hardness, giving an FPT-AS by rounding over tw, and establishing a kernelization lower bound under vertex cover. *Future work.* Is Critical Node Cut FPT under $k + \mathrm{td}$, or under cvd? Can we (dis)prove the optimality of the $n^{\mathcal{O}(\mathrm{tw})}$ or $n^{\mathcal{O}(2^{\mathrm{cw}})}$ algorithms? For the latter, only a handful of natural problems are known to have such ETH-optimal running times [2, 129, 176].

A. Résumé étendu en français

Cette thèse se situe à l'intersection de la complexité paramétrée, de la complexité fine et de l'approximation, et étudie, dans ce cadre, plusieurs problèmes NP-difficiles sur les graphes. Elle adopte ainsi trois points de vue complémentaires pour analyser ces problèmes. Son hypothèse directrice est la suivante : la difficulté algorithmique d'un problème ne provient pas uniquement de la taille brute de l'entrée, mais aussi de la structure combinatoire du graphe considéré. Des paramètres tels que la largeur d'arbre, la profondeur d'arbre, la taille d'une couverture de sommets, la largeur modulaire, ou encore le nombre de feuilles maximal, permettent de mesurer cette structure et d'expliquer pourquoi certains cas apparemment difficiles deviennent traitables alors que d'autres restent inaccessibles [192, 249].

L'objectif scientifique de la thèse est de comprendre, avec le plus de précision possible, comment ces paramètres structuraux gouvernent la tractabilité. Plus précisément, il s'agit : (i) d'identifier la frontière entre les paramétrisations qui admettent des algorithmes FPT et celles pour lesquelles on peut démontrer une dureté W[1] ; (ii) de déterminer la dépendance optimale au paramètre dans le temps d'exécution, au moyen de bornes inférieures serrées sous ETH, SETH ou pw-SETH ; et (iii) d'examiner si l'approximation paramétrée permet de contourner des barrières de calcul exact, ou si, au contraire, même cette relaxation reste hors de portée, comme l'illustrent notamment les résultats développés dans Chapters 3 to 7.

Au-delà des résultats obtenus pour chaque problème étudié, la contribution principale de la thèse est d'ordre méthodologique. Elle met au point plusieurs techniques réutilisables qui enrichissent l'arsenal de la complexité paramétrée structurale : une nouvelle variante de Unary Bin Packing pour construire des réductions de dureté W[1] sous des paramètres très contraints ; une construction récursive adaptée à la profondeur d'arbre pour établir des bornes inférieures ETH serrées ; une nouvelle utilisation des familles d-détectantes pour obtenir des bornes inférieures serrées sous le paramètre couverture de sommets ; et enfin une construction simple en trois étapes menant à des schémas d'approximation FPT efficaces. Cette dimension transversale constitue le coeur de la démarche. Les chapitres n'apparaissent pas comme une juxtaposition de cas particuliers, mais comme les différentes manifestations d'un même programme de recherche, consistant à comprendre où se situent les vraies barrières algorithmiques et quelles techniques permettent de les atteindre ou de les contourner.

A.1. Contexte scientifique et motivation

Complexité classique, structure et limites du point de vue polynomial

La théorie de la complexité cherche à distinguer les problèmes qui peuvent être résolus efficacement de ceux pour lesquels aucun algorithme polynomial n'est connu. Dans ce cadre, la frontière emblématique est celle entre P et NP. Cependant, pour une grande variété de problèmes combinatoires importants, en particulier sur les graphes, cette dichotomie s'avère trop grossière. Savoir qu'un problème est NP-difficile n'explique ni quels aspects d'une instance sont responsables de cette difficulté, ni quelles restrictions structurelles pourraient rendre le problème plus accessible [82, 146, 269].

Les problèmes étudiés dans cette thèse illustrent bien cette tension. Tous sont NP-difficiles, mais ils réagissent de manière très différente lorsqu'on restreint la structure de l'entrée. Sur des graphes proches d'arbres, de forêts d'étoiles, ou de graphes très modulaires, on peut parfois espérer des algorithmes efficaces. À l'inverse, même des graphes très contraints peuvent encore encoder une difficulté suffisante pour exclure l'existence d'algorithmes FPT. L'enjeu n'est donc pas simplement de trouver des algorithmes ou des réductions de dureté, mais de cartographier avec finesse le paysage de complexité induit par plusieurs paramètres structuraux.

Le paradigme de la complexité paramétrée

La complexité paramétrée fournit le langage naturel pour mener une telle étude. On enrichit l'entrée d'un paramètre k, supposé petit sur les instances intéressantes, et on cherche des algorithmes de temps $f(k) \cdot n^{\mathcal{O}(1)}$, où n est la taille de l'entrée. Lorsqu'un tel algorithme existe, le problème est dit *traitable à paramètres fixes* ou FPT. Cette notion raffine profondément l'analyse classique : un problème NP-difficile peut rester inabordable au sens polynomial tout en devenant raisonnablement calculable dès lors qu'un bon paramètre reste borné [92, 103, 123].

Dans le cadre des graphes, les paramètres structuraux sont particulièrement pertinents. La largeur d'arbre mesure à quel point un graphe est proche d'un arbre ; la largeur de chemin en constitue une version linéarisée ; la profondeur d'arbre capture plutôt une hiérarchie enracinée très restrictive ; la taille d'une couverture de sommets quantifie la distance à une structure quasi indépendante ; la largeur modulaire et le nombre maximal de feuilles mesurent d'autres formes de simplicité combinatoire. L'un des thèmes centraux de la thèse consiste à comparer ces paramètres entre eux, afin de déterminer lesquels suffisent à rendre un problème FPT, et lesquels restent trop généraux [192, 249].

Il est important de souligner que ces paramètres n'ordonnent pas seulement les graphes du plus simple au plus complexe. Ils décrivent des formes différentes de structure. Un graphe de petite largeur d'arbre autorise en général une décomposition locale en morceaux qui communiquent peu entre eux. Un graphe de petite profondeur d'arbre possède plutôt une structure hiérarchique fortement imbriquée. Un graphe de petite couverture de sommets concentre sa complexité sur un petit noyau, tandis qu'une grande partie de ses sommets restent en dehors de toute arête. Du point de vue algorithmique, ces différences sont fondamentales : elles déterminent non seulement la possibilité d'un algorithme FPT,

mais aussi la nature des informations à mémoriser dans une programmation dynamique, la forme des réductions de dureté admissibles, et même le type de raisonnement combinatoire pertinent.

Cette attention aux différences qualitatives entre paramètres explique le choix des études de cas retenues dans la thèse. Il ne s'agit ni d'étudier un seul problème sous tous les paramètres possibles, ni d'examiner un paramètre unique sur tous les problèmes possibles. Le fil conducteur consiste plutôt à choisir des couples problème-paramètre suffisamment riches pour faire apparaître des phénomènes distincts : des transitions FPT/W[1], des changements dans la base d'une exponentielle, des écarts entre calcul exact et approximation, ou encore des contrastes entre paramètres voisins. Ce positionnement donne à l'ensemble de la thèse son unité intellectuelle.

La contrepartie naturelle de FPT est la dureté W[1]. Prouver qu'un problème est W[1]-difficile pour une paramétrisation donnée constitue un argument fort selon lequel il n'existe vraisemblablement pas d'algorithme FPT sous ce paramètre. Une telle preuve joue donc le rôle d'une frontière de tractabilité. Toutefois, obtenir une dureté W[1] sous des paramètres structuraux très restrictifs est souvent délicat : les réductions classiques gonflent trop fortement les paramètres, ou bien nécessitent des gadgets qui détruisent précisément la structure qu'on cherche à contrôler. L'une des contributions majeures de la thèse est précisément de surmonter cette difficulté [92, 103].

Pourquoi la complexité fine est indispensable

Même lorsqu'un problème est FPT, cette information reste incomplète. Deux algorithmes FPT peuvent différer radicalement : l'un en temps $2^k n^{\mathcal{O}(1)}$, l'autre en temps $k^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$, voire pire. Or, dans de nombreux cas, les algorithmes naturels issus d'une programmation dynamique sur une décomposition arborée semblent difficiles à améliorer. Pour démontrer que cette impression correspond à une barrière réelle, il faut quitter le terrain de la seule classification FPT/W[1] et recourir à la complexité fine.

Les hypothèses ETH et SETH permettent d'obtenir des bornes inférieures conditionnelles sur le temps d'exécution. Dans le cadre paramétré, elles servent à montrer qu'une dépendance exponentielle, ou une base particulière dans l'exponentielle, est essentiellement optimale. La thèse s'inscrit pleinement dans cette perspective : pour plusieurs problèmes, elle montre que les algorithmes dynamiques "naturels" sur largeur d'arbre, largeur de chemin, profondeur d'arbre ou couverture de sommets sont en réalité optimaux sous des hypothèses standard. Elle s'appuie également sur l'hypothèse pw-SETH, récemment introduite, qui permet de raffiner encore les résultats sur les paramètres linéaires comme la largeur de chemin et d'obtenir des équivalences fines entre problèmes [53, 172, 173, 210].

Approximation et approximation paramétrée

Lorsqu'un calcul exact est hors d'atteinte, une stratégie naturelle consiste à rechercher une solution proche de l'optimum. L'approximation classique s'intéresse à des algorithmes polynomiaux garantissant un certain facteur. Cependant, pour certains problèmes, les garanties classiques restent faibles, et les barrières d'inapproximabilité sont sévères.

L'approximation paramétrée propose alors un compromis plus subtil : autoriser une dépendance forte en un paramètre structural tout en conservant une garantie de proximité à l'optimum.

Ce point de vue est particulièrement fécond dans le cadre de cette thèse. Il permet d'identifier des situations où l'approximation "sauve" un problème que le calcul exact ne peut traiter, mais aussi des situations où même un schéma d'approximation FPT ne peut exister. L'approximation n'est donc pas envisagée comme un simple prolongement annexe, mais comme une composante à part entière de l'étude de la structure : selon le paramètre choisi, elle peut soit franchir une barrière fondamentale, soit se heurter à une nouvelle frontière [115, 202, 236, 282, 284].

Cette perspective modifie aussi la manière d'évaluer un résultat positif. Dans un cadre classique, obtenir un facteur constant ou un PTAS constitue déjà une information importante. Dans un cadre paramétré, il faut en plus se demander comment le coût de cette approximation dépend du paramètre structural. Un schéma d'approximation dont le temps d'exécution explose inutilement avec la largeur d'arbre ou la profondeur d'arbre a finalement une portée limitée. L'enjeu n'est donc pas seulement d'approcher l'optimum, mais de le faire en respectant la structure de l'instance. Cette exigence est au centre des résultats de la thèse sur Maximum Node-Disjoint Paths et sur Critical Node Cut (voir Chapters 5 and 7).

A.2. Question de recherche et positionnement de la thèse

Une étude guidée par les paramètres structuraux

La question générale posée dans cette thèse peut se formuler ainsi : *dans quelle mesure la structure du graphe permet-elle de rendre traitables des problèmes NP-difficiles, et quel est le coût paramétré optimal de cette traitabilité ?* Cette question se décline en trois axes.

Le premier axe consiste à déterminer la frontière entre tractabilité et dureté. Pour un problème donné, certains paramètres permettent des algorithmes FPT, tandis que des paramètres un peu plus généraux conduisent déjà à la dureté W[1]. Comprendre cette transition revient à mesurer le "prix de la généralité" : jusqu'où peut-on relâcher la structure avant que toute approche FPT devienne illusoire ?

Le deuxième axe vise l'optimalité des algorithmes. Lorsqu'une programmation dynamique sur largeur d'arbre donne un algorithme de temps $c^{\mathrm{tw}}n^{\mathcal{O}(1)}$, peut-on réduire la base c ? Lorsqu'un algorithme dépend comme $\mathrm{vc}^{\mathcal{O}(\mathrm{vc})}$, est-ce une faiblesse technique ou une nécessité intrinsèque ? Les réponses obtenues dans la thèse montrent que, dans plusieurs cas, les algorithmes les plus évidents sont déjà les meilleurs possibles.

Le troisième axe concerne les cas où le calcul exact est bloqué. Il s'agit alors de comprendre si une approximation paramétrée efficace peut surmonter l'obstacle. Cette approche conduit à des dichotomies fines : un problème peut être inabordable exactement sous un certain paramètre, mais admettre un schéma d'approximation FPT sous un paramètre plus restrictif ; inversement, un léger relâchement du paramètre peut rendre impossible même l'approximation paramétrée.

Choix des problèmes étudiés

Les cinq familles de problèmes choisies dans la thèse couvrent plusieurs phénomènes typiques de l'algorithmique sur graphes.

Les problèmes à degré cible, Bounded Degree Vertex Deletion et Defective Coloring, forment un premier bloc naturel : ils généralisent respectivement Vertex Cover et Coloring et constituent un terrain privilégié pour comparer plusieurs paramètres structuraux à partir de programmations dynamiques relativement naturelles (voir Chapter 3).

Un deuxième bloc est constitué de Induced Matching et Acyclic Matching, où la difficulté ne réside plus seulement dans la sélection d'une sous-structure, mais dans les contraintes globales qu'elle doit satisfaire. Ces problèmes permettent d'étudier avec précision le rôle de la largeur d'arbre, de la largeur de chemin et de la largeur de clique, ainsi que les limites d'algorithmes fondés sur des représentations d'états (voir Chapter 4).

Maximum Node-Disjoint Paths occupe une place particulière, car il se situe à l'intersection de la complexité paramétrée et de l'approximation. Il fournit le cadre le plus naturel pour examiner si un schéma d'approximation FPT peut prendre le relais lorsque le calcul exact échoue (voir Chapter 5).

Vertex Integrity et Component Order Connectivity, d'une part, et Critical Node Cut, d'autre part, relèvent quant à eux de la famille des problèmes de coupure. Les premiers se prêtent particulièrement bien à l'étude de la frontière FPT/W[1] sous des paramètres très restrictifs, tandis que le second prolonge cette perspective vers un objectif plus global, en faisant aussi apparaître des questions d'approximation et de noyautisation (voir Chapters 6 and 7).

Message principal de la thèse

Le message scientifique central de la thèse peut se résumer ainsi : les paramètres structuraux constituent un outil particulièrement puissant pour affiner l'analyse de problèmes NP-difficiles sur les graphes. L'effet algorithmique de cette structure dépend toutefois de manière très sensible du paramètre retenu. Un même problème peut ainsi changer de statut lorsque l'on passe de la profondeur d'arbre à la largeur de chemin, ou de la largeur d'arbre à la couverture de sommets.

Il en résulte une seconde idée forte : les barrières mises en évidence dans le manuscrit ne relèvent pas d'accidents techniques, mais correspondent, dans de nombreux cas, à des limitations intrinsèques attestées par des bornes inférieures fines. Lorsque l'approximation permet néanmoins de progresser, cette possibilité apparaît elle aussi comme un phénomène structurel précis.

En ce sens, la thèse s'inscrit dans une ligne de recherche où conception d'algorithmes et preuves d'optimalité se répondent constamment. L'un des apports du travail est précisément de faire apparaître cette interaction comme un principe d'organisation des résultats, et non comme une considération secondaire.

A.3. Contributions méthodologiques principales

Une nouvelle variante de Unary Bin Packing pour la dureté W[1]

La première innovation conceptuelle importante concerne les réductions de dureté W[1] sous paramètres structuraux restreints. Les réductions usuelles vers des problèmes de coupure utilisent souvent des gadgets lourds, qui augmentent la largeur d'arbre, la largeur de chemin, le degré maximal, ou le nombre d'arêtes de retour. Il devient alors très difficile d'obtenir des résultats négatifs forts pour des combinaisons de paramètres comme $\text{fes} + \Delta + \text{pw}$.

L'idée développée dans la thèse est de repartir d'un problème arithmétique, Unary Bin Packing, et d'observer que sa dureté W[1] subsiste même dans une variante extrêmement contrainte : chaque objet ne peut être placé que dans exactement deux bacs possibles. Cette observation est loin d'être anodine. Elle offre une interface combinatoire très fine entre des choix locaux binaires, d'une part, et des contraintes globales d'équilibrage, d'autre part. Grâce à l'encodage unaire, il devient en outre possible d'exploiter des algorithmes pseudo-polynomiaux pour Subset Sum [22, 179].

Cette combinaison conduit à un schéma de réduction particulièrement souple. Les gadgets obtenus gardent une structure presque arborescente, contrôlent le degré maximal, et permettent d'isoler très précisément les arêtes responsables des cycles. La conséquence est une amélioration significative des résultats de dureté pour Vertex Integrity, Component Order Connectivity et Critical Node Cut (voir Theorems 6.15, 6.16, and 7.2). Plus généralement, cette technique suggère une nouvelle façon de construire des réductions paramétrées "structurellement sobres", où l'effort principal porte sur le maintien de la simplicité du graphe cible plutôt que sur la seule correction logique de la réduction.

Une construction récursive adaptée à la profondeur d'arbre

La profondeur d'arbre est un paramètre plus restrictif que la largeur d'arbre, mais sa nature hiérarchique la rend difficile à exploiter dans les preuves de bornes inférieures. De nombreuses réductions classiques préservent mal ce paramètre et n'obtiennent que des bornes faibles, laissant parfois penser qu'un algorithme beaucoup plus rapide pourrait exister.

La thèse introduit ici une construction récursive fondée sur des *copy gadgets*, conçue spécialement pour garder la profondeur d'arbre linéaire. L'idée est de reproduire l'information de manière hiérarchique, en synchronisant les copies sans perdre le contrôle de la structure récursive globale. Plutôt que de considérer la profondeur d'arbre comme une simple contrainte technique, la construction épouse sa définition même. Cette approche permet d'établir des bornes ETH serrées de la forme $n^{o(\text{td})}$ impossibles pour plusieurs problèmes, notamment dans Chapters 3 and 5.

Méthodologiquement, ce résultat est important pour deux raisons. Premièrement, il montre que les paramètres "très restrictifs" ne se traduisent pas nécessairement par des accélérations qualitatives. Deuxièmement, il fournit un cadre potentiellement réutilisable pour d'autres problèmes où la profondeur d'arbre joue un rôle naturel. La thèse défend

ainsi l'idée que la bonne manière d'obtenir des bornes inférieures serrées pour un paramètre consiste souvent à construire des gadgets compatibles avec sa géométrie combinatoire propre.

Familles d-détectantes et bornes serrées pour la couverture de sommets

Un autre apport technique majeur de la thèse réside dans l'utilisation des familles d-détectantes pour établir des bornes inférieures sous le paramètre couverture de sommets. Ce paramètre est très restrictif. De nombreux problèmes qui sont difficiles sous largeur d'arbre deviennent FPT sous couverture de sommets. Cependant, la bonne échelle de complexité dans ce cadre n'est pas toujours évidente. Un algorithme en temps $\mathrm{vc}^{\mathcal{O}(\mathrm{vc})}$ peut-il être amélioré en $\mathrm{vc}^{o(\mathrm{vc})}$?

Pour Bounded Degree Vertex Deletion et Defective Coloring, la thèse montre que la réponse est négative sous ETH (voir Theorems 3.49 and 3.54). La nouveauté ne tient pas seulement au résultat, mais à la méthode. Les familles d-détectantes avaient déjà servi à compresser des informations dans des preuves de bornes inférieures, mais leur emploi dans ce contexte structurel est original. Elles permettent de coder efficacement des configurations nombreuses tout en maintenant une petite couverture de sommets. Autrement dit, elles jouent un rôle de *compression contrôlée* de l'instance dure vers une instance structurellement restreinte.

Ce point est particulièrement significatif dans l'économie générale de la thèse : il montre que les bornes inférieures fines ne sont pas réservées aux paramètres de type largeur d'arbre ou largeur de chemin. Même sous des paramètres très forts, on peut atteindre des résultats serrés si l'on dispose des bons outils combinatoires.

Une construction simple pour des schémas d'approximation FPT

La dernière grande contribution méthodologique concerne l'approximation paramétrée. Pour Maximum Node-Disjoint Paths, la thèse propose une construction en trois étapes, fondée sur le codage par couleurs, une réduction soignée de la taille pertinente de l'instance, et un argument de type *win/win*. Cette approche conduit à un schéma d'approximation FPT efficace sous le paramètre profondeur d'arbre (voir Theorem 5.5).

L'intérêt de cette construction dépasse le seul problème étudié. Elle fournit un modèle simple pour concevoir des schémas d'approximation FPT lorsque les solutions optimales utilisent un sous-ensemble relativement "localisable" de la structure. La thèse insiste sur ce caractère transférable : une bonne partie de l'apport ne réside pas uniquement dans le théorème final, mais dans la mise en évidence d'un patron algorithmique générique.

Une lecture unifiée des techniques développées

Ces quatre contributions méthodologiques peuvent sembler de nature très différente : l'une concerne la dureté W[1], deux autres des bornes inférieures fines, et la dernière l'approximation paramétrée. Pourtant, elles partagent une même idée directrice : adapter l'outil employé à la structure précise du paramètre étudié.

Dans le cas de la variante de Unary Bin Packing, l'objectif est de préserver des paramètres structurels très faibles pendant la réduction. Dans le cas de la profondeur d'arbre, l'enjeu est de faire coïncider la forme de la construction avec la récursivité intrinsèque du paramètre. Pour la couverture de sommets, il faut concentrer une quantité importante d'information dans un petit noyau de sommets distingués. Enfin, pour l'approximation paramétrée, il s'agit de tirer parti du fait que la structure borne non pas l'espace total des solutions, mais l'espace des configurations "importantes" qu'il faut explorer.

Cette lecture unifiée aide à comprendre l'originalité de la thèse. Elle n'accumule pas seulement des astuces techniques indépendantes. Elle défend une manière de travailler sur les paramètres structuraux : prendre au sérieux leur géométrie combinatoire propre et concevoir, pour chacun d'eux, des outils à la bonne échelle.

A.4. Panorama des résultats par chapitre

Problèmes à degré cible : Bounded Degree Vertex Deletion et Defective Coloring

Le premier chapitre de recherche est consacré à deux problèmes classiques et étroitement liés. Dans Bounded Degree Vertex Deletion, on cherche à supprimer au plus k sommets pour obtenir un graphe de degré maximum au plus Δ. Dans Defective Coloring, on souhaite colorier les sommets en un nombre donné de couleurs de sorte que chaque classe induise un sous-graphe de degré maximum au plus Δ. Ces deux problèmes généralisent respectivement Vertex Cover et Coloring lorsque $\Delta = 0$. Ils sont étudiés en détail dans Chapter 3.

Algorithmes naturels et optimalité sous largeur d'arbre. Le point de départ est l'existence d'algorithmes dynamiques simples sur une décomposition en arbre. Pour Bounded Degree Vertex Deletion, l'état associé à un sommet du sac peut essentiellement enregistrer son degré résiduel autorisé ou sa suppression ; pour Defective Coloring, il faut mémoriser à la fois une couleur et une information de degré local. On obtient ainsi des algorithmes de type $(\Delta + 2)^{\mathrm{tw}} n^{\mathcal{O}(1)}$ et $(\chi_d(\Delta + 1))^{\mathrm{tw}} n^{\mathcal{O}(1)}$.

La contribution essentielle du chapitre est de montrer que ces algorithmes sont essentiellement optimaux sous SETH, même si l'on remplace la largeur d'arbre par la largeur de chemin. Autrement dit, les tables de programmation dynamique "évidentes" sont déjà les meilleures possibles au niveau exponentiel. Ce type de résultat est précieux : il ne dit pas seulement qu'un problème est FPT, il explique pourquoi des années de recherche n'ont pas permis de réduire la base de l'exponentielle.

Profondeur d'arbre : pas d'accélération qualitative. Une fois établie l'optimalité sous largeur d'arbre, il est naturel de se demander si la paramétrisation plus restrictive par la profondeur d'arbre permettrait des gains qualitatifs. Des bornes inférieures antérieures étaient trop faibles pour exclure une telle possibilité. La thèse ferme cette porte.

À l'aide de la construction récursive mentionnée plus haut, elle montre que ni Bounded Degree Vertex Deletion ni Defective Coloring ne peuvent être résolus en temps $n^{o(\mathrm{td})}$ sous ETH. Le résultat est conceptuellement fort : il montre que la profondeur d'arbre, malgré son caractère très restrictif, ne suffit pas à obtenir une accélération sublinéaire dans l'exposant pour ces problèmes. La structure hiérarchique du graphe reste encore assez riche pour coder une difficulté essentiellement linéaire par rapport au paramètre.

Couverture de sommets : bornes serrées sous ETH. Le chapitre s'intéresse ensuite au paramètre couverture de sommets, sous lequel les deux problèmes deviennent FPT. Les meilleurs algorithmes connus ont alors une dépendance de la forme $\mathrm{vc}^{\mathcal{O}(\mathrm{vc})}$. La question est de savoir si cette dépendance super-exponentielle peut être abaissée.

La réponse apportée par la thèse est négative : une amélioration en $\mathrm{vc}^{o(\mathrm{vc})}$ contredirait ETH (voir Theorems 3.49 and 3.54). Le résultat complète ainsi un panorama presque exhaustif : pour la largeur d'arbre, la largeur de chemin, la profondeur d'arbre et la couverture de sommets, la complexité de ces deux problèmes est déterminée de manière serrée. Le chapitre joue donc un rôle fondateur dans l'économie générale de la thèse. Il introduit deux des grandes techniques réutilisées ensuite, tout en établissant un premier ensemble de résultats "optimaux".

Portée méthodologique du chapitre. Ce premier chapitre ne constitue pas seulement une étude de cas. Il introduit aussi une partie du vocabulaire technique et des méthodes qui seront réemployés dans la suite du manuscrit : la tension entre algorithmique naturelle et bornes fines, l'importance d'une distinction soigneuse entre paramètres structurels, ainsi que le rôle de constructions spécifiquement adaptées au paramètre considéré. En ce sens, il joue un rôle de laboratoire pour le reste de la thèse.

Surtout, il montre qu'un résultat négatif peut être informatif au même titre qu'un résultat positif. Établir qu'un problème est difficile sous largeur d'arbre n'est qu'une première étape. Montrer ensuite que cette difficulté demeure linéaire sous profondeur d'arbre et devient super-exponentielle sous couverture de sommets permet d'en mesurer beaucoup plus précisément le coût algorithmique.

Induced Matching et Acyclic Matching

Le deuxième chapitre porte sur deux variantes sophistiquées du problème de matching maximum. Dans Induced Matching, les arêtes choisies doivent former un matching induit. Dans Acyclic Matching, les sommets incidents aux arêtes du matching doivent induire une forêt. Ces contraintes introduisent une forte composante globale, malgré l'apparente localité du matching lui-même (voir Chapter 4).

Optimalité fine sous largeur de chemin. Des résultats récents avaient déjà donné des algorithmes paramétrés par la largeur d'arbre, mais des écarts subsistaient entre bornes supérieures et inférieures. La thèse resserre ce tableau de manière décisive.

Pour Acyclic Matching, elle affine l'analyse d'un algorithme connu et montre que sa complexité est en réalité $5^{\mathrm{tw}}n^{\mathcal{O}(1)}$, meilleure que l'estimation annoncée précédemment. Elle établit ensuite une borne inférieure sous pw-SETH montrant que cette base 5 ne peut être améliorée, même pour la largeur de chemin (voir Theorems 4.17 and 4.18).

Pour Induced Matching, le résultat principal est que l'algorithme $3^{\mathrm{tw}}n^{\mathcal{O}(1)}$ est optimal sous pw-SETH. Plus encore, la thèse obtient une forme d'équivalence : améliorer cette base pour la largeur de chemin reviendrait à réfuter pw-SETH. Ce type d'énoncé est particulièrement satisfaisant du point de vue de la complexité fine, car il établit une correspondance précise entre progrès algorithmique et renversement d'hypothèses fondamentales.

Largeur de clique et représentations d'états. Le chapitre ne se limite pas aux paramètres de type arbre. Il étudie également la largeur de clique, paramètre plus général, sous lequel les contraintes globales des problèmes de matching deviennent plus délicates à gérer. Pour Induced Matching, la thèse construit un algorithme FPT en temps $3^{\mathrm{cw}}n^{\mathcal{O}(1)}$, en s'appuyant sur l'idée de *représentations d'états*.

Ici encore, la portée du résultat dépasse l'énoncé isolé. Ce résultat montre que des outils développés pour d'autres problèmes et d'autres paramètres peuvent être adaptés avec succès à un cadre plus général. Le chapitre illustre ainsi une autre facette de la démarche adoptée dans la thèse : les bons algorithmes paramétrés reposent souvent sur une compréhension très fine de l'information qu'il faut vraiment transporter d'un sous-problème à l'autre.

Portée des résultats. Ces résultats montrent que les techniques mobilisées pour les problèmes à degré cible ne sont pas spécifiques à ce cadre. Elles peuvent être prolongées vers des problèmes de sélection de sous-structures plus délicats, tout en conservant un niveau d'analyse fine très élevé. Ils soulignent aussi l'intérêt de pw-SETH pour distinguer avec précision la difficulté de problèmes étudiés sous des paramètres linéaires.

Maximum Node-Disjoint Paths

Le troisième chapitre est consacré à Maximum Node-Disjoint Paths, problème d'optimisation où l'on cherche à connecter le plus grand nombre possible de paires terminales par des chemins intérieurement disjoints. Ce problème est central en algorithmique des graphes, à la fois pour ses liens avec la théorie des mineurs et parce qu'il se situe à la rencontre de l'optimisation, de la structure et de l'approximation (voir Chapter 5).

Pourquoi le problème est particulièrement intéressant. Sous de nombreuses paramétrisations structurales naturelles, le problème résiste au calcul exact. Cette résistance en fait un candidat idéal pour l'approximation paramétrée : si le calcul exact est trop ambitieux, peut-on au moins approcher efficacement l'optimum lorsque la structure du graphe est simple ? Le chapitre répond à cette question de manière particulièrement nette.

Un schéma d'approximation FPT sous profondeur d'arbre. Le résultat principal est un schéma d'approximation FPT efficace pour la paramétrisation par la profondeur d'arbre. Pour tout $\varepsilon > 0$, l'algorithme calcule une solution $(1-\varepsilon)$-approchée en temps $f(\mathrm{td}, \varepsilon)n^{\mathcal{O}(1)}$. Il s'agit d'un résultat fort, car il montre qu'un problème récalcitrant sous le point de vue exact devient approchable de manière très précise dès lors que la structure hiérarchique du graphe est suffisamment contrainte. Ce résultat est formalisé dans Theorem 5.5.

L'intérêt de la preuve tient à sa simplicité conceptuelle. L'algorithme combine codage par couleurs, contrôle de la taille utile de l'instance, et argument *win/win*. Si une solution optimale utilise peu d'objets critiques, on peut les deviner ou les isoler efficacement. Dans le cas contraire, la richesse même de la solution permet de perdre une petite fraction de l'optimum tout en conservant une garantie globale. La conception du schéma illustre parfaitement la logique de l'approximation paramétrée : la structure n'annule pas la difficulté du problème, mais elle permet de la "régulariser" suffisamment pour rendre possible une approximation quasi optimale.

Une frontière nette d'inapproximabilité sous largeur de chemin. Le chapitre établit également que ce phénomène a des limites précises. Sous l'hypothèse PIH, le chapitre exclut déjà l'existence d'une approximation à facteur constant, pour une certaine constante, en temps $f(\mathrm{pw}, \mathrm{fvs})n^{\mathcal{O}(1)}$ sous la paramétrisation combinée par la largeur de chemin et le nombre de sommets de retour. En particulier, un schéma d'approximation FPT pour une paramétrisation plus générale fondée sur la largeur de chemin serait incompatible avec cette borne négative. Autrement dit, il existe une véritable frontière structurelle entre "difficile mais approximable" et "difficile même à approximer". Cette borne d'inapproximabilité est démontrée dans Theorem 5.6.

Cette dichotomie constitue l'un des apports conceptuels les plus marquants du travail. Elle montre que l'approximation paramétrée n'est pas seulement un outil de secours lorsque le calcul exact échoue. Elle possède sa propre géographie, et cette géographie est fortement corrélée aux paramètres structuraux du graphe.

Portée du chapitre. Le chapitre complète ce tableau par plusieurs résultats de tractabilité et de dureté, notamment sous profondeur d'arbre et sous combinaisons avec largeur de chemin, comme le montrent Sections 5.1, 5.2, and 5.4. Sa fonction dans l'économie générale de la thèse est toutefois plus large : il constitue la démonstration la plus nette du fait que l'approximation paramétrée peut, dans certains régimes, ouvrir un espace algorithmique qui reste fermé au calcul exact.

Vertex Integrity et Component Order Connectivity

Le quatrième chapitre est consacré à deux problèmes de coupure intimement liés. L'idée commune est de supprimer un petit ensemble de sommets afin de contrôler la taille des composantes connexes restantes. Ces problèmes se situent à mi-chemin entre les problèmes de séparation locale et les objectifs plus globaux de désagrégation du graphe (voir Chapter 6).

Un nouveau saut dans la compréhension de la dureté W[1]. Le résultat principal est une amélioration substantielle des bornes de dureté W[1]. La thèse montre que Vertex Integrity reste W[1]-difficile sous le paramètre combiné $\text{fes} + \Delta + \text{pw}$, et que Component Order Connectivity le reste sous le paramètre $p + \text{fes} + \Delta + \text{pw}$. Il s'agit d'un renforcement important par rapport aux résultats antérieurs : la difficulté persiste alors même que le graphe est proche d'une forêt, de degré borné, et de petite largeur de chemin (voir Theorems 6.15 and 6.16).

Ce résultat est précisément rendu possible par la nouvelle réduction fondée sur la variante à deux choix de Unary Bin Packing. Le chapitre constitue ainsi le premier terrain d'application majeur de cette technique, et il en révèle toute la puissance. L'amélioration obtenue n'est pas seulement quantitative. Elle modifie qualitativement notre compréhension de la frontière FPT/W[1] pour ces problèmes.

Résultats positifs complémentaires. Le chapitre ne se limite pas aux résultats négatifs. Il obtient également plusieurs résultats de tractabilité, par exemple sous nombre maximal de feuilles, largeur modulaire, ou couverture de sommets pour la version pondérée (voir Theorems 6.20, 6.21, 6.24, and 6.28). Il montre en outre que l'approximation paramétrée permet de contourner la barrière de l'exactitude sous largeur d'arbre pour Vertex Integrity et Component Order Connectivity, via des schémas d'approximation FPT de type $(1 + \varepsilon)$ (voir Theorems 6.29 and 6.30). Ces résultats jouent un rôle important dans l'équilibre de la thèse : ils montrent que la difficulté démontrée sous certains paramètres très restrictifs n'empêche pas l'existence d'algorithmes FPT sous d'autres mesures structurelles, parfois encore plus adaptées au problème.

L'ensemble dessine donc une frontière fine plutôt qu'un verdict uniforme. Il ne suffit pas de dire que le problème est "difficile" ou "facile". Il faut préciser pour quelle notion de structure. Cette conclusion est emblématique de la philosophie générale de la thèse.

Interprétation conceptuelle. Vertex Integrity est aussi, à l'origine, un paramètre structural. Étudier la complexité du problème qui consiste à calculer ce paramètre ajoute un niveau de réflexivité intéressant : on analyse algorithmiquement une mesure elle-même conçue pour capturer la structure. Le chapitre montre que, même dans ce contexte, la situation algorithmique est subtile : certaines mesures voisines suffisent à restaurer la tractabilité, tandis que des combinaisons de paramètres très sévères ne suffisent pas.

Portée du chapitre. Dans l'économie de la thèse, ce chapitre occupe une position charnière. Il montre que la nouvelle technique de réduction ne conduit pas à un résultat isolé, mais qu'elle s'insère dans une analyse plus large où coexistent résultats positifs, résultats négatifs et comparaison systématique entre plusieurs paramètres. Il prépare ainsi le terrain pour le chapitre suivant, dans lequel la même idée de réduction est étendue à un problème à objectif encore plus global.

Critical Node Cut

Le cinquième chapitre prolonge l'étude des problèmes de coupure, mais avec un objectif global différent. Dans Critical Node Cut, on cherche à supprimer au plus k sommets afin de réduire le nombre total de paires de sommets encore connectées. Le problème généralise Vertex Cover et trouve des motivations dans la conception de réseaux, l'épidémiologie ou l'analyse des réseaux sociaux (voir Chapter 7).

Dureté sous paramètres très restreints. Le résultat principal est une preuve de dureté W[1] pour le paramètre combiné $k+\text{fes}+\Delta+\text{pw}$. Là encore, il s'agit d'un net renforcement par rapport à la littérature antérieure, qui ne considérait qu'une dureté sous $k + \text{tw}$. Le chapitre montre donc que la difficulté du problème subsiste même sur des graphes très structurés (voir Theorem 7.2).

Par rapport au chapitre précédent, l'analyse devient cependant plus délicate. Pour Critical Node Cut, l'objectif dépend d'une quantité globale : le nombre de paires encore connectées après suppression. Il faut donc contrôler non seulement la taille locale des composantes, mais également l'effet agrégé de toute la construction. La réussite de la réduction atteste ainsi de la robustesse de la technique introduite dans la thèse.

Algorithmes exacts, approximation et noyaux. Le chapitre propose également plusieurs résultats positifs : algorithmes FPT sous nombre maximal de feuilles, intégrité de sommet ou largeur modulaire, algorithme polynomial pour largeur de clique constante, et schéma d'approximation FPT sous largeur d'arbre. Ce dernier s'appuie sur une technique de Lampis et montre qu'une approximation très fine reste possible lorsque la structure de type arbre est disponible (voir Theorems 7.8, 7.11 to 7.13, and 7.17).

Enfin, le chapitre prouve l'absence de noyau polynomial sous le paramètre couverture de sommets, sauf effondrement d'hypothèses standards (voir Theorem 7.22). Cette dernière dimension enrichit encore le panorama. La thèse ne se contente donc pas de classer les problèmes selon FPT/W[1] ni de donner des bornes de temps fines. Elle examine aussi la possibilité de prétraitements efficaces, ce qui complète la compréhension structurelle du problème.

Portée du chapitre. Critical Node Cut occupe une place particulière dans l'ensemble de la thèse, parce qu'il mobilise presque tous les thèmes qui la traversent. Comme Vertex Integrity, il s'interprète naturellement comme un problème de coupure, mais son objectif est plus global. Il combine ainsi une forte résistance sous des paramètres très contraints, des résultats positifs sous plusieurs mesures structurelles, des questions d'approximation paramétrée, ainsi qu'un aspect de noyautisation. En cela, il joue un rôle de synthèse et montre que les outils développés dans le manuscrit ne sont pas attachés à une seule famille de problèmes.

Cohérence d'ensemble des cinq chapitres

Pris séparément, chacun des chapitres apporte des résultats nouveaux sur des problèmes bien établis. Pris ensemble, ils dessinent une image beaucoup plus cohérente. On y voit apparaître trois motifs récurrents : la comparaison entre paramètres structuraux voisins ; la mise en évidence de dépendances optimales au paramètre ; et l'usage de l'approximation comme voie alternative lorsque le calcul exact se heurte à des barrières solides.

Cette cohérence est renforcée par la réutilisation des techniques. La réduction fondée sur Unary Bin Packing irrigue plusieurs chapitres. Les idées de complexité fine développées pour les problèmes à degré cible réapparaissent dans les problèmes de matching. De même, l'approximation paramétrée, introduite pour Maximum Node-Disjoint Paths, trouve un écho dans Critical Node Cut (voir Chapters 3 to 7). Il se dégage ainsi une vision cumulative de la recherche : chaque résultat particulier compte, mais sa valeur augmente encore lorsqu'il vient enrichir une boîte à outils commune.

A.5. Analyse transversale des apports scientifiques

Délimiter finement la frontière FPT / W[1]

Une première leçon générale de la thèse est que la frontière entre FPT et dureté W[1] peut être beaucoup plus proche qu'on ne le pense. Des problèmes qui deviennent traitables sous un paramètre donné restent W[1]-difficiles sous un paramètre seulement un peu plus général, voire sous une combinaison de paramètres très restrictifs. Ce phénomène apparaît nettement pour Vertex Integrity, Component Order Connectivity et Critical Node Cut, à travers Theorems 6.15, 6.16, and 7.2.

Cette observation a une portée conceptuelle importante : l'existence d'une structure visible dans le graphe ne suffit pas, à elle seule. Tout dépend du type précis de structure mesuré. Deux paramètres qui semblent proches intuitivement peuvent conduire à des statuts algorithmiques très différents. La thèse contribue ainsi à une cartographie plus nuancée des paramètres structuraux, où l'on ne parle pas simplement de "graphes simples", mais de formes distinctes de simplicité ayant des effets algorithmiques différents.

Montrer que les programmations dynamiques naturelles sont optimales

Un deuxième apport transversal consiste à justifier la pertinence des programmations dynamiques classiques sur décompositions structurales. Souvent, face à un algorithme de temps $c^{\mathrm{tw}}n^{\mathcal{O}(1)}$, on peut soupçonner qu'il n'est qu'une première approximation technique, susceptible d'être améliorée par une idée plus subtile. Les bornes obtenues dans la thèse montrent au contraire que, pour plusieurs problèmes, la taille même des états est inévitable.

Cette conclusion a une valeur pratique et théorique. Pratique, parce qu'elle oriente la recherche future : plutôt que d'espérer une réduction miraculeuse de la base exponentielle, il faut probablement changer de paramètre, chercher une approximation, ou enrichir le

modèle. Théorique, parce qu'elle fait apparaître une correspondance profonde entre la structure combinatoire des solutions partielles et les hypothèses de la complexité fine.

Aller au-delà de la largeur d'arbre

La largeur d'arbre occupe une place centrale en algorithmique paramétrée, mais la thèse montre qu'il est crucial de ne pas s'y limiter. D'une part, des paramètres plus restrictifs comme la profondeur d'arbre ou la couverture de sommets permettent de mieux comprendre les raisons structurelles de la difficulté. D'autre part, ces paramètres exigent des techniques de preuve spécifiques.

La contribution de la thèse est donc aussi méthodologique au sens large : elle illustre comment adapter les preuves de bornes inférieures aux paramètres considérés, au lieu d'essayer d'appliquer uniformément des schémas conçus pour la seule largeur d'arbre. Cette attention à la géométrie propre de chaque paramètre constitue l'un des fils conducteurs les plus importants de la thèse.

Approximation comme outil structurel

Une autre idée forte de la thèse est que l'approximation n'est pas seulement une manière de renoncer à l'exactitude. Dans le cadre paramétré, elle devient un instrument d'analyse de la structure. Le fait qu'un schéma d'approximation FPT existe sous profondeur d'arbre pour Maximum Node-Disjoint Paths, alors qu'une borne négative forte subsiste pour des paramétrisations plus générales fondées sur la largeur de chemin, montre que la capacité d'approximer elle-même dépend finement du paramètre (voir Theorems 5.5 and 5.6).

Cette perspective ouvre un espace de recherche particulièrement riche. Elle invite à ne plus poser la seule question "le problème est-il FPT ?", mais également : "si ce n'est pas le cas, quel niveau d'approximation devient possible sous quel paramètre ?" La thèse répond de manière exemplaire à cette question sur plusieurs cas d'étude, et suggère une méthodologie généralisable.

Comparer les paramètres, et pas seulement les problèmes

Une dernière leçon transversale concerne la comparaison elle-même des paramètres. Dans beaucoup de travaux, le paramètre est fixé une fois pour toutes et sert simplement de cadre d'analyse. Ici, au contraire, la comparaison entre paramètres est un objet scientifique à part entière. Il ne s'agit pas d'un simple décor. Demander si la largeur de chemin suffit là où la largeur d'arbre suffisait déjà, ou si la profondeur d'arbre permet de gagner qualitativement sur la largeur d'arbre, revient à étudier la robustesse de la structure vis-à-vis d'un problème donné.

Cette manière de faire conduit à des conclusions particulièrement fines. Elle montre par exemple que des paramètres apparemment "plus forts" ne donnent pas toujours des algorithmes qualitativement meilleurs, tandis que dans d'autres cas ils ouvrent réellement de nouvelles possibilités, notamment du côté de l'approximation. La thèse contribue

ainsi non seulement à la complexité de plusieurs problèmes, mais aussi à une meilleure compréhension comparative des paramètres eux-mêmes.

A.6. Contribution principale et originalité

Une boîte à outils réutilisable

Si l'on doit résumer en une idée la contribution principale de la thèse, ce n'est pas seulement l'obtention d'une collection de théorèmes nouveaux sur cinq familles de problèmes. C'est la mise au point d'une *boîte à outils* pour l'étude de la complexité paramétrée structurale. Cette boîte à outils comprend à la fois des techniques de réduction, des techniques de bornes inférieures fines, et des patrons de conception d'algorithmes d'approximation.

Cette dimension est essentielle. Dans un domaine où les résultats peuvent facilement rester dispersés d'un problème à l'autre, la thèse cherche au contraire à faire émerger des principes généraux. La variante à deux choix de Unary Bin Packing n'est pas présentée comme une curiosité isolée, mais comme une méthode pour produire de nouvelles duretés W[1] sous contraintes structurelles. La construction récursive pour la profondeur d'arbre n'est pas un gadget ad hoc, mais une manière nouvelle d'exploiter la définition récursive du paramètre. Les familles d-détectantes deviennent un outil de compression pour les bornes inférieures sous couverture de sommets. Enfin, le schéma d'approximation pour Maximum Node-Disjoint Paths montre qu'un algorithme peut être à la fois techniquement simple et conceptuellement puissant.

Des résultats serrés plutôt que de simples séparations

Un autre aspect original de la thèse est l'insistance sur les résultats *serrés*. Il ne s'agit pas simplement de prouver qu'un problème est FPT, ou qu'il est W[1]-difficile, mais d'aller jusqu'au point où borne supérieure et borne inférieure se répondent. Cette quête d'optimalité apparaît dans presque tous les chapitres : bases exponentielles optimales sous largeur d'arbre ou largeur de chemin, exposants linéaires optimaux sous profondeur d'arbre, dépendances super-exponentielles inévitables sous couverture de sommets.

Ce choix donne à l'ensemble de la thèse une grande cohérence scientifique. Plutôt que d'accumuler des résultats ponctuels, il construit progressivement une image précise du paysage de complexité. Dans cette image, les algorithmes et les bornes inférieures ne sont pas des objets séparés : ils se répondent et se justifient mutuellement. L'originalité du manuscrit tient enfin à l'articulation étroite entre complexité paramétrée, complexité fine paramétrée et approximation paramétrée. Les résultats de dureté W[1] indiquent où l'on ne doit plus espérer d'algorithmes exacts FPT. Les bornes sous ETH, SETH ou pw-SETH expliquent pourquoi certains algorithmes FPT connus ne peuvent vraisemblablement pas être améliorés. L'approximation paramétrée intervient enfin lorsque le calcul exact est bloqué, tout en demeurant elle-même soumise à des frontières structurelles.

A.7. Perspectives ouvertes

Comme tout travail de recherche, cette thèse ferme certaines questions mais en ouvre d'autres. Plusieurs directions ressortent naturellement.

Une première consiste à étendre la nouvelle technique de réduction issue de Unary Bin Packing à d'autres problèmes de coupure ou de suppression. Les résultats obtenus ici suggèrent que de nombreuses duretés W[1] pourraient être renforcées sous des paramètres plus stricts qu'on ne le pensait.

Une deuxième direction concerne les bornes inférieures sous des paramètres intermédiaires. Par exemple, entre largeur d'arbre, nombre d'arêtes de retour, profondeur d'arbre et couverture de sommets, il reste souvent des zones mal comprises. La mise au point de gadgets adaptés à chacun de ces paramètres pourrait conduire à de nouveaux résultats serrés.

Une troisième direction, particulièrement prometteuse, est l'approximation paramétrée. Le schéma obtenu pour Maximum Node-Disjoint Paths et les résultats sur Critical Node Cut suggèrent que d'autres problèmes structurellement difficiles pourraient néanmoins admettre des schémas d'approximation FPT sous des paramètres bien choisis. À l'inverse, il serait intéressant de mieux comprendre quelles hypothèses d'inapproximabilité sont réellement nécessaires pour exclure de tels schémas.

Enfin, plusieurs questions ouvertes propres aux chapitres demeurent. Pour les problèmes à degré cible, on peut chercher des bornes inférieures serrées sous d'autres paramètres. Pour les problèmes de matching, la complexité optimale sous largeur de clique ou sous largeur de coupure mérite d'être davantage clarifiée. Pour Maximum Node-Disjoint Paths, la possibilité d'algorithmes FPT ou d'algorithmes d'approximation à facteur constant sous certains paramètres reste ouverte.

A.8. Conclusion générale

Cette thèse propose une étude unifiée de plusieurs problèmes NP-difficiles sur les graphes à travers le prisme des paramètres structuraux. Elle montre que l'analyse de leur complexité gagne à articuler trois niveaux complémentaires : la frontière FPT/W[1], l'optimalité fine des temps d'exécution, et la capacité de l'approximation à contourner, dans certains régimes, les barrières du calcul exact.

Les résultats obtenus dessinent, pour les problèmes considérés, un paysage précis. La thèse identifie les paramètres qui permettent la tractabilité à paramètres fixes et ceux pour lesquels une telle tractabilité est hors de portée sous des hypothèses usuelles. Lorsqu'un algorithme FPT existe, elle en établit souvent le caractère optimal au regard d'hypothèses standard de complexité fine. Lorsqu'aucun calcul exact efficace n'est plausible, elle montre que l'approximation peut parfois rétablir une forme de tractabilité, mais dans des régimes structurels soigneusement délimités.

L'apport principal de la thèse tient toutefois à sa portée méthodologique. Les techniques développées au fil du manuscrit constituent une contribution autonome à la boîte à outils de la complexité paramétrée structurale. Elles montrent qu'une compréhension plus fine

des problèmes étudiés passe par une adaptation des méthodes à la géométrie propre des paramètres, par une mise en regard systématique des algorithmes et des bornes inférieures, et par une intégration pleine de l'approximation dans l'étude de la structure.

En définitive, la thèse défend l'idée suivante : pour comprendre les problèmes difficiles sur les graphes, il ne suffit pas de demander s'ils sont ou non tractables. Il faut encore déterminer sous quels paramètres ils le deviennent, à quel coût, et dans quelle mesure l'approximation peut modifier ce diagnostic. C'est à cette cartographie, à la fois précise, structurée et outillée, que ce travail entend contribuer.

Bibliography

[1] Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. Seth-based lower bounds for subset sum and bicriteria path. *ACM Trans. Algorithms*, 18(1):6:1–6:22, 2022. doi:10.1145/3450524.

[2] Pierre Aboulker, Édouard Bonnet, Eun Jung Kim, and Florian Sikora. Grundy coloring and friends, half-graphs, bicliques. *Algorithmica*, 85(1):1–28, 2023. doi:10.1007/S00453-022-01001-2.

[3] Bernardetta Addis, Marco Di Summa, and Andrea Grosso. Identifying critical nodes in undirected graphs: Complexity results and polynomial algorithms for the case of bounded treewidth. *Discret. Appl. Math.*, 161(16-17):2349–2360, 2013. doi:10.1016/J.DAM.2013.03.021.

[4] Isolde Adler, Stavros G. Kolliopoulos, Philipp Klaus Krause, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Irrelevant vertices for the planar disjoint paths problem. *J. Comb. Theory, Ser. B*, 122:815–843, 2017. doi:10.1016/J.JCTB.2016.10.001.

[5] Akanksha Agrawal, Daniel Lokshtanov, and Amer E. Mouawad. Critical node cut parameterized by treewidth and solution size is W[1]-hard. In *Graph-Theoretic Concepts in Computer Science - 43rd International Workshop, WG 2017*, volume 10520 of *Lecture Notes in Computer Science*, pages 32–44. Springer, 2017. doi:10.1007/978-3-319-68705-6_3.

[6] Akanksha Agrawal, Dániel Marx, Daniel Neuen, and Jasper Slusallek. Computing square colorings on bounded-treewidth and planar graphs. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023*, pages 2087–2110. SIAM, 2023. doi:10.1137/1.9781611977554.CH80.

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

[8] James A. Andrews and Michael S. Jacobson. On a generalization of chromatic number. *Congressus Numerantium*, 47:33–48, 1985.

[9] Patrizio Angelini, Michael A. Bekos, Felice De Luca, Walter Didimo, Michael Kaufmann, Stephen G. Kobourov, Fabrizio Montecchiani, Chrysanthi N. Raftopoulou, Vincenzo Roselli, and Antonios Symvonis. Vertex-coloring with defects. *J. Graph Algorithms Appl.*, 21(3):313–340, 2017. doi:10.7155/jgaa.00418.

[10] Dan Archdeacon. A note on defective colorings of graphs in surfaces. *J. Graph Theory*, 11(4):517–519, 1987. doi:10.1002/jgt.3190110408.

[11] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[12] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998. doi:10.1145/278298.278306.

[13] Ashwin Arulselvan, Clayton W. Commander, Lily Elefteriadou, and Panos M. Pardalos. Detecting critical nodes in sparse graphs. *Comput. Oper. Res.*, 36(7):2193–2200, 2009. doi:10.1016/J.COR.2008.08.016.

[14] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *SIAM J. Comput.*, 47(3):1087–1097, 2018. doi:10.1137/15M1053128.

[15] Mitali Bafna, Karthik C. S., and Dor Minzer. Near optimal constant inapproximability under ETH for fundamental problems in parameterized complexity. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025*, pages 2118–2129. ACM, 2025. doi:10.1145/3717823.3718257.

[16] Balabhaskar Balasundaram, Sergiy Butenko, and Illya V. Hicks. Clique relaxations in social network analysis: The maximum k-plex problem. *Oper. Res.*, 59(1):133–142, 2011. doi:10.1287/opre.1100.0851.

[17] C. A. Barefoot, Roger Entringer, and Henda Swart. Vulnerability in graphs—a comparative survey. In *Proceedings of the first Carbondale combinatorics conference (Carbondale, Ill., 1986)*, volume 1, pages 13–22, 1987.

[18] Julien Baste, Maximilian Fürst, and Dieter Rautenbach. Linear programming based approximation for unweighted induced matchings - breaking the Δ barrier. *Discret. Optim.*, 38:100593, 2020. doi:10.1016/J.DISOPT.2020.100593.

[19] Julien Baste, Maximilian Fürst, and Dieter Rautenbach. Acyclic matchings in graphs of bounded maximum degree. *Discret. Math.*, 345(7):112885, 2022. doi:10.1016/J.DISC.2022.112885.

[20] Julien Baste and Dieter Rautenbach. Degenerate matchings and edge colorings. *Discret. Appl. Math.*, 239:38–44, 2018. doi:10.1016/J.DAM.2018.01.002.

[21] MohammadHossein Bateni, Mohammad Taghi Hajiaghayi, and Dániel Marx. Approximation schemes for steiner forest on planar graphs and graphs of bounded treewidth. *J. ACM*, 58(5):21:1–21:37, 2011. doi:10.1145/2027216.2027219.

[22] Richard E. Bellman. *Dynamic programming*. Princeton University Press, Princeton, NJ, 1957.

[23] Rémy Belmonte, Tesshu Hanaka, Ioannis Katsikarelis, Michael Lampis, Hirotaka Ono, and Yota Otachi. Parameterized complexity of safe set. *J. Graph Algorithms Appl.*, 24(3):215–245, 2020. doi:10.7155/JGAA.00528.

[24] Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, and Yota Otachi. Grundy distinguishes treewidth from pathwidth. *SIAM J. Discret. Math.*, 36(3):1761–1787, 2022. doi:10.1137/20M1385779.

[25] Rémy Belmonte, Michael Lampis, and Valia Mitsou. Parameterized (approximate) defective coloring. *SIAM J. Discret. Math.*, 34(2):1084–1106, 2020. doi:10.1137/18M1223666.

[26] Rémy Belmonte, Michael Lampis, and Valia Mitsou. Defective coloring on classes of perfect graphs. *Discret. Math. Theor. Comput. Sci.*, 24, 2022. doi:10.46298/dmtcs.4926.

[27] Huck Bennett, Mahdi Cheraghchi, Venkatesan Guruswami, and João Ribeiro. Parameterized inapproximability of the minimum distance problem over all fields and the shortest vector problem in all \({\ell_{{p}}}\) norms. *SIAM J. Comput.*, 53(5):1439–1475, 2024. doi:10.1137/23M1573021.

[28] Matthias Bentert, Klaus Heeger, and Tomohiro Koana. Fully polynomial-time algorithms parameterized by vertex integrity using fast matrix multiplication. In *31st Annual European Symposium on Algorithms, ESA 2023*, volume 274 of *LIPIcs*, pages 16:1–16:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ESA.2023.16.

[29] Benjamin Bergougnoux and Mamadou Moustapha Kanté. Fast exact algorithms for some connectivity problems parameterized by clique-width. *Theor. Comput. Sci.*, 782:30–53, 2019. doi:10.1016/J.TCS.2019.02.030.

[30] Benjamin Bergougnoux and Mamadou Moustapha Kanté. More applications of the d-neighbor equivalence: Acyclicity and connectivity constraints. *SIAM J. Discret. Math.*, 35(3):1881–1926, 2021. doi:10.1137/20M1350571.

[31] Umberto Bertele and Francesco Brioschi. *Nonserial dynamic programming*. Academic Press, Inc., 1972.

[32] Nadja Betzler, Hans L. Bodlaender, Robert Bredereck, Rolf Niedermeier, and Johannes Uhlmann. On making a distinguished vertex of minimum degree by vertex deletion. *Algorithmica*, 68(3):715–738, 2014. doi:10.1007/s00453-012-9695-6.

[33] Nadja Betzler, Robert Bredereck, Rolf Niedermeier, and Johannes Uhlmann. On bounded-degree vertex deletion parameterized by treewidth. *Discret. Appl. Math.*, 160(1-2):53–60, 2012. doi:10.1016/j.dam.2011.08.013.

[34] Nadja Betzler and Johannes Uhlmann. Parameterized complexity of candidate control in elections and related digraph problems. *Theor. Comput. Sci.*, 410(52):5425–5442, 2009. doi:10.1016/j.tcs.2009.05.029.

[35] Arnab Bhattacharyya, Édouard Bonnet, László Egri, Suprovat Ghoshal, Karthik C. S., Bingkai Lin, Pasin Manurangsi, and Dániel Marx. Parameterized intractability of even set and shortest vector problem. *J. ACM*, 68(3):16:1–16:40, 2021. doi:10.1145/3444942.

[36] Sriram Bhyravarapu, Lawqueen Kanesh, A. Mohanapriya, Nidhi Purohit, N. Sadagopan, and Saket Saurabh. On the parameterized complexity of minus domination. In *SOFSEM 2024: Theory and Practice of Computer Science - 49th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2024*, volume 14519 of *Lecture Notes in Computer Science*, pages 96–110. Springer, 2024. doi:10.1007/978-3-031-52113-3_7.

[37] 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*, pages 67–74. ACM, 2007. doi:10.1145/1250790.1250801.

[38] Hans 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.

[39] 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.

[40] 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.

[41] Hans L. Bodlaender, Carla Groenland, Jesper Nederlof, and Céline M. F. Swennenhuis. Parameterized problems complete for nondeterministic FPT time and logarithmic space. *Inf. Comput.*, 300:105195, 2024. doi:10.1016/J.IC.2024.105195.

[42] Hans L. Bodlaender and Torben Hagerup. Parallel algorithms with optimal speedup for bounded treewidth. *SIAM J. Comput.*, 27(6):1725–1746, 1998. doi:10.1137/S0097539795289859.

[43] Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21(2):358–402, 1996. doi:10.1006/JAGM.1996.0049.

[44] Hans L. Bodlaender, Erik Jan van Leeuwen, Johan M. M. van Rooij, and Martin Vatshelle. Faster algorithms on branch and clique decompositions. In *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010*, volume 6281 of *Lecture Notes in Computer Science*, pages 174–185. Springer, 2010. doi:10.1007/978-3-642-15155-2_17.

[45] Narek Bojikian, Vera Chekan, Falko Hegerfeld, and Stefan Kratsch. Tight bounds for connectivity problems parameterized by cutwidth. In *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023*, volume 254 of *LIPIcs*, pages 14:1–14:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.STACS.2023.14.

[46] Narek Bojikian, Vera Chekan, and Stefan Kratsch. Tight bounds for some classical problems parameterized by cutwidth. In *33rd Annual European Symposium on Algorithms, ESA 2025*, volume 351 of *LIPIcs*, pages 13:1–13:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPICS.ESA.2025.13.

[47] Narek Bojikian and Stefan Kratsch. A tight monte-carlo algorithm for steiner tree parameterized by clique-width. In *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024*, volume 297 of *LIPIcs*, pages 29:1–29:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ICALP.2024.29.

[48] Narek Bojikian and Stefan Kratsch. Tight bounds for connected odd cycle transversal parameterized by clique-width. In *20th International Symposium on Parameterized and Exact Computation, IPEC 2025*, volume 358 of *LIPIcs*, pages 19:1–19:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPICS.IPEC.2025.19.

[49] Marthe Bonamy, Lukasz Kowalik, Michal Pilipczuk, Arkadiusz Socala, and Marcin Wrochna. Tight lower bounds for the complexity of multicoloring. *ACM Trans. Comput. Theory*, 11(3):13:1–13:19, 2019. doi:10.1145/3313906.

[50] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In *11th International Symposium on Parameterized and Exact Computation, IPEC 2016*, volume 63 of *LIPIcs*, pages 8:1–8:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.IPEC.2016.8.

[51] Andreas Brandstädt, Elaine M. Eschen, and R. Sritharan. The induced matching and chain subgraph cover problems for convex bipartite graphs. *Theor. Comput. Sci.*, 381(1-3):260–265, 2007. doi:10.1016/J.TCS.2007.04.006.

[52] Andreas Brandstädt and Chính T. Hoàng. Maximum induced matchings for chordal graphs in linear time. *Algorithmica*, 52(4):440–447, 2008. doi:10.1007/S00453-007-9045-2.

[53] Karl Bringmann. Fine-grained complexity theory (tutorial). In *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019*, volume 126 of *LIPIcs*, pages 4:1–4:7. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.STACS.2019.4.

[54] Kathie Cameron. Induced matchings. *Discret. Appl. Math.*, 24(1-3):97–102, 1989. doi:10.1016/0166-218X(92)90275-F.

[55] Kathie Cameron, R. Sritharan, and Yingwen Tang. Finding a maximum induced matching in weakly chordal graphs. *Discret. Math.*, 266(1-3):133–142, 2003. doi:10.1016/S0012-365X(02)00803-8.

[56] Parinya Chalermsook, Bundit Laekhanukit, and Danupon Nanongkai. Graph products revisited: Tight approximation hardness of induced matching, poset dimension and more. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013*, pages 1557–1576. SIAM, 2013. doi:10.1137/1.9781611973105.112.

[57] Parinya Chalermsook, Bundit Laekhanukit, and Danupon Nanongkai. Independent set, induced matching, and pricing: Connections and tight (subexponential time) approximation hardnesses. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, pages 370–379. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.47.

[58] Jou-Ming Chang. Induced matchings in asteroidal triple-free graphs. *Discret. Appl. Math.*, 132(1-3):67–78, 2003. doi:10.1016/S0166-218X(03)00390-1.

[59] Maw-Shang Chang, Li-Hsuan Chen, and Ling-Ju Hung. Moderately exponential time algorithms for the maximum induced matching problem. *Optim. Lett.*, 9(5):981–998, 2015. doi:10.1007/S11590-014-0813-Z.

[60] Juhi Chaudhary, Harmender Gahlawat, Michal Wlodarczyk, and Meirav Zehavi. Kernels for the disjoint paths problem on subclasses of chordal graphs. *J. Comput. Syst. Sci.*, 156:103715, 2026. doi:10.1016/J.JCSS.2025.103715.

[61] Juhi Chaudhary and Meirav Zehavi. P-matchings parameterized by treewidth. *SIAM J. Discret. Math.*, 39(2):1280–1311, 2025. doi:10.1137/23M160013X.

[62] Juhi Chaudhary and Meirav Zehavi. Parameterized results on acyclic matchings with implications for related problems. *J. Comput. Syst. Sci.*, 148:103599, 2025. doi:10.1016/J.JCSS.2024.103599.

[63] 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.

[64] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010. doi:10.1016/J.TCS.2010.06.026.

[65] Yijia Chen, Yi Feng, Bundit Laekhanukit, and Yanlin Liu. Simple combinatorial construction of the $k^{o(1)}$-lower bound for approximating the parameterized k-clique. In *2025 Symposium on Simplicity in Algorithms, SOSA 2025*, pages 263–280. SIAM, 2025. doi:10.1137/1.9781611978315.21.

[66] Yijia Chen and Bingkai Lin. The constant inapproximability of the parameterized dominating set problem. *SIAM J. Comput.*, 48(2):513–533, 2019. doi:10.1137/17M1127211.

[67] Yijia Chen and Bingkai Lin. Parameterized inapproximability: From clique to PIH. *Comput. Sci. Rev.*, 59:100834, 2026. doi:10.1016/J.COSREV.2025.100834.

[68] Rajesh Chitnis, Andreas Emil Feldmann, and Pasin Manurangsi. Parameterized approximation algorithms for bidirected steiner network problems. *ACM Trans. Algorithms*, 17(2):12:1–12:68, 2021. doi:10.1145/3447584.

[69] Rajesh Chitnis, Andreas Emil Feldmann, and Ondrej Suchý. A tight lower bound for planar steiner orientation. *Algorithmica*, 81(8):3200–3216, 2019. doi:10.1007/S00453-019-00580-X.

[70] Miroslav Chlebík and Janka Chlebíková. Complexity of approximating bounded variants of optimization problems. *Theor. Comput. Sci.*, 354(3):320–338, 2006. doi:10.1016/J.TCS.2005.11.029.

[71] Miroslav Chlebík and Janka Chlebíková. Approximation hardness of dominating set problems in bounded degree graphs. *Inf. Comput.*, 206(11):1264–1275, 2008. doi:10.1016/J.IC.2008.07.003.

[72] Kyungjin Cho, Eunjin Oh, and Seunghyeok Oh. Parameterized algorithm for the disjoint path problem on planar graphs: Exponential in k^2 and linear in n. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023*, pages 3734–3758. SIAM, 2023. doi:10.1137/1.9781611977554.ch144.

[73] Ilkyoo Choi and Louis Esperet. Improper coloring of graphs on surfaces. *J. Graph Theory*, 91(1):16–34, 2019. doi:10.1002/jgt.22418.

[74] Huairui Chu and Bingkai Lin. FPT approximation using treewidth: Capacitated vertex cover, target set selection and vector dominating set. In *34th International Symposium on Algorithms and Computation, ISAAC 2023*, volume 283 of *LIPIcs*, pages 19:1–19:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ISAAC.2023.19.

[75] Julia Chuzhoy and David H. K. Kim. On approximating node-disjoint paths in grids. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015*, volume 40 of *LIPIcs*, pages 187–211. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.187.

[76] Julia Chuzhoy, David H. K. Kim, and Shi Li. Improved approximation for node-disjoint paths in planar graphs. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016*, pages 556–569. ACM, 2016. doi:10.1145/2897518.2897538.

[77] Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. Improved approximation for node-disjoint paths in grids with sources on the boundary. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, volume 107 of *LIPIcs*, pages 38:1–38:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICS.ICALP.2018.38.

[78] Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. Almost polynomial hardness of node-disjoint paths in grids. *Theory Comput.*, 17:1–57, 2021. doi:10.4086/toc.2021.v017a006.

[79] Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. New hardness results for routing on disjoint paths. *SIAM J. Comput.*, 51(2):17–189, 2022. doi:10.1137/17m1146580.

[80] Lane H. Clark, Roger C. Entringer, and Michael R. Fellows. Computational complexity of integrity. *J. Combin. Math. Combin. Comput.*, 2:179–191, 1987.

[81] Vincent Cohen-Addad, Anupam Gupta, Amit Kumar, Euiwoong Lee, and Jason Li. Tight FPT approximations for k-median and k-means. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019*, volume 132 of *LIPIcs*, pages 42:1–42:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.ICALP.2019.42.

[82] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158. ACM, 1971. doi:10.1145/800157.805047.

[83] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

[84] Derek G. Corneil and Udi Rotics. On the relationship between clique-width and treewidth. *SIAM J. Comput.*, 34(4):825–847, 2005. doi:10.1137/S0097539701385351.

[85] 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.

[86] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000. doi:10.1007/S002249910009.

[87] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discret. Appl. Math.*, 101(1-3):77–114, 2000. doi:10.1016/S0166-218X(99)00184-5.

[88] Lenore J. Cowen, Robert Cowen, and Douglas R. Woodall. Defective colorings of graphs in surfaces: Partitions into subgraphs of bounded valency. *J. Graph Theory*, 10(2):187–195, 1986. doi:10.1002/jgt.3190100207.

[89] 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*, pages 1080–1099. SIAM, 2018. doi:10.1137/1.9781611975031.70.

[90] 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*, pages 1650–1669. SIAM, 2016. doi:10.1137/1.9781611974331.CH113.

[91] Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3):41:1–41:24, 2016. doi:10.1145/2925416.

[92] 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.

[93] 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.

[94] 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. *ACM Trans. Algorithms*, 18(2):17:1–17:31, 2022. doi:10.1145/3506707.

[95] Marek Cygan and Marcin Pilipczuk. Exact and approximate bandwidth. *Theor. Comput. Sci.*, 411(40-42):3701–3713, 2010. doi:10.1016/j.tcs.2010.06.018.

[96] Konrad K. Dabrowski, Marc Demange, and Vadim V. Lozin. New results on maximum induced matchings in bipartite graphs and beyond. *Theor. Comput. Sci.*, 478:33–40, 2013. doi:10.1016/J.TCS.2013.01.027.

[97] Marc Demange and Tínaz Ekim. A note on the NP-hardness of two matching problems in induced subgrids. *Discret. Math. Theor. Comput. Sci.*, 15(2):233–242, 2013. doi:10.46298/DMTCS.606.

[98] Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate texts in mathematics*. Springer, 2025. doi:10.1007/978-3-662-70107-2.

[99] Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007. doi:10.1145/1236457.1236459.

[100] Irit Dinur. Mildly exponential reduction from gap 3sat to polynomial-gap label-cover. *Electron. Colloquium Comput. Complex.*, TR16-128, 2016. URL: https://eccc.weizmann.ac.il/report/2016/128, arXiv:TR16-128.

[101] Irit Dinur and Pasin Manurangsi. Eth-hardness of approximating 2-csps and directed steiner network. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018*, volume 94 of *LIPIcs*, pages 36:1–36:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICS.ITCS.2018.36.

[102] Martin Doucha and Jan Kratochvíl. Cluster vertex deletion: A parameterization between vertex cover and clique-width. In *Mathematical Foundations of Computer Science 2012 - 37th International Symposium, MFCS 2012*, volume 7464 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2012. doi:10.1007/978-3-642-32589-2_32.

[103] 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.

[104] Pål Grønås Drange, Markus S. Dregi, and Pim van 't Hof. On the computational complexity of vertex integrity and component order connectivity. *Algorithmica*, 76(4):1181–1202, 2016. doi:10.1007/S00453-016-0127-X.

[105] Louis Dublois, Michael Lampis, and Vangelis Th. Paschos. New algorithms for mixed dominating set. *Discret. Math. Theor. Comput. Sci.*, 23(1), 2021. doi:10.46298/dmtcs.6824.

[106] Louis Dublois, Michael Lampis, and Vangelis Th. Paschos. Upper dominating set: Tight algorithms for pathwidth and sub-exponential approximation. *Theor. Comput. Sci.*, 923:271–291, 2022. doi:10.1016/j.tcs.2022.05.013.

[107] William Duckworth, David F. Manlove, and Michele Zito. On the approximability of the maximum induced matching problem. *J. Discrete Algorithms*, 3(1):79–91, 2005. doi:10.1016/J.JDA.2004.05.001.

[108] Pavel Dvorák, Eduard Eiben, Robert Ganian, Dusan Knop, and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP: programs with few global variables and constraints. *Artif. Intell.*, 300:103561, 2021. doi:10.1016/J.ARTINT.2021.103561.

[109] Pavel Dvorák, Andreas Emil Feldmann, Dusan Knop, Tomás Masarík, Tomás Toufar, and Pavel Veselý. Parameterized approximation schemes for steiner trees with small number of steiner vertices. *SIAM J. Discret. Math.*, 35(1):546–574, 2021. doi:10.1137/18M1209489.

[110] Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. An algorithmic theory of integer programming, 2022. URL: https://arxiv.org/abs/1904.01361, arXiv:1904.01361.

[111] Khaled M. Elbassioni, Rajiv Raman, Saurabh Ray, and René Sitters. On the approximability of the maximum feasible subsystem problem with 0/1-coefficients. In *Pro-*

ceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, pages 1210–1219. SIAM, 2009. doi:10.1137/1.9781611973068.131.

[112] Alina Ene, Matthias Mnich, Marcin Pilipczuk, and Andrej Risteski. On routing disjoint paths in bounded treewidth graphs. In *15th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2016*, volume 53 of *LIPIcs*, pages 15:1–15:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.SWAT.2016.15.

[113] Rok Erman, Lukasz Kowalik, Matjaz Krnc, and Tomasz Walen. Improved induced matchings in sparse graphs. *Discret. Appl. Math.*, 158(18):1994–2003, 2010. doi:10.1016/J.DAM.2010.08.026.

[114] Baris Can Esmer, Jacob Focke, Dániel Marx, and Pawel Rzazewski. List homomorphisms by deleting edges and vertices: Tight complexity bounds for bounded-treewidth graphs. In *32nd Annual European Symposium on Algorithms, ESA 2024*, volume 308 of *LIPIcs*, pages 39:1–39:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ESA.2024.39.

[115] Andreas Emil Feldmann, Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6):146, 2020. doi:10.3390/A13060146.

[116] Andreas Emil Feldmann and Michael Lampis. Parameterized algorithms for steiner forest in bounded width graphs. *ACM Trans. Algorithms*, 21(4):47:1–47:26, 2025. doi:10.1145/3748724.

[117] Michael R. Fellows, Jiong Guo, Hannes Moser, and Rolf Niedermeier. A generalization of nemhauser and trotter's local optimization theorem. *J. Comput. Syst. Sci.*, 77(6):1141–1158, 2011. doi:10.1016/j.jcss.2010.12.001.

[118] Michael R. Fellows, Bart M. P. Jansen, and Frances A. Rosamond. Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. *Eur. J. Comb.*, 34(3):541–566, 2013. doi:10.1016/J.EJC.2012.04.008.

[119] Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Matthias Mnich, Frances A. Rosamond, and Saket Saurabh. The complexity ecology of parameters: An illustration using bounded max leaf number. *Theory Comput. Syst.*, 45(4):822–848, 2009. doi:10.1007/S00224-009-9167-9.

[120] Michael R. Fellows, Frances A. Rosamond, Udi Rotics, and Stefan Szeider. Clique-width is NP-complete. *SIAM J. Discret. Math.*, 23(2):909–939, 2009. doi:10.1137/070687256.

[121] Michael R. Fellows and Sam Stueckle. The immersion order, forbidden subgraphs and the complexity of network integrity. *J. Combin. Math. Combin. Comput.*, 6:23–32, 1989.

[122] Krzysztof Fleszar, Matthias Mnich, and Joachim Spoerhase. New algorithms for maximum disjoint paths based on tree-likeness. *Math. Program.*, 171(1-2):433–461, 2018. doi:10.1007/s10107-017-1199-3.

[123] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. doi:10.1007/3-540-29953-X.

[124] Jacob Focke, Dániel Marx, Fionn Mc Inerney, Daniel Neuen, Govind S. Sankar, Philipp Schepper, and Philip Wellnitz. Tight complexity bounds for counting generalized dominating sets in bounded-treewidth graphs - part I: algorithmic results. *ACM Trans. Algorithms*, 21(3):27:1–27:45, 2025. doi:10.1145/3731452.

[125] Jacob Focke, Dániel Marx, Fionn Mc Inerney, Daniel Neuen, Govind S. Sankar, Philipp Schepper, and Philip Wellnitz. Tight complexity bounds for counting generalized dominating sets in bounded-treewidth graphs part II: hardness results. *ACM Trans. Comput. Theory*, 17(2):10:1–10:101, 2025. doi:10.1145/3708509.

[126] Jacob Focke, Dániel Marx, and Pawel Rzazewski. Counting list homomorphisms from graphs of bounded treewidth: Tight complexity bounds. *ACM Trans. Algorithms*, 20(2):11, 2024. doi:10.1145/3640814.

[127] Fedor V. Fomin, Petr A. Golovach, Tanmay Inamdar, and Tomohiro Koana. FPT approximation and subexponential algorithms for covering few or many edges. *Inf. Process. Lett.*, 185:106471, 2024. doi:10.1016/J.IPL.2024.106471.

[128] Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Intractability of clique-width parameterizations. *SIAM J. Comput.*, 39(5):1941–1956, 2010. doi:10.1137/080742270.

[129] Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Clique-width III: hamiltonian cycle and the odd case of graph coloring. *ACM Trans. Algorithms*, 15(1):9:1–9:27, 2019. doi:10.1145/3280824.

[130] Fedor V. Fomin and Tuukka Korhonen. Fast fpt-approximation of branchwidth. *SIAM J. Comput.*, 53(4):1085–1131, 2024. doi:10.1137/22M153937X.

[131] 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.

[132] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019. doi:10.1017/9781107415157.

[133] Lance Fortnow. The status of the P versus NP problem. *Commun. ACM*, 52(9):78–86, 2009. doi:10.1145/1562164.1562186.

[134] Lance Fortnow. *The Golden Ticket - P, NP, and the Search for the Impossible.* Princeton University Press, 2013.

[135] Florent Foucaud, Esther Galby, Liana Khazaliya, Shaohua Li, Fionn Mc Inerney, Roohani Sharma, and Prafullkumar Tale. Problems in NP can admit double-exponential lower bounds when parameterized by treewidth or vertex cover. In *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024*, volume 297 of *LIPIcs*, pages 66:1–66:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ICALP.2024.66.

[136] Shinya Fujita and Michitaka Furuya. Safe number and integrity of graphs. *Discret. Appl. Math.*, 247:398–406, 2018. doi:10.1016/J.DAM.2018.03.074.

[137] Toshihiro Fujito. A unified approximation algorithm for node-deletion problems. *Discret. Appl. Math.*, 86(2-3):213–231, 1998. doi:10.1016/S0166-218X(98)00035-3.

[138] Toshihiro Fujito. Approximating bounded degree deletion via matroid matching. In *Algorithms and Complexity - 10th International Conference, CIAC 2017*, volume 10236 of *Lecture Notes in Computer Science*, pages 234–246, 2017. doi:10.1007/978-3-319-57586-5_20.

[139] Maximilian Fürst, Marilena Leichter, and Dieter Rautenbach. Locally searching for large induced matchings. *Theor. Comput. Sci.*, 720:64–72, 2018. doi:10.1016/J.TCS.2018.02.006.

[140] Maximilian Fürst and Dieter Rautenbach. On some hard and some tractable cases of the maximum acyclic matching problem. *Ann. Oper. Res.*, 279(1-2):291–300, 2019. doi:10.1007/S10479-019-03311-1.

[141] Ajinkya Gaikwad and Soumen Maity. Offensive alliances in graphs. *Theor. Comput. Sci.*, 989:114401, 2024. doi:10.1016/J.TCS.2024.114401.

[142] Ajinkya Gaikwad and Soumen Maity. On structural parameterizations of the harmless set problem. *Algorithmica*, 86(5):1475–1511, 2024. doi:10.1007/S00453-023-01199-9.

[143] Jakub Gajarský, Michael Lampis, and Sebastian Ordyniak. Parameterized algorithms for modular-width. In *Parameterized and Exact Computation - 8th International Symposium, IPEC 2013*, volume 8246 of *Lecture Notes in Computer Science*, pages 163–176. Springer, 2013. doi:10.1007/978-3-319-03898-8_15.

[144] Robert Ganian, Thekla Hamm, Viktoriia Korchemna, Karolina Okrasa, and Kirill Simonov. The fine-grained complexity of graph homomorphism parameterized by clique-width. *ACM Trans. Algorithms*, 20(3):19, 2024. doi:10.1145/3652514.

[145] Robert Ganian, Fabian Klute, and Sebastian Ordyniak. On structural parameterizations of the bounded-degree vertex deletion problem. *Algorithmica*, 83(1):297–336, 2021. doi:10.1007/s00453-020-00758-8.

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

[147] Archontia C. Giannopoulou, Michal Pilipczuk, Jean-Florent Raymond, Dimitrios M. Thilikos, and Marcin Wrochna. Cutwidth: Obstructions and algorithmic aspects. *Algorithmica*, 81(2):557–588, 2019. doi:10.1007/S00453-018-0424-7.

[148] Tatsuya Gima, Tesshu Hanaka, Masashi Kiyomi, Yasuaki Kobayashi, and Yota Otachi. Exploring the gap between treedepth and vertex cover through vertex integrity. *Theor. Comput. Sci.*, 918:60–76, 2022. doi:10.1016/J.TCS.2022.03.021.

[149] Tatsuya Gima, Tesshu Hanaka, Yasuaki Kobayashi, Ryota Murai, Hirotaka Ono, and Yota Otachi. Structural parameterizations of vertex integrity. *Theor. Comput. Sci.*, 1024:114954, 2025. doi:10.1016/J.TCS.2024.114954.

[150] Tatsuya Gima and Yota Otachi. Extended MSO model checking via small vertex integrity. *Algorithmica*, 86(1):147–170, 2024. doi:10.1007/S00453-023-01161-9.

[151] Wayne Goddard, Sandra Mitchell Hedetniemi, Stephen T. Hedetniemi, and Renu C. Laskar. Generalized subgraph-restricted matchings in graphs. *Discret. Math.*, 293(1-3):129–138, 2005. doi:10.1016/J.DISC.2004.08.027.

[152] Martin Charles Golumbic and Moshe Lewenstein. New results on induced matchings. *Discret. Appl. Math.*, 101(1-3):157–165, 2000. doi:10.1016/S0166-218X(99)00194-8.

[153] Guilherme C. M. Gomes, Bruno Porto Masquio, Paulo E. D. Pinto, Vinícius Fernandes dos Santos, and Jayme Luiz Szwarcfiter. Disconnected matchings. *Theor. Comput. Sci.*, 956:113821, 2023. doi:10.1016/J.TCS.2023.113821.

[154] Jakob Greilhuber, Philipp Schepper, and Philip Wellnitz. Residue domination in bounded-treewidth graphs. In *42nd International Symposium on Theoretical Aspects of Computer Science, STACS 2025*, volume 327 of *LIPIcs*, pages 41:1–41:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPICS.STACS.2025.41.

[155] Carla Groenland, Isja Mannens, Jesper Nederlof, Marta Piecyk, and Pawel Rzazewski. Towards tight bounds for the graph homomorphism problem parameterized by cutwidth via asymptotic matrix parameters. In *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024*, volume 297 of *LIPIcs*, pages 77:1–77:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ICALP.2024.77.

[156] Carla Groenland, Isja Mannens, Jesper Nederlof, and Krisztina Szilágyi. Tight bounds for counting colorings and connected edge sets parameterized by cutwidth. In *39th International Symposium on Theoretical Aspects of Computer Science, STACS*

2022, volume 219 of *LIPIcs*, pages 36:1–36:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.STACS.2022.36.

[157] Venkatesan Guruswami, Bingkai Lin, Xuandi Ren, Yican Sun, and Kewen Wu. Almost optimal time lower bound for approximating parameterized clique, csp, and more, under ETH. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025*, pages 2136–2144. ACM, 2025. doi:10.1145/3717823.3718130.

[158] Venkatesan Guruswami, Bingkai Lin, Xuandi Ren, Yican Sun, and Kewen Wu. Parameterized inapproximability hypothesis under ETH. *J. ACM*, 72(5):32:1–32:40, 2025. doi:10.1145/3749982.

[159] Gregory Z. Gutin, Mark Jones, and Magnus Wahlström. The mixed chinese postman problem parameterized by pathwidth and treedepth. *SIAM J. Discret. Math.*, 30(4):2177–2205, 2016. doi:10.1137/15M1034337.

[160] Michel Habib and Lalla Mouatadid. Maximum induced matching algorithms via vertex ordering characterizations. *Algorithmica*, 82(2):260–278, 2020. doi:10.1007/S00453-018-00538-5.

[161] Sahab Hajebi and Ramin Javadi. On the parameterized complexity of the acyclic matching problem. *Theor. Comput. Sci.*, 958:113862, 2023. doi:10.1016/J.TCS.2023.113862.

[162] Rudolf Halin. S-functions for graphs. *Journal of geometry*, 8(1-2):171–186, 1976.

[163] Tesshu Hanaka, Ioannis Katsikarelis, Michael Lampis, Yota Otachi, and Florian Sikora. Parameterized orientable deletion. *Algorithmica*, 82(7):1909–1938, 2020. doi:10.1007/s00453-020-00679-6.

[164] Tesshu Hanaka, Michael Lampis, Manolis Vasilakis, and Kanae Yoshiwatari. Parameterized vertex integrity revisited. In *49th International Symposium on Mathematical Foundations of Computer Science, MFCS 2024*, volume 306 of *LIPIcs*, pages 58:1–58:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.MFCS.2024.58.

[165] Frédéric Havet, Ross J. Kang, and Jean-Sébastien Sereni. Improper coloring of unit disk graphs. *Networks*, 54(3):150–164, 2009. doi:10.1002/net.20318.

[166] Falko Hegerfeld and Stefan Kratsch. Solving connectivity problems parameterized by treedepth in single-exponential time and polynomial space. In *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020*, volume 154 of *LIPIcs*, pages 29:1–29:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.STACS.2020.29.

[167] Falko Hegerfeld and Stefan Kratsch. Towards exact structural thresholds for parameterized complexity. In *17th International Symposium on Parameterized and*

Exact Computation, IPEC 2022, volume 249 of *LIPIcs*, pages 17:1–17:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.IPEC.2022.17.

[168] Falko Hegerfeld and Stefan Kratsch. Tight algorithms for connectivity problems parameterized by clique-width. In *31st Annual European Symposium on Algorithms, ESA 2023*, volume 274 of *LIPIcs*, pages 59:1–59:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ESA.2023.59.

[169] Danny Hermelin, Moshe Kaspi, Christian Komusiewicz, and Barak Navon. Parameterized complexity of critical node cuts. *Theor. Comput. Sci.*, 651:62–75, 2016. doi:10.1016/J.TCS.2016.08.018.

[170] Danny Hermelin, Stefan Kratsch, Karolina Soltys, Magnus Wahlström, and Xi Wu. A completeness theory for polynomial (turing) kernelization. *Algorithmica*, 71(3):702–730, 2015. doi:10.1007/S00453-014-9910-8.

[171] Christoph Hunkenschröder, Martin Koutecký, Asaf Levin, and Tung Anh Vu. (near)-optimal algorithms for sparse separable convex integer programs. In *Integer Programming and Combinatorial Optimization - 26th International Conference, IPCO 2025*, volume 15620 of *Lecture Notes in Computer Science*, pages 297–311. Springer, 2025. doi:10.1007/978-3-031-93112-3_22.

[172] Russell 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.

[173] 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.

[174] Tanmay Inamdar, Daniel Lokshtanov, Saket Saurabh, and Vaishali Surianarayanan. Parameterized complexity of fair bisection: (fpt-approximation meets unbreakability). In *31st Annual European Symposium on Algorithms, ESA 2023*, volume 274 of *LIPIcs*, pages 63:1–63:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ESA.2023.63.

[175] Lars Jaffke and Bart M. P. Jansen. Fine-grained parameterized complexity analysis of graph coloring problems. *Discret. Appl. Math.*, 327:33–46, 2023. doi:10.1016/j.dam.2022.11.011.

[176] Lars Jaffke, Paloma T. Lima, and Daniel Lokshtanov. b-coloring parameterized by clique-width. *Theory Comput. Syst.*, 68(4):1049–1081, 2024. doi:10.1007/S00224-023-10132-0.

[177] Satyabrata Jana, Daniel Lokshtanov, Soumen Mandal, Ashutosh Rai, and Saket Saurabh. Parameterized approximation scheme for feedback vertex set. In *48th International Symposium on Mathematical Foundations of Computer Science, MFCS*

2023, volume 272 of *LIPIcs*, pages 56:1–56:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.MFCS.2023.56.

[178] Bart M. P. Jansen and Jesper Nederlof. Computing the chromatic number using graph decompositions via matrix rank. *Theor. Comput. Sci.*, 795:520–539, 2019. doi:10.1016/J.TCS.2019.08.006.

[179] Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *J. Comput. Syst. Sci.*, 79(1):39–49, 2013. doi:10.1016/J.JCSS.2012.04.004.

[180] Iyad A. Kanj, Michael J. Pelsmajer, Marcus Schaefer, and Ge Xia. On the induced matching problem. *J. Comput. Syst. Sci.*, 77(6):1058–1070, 2011. doi:10.1016/J.JCSS.2010.09.001.

[181] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

[182] Richard M. Karp. On the computational complexity of combinatorial problems. *Networks*, 5(4):45–68, 1975. doi:10.1002/NET.1975.5.1.45.

[183] Karthik C. S. and Subhash Khot. Almost polynomial factor inapproximability for parameterized k-clique. In *37th Computational Complexity Conference, CCC 2022*, volume 234 of *LIPIcs*, pages 6:1–6:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.CCC.2022.6.

[184] Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. *J. ACM*, 66(5):33:1–33:38, 2019. doi:10.1145/3325116.

[185] Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. On equivalence of parameterized inapproximability of k-median, k-max-coverage, and 2-csp. *Algorithmica*, 87(12):1711–1731, 2025. doi:10.1007/S00453-025-01338-4.

[186] Ioannis Katsikarelis, Michael Lampis, and Vangelis Th. Paschos. Structural parameters, tight bounds, and approximation for (k, r)-center. *Discret. Appl. Math.*, 264:90–117, 2019. doi:10.1016/j.dam.2018.11.002.

[187] Ioannis Katsikarelis, Michael Lampis, and Vangelis Th. Paschos. Structurally parameterized d-scattered set. *Discret. Appl. Math.*, 308:168–186, 2022. doi:10.1016/j.dam.2020.03.052.

[188] Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce A. Reed. The disjoint paths problem in quadratic time. *J. Comb. Theory, Ser. B*, 102(2):424–435, 2012. doi:10.1016/j.jctb.2011.07.004.

[189] Ken-ichi Kawarabayashi and Bingkai Lin. A nearly 5/3-approximation FPT algorithm for min-k-cut. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*, pages 990–999. SIAM, 2020. doi:10.1137/1.9781611975994.59.

[190] Daniel J. Kleitman and Douglas B. West. Spanning trees with many leaves. *SIAM J. Discret. Math.*, 4(1):99–106, 1991. doi:10.1137/0404010.

[191] Boris Klemz and Günter Rote. Linear-time algorithms for maximum-weight induced matchings and minimum chain covers in convex bipartite graphs. *Algorithmica*, 84(4):1064–1080, 2022. doi:10.1007/S00453-021-00904-W.

[192] Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994. doi:10.1007/BFB0045375.

[193] C. W. Ko and F. Bruce Shepherd. Bipartite domination and simultaneous matroid covers. *SIAM J. Discret. Math.*, 16(4):517–523, 2003. doi:10.1137/S089548019828371X.

[194] Tomohiro Koana. Induced matching below guarantees: Average paves the way for fixed-parameter tractability. In *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023*, volume 254 of *LIPIcs*, pages 39:1–39:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.STACS.2023.39.

[195] Daniel Kobler and Udi Rotics. Finding maximum induced matchings in subclasses of claw-free and P_5-free graphs, and in graphs with matching and induced matching of equal maximum size. *Algorithmica*, 37(4):327–346, 2003. doi:10.1007/S00453-003-1035-4.

[196] Stavros G. Kolliopoulos and Clifford Stein. Approximating disjoint-path problems using packing integer programs. *Math. Program.*, 99(1):63–87, 2004. doi:10.1007/s10107-002-0370-6.

[197] Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 184–192. IEEE, 2021. doi:10.1109/FOCS52979.2021.00026.

[198] Tuukka Korhonen and Daniel Lokshtanov. An improved parameterized algorithm for treewidth. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023*, pages 528–541. ACM, 2023. doi:10.1145/3564246.3585245.

[199] Tuukka Korhonen, Michal Pilipczuk, and Giannos Stamoulis. Minor containment and disjoint paths in almost-linear time. In *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024*, pages 53–61. IEEE, 2024. doi:10.1109/FOCS61266.2024.00014.

[200] Martin Koutecký, Asaf Levin, and Shmuel Onn. A parameterized strongly polynomial algorithm for block structured integer programs. In *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 für Informatik, 2018. doi:10.4230/LIPICS.ICALP.2018.85.

[201] Chandra Mohan Krishnamurthy and R. Sritharan. Maximum induced matching problem on hhd-free graphs. *Discret. Appl. Math.*, 160(3):224–230, 2012. doi:10.1016/J.DAM.2011.08.027.

[202] Ariel Kulik and Hadas Shachnai. Techniques in parameterized approximation. *Comput. Sci. Rev.*, 59:100833, 2026. doi:10.1016/J.COSREV.2025.100833.

[203] Ravi Kumar, Uma Mahadevan, and D. Sivakumar. A graph-theoretic approach to extract storylines from search results. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004*, pages 216–225. ACM, 2004. doi:10.1145/1014052.1014078.

[204] Mohammed Lalou, Mohammed Amin Tahraoui, and Hamamache Kheddouci. The critical node detection problem in networks: A survey. *Comput. Sci. Rev.*, 28:92–117, 2018. doi:10.1016/J.COSREV.2018.02.002.

[205] Michael Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012. doi:10.1007/S00453-011-9554-X.

[206] Michael Lampis. Parameterized maximum path coloring. *Theor. Comput. Sci.*, 511:42–53, 2013.

[207] Michael Lampis. Parameterized approximation schemes using graph widths. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014*, volume 8572 of *Lecture Notes in Computer Science*, pages 775–786. Springer, 2014. doi:10.1007/978-3-662-43948-7_64.

[208] Michael Lampis. Finer tight bounds for coloring on clique-width. *SIAM J. Discret. Math.*, 34(3):1538–1558, 2020. doi:10.1137/19M1280326.

[209] Michael Lampis. *Structural Graph Parameters, Fine-Grained Complexity, and Approximation*. 2022. URL: https://tel.archives-ouvertes.fr/tel-03848575.

[210] Michael Lampis. The primal pathwidth SETH. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025*, pages 1494–1564. SIAM, 2025. doi:10.1137/1.9781611978322.47.

[211] Michael Lampis. Circuits and backdoors: Five shades of the SETH. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2026*, pages 1945–2001. SIAM, 2026. doi:10.1137/1.9781611978971.71.

[212] Michael Lampis and Valia Mitsou. Treewidth with a quantifier alternation revisited. In *12th International Symposium on Parameterized and Exact Computation, IPEC 2017*, volume 89 of *LIPIcs*, pages 26:1–26:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICS.IPEC.2017.26.

[213] Michael Lampis and Valia Mitsou. Fine-grained meta-theorems for vertex integrity. *Log. Methods Comput. Sci.*, 20(4), 2024. doi:10.46298/lmcs-20(4:18)2024.

[214] Michael Lampis and Manolis Vasilakis. Structural parameterizations for two bounded degree problems revisited. In *31st Annual European Symposium on Algorithms, ESA 2023*, volume 274 of *LIPIcs*, pages 77:1–77:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ESA.2023.77.

[215] Michael Lampis and Manolis Vasilakis. Structural parameterizations for two bounded degree problems revisited. *ACM Trans. Comput. Theory*, 16(3):17:1–17:51, 2024. doi:10.1145/3665156.

[216] Michael Lampis and Manolis Vasilakis. Parameterized maximum node-disjoint paths. In *20th International Symposium on Parameterized and Exact Computation, IPEC 2025*, volume 358 of *LIPIcs*, pages 3:1–3:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPICS.IPEC.2025.3.

[217] Michael Lampis and Manolis Vasilakis. Structural parameterizations for induced and acyclic matching. In *Graph-Theoretic Concepts in Computer Science - 51st International Workshop, WG 2025*, volume 16124 of *Lecture Notes in Computer Science*, pages 360–376. Springer, 2026. doi:10.1007/978-3-032-11835-6_26.

[218] Bingkai Lin. The parameterized complexity of the k-biclique problem. *J. ACM*, 65(5):34:1–34:23, 2018. doi:10.1145/3212622.

[219] Bingkai Lin, Xuandi Ren, Yican Sun, and Xiuhan Wang. On lower bounds of approximating parameterized k-clique. In *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022*, volume 229 of *LIPIcs*, pages 90:1–90:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.ICALP.2022.90.

[220] Bingkai Lin, Xuandi Ren, Yican Sun, and Xiuhan Wang. Constant approximating parameterized k-setcover is w[2]-hard. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023*, pages 3305–3316. SIAM, 2023. doi:10.1137/1.9781611977554.CH126.

[221] Bingkai Lin, Xuandi Ren, Yican Sun, and Xiuhan Wang. Improved hardness of approximating k-clique under ETH. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 285–306. IEEE, 2023. doi:10.1109/FOCS57990.2023.00025.

[222] Min Chih Lin, Julián Mestre, and Saveliy Vasiliev. Approximating weighted induced matchings. *Discret. Appl. Math.*, 243:304–310, 2018. doi:10.1016/J.DAM.2018.01.009.

[223] Bernt Lindström. On a combinatorial problem in number theory. *Canadian Mathematical Bulletin*, 8(4):477–490, 1965. doi:10.4153/CMB-1965-034-2.

[224] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Trans. Algorithms*, 14(2):13:1–13:30, 2018. doi:10.1145/3170442.

[225] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. *SIAM J. Comput.*, 47(3):675–702, 2018. doi:10.1137/16M1104834.

[226] Daniel Lokshtanov, Pranabendu Misra, Michal Pilipczuk, Saket Saurabh, and Meirav Zehavi. An exponential time parameterized algorithm for planar disjoint paths. *SIAM J. Comput.*, 54(2):321–418, 2025. doi:10.1137/20M1355902.

[227] Daniel Lokshtanov, Pranabendu Misra, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Fpt-approximation for FPT problems. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 199–218. SIAM, 2021. doi:10.1137/1.9781611976465.14.

[228] Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Parameterized complexity and approximability of directed odd cycle transversal. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*, pages 2181–2200. SIAM, 2020. doi:10.1137/1.9781611975994.134.

[229] Daniel Lokshtanov, Abhishek Sahu, Saket Saurabh, Vaishali Surianarayanan, and Jie Xue. Parameterized approximation for capacitated d-hitting set with hard capacities. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025*, pages 1565–1592. SIAM, 2025. doi:10.1137/1.9781611978322.48.

[230] Daniel Lokshtanov, Saket Saurabh, and Vaishali Surianarayanan. A parameterized approximation scheme for min k-cut. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 798–809. IEEE, 2020. doi:10.1109/FOCS46700.2020.00079.

[231] Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Efficient graph minors theory and parameterized algorithms for (planar) disjoint paths. In *Treewidth, Kernels, and Algorithms - Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, volume 12160 of *Lecture Notes in Computer Science*, pages 112–128. Springer, 2020. doi:10.1007/978-3-030-42071-0_9.

[232] László Lovász and Michael D Plummer. *Matching theory*, volume 367. American Mathematical Soc., 2009.

[233] Vadim V. Lozin. On maximum induced matchings in bipartite graphs. *Inf. Process. Lett.*, 81(1):7–11, 2002. doi:10.1016/S0020-0190(01)00185-5.

[234] David F. Manlove. *Algorithmics of Matching Under Preferences*, volume 2 of *Series on Theoretical Computer Science*. WorldScientific, 2013. doi:10.1142/8591.

[235] Pasin Manurangsi and Prasad Raghavendra. A birthday repetition theorem and complexity of approximating dense csps. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017*, volume 80 of *LIPIcs*, pages 78:1–78:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICS.ICALP.2017.78.

[236] Dániel Marx. Parameterized complexity and approximation algorithms. *Comput. J.*, 51(1):60–78, 2008. doi:10.1093/COMJNL/BXM048.

[237] Dániel Marx and Valia Mitsou. Double-exponential and triple-exponential bounds for choosability problems parameterized by treewidth. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016*, volume 55 of *LIPIcs*, pages 28:1–28:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICS.ICALP.2016.28.

[238] Dániel Marx, Govind S. Sankar, and Philipp Schepper. Degrees and gaps: Tight complexity results of general factor problems parameterized by treewidth and cutwidth. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, volume 198 of *LIPIcs*, pages 95:1–95:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ICALP.2021.95.

[239] Dániel Marx and Paul Wollan. An exact characterization of tractable demand patterns for maximum disjoint path problems. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 642–661. SIAM, 2015. doi:10.1137/1.9781611973730.44.

[240] Benjamin McClosky and Illya V. Hicks. Combinatorial algorithms for the maximum k-plex problem. *J. Comb. Optim.*, 23(1):29–49, 2012. doi:10.1007/s10878-010-9338-2.

[241] Ross M. McConnell and Jeremy P. Spinrad. Modular decomposition and transitive orientation. *Discret. Math.*, 201(1-3):189–241, 1999. doi:10.1016/S0012-365X(98)00319-7.

[242] Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science*, pages 17–27. IEEE Computer Society, 1980. doi:10.1109/SFCS.1980.12.

[243] Robert T. Moenck. Practical fast polynomial multiplication. In *Proceedings of the third ACM Symposium on Symbolic and Algebraic Manipulation, SYMSAC 1976*, pages 136–148. ACM, 1976. doi:10.1145/800205.806332.

[244] Hannes Moser, Rolf Niedermeier, and Manuel Sorge. Exact combinatorial algorithms and experiments for finding maximum k-plexes. *J. Comb. Optim.*, 24(3):347–373, 2012. doi:10.1007/s10878-011-9391-5.

[245] Hannes Moser and Somnath Sikdar. The parameterized complexity of the induced matching problem. *Discret. Appl. Math.*, 157(4):715–727, 2009. doi:10.1016/J.DAM.2008.07.011.

[246] Hannes Moser and Dimitrios M. Thilikos. Parameterized complexity of finding regular induced subgraphs. *J. Discrete Algorithms*, 7(2):181–190, 2009. doi:10.1016/J.JDA.2008.09.005.

[247] Wojciech Nadara, Michal Pilipczuk, and Marcin Smulewicz. Computing treedepth in polynomial space and linear FPT time. In *30th Annual European Symposium on Algorithms, ESA 2022*, volume 244 of *LIPIcs*, pages 79:1–79:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.ESA.2022.79.

[248] Jesper Nederlof, Michal Pilipczuk, Céline M. F. Swennenhuis, and Karol Wegrzycki. Hamiltonian cycle parameterized by treedepth in single exponential time and polynomial space. *SIAM J. Discret. Math.*, 37(3):1566–1586, 2023. doi:10.1137/22M1518943.

[249] Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

[250] Naomi Nishimura, Prabhakar Ragde, and Dimitrios M. Thilikos. Fast fixed-parameter tractable algorithms for nontrivial generalizations of vertex cover. *Discret. Appl. Math.*, 152(1-3):229–245, 2005. doi:10.1016/j.dam.2005.02.029.

[251] Naoto Ohsaka. On the parameterized intractability of determinant maximization. *Algorithmica*, 86(6):1731–1763, 2024. doi:10.1007/S00453-023-01205-0.

[252] Karolina Okrasa and Pawel Rzazewski. Fine-grained complexity of the graph homomorphism problem for bounded-treewidth graphs. *SIAM J. Comput.*, 50(2):487–508, 2021. doi:10.1137/20M1320146.

[253] Michael Okun and Amnon Barak. A new approach for approximating node deletion problems. *Inf. Process. Lett.*, 88(5):231–236, 2003. doi:10.1016/j.ipl.2003.08.005.

[254] Yury L. Orlovich, Gerd Finke, Valery S. Gordon, and Igor E. Zverovich. Approximability results for the maximum and minimum maximal induced matching problems. *Discret. Optim.*, 5(3):584–593, 2008. doi:10.1016/J.DISOPT.2007.11.010.

[255] Sang-il Oum. Approximating rank-width and clique-width quickly. *ACM Trans. Algorithms*, 5(1):10:1–10:20, 2008. doi:10.1145/1435375.1435385.

[256] B. S. Panda and Juhi Chaudhary. Acyclic matching in some subclasses of graphs. *Theor. Comput. Sci.*, 943:36–49, 2023. doi:10.1016/J.TCS.2022.12.008.

[257] B. S. Panda and Dinabandhu Pradhan. Acyclic matchings in subclasses of bipartite graphs. *Discret. Math. Algorithms Appl.*, 4(4), 2012. doi:10.1142/S1793830912500504.

[258] Venkatesh Raman, Saket Saurabh, and Sriganesh Srihari. Parameterized algorithms for generalized domination. In *Combinatorial Optimization and Applications, Second International Conference, COCOA 2008*, volume 5165 of *Lecture Notes in Computer Science*, pages 116–126. Springer, 2008. doi:10.1007/978-3-540-85097-7_11.

[259] Dieter Rautenbach. Two greedy consequences for maximum induced matchings. *Theor. Comput. Sci.*, 602:32–38, 2015. doi:10.1016/J.TCS.2015.08.002.

[260] Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. A faster parameterized algorithm for treedepth. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014*, volume 8572 of *Lecture Notes in Computer Science*, pages 931–942. Springer, 2014. doi:10.1007/978-3-662-43948-7_77.

[261] Neil Robertson and Paul D. Seymour. Graph minors. III. planar tree-width. *J. Comb. Theory B*, 36(1):49–64, 1984. doi:10.1016/0095-8956(84)90013-3.

[262] Neil Robertson and Paul 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.

[263] Neil Robertson and Paul D. Seymour. Graph minors .XIII. The disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

[264] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 216–226. ACM, 1978. doi:10.1145/800133.804350.

[265] Petra Scheffler. A linear algorithm for the pathwidth of trees. In *Topics in Combinatorics and Graph Theory: Essays in Honour of Gerhard Ringel*, pages 613–620. Springer, 1990.

[266] Petra Scheffler. *A practical linear time algorithm for disjoint paths in graphs with bounded tree-width.* TU, Fachbereich 3, 1994.

[267] Jannik Schestag. Critical node problem with vulnerable vertices. Master's thesis, Philipps-Universität Marburg, 2021.

[268] Jannik Schestag, Niels Grüttemeier, Christian Komusiewicz, and Frank Sommer. On critical node problems with vulnerable vertices. *J. Graph Algorithms Appl.*, 28(1):1–26, 2024. doi:10.7155/JGAA.V28I1.2922.

[269] Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.

[270] Roberto Solis-Oba, Paul S. Bonsma, and Stefanie Lowski. A 2-approximation algorithm for finding a spanning tree with maximum number of leaves. *Algorithmica*, 77(2):374–388, 2017. doi:10.1007/S00453-015-0080-0.

[271] Larry J. Stockmeyer and Vijay V. Vazirani. NP-completeness of some generalizations of the maximum matching problem. *Inf. Process. Lett.*, 15(1):14–19, 1982. doi:10.1016/0020-0190(82)90077-1.

[272] Marco Di Summa, Andrea Grosso, and Marco Locatelli. Complexity of the critical node problem over trees. *Comput. Oper. Res.*, 38(12):1766–1774, 2011. doi:10.1016/J.COR.2011.02.016.

[273] Atsushi Takahashi, Shuichi Ueno, and Yoji Kajitani. Mixed searching and proper-path-width. *Theor. Comput. Sci.*, 137(2):253–268, 1995. doi:10.1016/0304-3975(94)00160-K.

[274] Marc Tedder, Derek G. Corneil, Michel Habib, and Christophe Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, volume 5125 of *Lecture Notes in Computer Science*, pages 634–645. Springer, 2008. doi:10.1007/978-3-540-70575-8_52.

[275] 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.

[276] Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discret. Appl. Math.*, 8(1):85–89, 1984. doi:10.1016/0166-218X(84)90081-7.

[277] Dekel Tsur. Faster parameterized algorithm for cluster vertex deletion. *Theory Comput. Syst.*, 65(2):323–343, 2021. doi:10.1007/S00224-020-10005-W.

[278] Martijn van Ee. Some notes on bounded starwidth graphs. *Inf. Process. Lett.*, 125:9–14, 2017. doi:10.1016/J.IPL.2017.04.011.

[279] 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. *J. Graph Algorithms Appl.*, 24(3):461–482, 2020. doi:10.7155/jgaa.00542.

[280] Johan M. M. van Rooij. A generic convolution algorithm for join operations on tree decompositions. In *Computer Science - Theory and Applications - 16th International Computer Science Symposium in Russia, CSR 2021*, volume 12730 of *Lecture Notes in Computer Science*, pages 435–459. Springer, 2021. doi:10.1007/978-3-030-79416-3_27.

[281] 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. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2009. doi:10.1007/978-3-642-04128-0_51.

[282] Vijay V. Vazirani. *Approximation algorithms*. Springer, 2001.

[283] Mario Ventresca and Dionne M. Aleman. A derandomized approximation algorithm for the critical node detection problem. *Comput. Oper. Res.*, 43:261–270, 2014. doi:10.1016/J.COR.2013.09.012.

[284] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

[285] Michal Wlodarczyk. Parameterized inapproximability for steiner orientation by gap amplification. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020*, volume 168 of *LIPIcs*, pages 104:1–104:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.ICALP.2020.104.

[286] Michal Wlodarczyk. Constant approximating disjoint paths on acyclic digraphs is w[1]-hard. In *35th International Symposium on Algorithms and Computation, ISAAC 2024*, volume 322 of *LIPIcs*, pages 57:1–57:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ISAAC.2024.57.

[287] Michal Wlodarczyk and Meirav Zehavi. Planar disjoint paths, treewidth, and kernels. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023*, pages 649–662. IEEE, 2023. doi:10.1109/FOCS57990.2023.00044.

[288] Mingyu Xiao. A parameterized algorithm for bounded-degree vertex deletion. In *Computing and Combinatorics - 22nd International Conference, COCOON 2016*, volume 9797 of *Lecture Notes in Computer Science*, pages 79–91. Springer, 2016. doi:10.1007/978-3-319-42634-1_7.

[289] Mingyu Xiao. On a generalization of nemhauser and trotter's local optimization theorem. *J. Comput. Syst. Sci.*, 84:97–106, 2017. doi:10.1016/j.jcss.2016.08.003.

[290] Mingyu Xiao and Shaowei Kou. Parameterized algorithms and kernels for almost induced matching. *Theor. Comput. Sci.*, 846:103–113, 2020. doi:10.1016/J.TCS.2020.09.026.

[291] Mingyu Xiao and Huan Tan. Exact algorithms for maximum induced matching. *Inf. Comput.*, 256:196–211, 2017. doi:10.1016/J.IC.2017.07.006.

[292] Meirav Zehavi. The k-leaf spanning tree problem admits a klam value of 39. *Eur. J. Comb.*, 68:175–203, 2018. doi:10.1016/J.EJC.2017.07.018.

[293] Michele Zito. Induced matchings in regular graphs and trees. In *Graph-Theoretic Concepts in Computer Science, 25th International Workshop, WG '99, Proceedings*, volume 1665 of *Lecture Notes in Computer Science*, pages 89–100. Springer, 1999. doi:10.1007/3-540-46784-X_10.

[294] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory Comput.*, 3(1):103–128, 2007. doi:10.4086/TOC.2007.V003A006.

RÉSUMÉ

Cette thèse étudie des problèmes sur les graphes NP-difficiles au travers de la paramétrisation structurelle et d'une analyse fine. Pour les problèmes considérés, nous (i) localisons la frontière entre la tractabilité à paramètres fixes (FPT) et la W[1]-dureté, en proposant des réductions d'intractabilité améliorées et en concevant de nouveaux algorithmes FPT ; (ii) faisons correspondre des algorithmes naturels de programmation dynamique à des bornes inférieures serrées fondées sur SETH ou pw-SETH ; (iii) établissons des bornes inférieures serrées sous ETH pour des paramétrisations au-delà de la largeur d'arbre ; et (iv) montrons que l'approximation permet, dans certains cas, de franchir des barrières du calcul exact, tandis que dans d'autres elle ne le peut pas. Pour atteindre ces résultats, nous développons de nouvelles techniques de réductions de dureté et de conception d'algorithmes, susceptibles d'intérêt indépendant. En particulier, nous introduisons (i) une nouvelle variante de UNARY BIN PACKING qui conduit à des résultats de W[1]-dureté renforcés ; (ii) des techniques pour obtenir des bornes inférieures améliorées pour des problèmes paramétrés par la profondeur d'arbre et la couverture de sommets ; et (iii) une construction simple pour concevoir des schémas d'approximation FPT. Nous mettons à profit ces techniques pour obtenir un ensemble de nouveaux résultats sur plusieurs problèmes sur les graphes NP-difficiles bien étudiés.

MOTS CLÉS

Complexité paramétrée · Complexité fine · Algorithmes d'approximation · Largeur d'arbre

ABSTRACT

This thesis studies NP-hard graph problems through structural parameterization and fine-grained analysis. For our problems of study we (i) locate the boundary between fixed-parameter tractability and W[1]-hardness, by giving improved hardness reductions and designing new FPT algorithms; (ii) match natural DP algorithms with tight SETH- or pw-SETH-lower bounds; (iii) obtain tight ETH-lower bounds for parameterizations beyond treewidth; and (iv) show that adding approximation into the mix can in some cases overcome exact barriers, yet in some other cases it cannot. On our way to these results we develop new techniques for both hardness reductions and algorithm design that may be of independent interest. In particular, we introduce (i) a new variant of UNARY BIN PACKING that allows for improved W[1]-hardness results; (ii) techniques for obtaining improved lower bounds for problems parameterized by tree-depth and vertex cover; and (iii) a simple construction for designing FPT approximation schemes. We leverage these techniques to obtain a number of new results for several well-studied NP-hard graph problems.

KEYWORDS

Parameterized complexity · Fine-grained complexity · Approximation algorithms · Treewidth