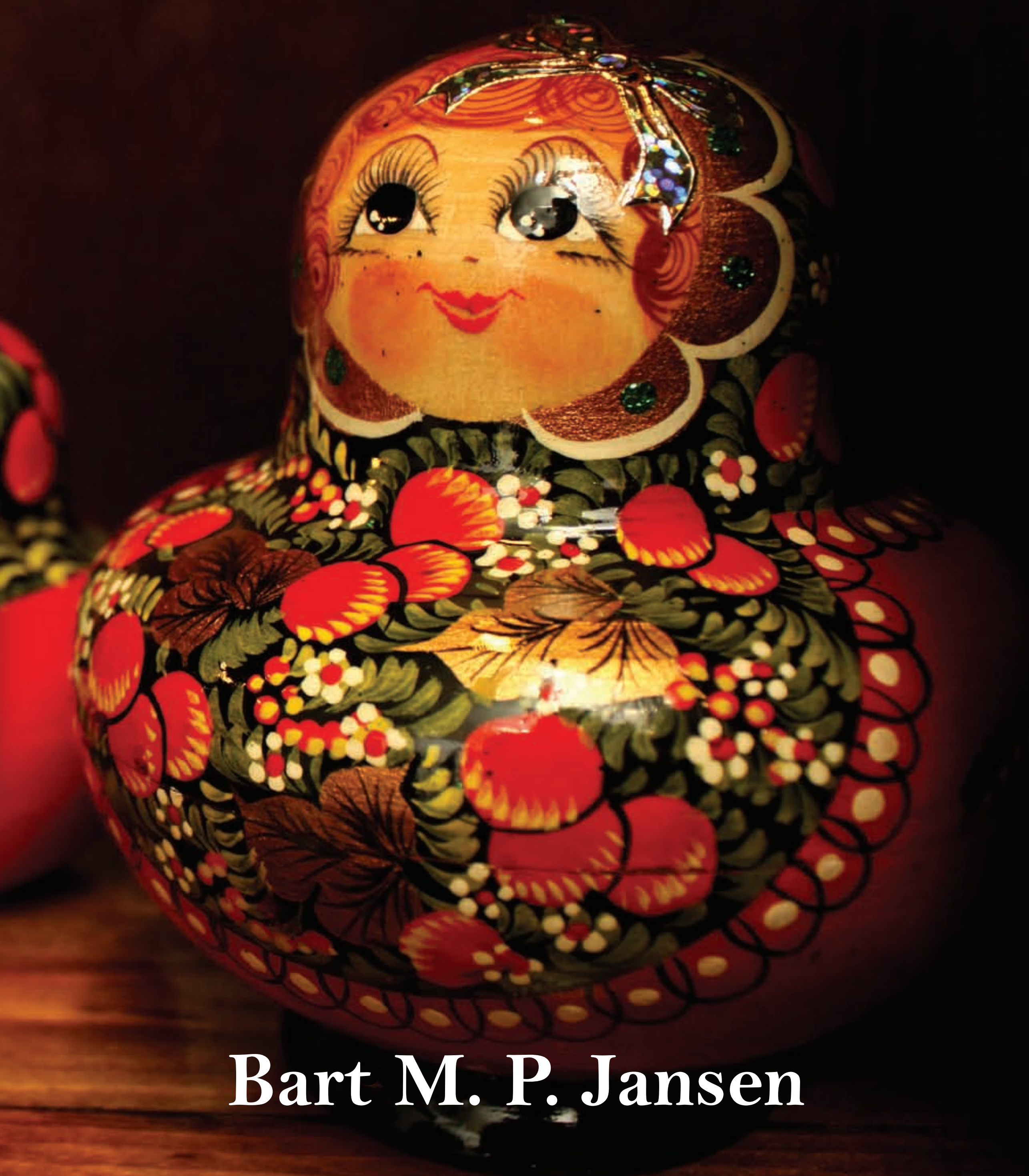
The Power of Data Reduction

Kernels for Fundamental Graph Problems



Bart M. P. Jansen

The Power of Data Reduction
Kernels for Fundamental Graph Problems

Bart Maarten Paul Jansen

The Power of Data Reduction: Kernels for Fundamental Graph Problems
Bart M. P. Jansen

ISBN 978-90-393-5966-2

9 789039 359662

The Power of Data Reduction
Kernels for Fundamental Graph Problems

De Kracht van Gegevensreductie
De Kern van Fundamentele Graafproblemen

(met een samenvatting in het Nederlands)

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de rector magnificus, prof. dr. G.J. van der Zwaan, ingevolge het besluit van het college voor promoties in het openbaar te verdedigen op maandag 1 juli 2013 des middags te 12.45 uur

door

Bart Maarten Paul Jansen

geboren op 5 augustus 1986
te Nijmegen

Promotor:	Prof. dr. J. van Leeuwen
Co-promotor:	Dr. H.L. Bodlaender

This research was supported by the Netherlands Organization for Scientific Research (NWO) through the project "KERNELS: Combinatorial Analysis of Data Reduction".

Mathematical talent is probably congenital, but aside from that the most important attribute of a genuine professional mathematician is scholarship. The scholar is always studying, always ready and eager to learn. The scholar knows the connections of this specialty with the subject as a whole; he knows not only the technical details of his specialty, but its history and its present standing; he knows about the others who are working on it and how far they have reached. He knows the literature, and he trusts nobody; he himself examines the original paper. He acquires firsthand knowledge not only of its intellectual content, but also of the date of the work, the spelling of the author's name, and the punctuation in the title; he insists on getting every detail of every reference absolutely straight. The scholar tries to be as broad as possible. No one can know all of mathematics, but the scholar can succeed in knowing the outline of it all: what are its parts and what are their places in the whole? These are the things, some of the things, that go to make up a pro.

—Paul R. Halmos, 1985

Contents

List of Figures

List of Symbols and Notation

Mathematics

Symbol	Interpretation
$:=$	Definition of a constant or concept.
$\mathbb{R}$	The real numbers.
$\mathbb{N}$	The non-negative integers.
$\mathbb{N}^+$	The positive integers.
$[n]$	The set $\{1, 2, \dots, n\}$ for $n \in \mathbb{N}$, with $[0] := \emptyset$.
$\emptyset$	The empty set.
$X \cup Y$	The union of sets X and Y.
$X \dot\cup Y$	The disjoint union of sets X and Y.
$X \setminus Y$	The set difference of X and Y.
$X \times Y$	The Cartesian product of sets X and Y.
X^i	The Cartesian product of i copies of X $(X \times X \times \dots \times X)$.
$\binom{n}{k}$	The binomial coefficient with indices $n \in \mathbb{N}$ and $k \in \mathbb{N}$.
$\binom{X}{k}$	The size-k subsets of the set X.
$\binom{X}{\leq k}$	The size-at-most-k subsets of the set X, including $\emptyset$.
$\lvert x \rvert$	The length of the encoding of structure x as a string of symbols.
1^k	The string consisting of k ones (in the context of strings).
Σ	A finite alphabet (may differ across repeated occurrences).
Σ^*	The set of finite strings over Σ.

Complexity Theory

Symbol	Interpretation
$\mathcal{O}(f(n))$	Landau notation for asymptotic upper bounds.
$\Omega(f(n))$	Landau notation for asymptotic lower bounds.
$\Theta(f(n))$	Landau notation for asymptotically tight bounds.
$\mathcal{O}^*(f(k))$	The class $\bigcup_{i=1}^{+\infty} \mathcal{O}(f(k)(n+k)^i)$.
SAT	The satisfiability problem for Boolean formulae.
P	The class of languages deterministically decidable in polynomial time.
NP	The class of languages nondeterministically decidable in polynomial time.
coNP	The class of languages co-nondeterministically decidable in polynomial time.
coNP/poly	The class of languages co-nondeterministically decidable in polynomial time, when given access to polynomial-size advice.
EXP	The class of languages deterministically decidable in exponential time.
$\widetilde{\mathcal{Q}}$	The classical variant of parameterized problem $\mathcal{Q}$.
XP	The class of parameterized languages decidable in strongly uniform slicewise polynomial time.
FPT	The class of languages that are strongly uniformly fixed-parameter tractable.
para-NP	The class of parameterized languages nondeterministically decidable in $f(k)n^{\mathcal{O}(1)}$ time.
$W[i]$	The i-th level of the bounded-weft W-hierarchy.
$W[P]$	The class of parameterized languages decidable by circuits of polynomial weft.
PH	The polynomial hierarchy.
Σ_3^p	The third level of the polynomial hierarchy.

In-depth material on complexity theory can be found in the recent textbooks on the subject [10, 107, 118].

Graph Theory

Symbol	Interpretation
G	An undirected, simple graph.
$V(G)$	The vertex set of G.
$E(G)$	The edge set of G, consisting of unordered pairs over $V(G)$.
$\{u, v\}$	An unordered pair, or edge in a graph.
$G - X$	Graph G after removing vertex set X and its incident edges.
$G[X]$	The subgraph of G induced by vertex set X.
$G' \subseteq G$	Graph G' is a subgraph of G.
$\Delta(G)$	The maximum degree of a vertex in G.
$\deg_G(v)$	The degree of vertex v in graph G.
$G \otimes H$	The join of graphs G and H.
$G \mathbin{\dot\cup} H$	The disjoint union of graphs G and H.
$G \cup H$	The union of graphs G and H.
$N_G(\dots)$	The open neighborhood of a vertex (set) in graph G.
$N_G[\dots]$	The closed neighborhood of a vertex (set) in graph G.

Graph Classes

Symbol	Interpretation
C_n	The cycle graph on n vertices.
P_n	The path graph on n vertices.
K_n	The complete graph on n vertices.
$K_{n,m}$	The biclique with partite sets of sizes n and m.
$\dot\bigcup$Cochordal	The closure of cochordal graphs under disjoint union.
$\dot\bigcup$Split	The closure of split graphs under disjoint union.

When analyzing the running time of a graph algorithm, we write n for the number of vertices and m for the number of edges. Full definitions of all graph-theoretical concepts can be found in Appendix A.

Graph Parameters

Symbol	Interpretation
$\textsc{vc}(G)$	The vertex cover number of G.
$\alpha(G)$	The independence number of G.
$\omega(G)$	The clique number of G.
$\textsc{fvs}(G)$	The feedback vertex number of G.
$\chi(G)$	The chromatic number of G.
$\mu(G)$	The size of a maximum matching in G.
$\gamma'(G)$	The minimum size of a maximal matching in G.
$\textsc{tw}(G)$	The treewidth of G.
$\textsc{pw}(G)$	The pathwidth of G.
$\textsc{cw}(G)$	The cliquewidth of G.

Problem Parameterizations

Symbol	Interpretation
Π [SOL]	Problem Π parameterized by the size of the desired solution.
Π [VC]	Problem Π parameterized by the size of a given vertex cover.
Π [FVS]	Problem Π parameterized by the size of a given feedback vertex set.
Π [CHORDAL MOD]	Problem Π parameterized by the size of a given chordal graph modulator.
Π [CLIQUE MOD]	Problem Π parameterized by the size of a given clique modulator.
Π [CLUSTER MOD]	Problem Π parameterized by the size of a given cluster graph modulator.
Π [COGRAPH MOD]	Problem Π parameterized by the size of a given cograph modulator.
Π [OUTERPLANAR MOD]	Problem Π parameterized by the size of a given outerplanar graph modulator.
Π [ML]	Problem Π parameterized by the max leaf number, formalized as a promise problem.

Part I

Foundations

The unreasonable success of heuristic algorithms is one of the great mysteries in computer science.

—Richard M. Karp, 2012

1 Introduction

Whether you are solving a Sudoku puzzle, trying to find the best move in a chess game, allocating classrooms for simultaneously administered exams, or considering different construction patterns when building a new facility, a quick "back of the envelope" calculation or deduction often makes it possible to simplify the problem by discarding possibilities that provably fail to produce the desired solution. After exhaustive application of such simplification steps, the core or kernel of the problem remains. In the case of solving Sudoku puzzles, the preprocessing steps with sophisticated names such as "Crosshatching", "Box Line Reduction", "Sub Group Exclusion", "Hidden Twin", and "Swordfish" [77], are highly specialized puzzle-solving strategies, but preprocessing in general is a ubiquitous first line of attack against hard problems. It is used almost without conscious effort by anyone trying to make a difficult decision, and often greatly reduces the effort needed to solve the remaining simplified problem.

Given the effectiveness of preprocessing in human problem solving, it should come as no surprise that computer programs also benefit tremendously from the use of such data reduction techniques. A classical example of industrial-strength preprocessing is exhibited by the CPLEX problem solver, whose advanced preprocessing routines allow it to discard superfluous constraints from the specification of the problem it tries to solve.

For concreteness, let us consider the following example of constraint reduction. Imagine we are planning the route and schedule for a UPS package delivery truck. The schedule is subject to various constraints: there is a limit on how many packages can be loaded in the truck, on how far the truck can travel before running out of gasoline, and on the time by which the vehicle has to return to the depot to ensure the driver makes it home in time for dinner. Under these restrictions, we might be interested in delivering as many packages as possible. Suppose that the fuel tank is so large that it is impossible to run out of fuel before exceeding the driver's working hours. As problem solvers, we can then safely forget about the fuel restriction without running the risk of leaving the driver stranded: any schedule that satisfies the remaining constraints (i.e., that gets the driver home in time to carve the meat) will not empty the fuel tank.

Frivolous as this example may be, applying the same type of reasoning in a logically consistent way — albeit on a larger scale — can give tremendous speedups in industrial computations. The effectiveness of CPLEX's acclaimed preprocessing routine is widely recognized, and there are countless accounts of the unexplained power of preprocessing; a classic example being Weihe's [209, 210] work on a train scheduling problem for the German and European railways. Faced with a covering problem — promote some train stations to maintenance stations, such that all scheduled trains pass through at least one maintenance station each day — he contemplated various algorithmic strategies to deal with this NP-hard task. After implementing one such strategy and performing experiments with the data, he discovered to his surprise that simple preprocessing reduces the real-world inputs to small, independent pieces that can easily be solved by hand! This experience triggered him to issue the following warning [209]:

> The power of rigorous data reduction in a preprocessing phase should not be underestimated.
>
> —Karsten Weihe, 1998

Preprocessing often works efficiently and effectively on problems that arise in practice, even in settings where the problem being solved is intractable from a theoretical point of view. This raises the following question: *why is preprocessing so effective on real-world input distributions of NP-hard problems?* In the case of the Sudoku puzzles discussed earlier, the answer is simple: the puzzles are tailor-made to contain logical brainteasers for enthusiasts, and it is much more rewarding to arrive at a solution through clever insight than by following a trial-and-error approach. The Sudoku puzzles that are published as being of moderate difficulty, are therefore purposefully crafted to be amenable to preprocessing strategies. But when it comes to industrial "puzzles" from the realm of applied computing, the situation is much more involved. During the first decades of its scientific life, the field of complexity theory has lacked the mathematical tools to even begin to answer the question. The advent of parameterized complexity theory in the nineteen nineties has made it possible to analyze efficient preprocessing rigorously, by examining the interplay between problem parameters and instance size. This opened up an entirely new arena of research dubbed *kernelization*, which was described as "The Lost Continent of Polynomial Time" [95]:

> Pre-processing is a humble strategy for coping with hard problems, almost universally employed. It has become clear, however, that far from being trivial and uninteresting, pre-processing has unexpected practical power for real world input distributions, and is mathematically a much deeper subject than has generally been understood.
>
> —Michael R. Fellows, 2006

The concept of kernelization and the language of parameterized complexity give us the tools needed to rigorously analyze efficient preprocessing.

1.1 Capturing Preprocessing by Kernelization

Without going into too much detail — we leave that for Chapter 2 — let us consider how the formalization of an effective preprocessing technique as a polynomial kernelization helps to explain its observed effectiveness. The main idea is that the problem instances we face in real life exhibit structure that is inherited from the processes that generate them (be it industrial settings, human input, or the historical evolution of the problem domain). It is a priori unclear in what form this structure is manifested, but it often causes the real-world input distribution of instances to be much more tractable than what is suggested by worst-case theoretical models. To allow an accurate analysis of the effect of preprocessing, we therefore consider various measures of input complexity as *parameters* and study whether the instance can be shrunk efficiently if its size exceeds a polynomial function of the chosen parameter. In the language of parameterized complexity, we ask ourselves whether the corresponding parameterized problem admits a *polynomial-size kernel*. If this is the case, then the proven size bound for the kernelization explains the amenability to preprocessing of instances in which the chosen parameter is small. When real-world instances have small complexity measured in terms of such a parameter, then this explains the power of data reduction for such inputs. Whether actual datasets have these properties can, of course, only be established by inspecting them; a task that is beyond the scope of this theoretical work.

This thesis is therefore concerned with the systematic identification of problem parameterizations that admit polynomial kernelizations. We aim to obtain theoretical explanations for the effectiveness of preprocessing, and to find new preprocessing schemes for important graph problems. Our efforts are organized programmatically in the *parameter ecology program* that is described at length in Chapter 2, which asks for an extended dialogue with a classical problem to determine the existence of polynomial kernels for an array of different parameters. To carry out this program, we develop general tools that can be used to derive upper- and lower bounds on the sizes of kernelizations.

When analyzing the contribution of various parameters to problem complexity, it is helpful to have a wide array of parameters at our disposal. Advances in structural graph theory over the last decades have resulted in sophisticated ways of expressing graph complexity in terms of treewidth, cliquewidth, and related graph measures. We use these structural graph measures as parameters for our problems. Consequently, we restrict our attention to fundamental graph problems.

1.2 Thesis Overview

This thesis is divided into four parts. The first part “Foundations” contains this introduction, along with a more technical opening in Chapter 2. We not only introduce the relevant aspects of parameterized complexity theory, but also present new views on the parameter ecology program and discuss its relevance for the study of preprocessing. In the second part we give two types of tools for kernelization. Chapter 3 reviews relevant techniques for proving lower bounds on the sizes of kernelizations, and introduces the framework of cross-composition as a new method to establish such bounds. The succeeding Chapter 4 gives general theorems that can be used to establish polynomial upper bounds on the sizes of kernels for graph problems parameterized by the size of a vertex cover.

Having set the stage, Part III contains the main contribution of this thesis. We use the tools developed in the earlier parts to systematically analyze the extent to which effective and efficient preprocessing is possible with respect to various parameters, as captured by the concept of polynomial kernelization. We organize our efforts programmatically by considering the parameter space as a partially ordered hierarchy, and study a variety of combinatorial problems. For each problem we try to pinpoint the limits to polynomial kernelizability, by searching for tractability boundaries in the parameter hierarchy.

When it comes to concrete kernelizations, we can only start by revisiting everyone’s favorite problem, the *drosophila melanogaster* [124] of parameterized algorithmics: Vertex Cover (Chapter 5). We analyze the potential for preprocessing with respect to the parameter feedback vertex number. The main result of the chapter is an example of transferring positive results to lower regions of the parameter hierarchy: while it was already known that the graph size of a Vertex Cover instance can be efficiently reduced to a polynomial in its vertex cover number, we show how to reduce the graph size to a polynomial in its feedback vertex number. The latter is a *smaller* parameter as the vertex cover number bounds the feedback vertex number from above, while the difference can be arbitrarily large.

The next chapter focuses on the problem of computing tree decompositions of graphs. We show that heuristic preprocessing rules that have been applied for years, can be analyzed using the framework of kernelization to give *provable* guarantees on the sizes of exhaustively reduced instances, with respect to several structural parameters. We also give a kernelization lower bound that applies to the decision problems Treewidth and Pathwidth, which ask whether the chosen width value of a given graph G is at most ℓ. We show that, unless an unlikely complexity-theoretic collapse occurs, it is impossible to do the following: given a graph G, an integer ℓ, and a set of k vertices whose removal from G leaves a clique, produce in polynomial time a pair (G', ℓ') of total bitsize polynomial in k such that G has width at most ℓ if and only if G' has width at most ℓ'. Existing fixed-parameter tractability results show that this compression task is possible if we demand that the bitsize of (G', ℓ') is bounded by an arbitrary computable function of k, rather than a polynomial in k. The result therefore gives a lower bound on the sizes of equivalent instances that can be computed in polynomial time.

Chapter 7 studies various types of graph coloring problems. We discover an interesting relationship between the existence of polynomial kernels for structural parameterizations of the problem, and the sizes of certain minimal critical structures relating to these parameterizations. Surprisingly, the behavior of these critical structures completely determines the existence of polynomial-size kernels: an upper bound on the sizes of critical structures yields kernelization routines whose outputs have bounded size, whereas explicit constructions of large critical structures can be transformed into conditional lower bounds on the sizes of reduced instances produced by a kernelization.

In Chapter 8 we consider various path and cycle problems. Such problems have been instrumental for the development of important upper- and lower bound tools in parameterized complexity theory. We prove a theorem that shows how to shrink one partite set of a bipartite graph, without changing the subsets of vertices in the other partite set that can be saturated by a matching. We use the theorem to efficiently reduce instances of various path and cycle problems on graphs with a vertex cover of size k, to $\mathcal{O}(k^2)$ vertices in total. We give related results for various other parameters, and also show that even graphs that are almost outerplanar can be hard to reduce. We prove a superpolynomial lower bound on the size of kernelizations for HAMILTONIAN CYCLE parameterized by the vertex-deletion distance to an outerplanar graph.

Finally, Part IV concludes the thesis. We reflect on the progress, and look towards the future by considering open problems and new research directions.

In our exposition we assume that the reader is well-versed in the language of structural graph theory. For completeness we provide an overview of the relevant concepts in Appendix A. Full definitions of all the problems considered in this work are given in Appendix B.

1.3 Published Papers

This thesis is based on the following refereed publications.

[1] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. “Cross-composition: A new technique for kernelization lower bounds”. In: *Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011.* Edited by Thomas Schwentick and Christoph Dürr. Volume 9 of

LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011, pages 165–176. DOI: 10.4230/LIPIcs.STACS.2011.165

[2] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. "Kernel bounds for path and cycle problems". In: *Theor. Comput. Sci.* 2012. Online First. DOI: 10.1016/j.tcs.2012.09.006

[3] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. "Kernel bounds for structural parameterizations of pathwidth". In: *Proceedings of the 13th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2012*. Edited by Fedor V. Fomin and Petteri Kaski. Volume 7357 of LNCS. Springer, 2012, pages 352–363. DOI: 10.1007/978-3-642-31155-0_31

[4] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. "Preprocessing for treewidth: A combinatorial analysis through kernelization". In: *Proceedings of the 38th International Colloquium on Automata, Languages and Programming, ICALP 2011, Part I*. Edited by Luca Aceto, Monika Henzinger, and Jiri Sgall. Volume 6755 of LNCS. Springer, 2011, pages 437–448. DOI: 10.1007/978-3-642-22006-7_37

[5] Michael R. Fellows, Bart M. P. Jansen, and Frances A. Rosamond. "Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity". In: *European J. Combin.* 34(3), 2013, pages 541–566. DOI: 10.1016/j.ejc.2012.04.008

[6] Fedor V. Fomin, Bart M. P. Jansen, and Michał Pilipczuk. "Preprocessing subgraph and minor problems: When does a small vertex cover help?" In: *Proceedings of the 7th International Symposium on Parameterized and Exact Computation, IPEC 2012*. Edited by Dimitrios M. Thilikos and Gerhard J. Woeginger. Volume 7535 of LNCS. Springer, 2012, pages 97–108. DOI: 10.1007/978-3-642-33293-7_11

[7] Bart M. P. Jansen and Hans L. Bodlaender. "Vertex cover kernelization revisited". In: *Theory Comput. Syst.* 2012. Online First, pages 1–37. DOI: 10.1007/s00224-012-9393-4

[8] Bart M. P. Jansen and Stefan Kratsch. "Data reduction for graph coloring problems". In: *Proceedings of the 18th International Symposium on Fundamentals of Computation Theory, FCT 2011*. Edited by Olaf Owe, Martin Steffen, and Jan Arne Telle. Volume 6914 of LNCS. Springer, 2011, pages 90–101. DOI: 10.1007/978-3-642-22953-4_8

The following refereed papers were also published as part of my Ph.D. research. They are not covered explicitly in this thesis.

[9] Michael R. Fellows, Bart M. P. Jansen, Daniel Lokshtanov, Frances A. Rosamond, and Saket Saurabh. "Determining the winner of a Dodgson election is hard". In: *Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010*. Edited by Kamal Lodaya and Meena Mahajan. Volume 8 of LIPIcs. Schloss

Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010, pages 459–468. DOI: `10.4230/LIPIcs.FSTTCS.2010.459`

[10] Pinar Heggernes, Pim van 't Hof, Bart M. P. Jansen, Stefan Kratsch, and Yngve Villanger. "Parameterized complexity of vertex deletion into perfect graph classes". In: *Theor. Comput. Sci.* 2012. Online First. DOI: `10.1007/978-3-642-22953-4_21`

[11] Bart M. P. Jansen. "Kernelization for maximum leaf spanning tree with positive vertex weights". In: *J. Graph Algorithms Appl.* 16(4), 2012, pages 811–846. DOI: `10.7155/jgaa.00279`

[12] Bart M. P. Jansen. "Polynomial kernels for hard problems on disk graphs". In: *Proceedings of the 12th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2010.* Edited by Haim Kaplan. Volume 6139 of LNCS. Springer, 2010, pages 310–321. DOI: `10.1007/978-3-642-13731-0_30`

[13] Bart M. P. Jansen and Stefan Kratsch. "On polynomial kernels for structural parameterizations of odd cycle transversal". In: *Proceedings of the 6th International Symposium on Parameterized and Exact Computation, IPEC 2011.* Edited by Dániel Marx and Peter Rossmanith. Volume 7112 of LNCS. Springer, 2011, pages 132–144. DOI: `10.1007/978-3-642-28050-4_11`

The ultimate goal is to quantitatively classify how parameters influence problem complexity. The more we know about these interactions, the more likely it becomes to master computational intractability.

—Rolf Niedermeier, 2010

2 Parameterized Complexity

This chapter introduces parameterized complexity theory, the language in which kernelization — a formalization of preprocessing — is naturally phrased. In Section 2.1 we present the main notions of the framework, assuming familiarity with the basic concepts of classical complexity theory as described in Garey and Johnson's famous monograph [117]. The *parameter ecology program* is discussed in Section 2.2. The program calls for a new, systematic way of studying complexity under various parameterizations. This study is facilitated by organizing the parameter space as a hierarchy, which is the topic of Section 2.3. It calls for a new type of race in parameterized analysis, comparable to the well-established races for the best running times and smallest kernel sizes, as outlined in Section 2.4. Finally, we explain in Section 2.5 how the systematic investigation of kernelization complexity outlined in the ecology program helps us to understand the power of preprocessing.

This chapter is based on the survey "Towards Fully Multivariate Algorithmics: Parameter Ecology and the Deconstruction of Computational Complexity" by Michael R. Fellows, Bart M. P. Jansen, and Frances A. Rosamond (European Journal of Combinatorics [101]).

2.1 The Main Ideas

Parameterized complexity starts from the premise that there are usually secondary measurements, apart from the primary measurement of overall input size, that can significantly affect the computational complexity of a problem, in qualitatively different ways that merit systematic investigation. Parameterized complexity makes room for an additional measurement called the *parameter*, usually denoted by an integer k. The theory revolves around the contrast between running times of the form $f(k)n^{\mathcal{O}(1)}$ versus $n^{g(k)}$. The subject of study is therefore a *parameterized problem*. There are two ways to formalize such problems [85, 107]; we adopt the notation introduced by Downey and Fellows [85].

Definition 2.1. A *parameterized problem* is a set $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet. In a tuple $(x, k) \in \Sigma^* \times \mathbb{N}$ we call x the *classical instance* and k the *parameter*.

To make the definition more concrete, we give an example relating to the Vertex Cover decision problem: given a graph G and integer ℓ, the question is whether G has a set of ℓ vertices that touch all its edges. The classical (i.e., not parameterized) language $\mathcal{L} \subseteq \Sigma^*$ corresponding to this problem is a set of strings that encode the yes-instances, and could be defined as follows.

$$\mathcal{L} := \big\{(G, \ell) \,\big|\, \text{graph } G \text{ has a vertex cover of size at most } \ell\big\}.$$

To make this definition rigorous, we should of course specify what the alphabet Σ is, and how tuples (G, ℓ) are encoded over this alphabet. Even though the particular choice of encoding does not matter in the end — as long as it is “reasonable”, as described by Garey and Johnson [117] — for the sake of this example we will be explicit. If the choose the three-letter alphabet $\Sigma := \{0, 1, \#\}$ then we can encode a pair (G, ℓ), where G is an n-vertex graph and $\ell \in [n]$, by a string as follows. The string contains the adjacency matrix of G in row-major order, using n bits per row, followed by the separator character $\#$, ending with the binary encoding of the number ℓ. This unambiguously defines $\mathcal{L}$ as a classical language formalizing the Vertex Cover problem.

Now suppose that we are interested in studying how the value of ℓ, the size of the desired vertex cover, affects the complexity of the problem. We would then consider the parameterized problem that is derived from the classical Vertex Cover problem by using the solution value as the parameter. The corresponding parameterized problem $\mathcal{Q} \subseteq \{0, 1, \#\}^* \times \mathbb{N}$ could be defined as follows.

$$\begin{aligned}\mathcal{Q} := \big\{(x, k) \,\big|\, & x \text{ encodes a tuple } (G, \ell) \text{ as described above, } k = \ell, \\ & \text{and } G \text{ has a vertex cover of size at most } \ell\big\}.\end{aligned}$$

The example illustrates why we call the component x the classical instance: restricting the set $\mathcal{Q}$ to its first component, we obtain the classical Vertex

Cover language $\mathcal{L}$. The parameter is added to the language to describe the measure of instance complexity whose contribution to the problem's complexity we are interested in. We will later see that using other structural measures as the parameter gives some technical difficulties, which are avoided here because the parameter corresponds directly to a value found in classical instances of the problem. We discuss such technicalities later in the chapter.

As in classical complexity theory, it is often more convenient to describe problems in natural language instead of defining them as sets of strings. Adding a line that describes the parameter to the instance-question format employed by Garey and Johnson, we might alternatively define the problem as follows.

Vertex Cover [sol]
Input: A graph G and an integer ℓ.
Parameter: The size of the desired vertex cover, i.e., $k := \ell$.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ that contains at least one endpoint of every edge?

The parameterized problem called Vertex Cover [sol] in this definition is more commonly referred to as k-Vertex Cover. While the latter notation is fine when discussing only a single parameterization of Vertex Cover, it is unsuitable for our purposes as we will be studying the problem with a range of different parameters.[1] The suffix [sol] indicates that we parameterize the problem by the size of the desired solution. This notational style will be applied throughout the thesis to refer to the parameterized problem derived from optimization problems by choosing the solution size as the parameter. This parameterization is also called the *natural parameterization* of optimization problems. More generally, we will use text in brackets after a problem name to specify the parameterization.

The above example shows how parameterized problems are derived from classical problems, by picking a numerical characteristic of the instance to use as the parameter. Many, if not all, interesting parameterized problems are defined in this way. In informal discussion we refer to the parameterized problem Vertex Cover [sol] as the parameterization of Vertex Cover with respect to the solution size, or simply as Vertex Cover parameterized by solution size. It is important to be aware of the distinction between classical and parameterized problems throughout this work.

2.1.1 The Positive Toolkit

The main objective when using parameterized complexity theory as a way of coping with NP-hardness, is to identify parameterizations that yield *fixed-parameter tractable* problems.

[1]Halmos [128] would also complain that the k-Vertex Cover notation is bad because it freezes the interpretation of the letter k.

Definition 2.2. A parameterized problem $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ is *strongly uniformly fixed-parameter tractable* if there is an algorithm ϕ, a constant $c \in \mathbb{N}$ and a computable function $f\colon \mathbb{N} \to \mathbb{N}$ such that for all $(x, k) \in \Sigma^* \times \mathbb{N}$:

- $\phi(x, k)$ terminates within $f(k)|x|^c$ steps, and
- $\phi(x, k)$ accepts if and only if $(x, k) \in \mathcal{Q}$.

Note that there are more relaxed notions of parameterized tractability, obtained by either dropping the requirement that f is computable, or by allowing a *different* algorithm for every value of k [85, Chapter 2]. The class of strongly uniformly fixed-parameter tractable parameterized problems is denoted FPT. Our investigation mainly revolves around FPT; we shall be explicit when discussing nonuniform or weak notions of parameterized tractability.

The class FPT generalizes P; parameterized complexity theory can be viewed as a natural two-dimensional generalization of the one-dimensional classical framework. For the purposes of this thesis, we are particularly interested in the following definition of preprocessing that can be expressed in the theory.

Definition 2.3. Let $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem and let $h\colon \mathbb{N} \to \mathbb{N}$ be a computable function. A *kernelization for $\mathcal{Q}$ of size $h(k)$* is an algorithm that on input $(x, k) \in \Sigma^* \times \mathbb{N}$ takes time polynomial in $|x| + k$ and outputs an instance (x', k') such that:

- the length of x' and the parameter value k' are bounded by $h(k)$, and
- $(x', k') \in \mathcal{Q}$ if and only if $(x, k) \in \mathcal{Q}$.

Such an algorithm is called a *polynomial kernelization* if h is a polynomial.

We shall also abbreviate kernelization by *kernel*. As has become customary, we also use the term kernel to refer to the output of the reduction algorithm when this does not lead to confusion.

Observe that the existence of a kernel for a parameterized problem expresses the fact that the complexity of the problem is governed by the parameter value k: in polynomial time we can "cut away" irrelevant parts of the problem until some core remains whose size only depends on k.

The following lemma shows that in general, kernelization is just the study of fixed-parameter tractability in disguise.[2]

Lemma 2.1 ([107, Theorem 1.39])**.** *A parameterized problem $\mathcal{Q}$ is FPT if and only if it is decidable and admits a kernelization whose size is bounded by a computable function.*

The search for a problem kernel becomes interesting (and, one could argue, useful) once we demand that the function h that bounds the size of the reduced instance is

[2]Bodlaender [21] discusses the history of the lemma; it is often stated incorrectly by omitting the requirement of decidability.

polynomial. The quest for polynomial kernels began in earnest after Alber et al. [4] found a kernelization algorithm that reduces an instance of Planar Dominating Set [sol] to an equivalent instance whose graph has at most $335k$ vertices. As an instance of Dominating Set is formed by a tuple (G, ℓ) where G is a graph and ℓ is an integer whose value does not exceed the vertex count of the graph, the size of an instance is determined by the size of G. We therefore often express the size of a kernel for a graph problem in terms of the number of vertices of reduced instances. In this case, we speak of a $335k$-vertex kernel. More generally, we speak of a *linear-vertex kernel* if the number of vertices in the reduced instance depends linearly on the parameter value.

When it comes to planar graphs, linear-vertex kernels often yield kernels of bitsize $\mathcal{O}(k)$: an n-vertex planar graph has $\mathcal{O}(n)$ edges and can be encoded in $\mathcal{O}(n)$ bits [141]. Since the edge count of unrestricted graphs may be quadratic in the number of vertices, a linear-vertex kernel generally does not yield a kernel of bitsize $\mathcal{O}(k)$ (but rather of bitsize $\mathcal{O}(k^2)$). We shall therefore be explicit in either expressing kernel sizes in the number of vertices, or in the number of bits of suitable encodings of the reduced instances. For some graph problems, the input consists of more than just a graph and an integer; for example, in the Disjoint Paths problem the input also contains a list of vertex pairs that are to be connected. In such cases it will often be easy to see that the size of the total instance is polynomially related to the vertex count of the graph, and therefore that a kernel with a polynomial number of vertices yields a polynomial (bitsize) kernelization as per Definition 2.3. When the total instance size is not polynomial in the vertex count of the associated graph, we will explicitly state how the kernel bounds should be interpreted.

Thus far we have seen that the positive FPT toolkit has two types of *deliverables*: efficient FPT algorithms, and kernels of small size. These two types of deliverables motivate two types of races in parameterized complexity research: the race for the smallest possible function $f(k)$ in the running time of an exact algorithm, and the race for the smallest possible function $h(k)$ to upper bound the size of a kernel. These races are well-established [93], and the current leader boards are exhibited on the FPT community wiki [192]. We will see in Section 2.3 that organizing the parameter space in a hierarchy naturally calls for a new kind of race, but first we continue our discussion of the main concepts of parameterized complexity.

2.1.2 The Negative Toolkit

Having defined the two types of good news that parameterized analysis potentially has to offer when facing an NP-hard problem, we shift our focus to the expressibility of negative results in the theory. We start by comparing the complexity of Vertex Cover [sol] discussed above to the natural parameterization of Dominating Set.

Dominating Set [sol]
Input: A graph G and an integer ℓ.
Parameter: The size of the desired dominating set, i.e., $k := \ell$.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ such that each vertex not in S, has a neighbor in S?

Although the classical versions of both problems are NP-complete, the *parameter k* contributes to the complexity of these two problems in two qualitatively different ways.

1. There is a simple *bounded search tree* algorithm for Vertex Cover [sol] that runs in time $\mathcal{O}(2^k n)$ [85].
2. The best known algorithm for Dominating Set [sol] uses fast matrix multiplication [92], and gives a running time of $\Theta(n^{\Theta(k)})$.

The running times suggest that the parameterized Dominating Set [sol] problem is harder than its counterpart Vertex Cover [sol]. The theory of parameterized complexity has notions that allow this intuition to be formalized.

Classically, evidence that a problem is unlikely to have a polynomial-time algorithm is given by determining that it is NP-hard, PSPACE-hard, EXP-hard, and so on. In parameterized complexity analysis there are analogous means to show likely parameterized intractability. The tower of parameterized complexity classes [85, 107] relevant to this work is:

$$\mathrm{FPT} \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[t] \subseteq W[P] \subseteq \mathrm{XP}.$$

The *bounded-weft W-hierarchy* contains the most important classes of presumed parameterized intractability. The class $W[i]$ contains all parameterized problems that reduce (in the sense formalized by Definition 2.5) to the weighted circuit satisfiability problem for constant-depth Boolean circuits of *weft i*. This problem asks whether a Boolean circuit of some constant (but arbitrarily large) depth can be satisfied by setting k inputs to true, and k is used as the parameter. A circuit family has *weft* at most i if there is a constant $c \in \mathbb{N}$ such that in any circuit in the family, the maximum number of gates of fan-in more than c on a path from an input to the output is at most i. We refer to Downey and Fellows [85] for details. The complexity class XP, pronounced slicewise P, is defined as follows.

Definition 2.4. A parameterized problem $\mathcal{Q}$ is contained in *strongly uniform XP* if there is a computable function $f\colon \mathbb{N} \to \mathbb{N}$ and an algorithm ϕ that decides membership of an instance (x, k) in $|x|^{f(k)} + f(k)$ steps.

We shall simply refer to this complexity class as XP. All the containment relations above are believed to be strict. The working hypothesis of parameterized complexity theory is that FPT $\neq W[1]$, and consequently that $W[1]$-hard problems cannot be solved in $f(k)n^c$ time for any computable function f. The hypothesis can be compared to widely-held belief that P $\neq$ NP, which says that NP-hard problems cannot be solved in polynomial time. It is known [50, 84] that if the *Exponential Time Hypothesis* of Impagliazzo and Paturi [139] holds, then FPT $\neq W[1]$ (cf. [188, Theorem 13.32]).

The only separation known so far is that FPT $\subsetneq$ XP, which is based on a straight-forward diagonalization [85, Chapter 15]. Although the evidence for intractability provided by a $W[1]$-hardness proof rests on the conjecture that FPT $\neq W[1]$, the separation allows XP-hardness proofs to be used to establish *provable intractability.* As FPT $\subsetneq$ XP, classifying a parameterized problem as XP-hard shows unconditionally that it is not fixed-parameter tractable. In the context of parameterized complexity theory, the relevant notion of reducibility for proving such hardness results is the following.

Definition 2.5. Let $\mathcal{Q}$ and $\mathcal{Q}'$ be parameterized problems. An algorithm ϕ is a *parameterized reduction* from $\mathcal{Q}$ to $\mathcal{Q}'$ if there is a computable function $f\colon \mathbb{N} \to \mathbb{N}$ and a constant $c \in \mathbb{N}$ such that, on input $(x, k) \in \Sigma^* \times \mathbb{N}$, algorithm ϕ outputs an instance (x', k') satisfying:

- $(x, k) \in \mathcal{Q}$ if and only if $(x', k') \in \mathcal{Q}'$,
- k' is bounded by $f(k)$, and
- the running time of ϕ is bounded by $f(k)|x|^c$.

To make our view of the parameterized complexity classes more concrete, we place some well-known problems in the complexity tower. The Vertex Cover [sol] problem is, of course, fixed-parameter tractable as can be seen from the given running time $\mathcal{O}(2^k n)$. The parameterized class $W[1]$ is strongly analogous to NP, because the Short Single-Tape Turing Machine Acceptance [sol] problem is complete for $W[1]$ (see [49]). The variant for two-tape Turing machines is $W[2]$-complete [51]. When it comes to more applied problems, the familiar Independent Set [sol] problem is complete for $W[1]$ and Dominating Set [sol] is complete for $W[2]$ (cf. [85]). These facts explain why the degree of the polynomials that bound the running times of the best-known algorithm for these problems still depends on k. The naturally parameterized Bandwidth problem is $W[t]$-hard for all t [25], but not known to belong to $W[P]$.

2.1.3 Nondeterminism and Advice

In this section we discuss complexity classes related to *nondeterminism* and computation with *advice*. Nondeterminism allows us to define *para-NP*, the last parameterized complexity class relevant for this thesis, while also giving the required background for the *co-nondeterministic* compositions and kernelizations that are briefly discussed in Chapter 3. Computations with advice are used in the complexity-theoretic hypothesis upon which the lower bounds for the sizes of kernelizations are based.

Nondeterminism

In our discussion we assume familiarity with the basic model of a *nondeterministic Turing machine* (NDTM), which is described in all texts on complexity theory [10, 117, 118].

Definition 2.6. Let M be a nondeterministic Turing machine. Machine M *nondeterministically decides* a language $\mathcal{L} \subseteq \Sigma^*$ if for each string $x \in \Sigma^*$, the computation of M with input x satisfies the following properties:

- if $x \in \mathcal{L}$ then at least one computation path ends in the ACCEPT state, and all other computation paths terminate (in the REJECT state).
- if $x \notin \mathcal{L}$ then all computation paths end in the REJECT state.

Machine M *co-nondeterministically decides* $\mathcal{L}$ if the computation instead satisfies the following:

- if $x \in \mathcal{L}$ then all computation paths end in the ACCEPT state.
- if $x \notin \mathcal{L}$ then at least one computation path ends in the REJECT state, and all other computation paths terminate (in the ACCEPT state).

The running time of a nondeterministic machine is measured by the length of its longest computation path.

Definition 2.7. A *polynomial-time nondeterministic Turing machine* is an NDTM, say M, for which there exists a polynomial p such that for all $x \in \Sigma^*$, each computation path of M with input x terminates within $p(|x|)$ steps.

To illustrate these concepts we show how they can be used to give alternative definitions for the familiar classes NP and coNP. The class NP can be defined as those languages $\mathcal{L}$ for which there is a polynomial-time NDTM that nondeterministically decides $\mathcal{L}$. The class coNP, often defined as $\{\overline{\mathcal{L}} \mid \mathcal{L} \in \text{NP}\}$, can also be defined as containing those languages that are co-nondeterministically decided by a polynomial-time NDTM. Nondeterministic computation is used in the definition of the final parameterized complexity class of this section; it is the parameterized analogue of NP [107].

Definition 2.8. A parameterized problem $\mathcal{Q}$ is in *para-NP* if there is a computable function $f\colon \mathbb{N} \to \mathbb{N}$, a constant $c \in \mathbb{N}$, and an NDTM that nondeterministically decides whether $(x, k) \in \mathcal{Q}$ within $f(k)|x|^c$ steps.

The class para-NP can be used to give evidence that a problem is not fixed-parameter tractable: para-NP-hard problems do not have FPT algorithms under the assumption that P $\neq$ NP. This follows from the fact that FPT = para-NP if and only if P = NP [107, Corollary 2.13]. Hence para-NP and XP both contain FPT, but are believed to be distinct classes. As a concrete example, observe that deciding whether a graph is ℓ-colorable is para-NP-complete when ℓ is the parameter [107, Example 2.17].

Advice and NP $\subseteq$ coNP/poly

We now turn to the advice classes NP/poly and coNP/poly, which are used to express the complexity-theoretic hypotheses that form the basis for several types of lower bounds on the sizes of kernelizations. We keep our discussion informal, as it is only meant to place the complexity-theoretic evidence for our lower bounds into context.

A (nondeterministic) Turing machine with advice is a Turing machine that has an extra read-only input tape, the advice tape, containing a string that only depends on the length of the input. The advice is given by an *advice function* $f\colon \mathbb{N} \to \Sigma^*$. When applying an NDTM with advice f to an input x, the string $f(|x|)$ is placed on the advice tape before the computation starts. This information can be used by the Turing machine during its computation, which proceeds as usual except for the fact that the machine may access the contents of the advice tape. An advice function is *polynomial* if there is a polynomial p such that $|f(n)| \leq p(n)$ for all $n \in \mathbb{N}$.

The class NP/poly contains the languages nondeterministically decided by an NDTM with polynomial advice; similarly coNP/poly contains the languages co-nondeterministically decided by an NDTM with polynomial advice. These advice classes are more powerful than their normal counterparts, as they contain sparse undecidable languages such as the unary encoding of the Halting Problem [10, §6.1.1]. It is conjectured that NP $\not\subseteq$ coNP/poly. This widely-believed complexity-theoretic conjecture is similar to the conjecture that P $\neq$ NP, but stronger: NP $\not\subseteq$ coNP/poly implies P $\neq$ NP, but the reverse is not known to hold. Note that NP $\not\subseteq$ coNP/poly is equivalent to coNP $\not\subseteq$ NP/poly. Most superpolynomial lower bounds on the sizes of kernelizations are conditioned on the assumption that NP $\not\subseteq$ coNP/poly.

There are several ways to motivate the belief in the conjecture. First consider the classes NP and coNP, forgetting about advice for the moment. NP is defined in terms of existential quantifiers (is there an accepting path?), while coNP deals with universal quantifiers (do all computation paths accept?). The existence of a short, efficiently verifiable certificate for membership of a string in the language puts that language in NP, while coNP requires such a certificate for *non*-membership of a string. On a very fundamental level it seems that these two notions are inherently different. For example, having a short and efficiently verifiable certificate for membership such as a satisfying assignment for a Boolean formula, does not lead to any obvious way of efficiently certifying the nonsatisfiability of a formula. This motivates our belief that NP and coNP are incomparable. Now, it seems that giving an NDTM access to polynomial-size advice — a single, short string to be used on the exponentially many inputs of a given length — does not add enough power to change this situation, suggesting that indeed NP $\not\subseteq$ coNP/poly.

A second justification for the belief in NP $\not\subseteq$ coNP/poly is given by its connections to the *polynomial hierarchy*. The polynomial hierarchy is an infinite tower of complexity classes, which may be defined by variants of the problem Truth

Checking For Quantified Boolean Formulae with increasing numbers of quantifier alternations. If the polynomial hierarchy collapses, then there is a value of i such that checking quantified Boolean formulae with $i + 1$ quantifiers can be efficiently reduced to checking formulae with i quantifiers. This seems impossible, and motivates our belief that increasing the number of alternations increases the difficulty of the problem. Hence it seems implausible for PH to collapse to any particular level. Yap [212] proved that NP $\subseteq$ coNP/poly implies that the polynomial hierarchy collapses to its third level[3] (PH $= \Sigma_3^p$), which again suggests that NP $\not\subseteq$ coNP/poly.

Uses of intractability theory

Having described various types of parameterized intractability, we highlight their role in this thesis. Rather than studying parameterized (in)tractability, we will mostly be interested in the existence of polynomial kernelizations for problems *within* FPT. This requires a new toolkit, which will be developed in Part II. The intractability concepts we just presented will nevertheless be useful: by Lemma 2.1 any quest for a polynomial kernel should start with an FPT classification. This is where the introduced notions come into play.

For further background on parameterized complexity we refer the reader to the textbooks [85, 107, 188], the double special issue of surveys of aspects of the field and various application areas [86], and to the recent survey by Downey and Thilikos [88]. Having given an overview of the main concepts of parameterized complexity theory, we discuss the notion of parameter ecology and show how it sets a roadmap for the study of parameterized problems.

2.2 The Complexity Ecology of Parameters

For Vertex Cover [sol] and Dominating Set [sol], the parameter is the size of the solution being sought. But a parameter that affects the complexity of a problem can be many things. A beautiful motivating example is provided by the problem ML Type Checking.

ML is a logic-based programming language for which relatively efficient compilers exist. One of the problems the compiler must solve is the checking of the compatibility of type declarations. This problem is known to be complete for EXP, so the situation appears discouraging from the standpoint of classical complexity theory [133]. The implemented compilers contain an FPT-time type checking routine with a running time of $\mathcal{O}(2^k n)$, where n is the size of the program and k is the maximum nesting depth of the type declarations [174]. Despite the severe intractability of type checking from a classical point of view, the implemented ML compilers work efficiently because they *exploit the bounded nesting depth* of the

[3]Cai et al. [47] improved Yap's result and showed that NP $\subseteq$ coNP/poly collapses PH to the class $\mathrm{S}_2^{\mathrm{NP}}$, but this does not contribute much to the intuition.

inputs. Programs written by humans typically have a maximum type-declaration nesting depth of $k \leq 5$, causing the runtime of $\mathcal{O}(2^k n)$ to be entirely acceptable. Moreover, the reason that programs occurring in practice have small nesting depth is because they would otherwise become incomprehensible to the programmer creating them!

Type checking is not an optimization problem; hence there is no obvious "natural" parameter. The example shows that relevant secondary measurements that affect problem complexity can take *many different forms*, such as the size of the solution, aspects of the structure of typical instances, aspects of the algorithmic approach, or the quality of an approximation (cf. [189]). For any real-world problem, there may be several such secondary measurements in various ranges of magnitude that should be considered, or measurements with qualitatively different roles, such as structural (e.g., treewidth) and solution size.

The example of ML Type Checking points to the phenomenon that the "inputs" to computational problems of interest to real-world algorithmics are often not at all arbitrary, but rather are produced by other natural computational processes (e.g., the thinking processes and abilities of the programmer) that are themselves subject to computational constraints. In this way, the natural input distributions encountered in abstractly defined computational problems often have inherited structural regularities and restrictions (relevant parameters, in the sense of parameterized complexity) due to the natural complexity constraints on the generative processes. This connection is what we refer to as the *ecology* of computation.

Having identified a reason to expect some kind of structure in real-world problem instances, we are led to explore how to exploit that structure in order to solve problems more efficiently. It therefore seems useful to know how all the various parameterized structural notions interact with the computational objectives one might have. The goal of the *parameter ecology program* is to determine systematically how different parameters affect the complexity of a problem. For example, faced with the apparent parameterized intractability of Dominating Set [sol], one could ask whether a bound on the treewidth of the input graphs allows Dominating Set to be solved more efficiently. It has long been known that this is the case: when a tree-decomposition of width k is supplied, a minimum size dominating set can be computed in $\mathcal{O}(3^k k^2 n)$ time [201]. The familiar paradigm of efficiently solving various problems for graphs of bounded treewidth represents just one line of research in this ecology program. Before discussing further examples, we consider how such structural parameterizations can be formalized in the framework of parameterized complexity theory.

2.2.1 Formalizing Structural Parameterizations

Properly formalizing the intuitive notion of nonstandard parameterization requires some care. We discuss potential issues, their solutions, and justify the conventions that have been chosen for this thesis. We focus on the Dominating Set problem as an example. Suppose we want to determine whether a restriction on the richness of the input graph, in terms of a bound on its bandwidth, allows DOMINATING SET to be solved more efficiently. We would then study the complexity of the following parameterization by bandwidth.

> DOMINATING SET [BW]
> **Input:** A graph G and integer ℓ.
> **Parameter:** A value k that bounds the bandwidth of G.
> **Question:** Does G have a dominating set of size at most ℓ?

What is the parameterized complexity of this problem? It has long been known that the bandwidth of a graph bounds its pathwidth [20, Theorem 44], which in turn bounds its treewidth. Recall from the example in the previous section that DOMINATING SET can be solved in FPT-time given a tree decomposition of width k, and note that a tree decomposition of a graph having bandwidth k (and hence treewidth at most k) can be found in $2^{\mathcal{O}(k^3)}n$ time using Bodlaender's algorithm [19]. Hence it would seem that DOMINATING SET [BW] can be solved in FPT-time by first computing a width-k tree decomposition, and then applying the algorithm for DOMINATING SET on graphs of bounded treewidth to the resulting decomposition.

This intuition about the difficulty of the problem is not captured by all possible formalizations. This is partly due to issues that are hidden by the instance-parameter-question notation. Recall that at the most elementary level, a parameterized problem is a set of pairs (x, k) that encode the YES-instances to the problem. Which set of tuples $\mathcal{Q}$ corresponds to DOMINATING SET [BW]? A first answer might look like this:

$$\mathcal{Q} := \big\{(x,k) \,\big|\, x \text{ encodes a graph } G \text{ of bandwidth at most } k \text{ and an integer } \ell\text{, s.t. } G \text{ has a dominating set of size at most } \ell\big\}.$$

Unfortunately the behavior of this formalization differs radically from our intended meaning, which is best illustrated by noting that membership in $\mathcal{Q}$ is $W[t]$-hard to decide for all t! Since a graph G on n vertices trivially has a dominating set of size n, by our current definition the instance $(x := \langle G, n\rangle, k)$ is contained in $\mathcal{Q}$ if and only if G has bandwidth at most k, and this results in the aforementioned complexity [25]. The problem comes from the fact that the distinction between parameter and solution value was lost in the formalization, and was effectively turned into the task of simultaneously deciding the bandwidth and the domination number of the input graph. The treewidth-based approach sketched above fails to decide $\mathcal{Q}$ correctly, as it may falsely output YES on treewidth-k graphs that have a dominating set of size ℓ, but whose bandwidth exceeds k.

It is clear that if we are interested in studying how structural measures of problem instances affect their complexity, we should choose a different way to formalize the corresponding parameterized problems. There are two routes to formal parameterized problems that capture our intended meaning, which we discuss below.

Using a Promise Problem

The most robust solution to our formalization issue is to model the parameterization as a promise problem (cf. [119]).

Definition 2.9. A *parameterized promise problem* is a pair $(\mathcal{Q}_Y, \mathcal{Q}_N)$ of disjoint subsets of $\Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet. The set $\mathcal{Q}_Y$ contains the YES-instances satisfying the promise, whereas $\mathcal{Q}_N$ contains the NO-instances satisfying the promise.

The corresponding notion of parameterized tractability is defined as follows.

Definition 2.10. A parameterized promise problem $(\mathcal{Q}_Y, \mathcal{Q}_N)$ is *strongly uniformly fixed-parameter tractable* if there is an algorithm ϕ, a constant $c \in \mathbb{N}$ and a computable function $f \colon \mathbb{N} \to \mathbb{N}$ such that for all $(x, k) \in \mathcal{Q}_Y \cup \mathcal{Q}_N$:

- $\phi(x, k)$ terminates within $f(k)|x|^c$ steps, and
- $\phi(x, k)$ accepts if and only if $(x, k) \in \mathcal{Q}_Y$.

Observe that by the definition above, the behavior of ϕ on inputs in $(\Sigma^* \times \mathbb{N}) \setminus (\mathcal{Q}_Y \cup \mathcal{Q}_n)$ may be arbitrary; the algorithm is not even required to halt. In slight abuse of notation we will say that a parameterized promise problem is FPT if it is strongly uniformly fixed-parameter tractable by the definition above. The notion of kernelization can also be lifted to promise problems.

Definition 2.11. Let $(\mathcal{Q}_Y, \mathcal{Q}_N)$ be a parameterized promise problem and let $h \colon \mathbb{N} \to \mathbb{N}$ be a computable function. A *kernelization for* $(\mathcal{Q}_Y, \mathcal{Q}_N)$ *of size* $h(k)$ is an algorithm that on input $(x, k) \in \mathcal{Q}_Y \cup \mathcal{Q}_N$ takes time polynomial in $|x| + k$ and outputs an instance $(x', k') \in \mathcal{Q}_Y \cup \mathcal{Q}_N$ such that:

- the length of x' and the parameter value k' are bounded by $h(k)$, and
- $(x', k') \in \mathcal{Q}_Y$ if and only if $(x, k) \in \mathcal{Q}_Y$.

The definition does not restrict a kernelization algorithm's behavior on inputs that violate the promise, but requires inputs that do satisfy the promise to be mapped to outputs satisfying the promise.

The notion of structural parameterization can be robustly formalized by parameterized promise problems. Let us consider how this plays out for DOMINATING SET parameterized by bandwidth. To formalize DOMINATING SET [BW] as a parameterized promise problem we can take $\mathcal{Q}_Y$ to be the set $\mathcal{Q}$ of YES-instances defined earlier. We define the set $\mathcal{Q}_N$ of well-formed NO-instances as the pairs (x, k) for which x encodes a tuple $\langle G, \ell \rangle$ where G has bandwidth at most k, and G does *not* have a dominating set of size ℓ.

Since the instances where the bandwidth of G exceeds k fall outside the range of well-formed YES- or NO-instances, the restriction to a promise problem effectively means that we do not have to verify the claimed bound on the bandwidth of the input graph. All we demand is that *if* the bandwidth of the input graph is suitably bounded, *then* the algorithm gives the correct answer in FPT-time. The promise version of Dominating Set parameterized by bandwidth is indeed contained in FPT by the treewidth approach sketched earlier.

Formalizing a structural parameterization as a promise problem thus sidesteps the issue of having to verify the parameter value. The formalization technique results in parameterized promise problems whose complexity indeed reflects the effect of the parameter value on the problem's complexity, which we are interested in studying. Unfortunately, the use of a promise problem means that whenever we apply theorems from the FPT-toolkit that were proven for arbitrary decision problems, we have to ensure that they still hold in the promise setting. The next approach allows us to stay within the established definitions of parameterized problems, but is not applicable to all types of parameters.

Supplying a Witness in the Input

When studying a structural parameterization of problem Π, we can conceptually simplify matters by studying two separate questions: (a) how hard is it to compute the relevant structural measure, either exactly or approximately, and (b) how hard is it to solve Π when a *witness* for the structure is given? The first problem is subject of study when considering the parameterized complexity or approximation complexity of the optimization problem corresponding to the structural measure. The most interesting question from the viewpoint of parameter ecology is question (b). We could therefore opt for the following alternative definition of our running example, giving a witness of the graph structure "for free".

> DS Parameterized by the Bandwidth of a Layout
> **Input:** A graph G, an integer ℓ, and a linear layout of G given by a bijection $f\colon V(G) \to \{1, \ldots, |V(G)|\}$.
> **Parameter:** The bandwidth $k := \max_{\{u,v\} \in E(G)} |f(u) - f(v)|$ of f.
> **Question:** Does G have a dominating set of size at most ℓ?

The formal parameterized language corresponding to this definition contains those pairs (x, k) where x encodes a tuple $\langle G, \ell, f \rangle$ such that f is a linear layout of bandwidth at most k for G, and G has a dominating set of size at most ℓ. By supplying a layout in the input, we can *test in polynomial time whether an instance obeys the claimed bound* on its bandwidth or not. Such a test allows us to classify this formal parameterized problem as FPT, by rejecting all instances where f is not a layout of bandwidth at most k and then applying the treewidth approach.

Since the bandwidth of the *given* layout is used as the parameter, rather than the optimum bandwidth, verifying the claimed structural restriction becomes trivial. In a similar way we may formalize parameterizations corresponding to the solution values of other NP-optimization problems; these problems have small, efficiently verifiable witnesses that can be added to the input specification.

However, this type of formalization only captures relevant questions for parameters corresponding to minimization problems; for maximization problems such a formalization might be para-NP-complete even though the promise-route to formalization would yield a problem in FPT. As an example, consider the Dominating Set problem parameterized by the number of leaves in a given spanning tree of the graph. As the maximum number of leaves in a spanning tree of a graph linearly bounds its treewidth (see Fig. 2.1), we expect this parameterized problem to be FPT using treewidth techniques [23, Theorem 4.5]. Now consider the following parameterized problem.

> DS Parameterized by the Leaves of a Spanning Tree
> **Input:** Graph G, an integer ℓ, and a spanning tree T of G.
> **Parameter:** The number of leaves k in tree T.
> **Question:** Does G have a dominating set of size at most ℓ?

It is easy to devise a polynomial-time transformation that, given a graph G, outputs a graph G' and a Hamiltonian path in G' such that the domination number of G' matches that of G.[4] Using this transformation, we can reduce the classical Dominating Set problem to our last parameterization of Dominating Set with a *constant* parameter value $k = 2$, by using the Hamiltonian path as a spanning tree with exactly two leaves. Hence the considered parameterized problem is para-NP-complete, rather than FPT as we would expect.

The underlying issue is that when a structural witness is part of the input, the supplied witness may be suboptimal. In the case of minimization problems, suboptimality results in a larger witness and therefore a larger value of k than the optimum k^*. Hence if the problem can be solved in $f(k^*)n^c$ time, then certainly the same can be done in $f(k)n^c$ time. But for maximization problems, a suboptimal witness results in a value of k that is smaller than the optimum, and which therefore allows an algorithm less running time compared to the bound following from the optimum.

In summary, adding a witness of the problem structure to the input specification of a parameterized problem solves the formalization issue for parameterizations corresponding to NP minimization problems. For maximization parameters, or for parameterizations that do not have short, efficiently verifiable witnesses, we nevertheless have to resort to promise problems.

Conventions of Problem Parameterization

We end our discussion of problem formalization by discussing the conventions adopted in this thesis. Parameterizations by a bound on the solution size of an NP minimization problem (such as the vertex cover number, the feedback vertex number, or the treewidth) are formalized by supplying an NP witness (e.g., a set of vertices that form a vertex cover) along with the input. For parameterizations

[4]From G construct G' as the split graph with clique $X := \{v', v'' \mid v \in V(G)\}$ and independent set $Y := \{v''' \mid v \in V(G)\}$, adding edges from v' and v'' to u''' whenever $u \in N_G[v]$. The Hamiltonian path consists of $(v'_1, v'''_1, v''_1, v'_2, v'''_2, v''_2, \dots, v''_n)$ for an arbitrary ordering of the vertices.

by the optimum values of maximization problems, or by values that cannot be verified in polynomial time, we turn to the setting of promise problems. In this thesis, we only use promise problems for the results in Chapter 8. We explicitly define parameterized problems whenever confusion may arise.

One may question the relevance of kernelization results for problems whose definition requires a witness to be given along with the input. Our choices may be justified as follows. When it comes to negative results, observe that having access to the witness only makes it *easier* to compress the problem. A kernelization lower bound for the version where a witness is supplied with the input, is therefore *stronger* than a bound for the version where a witness is not given. Even so, all our lower bounds apply to such problem formalizations.

When it comes to kernelization upper bounds, it takes some more consideration to justify preprocessing schemes that rely on having a witness to the parameter structure in the input. How would the preprocessing be applied in real-world settings, where the input to a computational problem is not accompanied by a map of its structure? For most parameterizations we consider (such as vertex cover number, feedback vertex number, and vertex-deletion distance to a clique) one may simply use a polynomial-time constant-factor approximation algorithm to find the instance structure [11, 137], and use this approximate solution as the witness. In settings where no such approximation algorithm is known, one may be able to find ad-hoc solutions to these issues. Whether or not this is possible, we feel that it is relevant to distinguish between the *approximation complexity* of structural parameterizations and their *kernelization complexity*. By asking for a witness set in the input we decouple these two types of complexity, and focus on the potential for preprocessing that forms the core subject of this work.

2.2.2 A Table View of Ecology: Problems vs. Parameters

With these conventions for problem parameterization we can continue our discussion of the parameter ecology program. Table 2.1 gives a sample of the research directions in the program: it shows how the complexity of six popular graph problems depends on the amount of structure in the graph, as measured by the values of the optima of those six problems. The entry in the second row and fourth column of the table represents the running example of the previous section: there is an FPT algorithm to optimally solve the DOMINATING SET problem for a graph G of bandwidth at most k. The table also shows that there is no polynomial kernel for this problem unless NP $\subseteq$ coNP/poly. The entry in the fourth row and second column indicates that it is unknown whether BANDWIDTH can be solved optimally by an FPT algorithm when the parameter is a bound on the domination number of the input.

Although the table gives just a few examples of the types of problems we are concerned with, the ecology program targets the *unbounded conceptual matrix* of problems versus parameters. The popular approach of efficiently solving problems on graphs of bounded treewidth treats just one row of this matrix. It is the hope

Table 2.1: The complexity ecology of parameters.

k	Problem to be solved					
	TW	BW	VC	DS	G	ML
TW	☺☹[19, 24]	☹☹	☺☹[24, 37]	☺☹[24, 37]	☺☹[24, 152]	☺☹[24, 37]
BW	☺☹[19, 24]	☹☹[25]	☺☹[24, 37]	☺☹[24, 37]	☺☹[152] ★	☺☹[24, 37]
VC	☺☺[35]	☺ ? [103]	☺☺[53, 187]	☺☹[37, 83]	☺ ? [152]	☺☹[37, 83]
DS	☹☹★	? ?	☹☹★	☹☹[85]	? ?	☹☹[85]
G	? ?	☹☹[117]	☹☹★	☹☹[117]	☺ ? [152]	☹☹[117]
ML	☺☺[153] ★	☺ ? [103]	☺☺[103]	☺☺[103]	☺☺[153] ★	☺☺[93]

Each column corresponds to a classical problem Π, and each row to a parameterization k of that problem. We use the shorthand: TW is Treewidth, BW is Bandwidth, VC is Vertex Cover, DS is Dominating Set, G is Orientable Genus and ML is Max Leaf. An entry shows the current state of knowledge concerning the existence of an FPT algorithm or polynomial kernel when the input graph is assumed to have a structural bound described by the *row*, and the problem described by the *column* is to be solved to optimality. If the first symbol is ☺ then the parameterized problem is in FPT, while ☹ signifies hardness for $W[1]$. The second symbol indicates whether there is a polynomial kernel (☺) or not (☹). The kernelization lower-bounds for problems in FPT are conditioned on NP $\not\subseteq$ coNP/poly (cf. [24, 89]). Entries marked with ★ are not stated explicitly in the literature, but follow from known results or unpublished insights.

that by exploring the matrix we develop tools that allow us to cope with the parameterized intractability of certain parameterized problems, for example of Dominating Set [sol], by finding different parameterizations that admit FPT algorithms. If the generative processes of real-world instances result in sufficient structure to allow them to be solved within reasonable time constraints, then one expects this to corresponds to a parameterization in FPT with a parameter value that is small on these inputs. By exploring the matrix of problems versus parameters we may understand the properties of Dominating Set instances that make them computationally difficult, while hopefully finding practical approaches to solve real-world instances of the problem by uncovering new parameterizations in FPT.

2.3 A Hierarchy of Parameters

The ecology program opens up a wide array of new research questions. One might wonder whether all the entries in the conceptually infinite matrix are really interesting. Indeed, not all the cells of Table 2.1 hold the same appeal. For example, many of the FPT classifications for the Bandwidth row of Table 2.1 are based on the fact that the bandwidth of a graph bounds its treewidth, together with FPT algorithms for bounded treewidth: these entries tell us nothing new. The table

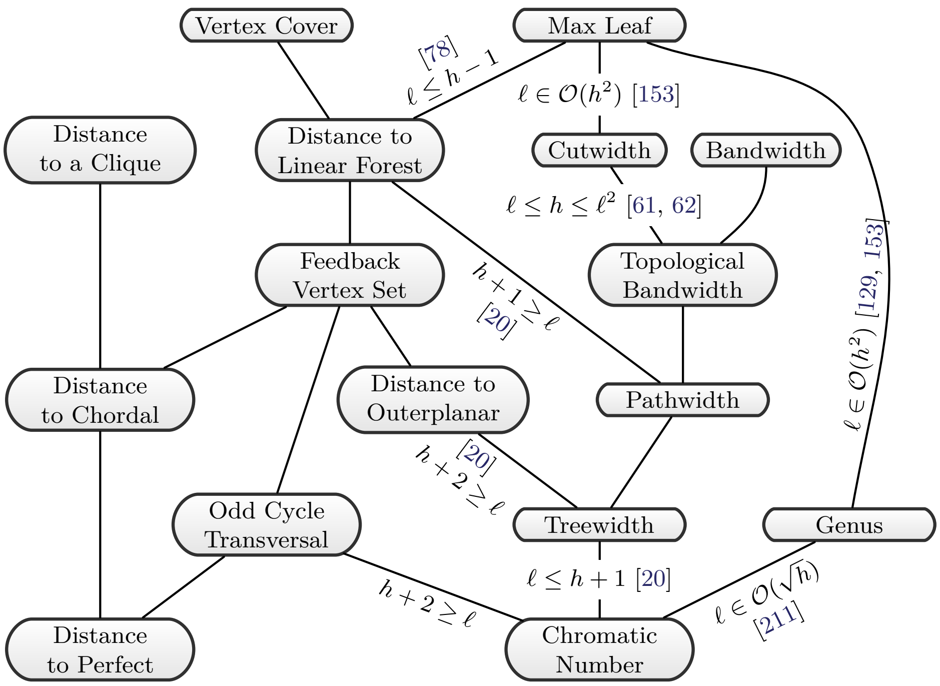


Figure 2.1: A hierarchy of parameters, with larger parameters drawn higher. An unlabeled line between two parameters means that the parameter drawn lowest is never larger than the one drawn highest. These unlabeled relationships (cf. [202]) follow from inclusions between graph classes [45] or bounds which can be found in Bodlaender's survey on treewidth [20]. When a line between parameters is labeled by a bound, the lower-drawn parameter is represented by ℓ and the higher-drawn parameter by h.

fails to take the *relationships* between various parameters into account, whereas a thorough understanding of the interplay between parameters is crucial for a proper grip on problem complexity. In Fig. 2.1 we therefore organize structural parameters of graphs into a hierarchy, based on the natural partial ordering on them. The parameters in the hierarchy correspond to the optimum values of classic optimization problems on graphs, structural measures, and to the (vertex-) deletion distance to well-known graph classes (see Appendix A). For example, the parameter *odd cycle transversal* is the size of the smallest vertex set intersecting all odd-length cycles, and the *distance to linear forest* is the minimum number of vertices whose removal results in a forest of maximum degree at most two. The figure gives relationships between these parameters: we see that graphs of vertex cover number k or max leaf number k both have treewidth $\mathcal{O}(k)$, while the vertex cover number of a graph can be arbitrarily large compared to its max leaf number, and vice versa. The following terminology facilitates a comparison of graph parameters.

Definition 2.12. A *graph parameter* π is a function π from the finite graphs to $\mathbb{N}$. A parameter π *is bounded in* parameter π' if there is a computable function $f\colon \mathbb{N} \to \mathbb{N}$ such that $\pi(G) \leq f(\pi'(G))$ for all graphs G. If this holds for a polynomial f, then π *is polynomially bounded in* π'. If π is (polynomially) bounded in π', but π' is not bounded in π, then π *(polynomially) dominates* π'.

Many of the entries in the Bandwidth row of Table 2.1 are explained by observing that Treewidth is bounded in (and in fact, dominates) Bandwidth: the treewidth of a graph never exceeds its bandwidth, while stars $K_{1,n}$ have treewidth one but bandwidth $\Theta(n)$. Complexity classifications propagate through the parameter hierarchy. If parameter π' is dominated by π, then an FPT algorithm for a parameterization by π implies[5] the same FPT status for the parameterization by π'. Hardness results propagate from larger parameters to smaller ones: a $W[1]$-hardness proof for parameterization π' also implies $W[1]$-hardness by the dominating parameter π. When the domination relation is based on a polynomial bound (which is the case for all relations in Fig. 2.1) then the (non-)existence of polynomial kernelizations propagates through the hierarchy as well: positive results transfer upwards to larger parameters, and negative results transfer downwards to smaller parameters. We will return to this in the next chapter, when we discuss polynomial-parameter transformations.

Using the propagation of results by the given relationships, the hierarchy *guides* us in our attack on intractability, pointing us to interesting questions and telling us over which flank a problem may be attacked. For example, as we know that Bandwidth is NP-complete on trees [117, GT40], which have trivial feedback vertex sets of size zero, we know that no parameterization below *feedback vertex set* can be FPT unless P = NP; hence we should look for FPT results for parameters that are larger than, or incomparable to, the feedback vertex number. When also taking the $W[t]$-hardness of the natural parameterization into account, the hierarchy shows that the remaining viable regions for tractable parameterizations are the vertex-deletion distance to a single clique, the distance to a linear forest, or the vertex cover/max leaf number. And indeed, early results in the parameter ecology program have shown fixed-parameter tractability for the latter two [103, 104].

The idea of considering "larger" parameterizations when encountering intractability has been applied in a few cases; especially regarding the parameter treewidth. Thanks to recent work we now know that besides Bandwidth, there are various other problems that are $W[1]$-hard for the parameter treewidth. Among them are List Coloring, Precoloring Extension, and Equitable Coloring [96]. Not all these problems behave the same using larger parameterizations like vertex cover number: although Precoloring Extension and Equitable Coloring become fixed-parameter tractable, List Coloring remains $W[1]$-hard for graphs of bounded vertex cover number [106].

[5]Assuming that the two parameterizations are similarly formalized, as discussed in Section 2.2.1.

These examples illustrate that the questions that the parameter hierarchy leads us to ask are not about toy problems, but are well motivated due to the inherent difficulty of the problems: $W[1]$-hardness of the parameterization by treewidth *forces us* to resort to larger parameters when looking for fixed-parameter tractability. The fact that the problem is $W[1]$-hard hard for smaller parameterizations leaves us little other choice.

2.4 Towards the Boundaries of Tractability

Although FPT classifications for parameterizations by vertex cover number give us some grip on problems such as Bandwidth that are $W[1]$-hard parameterized by treewidth, they should in no way be the end of our investigations.

Consider a graph parameter π that dominates[6] the parameter π'. The structural difference between the parameters implies that there are inputs where a *bigger* function of the *smaller* parameter yields a better overall run-time than the smaller function on the larger parameter. Similarly, there will be inputs where the kernel bound guaranteed by a bigger (yet polynomial) function of a smaller parameter beats the bound that is guaranteed for the larger parameterization.

Hence it is *always* meaningful to find out whether positive news (in the form of FPT-algorithms or polynomial kernels) can be transferred to smaller parameters — *even* if this (initially) requires a worse dependency on the parameter in the run-time or kernel size. When it comes to Bandwidth, the hierarchy shows that there is a range of parameters between vertex cover number and treewidth; we should not be satisfied until we know exactly where the complexity of the problem switches from FPT to $W[1]$-hard.

These observations motivate the search for positive FPT-deliverables (algorithms and kernels) for *increasingly smaller parameters.* Another justification for the search of positive results for smaller parameterizations, even at the cost of bigger functions $f(k)$ or kernel sizes, is the observed trend that once such results have been established, successive improvements are usually found that decrease the run-time and kernel size.[7]

These ideas call for a new type of *race* in parameterized complexity analysis [159]: the race for the *best* (i.e., smallest) parameter with respect to which the problem is still FPT, or still admits a polynomial kernel. Only if we know where the boundary lies between FPT and $W[1]$-hardness in the parameter landscape,

[6]This holds for all relationships exhibited in the hierarchy of Fig. 2.1, with the exception of the pair Cutwidth and Topological Bandwidth since one is *sandwiched* by the other: $\text{tbw}(G) \leq \text{cw}(G) \leq \text{tbw}(G)^2$. For concreteness, one could think of $\pi(G)$ as treewidth, and $\pi'(G)$ as the vertex cover number.

[7]There are, however, some intriguing cases where the function $f(k)$ for a smaller parameter provably *cannot* be decreased to match the dependency function of a larger parameter: compare the non-elementary lower-bound by Frick and Grohe [114] for Monadic Second-Order Logic model-checking parameterized by treewidth to the (doubly)exponential upper bounds parameterized by vertex cover and max leaf number, as given by Lampis [172].

and between exponential and polynomial-size kernels, do we fully understand what constitutes the difficulty of the problem. No parameterized analysis of a problem should therefore be complete without asking: can we extend our positive result to a smaller parameter?

The work on refined parameterizations of Vertex Cover discussed in Chapter 5 was one of the first kernelization results in this direction. The existence of a kernel with $2k$ vertices for the naturally parameterized Vertex Cover [sol] problem was known for a decade [53]. We show that even the parameterization by the dominating parameter "feedback vertex number" admits a polynomial kernel. For a practical example of the relevance of this smaller parameter, consider a path on n vertices: it has vertex cover number $k = \lfloor n/2 \rfloor$ but feedback vertex number zero. The kernelization with respect to the natural parameterization promises to shrink an instance (P_n, k) to an equivalent instance on at most $2k \approx n$ vertices; the instance is not guaranteed to shrink at all. But as the parameter feedback vertex number is zero, the kernelization with respect to that parameter promises that the instance will be shrunk to *constant* size! It now seems as if the polynomial kernelization by feedback vertex number was just the tip of the iceberg; further milestones in the race for the best Vertex Cover kernel will be discussed in Chapter 5.

Of course, the quest to extend positive news to smaller parameterizations is only one side of the story: we should consider simultaneously whether negative results can be lifted to larger parameters. When the upwards and downwards motions meet, we have succeeded in uncovering the boundary of tractability for the problem under consideration. Carrying out this program to uncover the kernelization complexity of fundamental graph problems with respect to the parameter hierarchy, forms the main topic of this thesis. In the next section we explain how this systematic study of kernelization complexity improves our understanding of preprocessing.

2.5 Explaining Preprocessing Through the Ecology Program

We close the chapter by explaining how the principles of the ecology program help us to understand the power of preprocessing. The relevance of the program for the study of effective preprocessing stems from the following hypothesis: if preprocessing is effective on a particular dataset, then the structure that is implicitly being exploited by the data reduction routines can be captured by a parameter that is small on the considered instances, and for which the resulting problem admits a small kernel. Thus we explain the effectiveness of preprocessing by proving the existence of polynomial kernels for various parameterizations.

Even when doubting the hypothesis above, the study of kernelization complexity within the parameter hierarchy has obvious merits. The classical focus of parameterized complexity theory on parameterizations by solution size is rather arbitrary, originating from the fact that it is the most "obvious" parameter, and one that may expected to be small in the popular examples of Vertex Cover and Dominating Set. But there is a host of problems such as Max Cut, Max Leaf, and Independent Set, where the optimum is typically not small.

The ecology program forces us to consider more meaningful parameterizations for these problems. By focusing on different parameterizations, we are challenged to exploit different types of structure in problem instances. Searching for small kernels in such settings therefore has the potential to lead to the discovery of new preprocessing strategies that focus on aspects of the inputs that were not relevant in the original setting. The data reduction rules developed for such scenarios may prove to be valuable when attacking real-world inputs.

Of course, the theoretical analyses presented in this thesis cannot explain success stories on particular datasets; only experiments with, and analysis of, the structure in that dataset can do so. One should note that the parameter hierarchy presented here is only the first step in the analysis of preprocessing: it is likely that the actual structure present in the instances cannot be captured by parameters as clean and abstract as the ones presented here. The work presented in this thesis paves the way for the study of more complex parameterizations.

Part II

Tools for Kernelization

Proofs of impossibility were effected by the ancients. In later mathematics, the question as to the impossibility of certain solutions plays a pre-eminent part.

—David Hilbert, 1900

3 Kernelization Lower Bounds

In this chapter we develop tools for proving lower bounds on kernel sizes, subject to certain complexity-theoretic assumptions. We review the history of kernel lower bound techniques in Section 3.1, to supply relevant background information for the material presented later on. In Section 3.2 we define polynomial-parameter transformations, which can be used to transfer kernelization lower bounds from one problem to another. The main contribution of this chapter is Section 3.3 where we introduce the technique of cross-composition and show that it yields lower bounds on the sizes of kernelization, under complexity-theoretic assumptions. We exhibit the power of cross-composition in Section 3.4 by proving lower bounds for classic problems such as CLIQUE parameterized by the vertex cover number, and FEEDBACK VERTEX SET and ODD CYCLE TRANSVERSAL parameterized by the vertex-deletion distance to a clique.

This chapter is based on the paper "Cross-Composition: A New Technique for Kernelization Lower Bounds" by Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch (28th International Symposium on Theoretical Aspects of Computer Science [29]).

3.1 A Brief History of Kernel Lower Bound Techniques

Since a parameterized problem is contained in FPT if and only if it is decidable and has a kernel (Lemma 2.1), the crudest type of kernelization lower bound is formed by a $W[1]$-hardness proof. Assuming that FPT is unequal to $W[1]$, no $W[1]$-hard problem has a kernel of size $f(k)$ for any computable function f. The situation becomes more interesting once we focus on problems inside the class FPT.

The first type of kernelization lower bounds for problems in FPT were given by Chen et al. [52]. This concerned problems whose *parameterized dual* is also fixed-parameter tractable. They showed how the duality can be exploited to establish lower bounds on the multiplicative constant in the sizes of linear kernels. For example, they proved that Vertex Cover [sol] restricted to planar graphs does not have a *parameter non-increasing kernelization*[1] with $4k/3$ vertices or less. As there are relatively few interesting problems in FPT whose duals are also tractable, the applicability of these methods is limited. The main interest at the time was therefore in the development of techniques to rule out kernels of *any* polynomial size, for problems in FPT. A prime candidate for not having a polynomial kernel was the Long Path [sol] problem.

Bodlaender et al. [24] then introduced the concept of an or-*composition* algorithm as a tool to give superpolynomial lower bounds on kernel sizes. Informally speaking — we defer formal definitions to later sections — an or-composition algorithm efficiently combines a series of inputs of a parameterized problem $\mathcal{Q}$, all sharing the same parameter k, into a single instance (x', k') of $\mathcal{Q}$ such that k' is polynomial in k, and (x', k') acts as the logical or of the inputs in the sense that (x', k') is a yes-instance if and only if (at least) one of the inputs is. They used a theorem of Fortnow and Santhanam [113] to show that if there is an or-composition for an NP-hard parameterized problem $\mathcal{Q}$, then $\mathcal{Q}$ does not admit a polynomial kernelization unless NP $\subseteq$ coNP/poly. This machinery made it possible to prove, e.g., that Long Path [sol] and the Clique problem parameterized by the treewidth of the graph do not admit polynomial kernels unless NP $\subseteq$ coNP/poly.[2] Chen et al. [57] further analyzed the complexity-theoretic consequences of Fortnow and Santhanam's work, and provided lower bounds against various types of preprocessing procedures related to kernelization. For example, they proved that sat parameterized by the number of variables does not have a parameter non-increasing polynomial kernelization unless P = NP.

It did not take long before the techniques of Bodlaender et al. were combined with the notion of a *polynomial-parameter transformation* to also prove lower bounds for problems for which no direct or-composition algorithm could be found. This idea was used implicitly by Fernau et al. [16, 105] to show that the naturally

[1] A kernelization is parameter non-increasing if it maps (x, k) to an output (x', k') with $k' \leq k$.

[2] In the remainder of this section we assume that NP $\not\subseteq$ coNP/poly when stating kernelization lower bounds.

parameterized *directed* variant of Max Leaf does not admit a polynomial kernel, and was formalized in a paper by Bodlaender et al. [43]: they showed that if the classical versions of the parameterized problems $\mathcal{Q}$ and $\mathcal{Q}'$ are NP-complete, and there is a polynomial-time transformation from $\mathcal{Q}$ to $\mathcal{Q}'$ that incurs only a polynomial blow-up in the parameter size, then if $\mathcal{Q}$ does not admit a polynomial kernel, $\mathcal{Q}'$ does not admit one either.

Polynomial-parameter transformations were used extensively by Dom et al. [83] who proved kernelization lower bounds for several important parameterized problems, such as Hitting Set and Set Cover parameterized by the size of the universe. Dell and van Melkebeek [80] were able to extend the techniques of Fortnow and Santhanam and proved, e.g., that Vertex Cover [sol] does not admit a kernel of bitsize $\mathcal{O}(k^{2-\varepsilon})$ for any $\varepsilon > 0$. Follow-up work has resulted in lower bounds on the degree of the polynomial in the kernel sizes of packing problems [79, 135], using composition-like techniques in which the output parameter is allowed to depend sublinearly on the number of inputs. The complementary witness lemma of Dell and van Melkebeek [80] shows that even *co-nondeterministic compositions* give kernel lower bounds.[3] A co-nondeterministic composition is a nondeterministic variant of a composition algorithm. It takes the same input as an or-composition, and outputs an instance whose parameter is bounded by $k^{\mathcal{O}(1)}$ on each computation path. The following co-nondeterministic form of correctness is required: if there is a yes-instance among the inputs, then all paths output a yes-instance; if all inputs are no-instances, then at least one path outputs a no-instance. Kratsch leveraged the power of co-nondeterminism to obtain a lower bound for a Ramsey-type problem [163].

The most recent proof-technical advance in the study of kernelization lower bounds is due to Drucker [89]. He proved the and-distillation conjecture, originally formulated by Bodlaender et al. [24], which concerns the natural and-variant of an or-composition algorithm. While the exact statement of the conjecture is not relevant for this overview, his result shows the following. If $\mathcal{Q}$ is a parameterized problem whose underlying classical version is NP-hard, then the combination of an and-composition and a polynomial kernel for $\mathcal{Q}$ implies NP $\subseteq$ coNP/poly and therefore a collapse of the polynomial hierarchy. Thus one may establish a conditional kernel lower bound for such $\mathcal{Q}$ by creating an and-composition.

3.2 Polynomial-Parameter Transformations

Although most kernelization lower bounds in this thesis are obtained through cross-composition, we shall occasionally use polynomial-parameter transformations to propagate a bound from one problem to another.

Definition 3.1 ([43])**.** Let $\mathcal{Q}, \mathcal{Q}' \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems. A *polynomial-parameter transformation* from $\mathcal{Q}$ to $\mathcal{Q}'$ is an algorithm that on

[3] In fact, this is already implicit in the results of Fortnow and Santhanam, as observed in unpublished work of Chen and Müller (cf. [130]).

input $(x, k) \in \Sigma^* \times \mathbb{N}$ takes time polynomial in $|x| + k$ and outputs an instance $(x', k') \in \Sigma^* \times \mathbb{N}$ such that:

- The parameter value k' is polynomially bounded in k.
- $(x', k') \in \mathcal{Q}'$ if and only if $(x, k) \in \mathcal{Q}$.

We denote the existence of such a transformation from $\mathcal{Q}$ to $\mathcal{Q}'$ by $\mathcal{Q} \leq_{ppt} \mathcal{Q}'$. The special case where k' is bounded linearly in k is called a *linear-parameter transformation.*

To formally state the requirements for obtaining kernelization lower bounds through the use of polynomial transformations, we need to associate parameterized problems with classical languages.

Definition 3.2 ([24]). The *unparameterized version* of a parameterized problem $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ is the language $\widetilde{\mathcal{Q}} := \{x\#1^k \mid (x, k) \in \mathcal{Q}\}$, where $\#$ is a new character that we add to the alphabet and 1 is an arbitrary letter in Σ. The *unparameterized instance* corresponding to $(x, k) \in \Sigma^* \times \mathbb{N}$ is the string $x\#1^k$.

If the classical problem $\widetilde{\mathcal{Q}}$ underlying a parameterized problem $\mathcal{Q}$ is NP-complete, then the polynomial-time reductions to- and from SAT that are guaranteed to exist by NP-completeness, can be used to transform an instance (x, k) of $\mathcal{Q}$ whose classical size $|x|$ is polynomial in k, to a SAT-instance $y \in \Sigma^*$ of size $k^{\mathcal{O}(1)}$, and vice versa: the encoding of the parameter in unary means that it cannot grow exponentially from a polynomial-time transformation. This observation can be used to prove the following theorem. Our formulation here is slightly simpler than the original, but is easily seen to be equivalent to the version due to Bodlaender et al. [43].

Theorem 3.1 ([43]). *Let $\mathcal{Q}$ and $\mathcal{Q}'$ be parameterized problems whose unparameterized versions $\widetilde{\mathcal{Q}}$ and $\widetilde{\mathcal{Q}'}$ are NP-complete. If $\mathcal{Q} \leq_{ppt} \mathcal{Q}'$ and $\mathcal{Q}'$ has a polynomial kernel, then $\mathcal{Q}$ also has a polynomial kernel.*

A corollary to the theorem yields a way to obtain kernelization lower bounds.

Corollary 3.1. *Let $\mathcal{Q}$ and $\mathcal{Q}'$ be parameterized problems whose unparameterized versions $\widetilde{\mathcal{Q}}$ and $\widetilde{\mathcal{Q}'}$ are NP-complete. If $\mathcal{Q} \leq_{ppt} \mathcal{Q}'$ and $\mathcal{Q}$ does not have a polynomial kernel, then $\mathcal{Q}'$ does not have a polynomial kernel either.*

The following lower bound is often a convenient starting point for polynomial-parameter transformations.

Proposition 3.1 ([80, 113]). CNF-SAT *parameterized by the number of variables does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.*

3.3 Cross-Composition

We now introduce the technique of *cross-composition* for obtaining kernelization lower bounds. It generalizes and strengthens the earlier methods of compositions [24] and polynomial-parameter transformations [43], and puts these two existing methods for obtaining kernelization lower bounds in a common framework. Whereas the existing notion of OR-composition works by composing multiple instances of a *parameterized* problem $\mathcal{Q}$ into a single instance of $\mathcal{Q}$ with a bounded parameter value, in the cross-composition framework it is sufficient to compose the OR of a series of instances of any *classical* NP-hard problem into an instance of the parameterized problem $\mathcal{Q}$ for which we want to prove a lower bound. The term *cross* in the name stems from this fact: the source- and target problem of the composition need no longer be the same. Since the input to a cross-composition algorithm is a list of *classical* rather than parameterized instances, the inputs do not have a parameter in which the parameter of the composition's output must be bounded; instead we require that the size of the output parameter is polynomially bounded in the size of the largest input instance. This makes our extension also fruitful when the source and target problems of the composition coincide, since it allows for a larger parameter value in the output instance than would be permissible in a "classic" OR/AND-composition. In addition we show that the output parameter for a composition of t instances may depend polynomially on $\log t$, which often simplifies the constructions and proofs. We also introduce the concept of a *polynomial equivalence relation* to remove the need for padding arguments, which were frequently used in compositions.

3.3.1 The Definition

In this section we define the concept of cross-composition and give all the terminology needed to apply the technique.

Definition 3.3. An equivalence relation $\mathcal{R}$ on Σ^* is called a *polynomial equivalence relation* if the following two conditions hold:

1. There is an algorithm that, given two strings $x, y \in \Sigma^*$, runs in time polynomial in $|x| + |y|$ and decides whether x and y belong to the same equivalence class.
2. For any finite set $S \subseteq \Sigma^*$ the equivalence relation $\mathcal{R}$ partitions the elements of S into a number of classes that is polynomially bounded in the length of the largest element of S.

The intended use of polynomial equivalence relations is to group inputs for a (cross-)composition such that the composition will only be applied to groups of instances that are somewhat similar, e.g., they ask for the same solution size or the considered graphs have the same numbers of vertices and edges. We point out that $\mathcal{R} = \Sigma^* \times \Sigma^*$ trivially fulfills the definition of a polynomial equivalence relation (all strings are equivalent and any set S is “partitioned” into a single class). Thus the use of polynomial equivalence relations in cross-compositions according to the following definition is optional, but often simplifies proofs.

Definition 3.4. Let $\mathcal{L} \subseteq \Sigma^*$ be a set and let $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. The set $\mathcal{L}$ *cross-composes* into $\mathcal{Q}$ if there is a polynomial equivalence relation $\mathcal{R}$ and an algorithm that, given t strings $x_1, x_2, \ldots, x_t$ belonging to the same equivalence class of $\mathcal{R}$, computes an instance $(y, k) \in \Sigma^* \times \mathbb{N}$ in time polynomial in $\sum_{i=1}^{t} |x_i|$ such that:

1. $(y, k) \in \mathcal{Q}$ if and only if there is at least one $i \in [t]$ such that $x_i \in \mathcal{L}$,
2. k is bounded by a polynomial in $\max_{i=1}^{t} |x_i| + \log t$.

3.3.2 How Cross-Compositions Yield Lower Bounds

The purpose of this section is to prove that cross-compositions yield kernelization lower bounds. To give this proof we need some concepts from earlier work [24, 80, 113]. Recall that SAT denotes the satisfiability problem for Boolean formulae.

Definition 3.5 ([113])**.** A *weak distillation* of SAT into a set $\mathcal{L} \subseteq \Sigma^*$ is an algorithm that, given t instances $x_1, \ldots, x_t$ of SAT, computes a string $y \in \Sigma^*$ in time polynomial in $\sum_{i=1}^{t} |x_i|$ such that:

1. $y \in \mathcal{L}$ if and only if there is at least one $i \in [t]$ such that $x_i \in$ SAT,
2. $|y|$ is bounded by a polynomial in $\max_{i=1}^{t} |x_i|$.

Fortnow and Santhanam gave the following evidence against the existence of weak distillations for SAT.

Theorem 3.2 (Theorem 1.2 [113])**.** *If there is a weak distillation of* SAT *into any set $\mathcal{L} \subseteq \Sigma^*$, then NP $\subseteq$ coNP/poly.*

The following notation will be useful in the proof.

Definition 3.6 ([80])**.** The OR *of a language* $\mathcal{L} \subseteq \Sigma^*$ is the set OR($\mathcal{L}$) consisting of all tuples $(x_1, \ldots, x_t)$ for which there is an index $i \in [t]$ with $x_i \in \mathcal{L}$.

We are now ready to state and prove the main theorem of this section. The precondition to the proof requires the NP-hardness of a set $\mathcal{L}$ under Karp reductions — as opposed, for example, to hardness under Turing reductions [10, 118]. This guarantees the existence of a polynomial-time many-one reduction from SAT to $\mathcal{L}$, which will be used in the proof.

Theorem 3.3. *Let $\mathcal{L} \subseteq \Sigma^*$ be a set that is NP-hard under Karp reductions. If $\mathcal{L}$ cross-composes into the parameterized problem $\mathcal{Q}$, and $\mathcal{Q}$ has a polynomial kernel, then there is a weak distillation of* SAT *into* OR($\widetilde{\mathcal{Q}}$) *and NP $\subseteq$ coNP/poly.*

Proof. The proof is by construction and generalizes the strategy of Bodlaender et al. [24]. Assuming the conditions in the statement of the theorem hold, we show how to build an algorithm that distills SAT into OR($\widetilde{\mathcal{Q}}$). By the definition of cross-composition there is a polynomial equivalence relation $\mathcal{R}$ and an algorithm C that composes $\mathcal{L}$-instances belonging to the same class of $\mathcal{R}$ into a $\mathcal{Q}$-instance.

The input to the distillation algorithm consists of a sequence $(x_1, \ldots, x_t)$ of instances of SAT, which we may assume are elements of Σ^*. Define $m := \max_{j=1}^{t} |x_j|$. If $t > (|\Sigma|+1)^m$ then there must be duplicate inputs among $x_1, \ldots, x_t$, since the number of distinct inputs of length $m' \leq m$ is $|\Sigma|^{m'}$. By discarding duplicates we may therefore assume that $t \leq (|\Sigma|+1)^m$, i.e., $\log t \in \mathcal{O}(m)$. By the assumption that $\mathcal{L}$ is NP-hard under Karp reductions, there is a polynomial-time many-one reduction from SAT to $\mathcal{L}$. We use this reduction to transform each SAT instance x_i for $i \in [t]$ into an equivalent $\mathcal{L}$-instance y_i. Since the transformation takes polynomial time, it cannot increase the size of an instance by more than a polynomial factor and therefore $|y_i|$ is polynomial in m for all i.

The algorithm now pairwise compares instances using the polynomial-time equivalence test of $\mathcal{R}$ (whose existence is guaranteed by Definition 3.3) to partition the $\mathcal{L}$-instances $(y_1, \ldots, y_t)$ into partite sets $Y_1, \ldots, Y_r$ such that all instances from the same partite set are equivalent under $\mathcal{R}$. The properties of a polynomial equivalence relation guarantee that r is polynomial in m and that this partitioning step takes polynomial time in the total input size.

We now use the cross-composition algorithm C on each of the partite sets $Y_1, \ldots, Y_r$, which is possible since all instances from the same set are equivalent under $\mathcal{R}$. Let (z_i, k_i) be the result of applying C to a sequence containing the contents of the set Y_i, for $i \in [r]$. From the definition of cross-composition and using $\log t \in \mathcal{O}(m)$ it follows that each k_i is polynomial in m, and that the computation of these parameterized instances takes polynomial time in the total input size. From Definition 3.4 it follows that (z_i, k_i) is a YES-instance of $\mathcal{Q}$ if and only if one of the instances in Y_i is a YES-instance of $\mathcal{L}$, which in turn happens if and only if one of the inputs x_i is a YES-instance of SAT.

Let K be a polynomial kernelization algorithm for $\mathcal{Q}$, whose existence we assumed in the statement of the theorem. We apply K to the instance (z_i, k_i) to obtain an equivalent instance (z'_i, k'_i) of $\mathcal{Q}$ for each $i \in [r]$. Since K is a polynomial kernelization we know that these transformations can be carried out in polynomial time and that $|z'_i|, k'_i \leq k_i^{\mathcal{O}(1)}$. Since k_i is polynomial in m it follows that $|z'_i|$ and k'_i are also polynomial in m for $i \in [r]$.

As the next step we convert each parameterized instance (z'_i, k'_i) to the unparameterized variant $\widetilde{z_i} := z'_i\#1^{k'_i}$. Since the values of the parameters are polynomial in m this transformation takes polynomial time, and afterwards we find that $|\widetilde{z_i}|$ is polynomial in m for each $i \in [r]$.

The last stage of the algorithm simply combines all unparameterized variants into one tuple $x^* := (\widetilde{z_1}, \widetilde{z_2}, \ldots, \widetilde{z_r})$. Since the size of each component is polynomial in m, and since the number of components r is polynomial in m, we have that

$|x^*|$ is polynomial in m. The tuple x^* forms an instance of $\textsc{or}(\widetilde{\mathcal{Q}})$, and by the definition of $\textsc{or}(\widetilde{\mathcal{Q}})$ we know that $x^* \in \textsc{or}(\widetilde{\mathcal{Q}})$ if and only if some element of the tuple is contained in $\widetilde{\mathcal{Q}}$. By tracing back the series of equivalences we therefore find that $x^* \in \textsc{or}(\widetilde{\mathcal{Q}})$ if and only if some input x_i is a $\textsc{yes}$-instance of $\textsc{sat}$. Since we can construct x^* in polynomial time and $|x^*|$ is polynomial in m, this constitutes a weak distillation of $\textsc{sat}$ into $\textsc{or}(\widetilde{\mathcal{Q}})$. By Theorem 3.2 this implies NP $\subseteq$ coNP/poly and proves the theorem. □

We state the following simple corollary as our main tool for giving kernelization lower bounds.

Corollary 3.2. *If the set $\mathcal{L} \subseteq \Sigma^*$ is NP-hard under Karp reductions and $\mathcal{L}$ cross-composes into the parameterized problem $\mathcal{Q}$, then there is no polynomial kernel for $\mathcal{Q}$ unless NP $\subseteq$ coNP/poly.*

A simple extension of Theorem 3.3 shows that cross-compositions also exclude the possibility of efficiently transforming instances of a parameterized problem into small and equivalent instances of a *different* parameterized problem; this notion is sometimes referred to as *compression* or *bikernelization* [6, 126, 166, 208]. Assuming NP $\not\subseteq$ coNP/poly, if an NP-hard set cross-composes into a parameterized problem $\mathcal{Q}$, then there is no polynomial-time algorithm that maps an instance (x, k) of $\mathcal{Q}$ to an equivalent instance (x', k') of *any* parameterized problem $\mathcal{Q}'$ with $|x'|, k' \leq k^{\mathcal{O}(1)}$.

Since the first publication of our cross-composition framework, it has found numerous applications [30, 35, 69–72, 74, 123, 145, 148]. Although all applications of the cross-composition technique presented here are for problems under structural parameterizations, the framework can also be used to obtain lower bounds for natural parameterizations. For example, Cygan et al. [71] employed cross-composition to obtain kernelization lower bounds for Edge Clique Cover, settling an open problem in kernelization, and also for various graph cut problems parameterized by the size of the cutset. The mentioned follow-up work together with other results on lower bounds for kernelization [79, 80, 135, 163] suggests several extensions, which are not included here. They are reviewed in the journal paper on cross-composition [33]. For example, one can define $\textsc{and}$-cross-composition as used by Cygan et al. [71], and there is a modification of $\textsc{or}$-cross-composition that allows proofs of polynomial lower bounds for kernelization in the style of Dell and van Melkebeek [80]. Regarding co-nondeterministic variants of $\textsc{or}$-cross-compositions we refer the reader to the exposition by Kratsch [164].

3.4 Applications of Cross-Composition

We exhibit the power of cross-composition by giving kernelization lower bounds for *structural* parameterizations of several important graph problems. Since many combinatorial problems are computationally easy on graphs of bounded treewidth [37], and since the treewidth of a graph is bounded by the vertex cover number, many problems are fixed-parameter tractable when parameterized by the cardinality of a given vertex cover. We show that this tractability does not always extend to polynomial kernelizability: CLIQUE does not admit a polynomial kernel under this parameterization, unless NP $\subseteq$ coNP/poly. It was already known [24] that CLIQUE does not admit a polynomial kernel parameterized by the treewidth of the graph, unless NP $\subseteq$ coNP/poly; since the vertex cover number is at least as large as the treewidth, the result we prove is stronger. By taking the edge-complement of the input graph, CLIQUE parameterized by the size of a given vertex cover is FPT-equivalent to VERTEX COVER parameterized by the size of a given vertex set whose removal leaves a clique (a *modulator* to a clique). Hence we also establish a kernel lower bound for the latter problem, which marks an interesting boundary in the search for the smallest parameterization of VERTEX COVER admitting a polynomial kernel [73, 145, 167]. We provide a polynomial-parameter transformation to extend the lower bound to FEEDBACK VERTEX SET and ODD CYCLE TRANSVERSAL parameterized by the *vertex-deletion distance* to a clique, contrasting the known polynomial kernelizability of these problems with respect to the natural parameterization by solution size [167, 207].

3.4.1 Clique Parameterized by Vertex Cover

In this section we prove a kernel lower bound for a strong structural parameterization of CLIQUE, and we consider some of its consequences. An instance of the NP-complete CLIQUE problem [117, GT19] is a tuple (G, ℓ) and asks whether the graph G contains a clique on ℓ vertices. This classical problem is the source language for the cross-composition that establishes our first kernelization lower bound. The parameterization for which we derive a lower bound is formally defined as follows.

CLIQUE [VC]
Input: A graph G, a vertex cover $X \subseteq V(G)$, and an integer ℓ.
Parameter: The size $k := |X|$ of the vertex cover.
Question: Does G contain a complete subgraph on ℓ vertices?

The suffix [VC] shows that we consider the parameterization by the size of a given vertex cover. As discussed in Section 2.2.1 we supply a vertex cover in the input of the problem to ensure that well-formed instances can be recognized efficiently.

Let us briefly consider the complexity of the chosen parameterization of the CLIQUE problem. If the graph G has a vertex cover X and contains a clique $C \subseteq V(G)$, then $|C \setminus X| \leq 1$: as the definition of a vertex cover ensures that there are no edges between vertices in $V(G) \setminus X$, any clique contains at most one vertex from $V(G) \setminus X$. The problem defined above can therefore be solved in $\mathcal{O}(2^{|X|}n^2)$ time by enumerating all subsets $C \subseteq X$, checking whether they induce a clique $G[C]$, and whether there is a vertex in $V(G) \setminus X$ adjacent to all members of C that may be added to the clique. We detect a maximum clique when its intersection with X is used as the subset C.

This same insight also yields a so-called *cheating kernel* [16, 105] or *Turing kernel* [134]. For an input instance (G, X, ℓ, k) of Clique [vc], create a list of instances that contains for each $v \in V(G) \setminus X$ the instance $(G[X \cup \{v\}], X, \ell, k)$, in addition to $(G[X], X, \ell, k)$. The preceding discussion shows that G has a clique of size ℓ if and only if one of the instances in the list has such a clique. As each instance in the list contains at most $|X| + 1 = k + 1$ vertices, this is an interesting example of a rare phenomenon: although we cannot, in polynomial time, reduce the size of an instance of Clique [vc] to polynomial in the parameter, we can create $\mathcal{O}(n^{\mathcal{O}(1)})$ instances of size polynomial in the parameter, such that the input is yes if and only if one of the outputs is.

Having provided all the necessary background for the following lower bound, we give its proof by cross-composition.

Theorem 3.4. Clique [vc] *does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.*

Proof. We prove the theorem by showing that Clique cross-composes into Clique [vc]; by Corollary 3.2 this is sufficient to establish the claim. We define a polynomial equivalence relation $\mathcal{R}$ such that all strings that do not encode a valid instance of Clique, or that have a target clique size ℓ exceeding the total number of vertices, are equivalent. Similarly we make all the trivial instances asking for a clique of size one or less equivalent. Of the remaining instances any two (well-formed) instances (G_1, ℓ_1) and (G_2, ℓ_2) are equivalent if and only if they satisfy $|V(G_1)| = |V(G_2)|$ and $\ell_1 = \ell_2$. Observe that this $\mathcal{R}$ partitions a set of well-formed instances on at most n vertices each, into $\mathcal{O}(n^2)$ equivalence classes: there are at most n choices for the vertex count of the graph, and at most n choices for the target clique size.

To formally establish that $\mathcal{R}$ is a polynomial equivalence relation, we need to show that it satisfies the conditions of Definition 3.3. This requires us to choose a particular encoding of Clique as a language over a finite alphabet. Similarly as in NP-completeness proofs, the choice of encoding does not really matter, as long as it is reasonable. For the concreteness of this lower bound as a first application of the technique, we will be specific. We choose the encoding of Clique instances into a three-letter alphabet $\{0, 1, \#\}$ in which an instance is encoded by giving the adjacency matrix of the graph row by row, followed by a hash character, ending with the binary encoding of ℓ.

It is easy to see that using this encoding, equivalence of two instances under the given definition can be tested in polynomial time. As a set of well-formed instances on at most n vertices each is partitioned into $\mathcal{O}(n^2)$ equivalence classes, and an instance on n vertices is encoded by a string of length $\Omega(n^2)$, it follows that the second condition of Definition 3.3 is also satisfied and therefore $\mathcal{R}$ is a polynomial equivalence relationship.

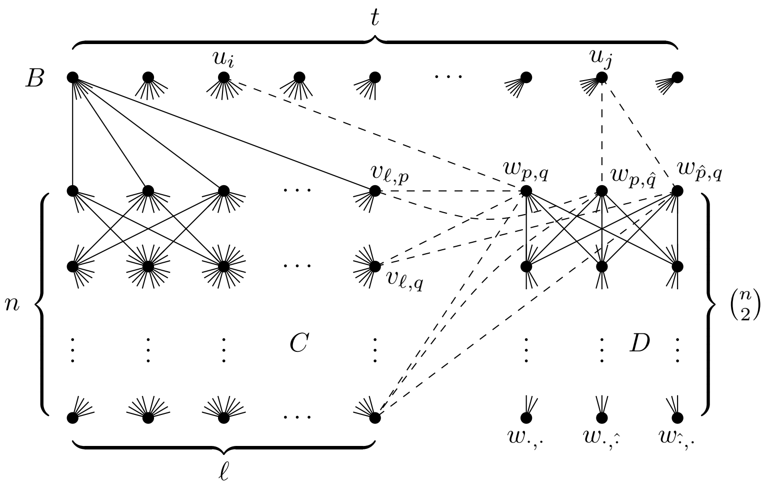


Figure 3.1: A sketch of the construction used in the proof of Theorem 3.4. The dashed edges show in an exemplary way how vertices $w_{p,q}$, $w_{p,\hat{q}}$, and $w_{\hat{p},q}$ are connected to vertices of B and C, e.g., $\{p, q\}$ is an edge of G_i but not of G_j.

We now give a cross-composition algorithm that composes t input instances x_1, $\ldots, x_t$ that are equivalent under $\mathcal{R}$ into a single instance of Clique [vc]. If the input instances are malformed or the size of the clique that is asked for exceeds the number of vertices in the graph, then we may output a single constant-size NO-instance. Similarly, if all instances ask for a clique of size at most one then we output a constant-size YES-instance. Hence in the remainder we may assume that all inputs are well-formed and encode structures $(G_1, \ell), \ldots, (G_t, \ell)$ such that $|V(G_i)| = n$ for all $i \in [t]$ and all instances agree on the value of $2 \leq \ell \leq n$. We construct a single instance of Clique [vc], which consists of a graph G' with a vertex cover $X' \subseteq V(G')$ of size k', and an integer ℓ'.

Let the vertices in each graph G_i be numbered arbitrarily from 1 to n. We construct the graph G' in three steps, as follows. See also Fig. 3.1.

1. Create ℓn vertices $v_{i,j}$ with $i \in [\ell]$ and $j \in [n]$. Connect two vertices $v_{i,j}$ and $v_{i',j'}$ if $i \neq i'$ and $j \neq j'$. Let C denote the set of these vertices. It is crucial that any clique in G' can only contain one vertex $v_{i,\cdot}$ or $v_{\cdot,j}$ for each choice of $i \in [\ell]$ respectively $j \in [n]$. Thus any clique contains at most ℓ vertices from C.
2. For each pair $1 \leq p < q \leq n$ of distinct vertices from $[n]$ (i.e., vertices of graphs G_i), create three vertices: $w_{p,q}$, $w_{p,\hat{q}}$, and $w_{\hat{p},q}$ and make them adjacent to C as follows:
   (a) $w_{p,q}$ is adjacent to all vertices from C,
   (b) $w_{p,\hat{q}}$ is adjacent to all vertices from C except for $v_{\cdot,j}$ with $j = q$, and

(c) $w_{\hat{p},q}$ is adjacent to all vertices from C except for $v_{\cdot,j}$ with $j = p$.

Furthermore we add all edges between vertices $w_{\cdot,\cdot}$ that correspond to distinct pairs from $[n]$. Let D denote these $3\binom{n}{2}$ vertices. Any clique can contain at most one $w_{\cdot,\cdot}$ vertex for each pair from $[n]$.

3. For each input instance x_i with graph G_i, make a new vertex u_i and connect it to all vertices in C. The adjacency to D is as follows:

   (a) Make u_i adjacent to $w_{p,q}$ if $\{p, q\}$ is an edge in G_i.
   (b) Otherwise make u_i adjacent to $w_{p,\hat{q}}$ and $w_{\hat{p},q}$.

   Let B denote this set of t vertices.

We define $\ell' := \ell + 1 + \binom{n}{2}$. Furthermore, we let $X' := C \cup D$, which is easily verified to be a vertex cover for G' of size $k' := |X'| = \ell n + 3\binom{n}{2}$. The value k' is the parameter to the problem; it is polynomial in n and hence in the size of the largest input instance. The cross-composition encodes (G', X', ℓ') as a string x' and outputs the parameterized instance (x', k'). It is easy to see that our construction of (x', k') can be performed in polynomial time.

Claim. *Instance (x', k') is* YES *for* CLIQUE [VC] *if and only if at least one of the instances x_i is* YES *for* CLIQUE.

Proof. ($\Leftarrow$) First we will assume that some $x_{i^*} = (G_{i^*}, \ell)$ is YES for CLIQUE, i.e., that G_{i^*} contains a clique on at least ℓ vertices. Let $S \subseteq [n]$ denote a clique of size exactly ℓ in G_{i^*}. We will construct a set S' of size $\ell' = \ell + 1 + \binom{n}{2}$ and show that it is a clique in G':

1. We add the vertex u_{i^*} to S'.
2. Let $S = \{p_1, \ldots, p_\ell\} \subseteq [n]$. For each p_j in S we add the vertex v_{j,p_j} to S'. By Step 1 all these vertices are pairwise adjacent, and by Step 3 they are adjacent to u_{i^*}.
3. For each pair p, q with $1 \leq p < q \leq n$ there are two cases:

   (a) If $\{p, q\}$ is an edge of G_{i^*} then the vertex u_{i^*} is adjacent to $w_{p,q}$ in G' (by Step 3) and $w_{p,q}$ is adjacent to all vertices of C (by Step 2). We add $w_{p,q}$ to S'.
   (b) If $\{p, q\}$ is not an edge of G_{i^*} then the vertex u_{i^*} is adjacent to both $w_{p,\hat{q}}$ and $w_{\hat{p},q}$. Since the clique S cannot contain both p and q when $\{p, q\}$ is a non-edge, we can add $w_{p,\hat{q}}$ or $w_{\hat{p},q}$ to S' while preserving the fact that $G'[S']$ is a clique. Recall that, e.g., $w_{p,\hat{q}}$ is adjacent to all vertices of C except those corresponding to q.

   In both cases we add one $w_{\cdot,\cdot}$-vertex to S', each corresponding to a different pair p, q; all these vertices are pairwise adjacent by Step 2.

We have identified the clique S' in G' of size $\ell' = \ell + 1 + \binom{n}{2}$, proving that (x', k') is a YES-instance.

($\Rightarrow$) Now assume that (x', k') is a YES-instance and let S' be a clique of size $\ell + 1 + \binom{n}{2}$ in G'. Since S' contains at most ℓ vertices from C (i.e., one $v_{i,\cdot}$ for each $i \in [\ell]$) and at most $\binom{n}{2}$ vertices from D, it must contain at least one vertex from B, say $u_{i^*} \in B$. Since B is an independent set, the clique S' must contain exactly ℓ vertices from C and exactly $\binom{n}{2}$ vertices from D. Let $S = \{j \in [n] \mid v_{i,j} \in S' \text{ for some } i \in [\ell]\}$. The set S has size ℓ since S' contains at most one vertex $v_{\cdot,j}$ for each $j \in [n]$. We will now argue that S is a clique in G_{i^*}. Let $p, q \in S$. The clique S' must contain a $w_{\cdot,\cdot}$-vertex corresponding to $\{p, q\}$ and it must contain vertices $v_{i,p}$ and $v_{i',q}$ for some $i, i' \in [\ell]$. Therefore it must contain $w_{p,q}$ since $w_{p,\hat{q}}$ has no edges to vertices $v_{\cdot,q}$ and $w_{\hat{p},q}$ has no edges to $v_{\cdot,p}$ by Step 2. Thus $u_{i^*} \in S'$ must be adjacent to $w_{p,q}$, which implies that G_{i^*} contains the edge $\{p, q\}$. Thus S is a clique in G_{i^*}. $\diamondsuit$

We proved that the instance (x', k') can be constructed in polynomial time and that it acts as the OR of the input instances. As the parameter value k' is bounded by a polynomial in the size of the largest input instance, our transformation is a valid cross-composition. This proves the theorem. $\square$

In the remainder of this work we will not be as explicit in giving the definition of the equivalence relationship $\mathcal{R}$ as we were in the proof of Theorem 3.4: we shall simply state the desired forms of the inputs when it is easy to see that such a form can be achieved by a suitable choice of $\mathcal{R}$ and encoding of the source problem. We restrict ourselves to giving cross-compositions for well-formed input instances, since we can simply output a constant-size NO-instance if the input is a sequence of malformed instances. The previous theorem yields an interesting corollary for the following structural parameterization of VERTEX COVER.

VERTEX COVER [CLIQUE MOD]
Input: A graph G, a modulator $X \subseteq V(G)$ such that $G - X$ is a clique, and an integer ℓ.
Parameter: The size $k := |X|$ of the modulator.
Question: Does G have a vertex cover of size at most ℓ?

As the vertex cover number of a clique K_t is $t - 1$, we can interpret this as a parameterization by the distance from trivially solvable instances as discussed by Niedermeier [189]. Our results imply that even though the parameter will often be very large, the problem is unlikely to admit a polynomial kernel.

Corollary 3.3. *If $\mathcal{F}$ is a class of graphs containing all cliques, then* VERTEX COVER *and* INDEPENDENT SET *parameterized by the minimum number of vertex deletions to obtain a graph in $\mathcal{F}$ do not admit polynomial kernels unless NP $\subseteq$ coNP/poly. In particular,* VERTEX COVER [CLIQUE MOD] *and* INDEPENDENT SET [CLIQUE MOD] *do not admit polynomial kernels unless NP $\subseteq$ coNP/poly.*

Proof. Consider an instance (G, X, ℓ, k) of CLIQUE [VC]. Since a clique in G is an independent set in $\overline{G}$, the CLIQUE instance is equivalent to asking whether the graph $\overline{G}$ has an independent set of size at least ℓ. Because X is a vertex cover for G we know that $G - X$ is an independent set, and therefore $\overline{G} - X$ is a clique. Hence the instance (G, X, ℓ, k) of Clique [vc] is equivalent to an instance $(\overline{G}, X, \ell, k)$ of Independent Set [clique mod]. Since $\overline{G}$ has an independent set of size ℓ if and only if it has a vertex cover of size $|V(G)| - \ell$ it follows that these two instances are also equivalent to the instance $(\overline{G}, X, |V(G)| - \ell, k)$ of Vertex Cover [clique mod]. Hence the described operations serve as polynomial-parameter transformations from Clique [vc] into the mentioned problems. This proves the claims in the second sentence of the corollary statement by Theorem 3.4 and Corollary 3.1, as it is easy to see that the corresponding unparameterized versions are NP-complete.

For the general statement, observe that for any $\mathcal{F}$ containing all cliques, the vertex-deletion distance to $\mathcal{F}$ does not exceed the distance to a clique. The identity mapping is therefore a trivial polynomial-parameter transformation from Vertex Cover [clique mod] to Vertex Cover parameterized by vertex-deletion distance to $\mathcal{F}$. □

3.4.2 Feedback Vertex Set and Odd Cycle Transversal

The problems Feedback Vertex Set (abbreviated fvs) and Odd Cycle Transversal (abbreviated oct) ask whether a given graph contains a small set of vertices that intersects all cycles, respectively all odd cycles. The study of their natural parameterization by solution size has led to important advances in the design of FPT algorithms and kernelizations. Both problems are fixed-parameter tractable [196, 207] and admit (randomized) polynomial kernels [167, 207]. We consider the kernelization complexity of these problems from a different angle. We study the nonstandard parameter "deletion distance to a clique" and use a polynomial-parameter transformation to transfer a lower bound result from the similarly parameterized vertex cover problem (Corollary 3.3). Thus the two cycle transversal problems do not admit polynomial kernels for this parameter, unless NP $\subseteq$ coNP/poly. We give kernelization lower bounds for fvs and oct under the following structural parameterization.

Feedback Vertex Set [clique mod]
Input: A graph G, a modulator $X \subseteq V(G)$ such that $G - X$ is a clique, and an integer ℓ.
Parameter: The size $k := |X|$ of the modulator.
Question: Is there a set $S \subseteq V(G)$ of at most ℓ vertices that intersects all cycles?

The problem Odd Cycle Transversal [clique mod] is defined analogously, by asking for a set that intersects all *odd* cycles. It is easy to give FPT algorithms for these two parameterizations. If G is a graph in which the removal of $X \subseteq V(G)$ leaves a clique, then an fvs or oct in G avoids at most two vertices of $V(G) \setminus X$, as otherwise $V(G) \setminus X$ contains a triangle not intersected by the transversal. Hence there are $\mathcal{O}(n^2)$ ways in which a valid transversal may intersect $V(G) \setminus X$. For each valid intersection with $V(G) \setminus X$ we may simply try all subsets of X to see which gives the smallest transversal that is valid for the entire graph, resulting in an algorithm of runtime $\mathcal{O}(2^{|X|} n^{\mathcal{O}(1)})$.

Despite the simplicity of the algorithms, the problems are unlikely to admit polynomial kernels. The following theorem strengthens the results in the extended abstract of our work on cross-composition [29], where we proved kernel lower bounds for FVS under the smaller parameterizations "vertex-deletion distance to a cluster graph" and "vertex-deletion distance to a cocluster graph" [29]. The theorem also subsumes results from an extended abstract by the author and Stefan Kratsch, where a kernel lower bound for OCT was given for parameterizations by (co-)cluster deletion distance [148].

Theorem 3.5. Feedback Vertex Set [clique mod] *and* Odd Cycle Transversal [clique mod] *do not admit polynomial kernels, unless NP $\subseteq$ coNP/poly.*

Proof. By Corollary 3.1 and Corollary 3.3 it is sufficient to prove that there are polynomial-parameter transformations from Vertex Cover [clique mod] to Feedback Vertex Set [clique mod] and Odd Cycle Transversal [clique mod], as it is easy to see that the unparameterized versions are NP-complete. We give a single construction that works for both cases.

Let (G, X, ℓ, k) be an instance of Vertex Cover [clique mod]. Recall that an edge clique cover of a graph is a collection of vertex sets, each inducing a clique, such that for each edge there is a clique in the cover containing both its endpoints. It is essential to our transformation that G has a small edge clique cover.

Claim. *The family of vertex subsets $\mathcal{G}$ defined by*

$$\mathcal{G} := E(G[X]) \cup \{V(G) \setminus X\} \cup \big\{\{v\} \cup (N_G(v) \setminus X) \mid v \in X\big\},$$

can be computed in polynomial time, and forms an edge clique cover of G consisting of at most $\binom{X}{2} + 1 + |X|$ cliques.

Proof. The endpoints of an edge trivially form a two-vertex clique, the problem definition ensures that $G - X$ is a clique, and each vertex together with its neighborhood in the clique $G - X$ also forms a clique: hence each subset induces a clique in G. Since each edge of G is either contained in $G[X]$, contained in $G - X$, or connects a vertex in X to a vertex outside X, the family of cliques covers all edges. It is easy to see that $|\mathcal{G}| \leq \binom{X}{2} + 1 + |X|$, and hence we can cover the edges in the input instance by a number of cliques that is polynomial in the parameter. It is trivial to compute $\mathcal{G}$ in polynomial time. $\diamondsuit$

We use the edge clique cover $\mathcal{G}$ to construct an output instance as follows. Initialize G' as a copy of G. For each clique $C \in \mathcal{G}$, add a vertex v_C to G' with $N_{G'}(v_C) := C$. Let X' be the union of X and all the vertices v_C added in this way. We find that $|X'| = |X| + |\mathcal{G}|$ and therefore $|X'|$ is bounded polynomially in the input parameter $|X|$; let $k' := |X'|$. As $G' - X' = G - X$, removal of the set X' from G' results in a clique. We interpret (G', X', ℓ, k') as an instance of the FVS and OCT problems parameterized by a clique modulator.

Claim. *The following statements are equivalent:*

1. *(G, X, ℓ, k) is a* YES*-instance of* Vertex Cover [clique mod]*.*
2. *(G', X', ℓ, k') is a* YES*-instance of* Feedback Vertex Set [clique mod]*.*
3. *(G', X', ℓ, k') is a* YES*-instance of* Odd Cycle Transversal [clique mod]*.*

Proof. (1⇒2) Suppose that (G, X, ℓ, k) is a YES-instance of Vertex Cover [clique mod], and let $S \subseteq V(G)$ be a vertex cover of G of size at most ℓ. By the definition of a vertex cover, $G - S$ is an independent set. Since every subset $C \in \mathcal{G}$ is a clique in G', it follows that $|C \setminus S| \leq 1$ for all $C \in \mathcal{G}$. Observe that we can obtain $G' - S$ from $G - S$ by introducing the vertices v_C for all $C \in \mathcal{G}$. Starting from the graph $G - S$ without edges, and using the fact that $|C \setminus S| \leq 1$ for all $C \in G'$ it is easy to see that adding the vertices v_C to obtain $G' - S$ comes down to repeatedly adding vertices of degree at most one to an acyclic graph. As such a process does not create cycles, $G' - S$ is a forest. Therefore S is an FVS for G' of size at most ℓ, which proves that (G', X', ℓ, k') a YES-instance of Feedback Vertex Set [clique mod].

(2⇒3) As any feedback vertex set is an odd cycle transversal, this implication is trivial.

(3⇒1) Suppose that there is a set $S' \subseteq V(G')$ of size at most ℓ whose removal from G' leaves a graph without odd cycles; if (G', X', ℓ, k') is a YES-instance of Odd Cycle Transversal [clique mod] then such a set exists.

We first show that without loss of generality, we may assume $v_C \notin S'$ for all $C \in \mathcal{G}$. To this end, assume that there is some $C^* \in \mathcal{G}$ such that $v_{C^*} \in S'$. If $C^* \setminus S' = \emptyset$, then $S' \setminus \{v_C^*\}$ is also an OCT for G' as all neighbors of v_{C^*} are contained in the deletion set. If, on the other hand, there exists some $u \in C^* \setminus S'$, then we claim $S^* := (S' \setminus \{v_{C^*}\}) \cup \{u\}$ is also an OCT in G'. Indeed, by construction of G' we have $N_{G'}[v_{C^*}] \subseteq N_{G'}[u]$ and therefore we can substitute u for v_{C^*} in any odd cycle in $G' - S^*$ to obtain an odd cycle in $G' - S'$. As the latter graph does not contain odd cycles, it follows that S^* is an OCT in G'. Since the replacement process can be performed independently for each $C \in \mathcal{G}$ we conclude that if (G', X', ℓ, k') is a YES-instance, then G' has an OCT S' of size at most ℓ that does not contain any vertex v_C for $C \in \mathcal{G}$.

Now observe that such a set S' forms a vertex cover of graph G: if $G - S$ contains an edge $\{x, y\}$, then the edge clique cover $\mathcal{G}$ contains a clique $C_{x,y} \supseteq \{x, y\}$. As $v_{C_{x,y}} \notin S'$ by construction, this implies $G' - S'$ contains a triangle on x, y and $v_{C_{x,y}}$; a contradiction. ♢

The claim shows that our construction is a valid polynomial-parameter transformation from Vertex Cover [clique mod] to Feedback Vertex Set [clique

MOD], and simultaneously from VERTEX COVER [CLIQUE MOD] to ODD CYCLE TRANSVERSAL [CLIQUE MOD]. Since it can be carried out in polynomial time, and the resulting parameter $|X'|$ is bounded polynomially in the input parameter $|X|$, the theorem follows. $\square$

3.5 Concluding Remarks

We have introduced the cross-composition framework and gave an application by providing a kernel lower bound for CLIQUE [VC]. Through a polynomial-parameter transformation, this lower bound was propagated to FVS and OCT parameterized by the vertex-deletion distance to a clique. Cross-composition will be indispensable for many other results in this thesis.

Through its many applications [30, 35, 69–72, 74, 123, 145, 148], our framework has already proven to be a fruitful tool in the study of kernelization complexity. The key to many cross-compositions is choosing a convenient NP-hard source problem. The importance of starting from a convenient source problem when proving kernel lower bounds can also be observed in the recent work of Dell and Marx [79] and Cygan et al. [71].

Two recent papers [163, 165] extend the cross-composition framework by the use of co-nondeterministic (cross-)compositions. We hope to see many applications of cross-composition in the future.

Algorithmic meta-theorems yield a better understanding of the scope of general algorithmic techniques and, in some sense, the limits of tractability.

—Martin Grohe, 2007

4 Meta-Theorems for Parameterizations by Vertex Cover

In this chapter we develop three general theorems that can be used to establish the existence of polynomial kernels for graph problems parameterized by the size of a vertex cover. This serves two purposes. From an applied point of view the theorems are useful because they show that a simple preprocessing rule gives provable size reductions for various graph problems. The theorems are appealing from a theoretical point of view because they unify many existing results. Thus the theorems also serve as a common explanation for the existence of polynomial kernels that were discovered in the past. The preconditions to the theorems tell us what kind of problem structure is sufficient to be able to perform efficient and effective preprocessing.

The chapter is organized as follows. We start out by giving some background on parameterizations by vertex cover in Section 4.1. In Section 4.2 we prove a proposition about bounded-size minor models in graphs that have a small vertex cover, which will be useful when applying the general kernelization theorems. The most important precondition to our general theorems is expressed by *characterizing a graph class by few adjacencies*. We introduce this notion in Section 4.3, explore its properties, and define a simple reduction rule whose correctness is based on the notion. In Section 4.4 we prove the first general theorem, aimed at vertex-deletion problems, and show its applicability to a variety of problems. Section 4.5 contains

This chapter is based on the paper "Preprocessing Subgraph and Minor Problems: When Does a Small Vertex Cover Help?" by Fedor V. Fomin, Bart M. P. Jansen, and Michał Pilipczuk (7th International Symposium on Parameterized and Exact Computation [109]).

the second general theorem, which can be applied to largest induced subgraph problems. Vertex-partitioning problems such as q-Coloring are captured by our third general theorem in Section 4.6.

4.1 Capturing Polynomial Kernelizability for Parameterizations by Vertex Cover

The parameter ecology program outlined in Section 2.2 calls for an investigation of the complexity of computational problems under various parameterizations. When it comes to graph problems, structural measures of graph complexity lead to important classes of parameterizations. In parameterized graph algorithms, one of the most important and relevant *complexity measures* of a graph is its treewidth. The algorithmic properties of problems parameterized by treewidth are, by now, well-understood. However, from the perspective of kernelization, this complexity measure is too general to obtain positive results: it is known for a multitude of graph problems such as Vertex Cover, Dominating Set, and 3-Coloring, that there are no polynomial kernels parameterized by the treewidth of the input graphs unless NP $\subseteq$ coNP/poly [24]. To find polynomial kernels for graph problems under structural parameterizations, we should therefore "move up" in the parameter hierarchy of Fig. 2.1. One of the largest graph complexity measures, and therefore one that should be understood completely before considering smaller parameterizations, is the minimum size of a vertex cover of the graph.

As a result, kernelization of graph problems parameterized by the *vertex cover number* was studied intensively [29, 35, 72, 73, 83, 147]. For example, it has been shown that graph problems such as Treewidth [35], Treewidth-η Transversal [73], and 3-Coloring [147], admit polynomial kernels parameterized by the size of a given vertex cover. On the other hand, under the assumption that NP $\not\subseteq$ coNP/poly it is possible to show that a number of problems including Dominating Set [83], Clique [29], Chromatic Number [29], Cutwidth [72], and Weighted Vertex Cover [146], do not admit polynomial kernels for this parameter. While different kernelization algorithms for various problems parameterized by vertex cover are known, we lack a general characterization of such problems. The main motivation of our work in this chapter is the quest for meta-theorems on kernelization algorithms for problems parameterized by vertex cover.

According to Grohe [121], meta-theorems expose the deep relations between logic and combinatorial structures, which is a fundamental issue of computational complexity. Such theorems also yield a better understanding of the scope of general algorithmic techniques and the limits of tractability. The canonical example here is Courcelle's theorem [65], which states that all problems expressible in Monadic Second-Order Logic are linear-time solvable on graphs of bounded treewidth. For more restricted parameters such as the vertex cover number, algorithmic meta-theorems are available with a better dependency on the parameter [116, 172]. In kernelization there are meta-theorems showing polynomial kernels for restricted graph families [28, 112]. A systematic way to understand the kernelization complexity of parameterizations by vertex cover would therefore be to obtain a meta-theorem capturing a large class of problems admitting polynomial kernels. But is there a logic capturing the known positive results we are interested in? If such a logic exists, it would have to be able to express Vertex Cover, which admits polynomial kernel, but not Clique, which does not [148]; it should capture Odd Cycle Transversal [148] and Long Cycle [31] but not Dominating Set [83]; and Treewidth [35] but not Cutwidth [72]. As a consequence, if a logic capturing the phenomenon of polynomial kernelizability for problems parameterized by vertex cover exists, it should be a very strange logic. We therefore take a different approach: we provide three theorems with general conditions capturing a wide variety of known kernelization results about parameterization by vertex cover.

Kernelization algorithms are often based on the idea of a *reduction rule*: an efficient procedure that, if appropriate conditions are satisfied, replaces a substructure of the input by a smaller structure. Such a rule is *correct* if the replacement does not change the answer to the decision problem. Many kernelizations consist of a series of carefully crafted reduction rules. Their effectiveness is proven by extremal arguments showing that when an instance is *exhaustively reduced* (i.e., no rule can be applied anymore), its size is bounded by a small function of its parameter value.

It has been observed before [31, 147] that for several graph problems parameterized by vertex cover, one can obtain a polynomial kernel using reduction rules that *mark* a bounded number of important vertices for each constant-size subset of the vertex cover, afterwards removing the vertices that were not marked at any point. Our first contribution here is to uncover a characteristic of graph problems that explains their amenability to such reduction strategies. We provide general kernelization theorems using this characteristic. For example, we target the following problem: given a graph G, find a minimum-size set of vertices that hits all induced subgraphs of G belonging to some graph family Π. Roughly speaking, this problem has a polynomial kernel parameterized by vertex cover if membership in Π is invariant under changing the presence of all but a constant number of (non)edges incident to each vertex (and some other technical conditions are met). The problems of finding the largest induced subgraph of G belonging to Π, and of finding a partition of the vertex set into a constant number of sets that each induce Π-free subgraphs, have polynomial kernels parameterized by vertex cover under similar conditions.

Our general theorems not only capture a wide variety of known results, they also imply results that were not known before. For example, as a corollary to our theorems we establish that the $\mathcal{F}$-Minor-Free Deletion problem has a polynomial kernel for every fixed $\mathcal{F}$, when parameterized by the size of a vertex cover. Our third general theorem, dealing with graph partitioning problems, is a significant generalization of the polynomial kernel for q-Coloring parameterized by vertex cover [147] since coloring a graph is equivalent to partitioning its vertex set into independent sets. We show that many different graph partitioning problems such as Partition into q Forests [117, GT14], Partition into q Cliques [117, GT15], and Partition into q Planar Graphs, have polynomial kernels parameterized by vertex cover. Although several partitioning problems were already listed by Garey and Johnson [117], little was previously known about the their kernelization complexity. Our theorems show that in many cases, effective preprocessing is possible for instances of such problems that have small vertex covers.

4.2 Minor Models in Graphs with Small Vertex Covers

The following proposition shows that in graphs that have a small vertex cover, the containment of a graph H as a minor (see Appendix A.3) is witnessed by a subgraph whose maximum degree is bounded by $\Delta(H)$, and whose vertex count is linear in $|V(H)|$ and the vertex cover number of the host graph. The proposition will be useful at various points in the chapter when applying our general theorems to $\mathcal{F}$-Minor-Free Deletion and its variants.

Proposition 4.1. *If graph G contains graph H as a minor, then there is a subgraph $G^* \subseteq G$ containing H as a minor such that $\Delta(G^*) \leq \Delta(H)$ and $|V(G^*)| \leq |V(H)| + \textsc{vc}(G^*) \cdot (\Delta(H) + 1)$.*

Proof. Let G be a graph containing a minor model ϕ of a graph H. We show how to find a subgraph G^* satisfying the claims.

For every edge $\{u, v\} \in E(H)$, mark an arbitrary edge in G between $\phi(u)$ and $\phi(v)$. In each branch set $\phi(v)$ for $v \in V(H)$, mark the edges of any inclusion-minimal tree T_v in G that contains all the vertices incident to edges marked so far. For each isolated vertex $v \in V(H)$, mark an arbitrary vertex in $\phi(v)$. Now obtain G^* from G by deleting unmarked edges and deleting unmarked vertices that are not incident on a marked edge. It is easy to verify that restricting ϕ to G^* gives an H-minor model in G^*. To see that $\Delta(G^*) \leq \Delta(H)$, consider a vertex $v \in V(G^*)$ and partition the edges incident on it into two types: those that were marked to build a tree T_u in a branch set $\phi(u)$ for some $u \in V(H)$, and those that connect two different branch sets. Suppose v is incident on r edges of the tree T_u. Then T_u has at least r leaves, and all these leaves connect $\phi(u)$ to different branch sets. Observe that each connection to a different branch set corresponds to a distinct neighbor of u in H. As u has at most $\Delta(H)$ neighbors in H, there are at most $\Delta(H)$ connections between the branch set $\phi(u)$ and other branch sets. Since at least r connections are made by leaves of T_u unequal to v, edges incident on v can make at most $\Delta(H) - r$ connections to other branch sets. As this accounts for all edges incident on v in G^* it follows that $\deg_{G^*}(v) \leq r + (\Delta(H) - r)$. As v was arbitrary this proves $\Delta(G^*) \leq \Delta(H)$. It remains to prove that $|V(G^*)|$ is suitably small.

Let $X \subseteq V(G^*)$ be a minimum vertex cover of G^*. All isolated vertices in G^* correspond to isolated vertices in H, so there are at most $|V(H)|$ of them. The remaining vertices of G^* that do not belong to X have at least one neighbor in X (as X is a vertex cover and the vertices are not isolated). Since each vertex in X has degree at most $\Delta(H)$, the total number of vertices in G^* is at most $|V(H)| + |X| + \Delta(H) \cdot |X| \leq |V(H)| + |X|(\Delta(H) + 1)$, which proves the claim. □

4.3 Characterization by Few Adjacencies

In this section we introduce a general reduction rule for problems parameterized by vertex cover, and show that the rule preserves the existence of certain kinds of induced subgraphs. The central concept is the following.

Definition 4.1. A graph property Π is *characterized by* $c_\Pi \in \mathbb{N}$ *adjacencies* if for all graphs $G \in \Pi$, for every vertex $v \in V(G)$, there is a set $D \subseteq V(G) \setminus \{v\}$ of size at most c_Π such that all graphs G' that are obtained from G by adding or removing edges between v and vertices in $V(G) \setminus D$, are also contained in Π.

The following proposition shows that various graph properties are characterized by few adjacencies.

Proposition 4.2. *The following properties are characterized by constantly many adjacencies (for any fixed finite set of graphs $\mathcal{F}$, graph H, or $r \geq 4$, respectively):*

1. *Having a Hamiltonian path (resp. cycle), or having an odd cycle ($c_\Pi = 2$).*
2. *Containing $H \in \mathcal{F}$ as a minor ($c_\Pi = \max_{H \in \mathcal{F}} \Delta(H)$).*
3. *Having a perfect H-packing ($c_\Pi = \Delta(H)$).*
4. *Having a chordless cycle of length at least r ($c_\Pi = r - 1$).*

Proof. We prove the claims one by one.

(1) First consider the property of being Hamiltonian. Take a graph G with a Hamiltonian cycle C, and consider an arbitrary vertex v in G. Let D contain the predecessor and successor of v on the cycle. Then it is easy to see that changing the presence of edges between v and $V(G) \setminus D$, preserves the cycle C. Hence by Definition 4.1 this proves that the property of Hamiltonicity is characterized by two adjacencies. As the length of the cycle is not affected, the same proof goes for the property of having an odd cycle. The property of having a Hamiltonian path is similar; for the endpoints we only have to preserve a single adjacency.

(2) Let $\mathcal{F}$ be a finite set of graphs. Let G contain $H' \in \mathcal{F}$ as a minor, and let $v \in V(G)$ be an arbitrary vertex. We give a set $D \subseteq V(G) \setminus \{v\}$ of size at most $\max_{H \in \mathcal{F}} \Delta(H)$ such that changing the adjacencies between v and $V(G) \setminus D$ preserves the fact that G has an H'-minor. By Proposition 4.1 a subgraph G^* of G with maximum degree at most $\Delta(H')$ exists that has an H'-minor model ϕ. If v is not contained in graph G^*, then changing the presence of edges incident to v

preserves the minor model ϕ in G. If v is contained in G^*, then pick $D := N_{G^*}(v)$ which has size at most $\Delta(H')$ by the degree bound of G^* guaranteed by the proposition. Changing adjacencies between v and $V(G) \setminus D$ preserves the fact that G^* is a subgraph of G, and therefore preserves the fact that G has H' as a minor; this implies membership in Π.

(3) Fix a graph H and let G be a graph with a perfect H-packing. For an arbitrary vertex $v \in V(G)$, consider a perfect H-packing in G and let G' be the subgraph in the packing that contains v. Picking $D := N_{G'}(v)$ it follows that $|D| \leq \Delta(H)$. Changing adjacencies between v and $V(G) \setminus D$ in G preserves the perfect H-packing we started from, as all edges incident on v needed to make the subgraph G' isomorphic to H are maintained. Hence the graph resulting from such modifications has a perfect H-packing and is contained in Π.

(4) Let G be a graph with a chordless cycle C of length at least r, and let v be an arbitrary vertex. If v does not lie on C then changing the presence of edges incident on v preserves C and results in a graph with a chordless cycle of length at least r. Suppose therefore that v lies on C, and label the vertices on C as $(v, v_2, \ldots, v_k)$ for some $k \geq r$. Define $D := \{v_2, \ldots, v_{r-1}\} \cup \{v_k\}$, i.e., D contains the predecessor of v and its $r-2$ successors. Now let G' be obtained from G by changing the adjacency between v and $V(G) \setminus D$. We prove that G' has a chordless cycle of length at least r. Let i be the smallest integer larger than two such that v is adjacent to v_i in G'. As we explicitly preserved the edge from v to v_k, this is well-defined. Because the vertices $\{v_2, \ldots, v_{r-1}\}$ are contained in D we know that $i > r-1$ because C is chordless. Since the only edges that were modified when moving from G to G' are incident on v, it follows from the choice of i that $(v, v_2, v_3, \ldots, v_i)$ is a chordless cycle in G' of length at least r; this completes the proof. □

To aid the intuition, we also consider some graph properties that do not satisfy the conditions of Definition 4.1. The properties of having chromatic number at least four, of being a cycle, or of *not* being a perfect graph, cannot be characterized by a constant number of adjacencies. To see this for graphs of chromatic number at least four, consider an odd wheel with a rim of length t: this is the graph built from an odd cycle C_t by adding a new vertex x, the *hub*, that is adjacent to all vertices of the cycle. As an odd cycle requires three colors in a proper coloring, the adjacency of the hub to all other vertices increases the chromatic number to four. Now observe that removing any edge between the hub and the cycle decreases the chromatic number to three, as the two endpoints of that edge can then share the same color. Hence any vertex set D that preserves the fact that the chromatic number is at least four, when changing adjacencies between x and vertices not in D, must contain all vertices of the cycle. Consequently, such sets cannot have constant size: having chromatic number at least four is not characterized by a finite number of adjacencies. Similar constructions can be made for the properties of being a cycle, and for imperfectness.

Before introducing the reduction rule that is based on characterizations by few adjacencies, we prove that the existence of such characterizations is closed under union and intersection.

Proposition 4.3. *Let Π and Π' be graph properties characterized by c_Π and $c_{\Pi'}$ adjacencies, respectively. The following holds:*

1. *The property $\Pi \cup \Pi'$ is characterized by $\max(c_\Pi, c_{\Pi'})$ adjacencies.*
2. *The property $\Pi \cap \Pi'$ is characterized by $c_\Pi + c_{\Pi'}$ adjacencies.*

Proof. We prove the two items separately.

(1) Let G be a graph in $\Pi \cup \Pi'$, and let v be an arbitrary vertex in G. We have to find a set D of size at most $\max(c_\Pi, c_{\Pi'})$ that satisfies the conditions of Definition 4.1 with respect to v. If $G \in \Pi$ then the characterization of Π by c_Π adjacencies guarantees the existence of a set $D \subseteq V(G) \setminus \{v\}$ of size at most c_Π such that changing adjacencies between v and $V(G) \setminus D$ preserves membership in Π, and hence in the union $\Pi \cup \Pi'$. If $G \in \Pi'$ we similarly find a set of size at most $c_{\Pi'}$ that preserves membership in Π' and therefore in the union. In either case we find a set of size at most $\max(c_\Pi, c_{\Pi'})$ that satisfies the conditions of Definition 4.1, establishing the characterization of $\Pi \cup \Pi'$.

(2) Let G be a graph in $\Pi \cap \Pi'$, and let v be an arbitrary vertex in G. Let $D \subseteq V(G) \setminus \{v\}$ be a set of size at most c_Π that preserves membership in Π, and let $D' \subseteq V(G) \setminus \{v\}$ be a set of size at most $c_{\Pi'}$ preserving membership in Π'. Now consider $D^* := D \cup D'$. Changing adjacencies between v and $V(G) \setminus D^*$ preserves membership in Π (since D^* contains D), and preserves membership in Π' (as D^* contains D'). Hence the set D^* of size at most $c_\Pi + c_{\Pi'}$ preserves membership in the intersection $\Pi \cap \Pi'$, which proves the claim. □

The closure properties of Proposition 4.3 can be used to quickly establish that a graph class is characterized by a constant number of adjacencies. Also observe that any finite graph property Π is trivially characterized by $\max_{G \in \Pi} |V(G)| - 1$ adjacencies (for $G \in \Pi$ and $v \in V(G)$, choose D as $V(G) \setminus \{v\}$). This will be useful to verify the preconditions to the general kernelization theorems.

The single reduction rule that we use to derive our general kernelization theorems, is the Reduce procedure presented as Algorithm 1. Recall that $\binom{X}{\leq c}$ denotes the collection of all subsets of X of size at most c, including the empty set. The utility of Reduce for kernelization stems from the fact that it efficiently shrinks a graph to a size bounded polynomially in the cardinality of the given vertex cover X.

Observation 4.1. *For every fixed constant c_Π,* Reduce (G, X, r, c_Π) *runs in polynomial time and results in a graph on $\mathcal{O}(|X| + r \cdot 2^{c_\Pi} \cdot |\binom{X}{\leq c_\Pi}|) = \mathcal{O}(|X| + r \cdot |X|^{c_\Pi})$ vertices.*

The soundness of the Reduce procedure for many types of kernelization comes from the following lemma. It shows that for graph properties Π that are characterized by few adjacencies, an application of Reduce with parameter $r =$

Algorithm 1 Reduce (Graph G, Vertex Cover $X \subseteq V(G), r \in \mathbb{N}, c \in \mathbb{N}$)

```
for each $Y \in \binom{X}{\leq c}$ and partition of $Y$ into $Y^+ \cup Y^-$ do
    $Z := \{v \in V(G) \setminus X \mid v$ is adjacent to all of $Y^+$ and to none of $Y^-\}$
    mark $r$ arbitrary vertices of $Z$ (if $|Z| < r$ then mark all of them)
end for
delete from $G$ all unmarked vertices that are not contained in $X$
```

$s + p$ preserves the existence of induced Π subgraphs of size up to p that avoid any set of size at most s.

Lemma 4.1. *Let Π be characterized by c_Π adjacencies, and let G be a graph with vertex cover X. If $G[P] \in \Pi$ for some $P \subseteq V(G) \setminus S$ and $S \subseteq V(G)$, then for any $r \geq |S| + |P|$ the graph G' resulting from* Reduce (G, X, r, c_Π) *contains $P' \subseteq V(G') \setminus S$ such that $G'[P'] \in \Pi$ and $|P'| = |P|$.*

Proof. Assume the conditions in the statement of the lemma hold, and let $R \subseteq V(G)$ be the vertices that are removed by the reduction procedure, i.e., $R := V(G) \setminus V(G')$. Let $p_1, p_2, \ldots, p_t$ be an arbitrary ordering of $P \cap R$. We inductively create a sequence of sets $P_0, P_1, \ldots, P_t$ with $P = P_0$ such that (a) $G[P_i] \in \Pi$, (b) $|P_i| = |P|$, (c) $P_i \cap S = \emptyset$, and (d) $P_i \cap R = \{p_{i+1}, p_{i+2}, \ldots, p_t\}$ for every $i \in \{0, 1, \ldots, t\}$. Note that P satisfies the constraints imposed on P_0, while existence of P_t proves the lemma. Hence, we only need to show how to construct P_i from P_{i-1} for $i \in [t]$.

Consider graph $G[P_{i-1}]$ and vertex $p_i \in P_{i-1}$. As $G[P_{i-1}] \in \Pi$, Definition 4.1 ensures that there exists a set D of at most c_Π vertices of P_{i-1} such that arbitrarily changing adjacencies between p_i and vertices of $P_{i-1} \setminus D$ in $G[P_{i-1}]$ preserves membership in Π. Let $D^+ := N_G(p_i) \cap D$ and $D^- := D \setminus D^+$. Since vertex p_i is contained in R and was removed by the reduction process, it follows from the deletion procedure that $p_i \notin X$ and therefore that $D^+ \subseteq N_G(p_i) \subseteq X$ since X is a vertex cover of G. Let $D^-_X := D^- \cap X$. Observe that p_i was a candidate for marking for the partition (D^+, D^-_X) of $D \cap X$, but as $p_i \in R$ it was not marked. Hence, there exist $r \geq |S| + |P|$ marked vertices in $V(G) \setminus X$ adjacent to all of D^+ and none of D^-_X. As $|P_{i-1}| = |P|$ and p_i is not marked, we can find a vertex $p'_i \in V(G) \setminus X$ that does not belong to P_{i-1} or S, is marked, and has the same neighborhood in $D \cap X$ as p_i. Since X is a vertex cover, both p_i and p'_i have all their neighbors in X. As p'_i is not adjacent to any member of D^-_X, it is not adjacent to D^-. Take $P_i := (P_{i-1} \cup \{p'_i\}) \setminus \{p_i\}$. Note that $|P_i| = |P_{i-1}| = |P|$ and $P_i \cap R = \{p_{i+1}, p_{i+2}, \ldots, p_t\}$. Moreover, a graph isomorphic to $G[P_i]$ can be obtained from $G[P_{i-1}]$ by changing adjacencies between p_i and vertices of $P_{i-1} \setminus D$. The only adjacencies that need to be changed are between p_i and $N_G(p_i) \triangle N_G(p'_i) \subseteq X$ ($\triangle$ denotes symmetric difference), but this set is disjoint with D and hence the changes preserve membership in Π. As P_i satisfies all induction claims, this completes the proof. $\square$

4.4 Kernelization for Vertex-Deletion Problems

We present a general theorem that gives polynomial kernels for vertex-deletion problems of the following form.

> Deletion Distance to Π-Free [vc]
> **Input:** A graph G with a vertex cover X, and an integer $\ell \geq 1$.
> **Parameter:** The size $k := |X|$ of the vertex cover.
> **Question:** Is there a set $S \subseteq V(G)$ of size at most ℓ such that $G - S$ does not contain a graph in Π as an induced subgraph?

Observe that the graph property Π need not be finite or decidable. As discussed in Section 2.2.1, we ask for a vertex cover along with the input. To apply the data reduction schemes presented in this chapter one may simply compute a 2-approximate vertex cover and use that as X. In the following, recall that a graph G is *vertex-minimal* with respect to Π if $G \in \Pi$ and for all $S \subsetneq V(G)$ the graph $G[S]$ is not contained in Π.

Theorem 4.1. *If* Π *is a graph property such that:*

(i) Π *is characterized by* c_Π *adjacencies,*
(ii) every graph in Π *contains at least one edge, and*
(iii) there is a nondecreasing polynomial $p\colon \mathbb{N} \to \mathbb{N}$ *such that all graphs* G *that are vertex-minimal with respect to* Π *satisfy* $|V(G)| \leq p(\operatorname{vc}(G))$,

then Deletion Distance to Π-Free [vc] *has a kernel with* $\mathcal{O}((k + p(k))k^{c_\Pi})$ *vertices, where* $k := |X|$.

Before proving the theorem, we briefly discuss its preconditions. Let us first show the necessity of (ii) by considering the property Π only consisting of the two-vertex graph without an edge. Then a graph G is a clique if and only if it does not contain the graph in Π as an induced subgraph, and hence a graph G has a clique of size at least ℓ if and only if we can delete at most $|V(G)| - \ell$ vertices from G to make it Π-free. Observe that Π is characterized by a single adjacency and trivially satisfies (iii) for $p(n) = 2$. But Clique parameterized by vertex cover does not admit a polynomial kernel unless NP ⊆ coNP/poly [29], which explains why (ii) is necessary.

To justify (i), consider the class Π containing the odd holes and odd anti-holes (induced cycles of odd length at least five, and their edge-complements). It is easy to verify that this Π satisfies conditions (ii) and (iii). Now observe that G has vertex-deletion distance at most ℓ to property Π if and only if G can be made perfect by ℓ vertex deletions, and that the kernelization complexity of Perfect Deletion parameterized by vertex cover is still open.

The third condition demands that the size of vertex-minimal graphs in Π is bounded polynomially in their vertex cover number. The condition is needed to make the proof go through. Observe that the restriction to a *polynomial*

Table 4.1: Problems that admit polynomial kernels when parameterized by the size of a given vertex cover, by applying Theorem 4.1.

Problem	Forbidden property Π	c_Π
Vertex Cover	$\{K_2\}$	1
Odd Cycle Transversal	Graphs containing an odd cycle	2
Chordal Deletion	Graphs with a chordless cycle	3
$\mathcal{F}$-Minor-Free Deletion	Graphs with an $H \in \mathcal{F}$-minor	$\max_{H \in \mathcal{F}} \Delta(H)$
Planarization	Graphs with a K_5 or $K_{3,3}$ minor	4
Treewidth-η Transversal	Graphs of treewidth $> \eta$	$f(\eta)$

function in the condition is crucial, as the existence of a (possibly exponential) function is trivial. For any graph property Π, the existence of a function $g\colon \mathbb{N} \to \mathbb{N}$ such that all graphs $G \in \Pi$ have an induced subgraph $G' \subseteq G$ contained in Π with $|V(G')| \leq g(\text{vc}(G'))$ is guaranteed by the fact that graphs of bounded vertex cover are well-quasi-ordered by the induced subgraph relation [98].

Having justified the preconditions to our general theorem, we give its proof.

Proof of Theorem 4.1. Consider some input instance (G, X, ℓ). First, observe that if $\ell \geq |X|$, then we clearly have a YES-instance: removal of X results in an empty graph, which is guaranteed not to contain induced subgraphs from Π due to Property (ii). Therefore, we may assume that $\ell < |X|$ as otherwise we output a trivial YES-instance.

We let G' be the result of Reduce $(G, X, \ell + p(|X|), c_\Pi)$ and return the instance (G', X, ℓ), which gives the right running time and size bound by Observation 4.1. We need to prove that the output instance (G', X, ℓ) is equivalent to the input instance (G, X, ℓ). As G' is an induced subgraph of G, it follows that if $G - S$ does not contain any graph in Π, then neither does $G' - (S \cap V(G'))$. Therefore, if (G, X, ℓ) is a YES-instance, then so is (G', X, ℓ). Assume then, that (G', X, ℓ) is a YES-instance and let S be a subset of vertices with $|S| \leq \ell$ such that $G' - S$ does not contain any induced subgraph from Π. We claim that $G - S$ does not contain such induced subgraphs either, i.e., that S is also a feasible solution for the instance (G, X, ℓ).

Assume for the sake of contradiction that there is a set $P \subseteq V(G) \setminus S$ such that $G[P] \in \Pi$. Consider a *minimal* such set P, which ensures by Property (iii) that $|P| \leq p(\text{vc}(G[P]))$. As $P \cap X$ is a vertex cover of $G[P]$, it follows that $|P| \leq p(|P \cap X|) \leq p(|X|)$. As we executed the reduction with parameter $r = \ell + p(|X|)$, Lemma 4.1 guarantees the existence of a set $P' \subseteq V(G') \setminus S$ such that $G'[P'] \in \Pi$. But this shows that the graph $G' - S$ contains an induced Π subgraph, contradicting the assumption that S is a solution for G' and thereby concluding the proof. □

Corollary 4.1. *All problems in Table 4.1 fit into the framework of Theorem 4.1 and admit polynomial kernels parameterized by the size of a given vertex cover.*

Proof. We consider the problems in the order of Table 4.1 and show how they fit into the framework.

VERTEX COVER [VC]. Observe that a graph G has a vertex cover of size ℓ if and only there is a set $S \subseteq V(G)$ of size ℓ such that $G - S$ is an independent set, or equivalently, $G - S$ does not have K_2 as an induced subgraph. So VERTEX COVER [VC] is equivalent to DELETION DISTANCE TO Π-FREE [VC] for $\Pi = \{K_2\}$. Since this Π contains only a single graph of degree one, it is easily seen to be characterized by the single adjacency of one vertex in K_2 to its neighbor (Property (i)). Obviously all graphs in Π contain at least one edge (Property (ii)), and since Π contains a single graph on two vertices, having a vertex cover of size one, the constant function $p(n) := 2$ suffices for Property (iii). Hence all preconditions for Theorem 4.1 are satisfied and the problem has a kernel with $\mathcal{O}(|X|^2)$ vertices.

ODD CYCLE TRANSVERSAL [VC]. A graph G is bipartite if and only if it does not contain a graph with an odd cycle as an induced subgraph. Hence by letting Π contain all graphs that contain an odd cycle (which is not the same as letting Π be the class of all odd cycles), ODD CYCLE TRANSVERSAL [VC] is equivalent to DELETION DISTANCE TO Π-FREE [VC]. By Proposition 4.2 this property Π is characterized by a constant number of adjacencies; the proof of the proposition shows $c_\Pi := 2$ suffices. Since all graphs with an odd cycle have at least one edge, the second condition is satisfied as well. For the last condition, consider a vertex-minimal graph G with an odd cycle; such a graph is Hamiltonian and has an odd number of vertices. Fix a minimum vertex cover X of G: walking along a Hamiltonian cycle of G, we must visit X before and after visiting every vertex from $V(G) \setminus X$ (as vertices of $V(G) \setminus X$ only have neighbors in X). Hence $|V(G) \setminus X| \leq X$ as every vertex in X is visited exactly one, and therefore $|V(G)| \leq 2|X|$ which proves that $p(n) := 2n$ suffices for the polynomial in Property (iii). We obtain a kernel with $\mathcal{O}(|X|^3)$ vertices.

CHORDAL DELETION [VC]. A graph G is chordal if all its cycles of length at least four have a chord; this can be stated equivalently as saying that G does not contain a graph with a chordless cycle as an induced subgraph. If we take Π to be the class of graphs that have a chordless cycle, we can express CHORDAL DELETION [VC] as an instantiation of DELETION DISTANCE TO Π-FREE [VC]. The proof of Proposition 4.2 shows the property is characterized by three adjacencies. As all graphs with a chordless cycle contain an edge, the second property is satisfied. Similarly as before, a vertex-minimal graph with a chordless cycle is Hamiltonian and hence $p(n) := 2n$ suffices for Property (iii). The resulting kernel has $\mathcal{O}(|X|^4)$ vertices.

$\mathcal{F}$-MINOR-FREE DELETION [VC]. If we let Π contain all graphs that contain a member of $\mathcal{F}$ as a minor, then a graph is Π-induced-subgraph-free if and only if it is $\mathcal{F}$-minor-free. By Proposition 4.2 this class Π is characterized by $c_\Pi := \max_{H \in \mathcal{F}} \Delta(H)$ adjacencies, so we satisfy Property (i). If $\mathcal{F}$ contains an empty graph, then $\mathcal{F}$-minor-free graphs have constant size and the problem is polynomial-time solvable; hence in interesting cases the graphs containing a minor from $\mathcal{F}$ have at least one edge (Property (ii)). Finally, consider a vertex-minimal graph G^*

that contains a graph $H \in \mathcal{F}$ as a minor. By Proposition 4.1 we have $|V(G^*)| \leq |V(H)| + \textsc{vc}(G^*) \cdot (\Delta(H) + 1)$. As $\mathcal{F}$ is fixed, the maximum degree and size of graphs in $\mathcal{F}$ are constants which shows that Property (iii) is satisfied, resulting in a kernel with $\mathcal{O}(|X|^{\Delta+1})$ vertices for $\Delta := \max_{H \in \mathcal{F}} \Delta(H)$.

Planarization [vc]. Since this problem is a special case of $\mathcal{F}$-Minor-Free Deletion [vc] for $\mathcal{F} := \{K_5, K_{3,3}\}$, and both forbidden minors are nonempty, the proof given above shows that this problem has a kernel with $\mathcal{O}(|X|^5)$ vertices.

Treewidth-η Transversal [vc]. Recall that the Treewidth-η Transversal problem asks for a vertex set whose removal results in a graph of treewidth at most η. Since treewidth does not increase when taking a minor [20, Lemma 16], the class of graphs of treewidth at most η is closed under minors. By the famous results of Robertson and Seymour [200], this implies that for each η there is a finite obstruction set $\mathcal{F}_\eta$ such that G has treewidth at most η if and only if G avoids all graphs in $\mathcal{F}_\eta$ as a minor. It is easy to see that the minimal obstruction sets $\mathcal{F}_\eta$ do not contain empty graphs, as empty graphs have treewidth zero and cannot be obstructions to having treewidth $\eta \geq 0$. Hence we may obtain a polynomial kernel for Treewidth-η Transversal [vc] by using the obstruction set $\mathcal{F}_\eta$ in the more general $\mathcal{F}$-Minor-Free Deletion [vc] scheme. The kernel size is $\mathcal{O}(|X|^{\Delta+1})$ where $\Delta := \max_{H \in \mathcal{F}_\eta} \Delta(H)$. □

Using Proposition 4.2 and Proposition 4.3 it is easy to apply Theorem 4.1 to many other vertex-deletion problems. For example, a graph is *distance hereditary* if and only if it excludes the house, gem, domino and holes (chordless cycles of length at least five) as induced subgraphs [45, Theorem 10.1.1]. (The house, gem and domino are fixed, constant-size graphs [45, Chapter 1].) Hence if we take Π to contain these constant graphs, together with the graphs that contain a chordless cycle of length at least five, then a graph is distance hereditary if and only if it is induced Π-free. Since Π is the union of a finite graph property $\{\text{house}, \text{gem}, \text{domino}\}$ with the graphs containing a chordless cycle of length at least five, and both are characterized by a finite number of adjacencies, it follows from Proposition 4.3 that Π is characterized by few adjacencies. It is easy to verify that the other preconditions to Theorem 4.1 are satisfied as well, which implies a polynomial kernel for Distance Hereditary Deletion [vc]. Using this recipe one can obtain polynomial kernels for a host of vertex-deletion problems, whose corresponding graph classes can be defined by combining the elements of Proposition 4.2 with a finite number of arbitrary forbidden induced subgraphs. We do not list all these possible applications here, but move on to our next general theorem.

4.5 Kernelization for Largest Induced Subgraph Problems

In this section we study the following class of problems, which is in some sense dual to the class considered earlier.

Largest Induced Π-Subgraph [vc]
Input: A graph G with a vertex cover X, and an integer $\ell \geq 1$.
Parameter: The size $k := |X|$ of the vertex cover.
Question: Is there a set $P \subseteq V(G)$ of size at least ℓ such that $G[P] \in \Pi$?

The following theorem gives sufficient conditions for the existence of polynomial kernels for such problems.

Theorem 4.2. *If* Π *is a graph property such that:*

(i) Π *is characterized by* c_Π *adjacencies, and*
(ii) there is a nondecreasing polynomial $p\colon \mathbb{N} \to \mathbb{N}$ *such that all graphs* $G \in \Pi$ *satisfy* $|V(G)| \leq p(\textsc{vc}(G))$,

then Largest Induced Π-Subgraph [vc] *has a kernel with* $\mathcal{O}(p(k) \cdot k^{c_\Pi})$ *vertices, where* $k := |X|$.

There is a natural example showing the necessity of the first condition in Theorem 4.2. If we take Π as the class of all cliques, then testing whether a graph G has an induced subgraph in Π on at least ℓ vertices is equivalent to asking whether G has a clique of size at least ℓ. Since the vertex count of a complete graph exceeds its vertex cover number by exactly one, the class of cliques satisfies (ii). The conditional superpolynomial kernel lower bound for Clique parameterized by vertex cover explains why (i) is necessary; the class of cliques is not characterized by any constant number of adjacencies.

The second condition of Theorem 4.2 is needed to ensure that the resulting problems have kernels at all. Observe that we do not require the set of graphs Π to be decidable. In the absence of the second condition, we could let Π contain all i-vertex graphs for which the i-th Turing machine halts on a blank tape. This class is trivially characterized by zero adjacencies, since membership in Π only depends on the number of vertices. If the Largest Induced Π-Subgraph [vc] problem for this class Π would have a kernel, then we could decide the Halting Problem as follows. To decide whether the i-th Turing machine halts, we create the empty graph G_i on i vertices with an empty vertex cover. By the definition of Π, the i-th machine halts if and only if G_i has an induced Π subgraph on i vertices. Running the supposed kernelization on this instance would yield an equivalent, constant-size instance as the parameter value is zero. We could then decide the problem by looking up the answer in a table for constant-size instances hard-coded into the algorithm, thereby solving the Halting Problem. The requirement that the size of the graphs in Π is bounded in terms of their vertex cover number, is therefore entirely natural. We need the dependence to be polynomial in order to obtain our polynomial kernel.

Having justified the preconditions, we present the proof of the theorem.

Table 4.2: Problems that admit polynomial kernels when parameterized by the size of a given vertex cover, by applying Theorem 4.2.

Problem	Desired property Π	c_Π
LONG CYCLE	Graphs with a Hamiltonian cycle	2
LONG PATH	Graphs with a Hamiltonian path	2
H-PACKING	Graphs with a perfect H-packing	$\Delta(H)$

Proof of Theorem 4.2. The kernelization reduces an instance (G, X, ℓ) by executing REDUCE $(G, X, p(|X|), c_\Pi)$ to obtain a graph G', and outputs the instance (G', X, ℓ). By Observation 4.1 this can be done in polynomial time and results in a graph whose size is appropriately bounded; it remains to prove that the two instances are equivalent.

Since G' is an induced subgraph of G, any solution contained in G' is also contained in G: so if (G', X, ℓ) is a YES-instance, then (G, X, ℓ) is as well. Assume then that (G, X, ℓ) is a YES-instance and let $P \subseteq V(G)$ be such that $G[P] \in \Pi$ and $|P| \geq \ell$. Clearly, $X \cap P$ is a vertex cover of $G[P]$, so $|P| \leq p(|X \cap P|) \leq p(|X|)$ by Property (ii). Since the reduction procedure is executed with a value $r = p(|X|)$ and $|P| \leq p(|X|)$, by applying Lemma 4.1 with an empty set for S we find that G' contains a set P' of the same size as P such that $G'[P] \in \Pi$. This proves that (G', X, ℓ) is a YES-instance and shows the correctness of the kernelization. □

Corollary 4.2. *All problems in Table 4.2 fit into the framework of Theorem 4.2 and admit polynomial kernels parameterized by the size of a given vertex cover.*

Proof. We consider the problems in the order of Table 4.2 and show how they fit into the framework.

LONG CYCLE [VC]. Observe that if G has a cycle on ℓ vertices $(v_1, \ldots, v_\ell)$ then the graph $G[\{v_1, \ldots, v_\ell\}]$ is Hamiltonian. So G has an ℓ-cycle if and only if G has an induced Hamiltonian subgraph on ℓ vertices. Hence LONG CYCLE [VC] is equivalent to LARGEST INDUCED Π-SUBGRAPH [VC] by letting Π be the class of Hamiltonian graphs. By Proposition 4.2 this class is characterized by two adjacencies. For all Hamiltonian graphs G' it holds that $|V(G')| \leq 2|\textsc{vc}(G')|$, since for any Hamiltonian cycle C and vertex cover X, any vertex outside X must have its unique predecessor on the cycle in X. Hence Property (ii) is satisfied as well and we obtain a kernel with $\mathcal{O}(|X|^3)$ vertices. The proof for LONG PATH [VC] is analogous.

H-PACKING [VC]. A graph G admits an H-packing of ℓ disjoint subgraphs if and only if G has an induced subgraph on $\ell \cdot |V(H)|$ vertices that admits a perfect H-packing. If H is an empty graph then the answer is trivial: there are ℓ vertex-disjoint subgraphs isomorphic to H if and only if the vertex count is at least $\ell \cdot |V(H)|$. We can therefore solve the case that H is an empty graph in polynomial time, and focus on the case that H is nonempty. Choosing Π as the graphs with a perfect H-packing allows us to model the packing problem as an instantiation of Largest Induced Π-Subgraph [vc], by scaling the target value ℓ by a factor $|V(H)|$. Proposition 4.2 shows that Π is characterized by $\Delta(H)$ adjacencies. Let us now prove that the second condition is satisfied for this Π, by utilizing the fact that we demand H to be nonempty. Consider a graph G with a perfect H-packing for a nonempty H, and let X be a minimum vertex cover of G. Each subgraph in the packing contains at least one edge, so each subgraph in the packing has size $|V(H)|$ and contains a vertex from X. Hence $|V(G)| \leq |X| \cdot |V(H)|$, which proves that $p(n) := n \cdot |V(H)|$ suffices for the polynomial. For fixed H this results in a kernel with $\mathcal{O}(|X|^{\Delta(H)+1})$ vertices. □

4.6 Kernelization for Graph Partitioning Problems

Having considered induced-subgraph testing and vertex-deletion problems in the previous two sections, we now consider partitioning problems. More concretely, we consider problems that ask for the existence of a partition of the vertex set into a constant number of partite sets such that each partite set induces a subgraph of a desired form. The parameterized problem we study is formally defined as follows.

Partition into q Disjoint Π-Free Subgraphs [vc]
Input: A graph G with vertex cover $X \subseteq V(G)$.
Parameter: The size $k := |X|$ of the vertex cover.
Question: Is there a partition of the vertex set into q sets $S_1 \cup S_2 \cup \ldots \cup S_q$ such that for each $i \in [q]$ the graph $G[S_i]$ does not contain a graph in Π as an induced subgraph?

Note that the value of q is treated as a constant in the above definition. To give an example of a problem that can be captured by this template, consider the 3-Coloring problem, which asks whether a graph admits a proper coloring with three colors. Such a coloring is nothing more than a partition of its vertex set into three independent sets. Observing that a graph is independent if and only if it excludes K_2 as an induced subgraph, we see that 3-Coloring parameterized by vertex cover can be phrased as Partition into 3 Disjoint $\{K_2\}$-Free Subgraphs [vc]. Further applications will be discussed after establishing a sufficient condition for polynomial kernelizability of the general problem.

The kernelization scheme once again uses the Reduce routine as its single reduction rule. Before presenting the kernel, we derive a lemma that shows how an application of Reduce affects instances of partitioning problems. In the following we say that a graph G can be partitioned into q disjoint Π-free subgraphs if there is a partition of $V(G)$ into $S_1 \cup \ldots \cup S_q$ such that for all $i \in [q]$ the graph $G[S_i]$ does not contain a member of Π as an induced subgraph.

Lemma 4.2. *Let* Π *be characterized by* c_Π *adjacencies, and let* $p\colon \mathbb{N} \to \mathbb{N}$ *be a nondecreasing polynomial such that all graphs* G^* *that are vertex-minimal with respect to* Π *satisfy* $|V(G^*)| \leq p(\textsc{vc}(G^*))$*. Let* G *be a graph with vertex cover* X*, and let* G' *be the graph resulting from* Reduce $(G, X, q \cdot p(|X|), q \cdot c_\Pi)$*. If* G' *can be partitioned into* q *disjoint* Π*-free subgraphs, then such a partition exists for* G *as well.*

Proof. Assume the conditions in the statement of the lemma hold, and let $R \subseteq V(G)$ be the vertices that are removed by the reduction procedure, i.e., $R := V(G) \setminus V(G')$. Let $r_1, r_2, \ldots, r_t$ be an arbitrary ordering of R. Assume that $\mathcal{S} = (S_1, S_2, \ldots, S_q)$ is a partition of $V(G')$ such that for each $i \in [q]$ the graph $G'[S_i]$ does not contain an induced subgraph from Π. We inductively create a sequence of set families $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_t$ with $\mathcal{S} = \mathcal{S}_0$ such that $\mathcal{S}_i$ is a partition of $V(G') \cup \{r_1, \ldots, r_i\}$ into q sets $S_i^1, \ldots, S_i^q$, and for all $j \in [q]$ the graph $G[S_i^j]$ does not contain a graph in Π as an induced subgraph. Note that $\mathcal{S}$ satisfies the constraints imposed on $\mathcal{S}_0$, while existence of $\mathcal{S}_t$ proves the lemma. Hence, we only need to show how to construct $\mathcal{S}_i$ from $\mathcal{S}_{i-1}$ for $i \in [t]$.

To construct the partition $\mathcal{S}_i$ out of the partition $\mathcal{S}_{i-1}$ we will show that there is a partite set S_{i-1}^j to which vertex r_i can be added, such that $G[S_{i-1}^j \cup \{r_i\}]$ does not contain a graph in Π. The partition $\mathcal{S}_i$ is then obtained by replacing S_{i-1}^j by $S_{i-1}^j \cup \{r_i\}$ in partition $\mathcal{S}_{i-1}$. Hence it remains to prove that a suitable partite set exists.

Assume for a contradiction that for all $j \in [q]$, the graph $G[S_{i-1}^j \cup \{r_i\}]$ contains an induced Π subgraph. For all $j \in [q]$ let $H_j \in \Pi$ be an induced subgraph of $G[S_{i-1}^j \cup \{r_i\}]$ that is vertex-minimal with respect to Π. By the induction hypothesis, each such subgraph in Π must contain r_i.

Since Π is characterized by c_Π adjacencies, it follows that for each H_j there is a set $D_j \subseteq V(H_j) \setminus \{r_i\}$ of size at most c_Π such that changing the adjacencies between r_i and $V(H_j) \setminus D_j$ in H_j preserves membership in Π. Now consider the union $D^* := \bigcup_{j=1}^q D_j$, and let $D_X^* := D \cap X$ be its intersection with X.

By the choice of parameters to Reduce and the fact that r_i was not marked, we know that for the subset D_X of X of size at most $q \cdot c_\Pi$, the procedure marked $q \cdot p(|X|)$ vertices $Z_{D_X^*} \subseteq V(G) \setminus X$ such that all $z \in Z_{D_X^*}$ have the same neighborhood into D_X^* as r_i, i.e., for which $N_G(z) \cap D_X^* = N_G(r_i) \cap D_X^*$. These vertices $Z_{D_X^*}$ were consequently preserved in G'. We will show that there is a vertex $z^* \in Z_{D_X^*}$ that is not contained in any forbidden graph H_j for $j \in [q]$. To see this, observe first that $r_i \notin Z_{D_X^*}$ since r_i was removed from the graph by the reduction procedure whereas all vertices in $Z_{D_X^*}$ were marked to survive in G'. Since X is a vertex cover of G, for each $j \in [q]$ the intersection $V(H_j) \cap X$ is a vertex cover of H_j. The precondition to the lemma therefore implies that $|V(H_j)| \leq p(\textsc{vc}(H_j)) \leq p(|X|)$. The total number of vertices in the union of the graphs H_j is therefore at most $q \cdot p(|X|)$. Since all these graphs contain r_i, while $r_i \notin Z_{D_X^*}$, the fact that $|Z_{D_X^*}| = q \cdot p(|X|)$ therefore implies that there is indeed a vertex $z^* \in Z_{D_X^*}$ that is not contained in any graph H_j for $j \in [q]$.

Let j^* be the index of the partite set of $\mathcal{S}_{i-1}$ that contains z^*, such that $z^* \in S^{j^*}_{i-1}$. We will use the characterization of Π by few adjacencies to show that r_i can be replaced by z^* in the forbidden graph H_{j^*} while preserving membership in Π, thereby obtaining the contradiction that $G[S^{j^*}_{i-1}]$ contains a graph in Π. Since neither z^* nor r_i is contained in the vertex cover X by the definition of Reduce— it only marks and deletes vertices outside X — it follows that $N_G(z^*) \subseteq X$ and $N_G(r_i) \subseteq X$. Hence $N_G(z^*) \cap (D^* \setminus X) = N_G(r_i) \cap (D^* \setminus X) = \emptyset$. By choice of z^* we have that $N_G(z^*) \cap D^*_X = N_G(r_i) \cap D^*_X$. Combining the last two statements shows that $N_G(z^*) \cap D^* = N_G(r_i) \cap D^*$. This shows that starting from the graph H_{j^*}, we can obtain the graph $G[(V(H_{j^*}) \setminus \{r_i\}) \cup \{z^*\}]$ by changing the label of r_i to z^*, and changing adjacencies between the resulting z^* and vertices outside the set D^*. But since D^* contains the set D_{j^*}, which preserves membership of H_{j^*} in Π, this transformation preserves membership in Π and therefore $G[(V(H_{j^*}) \setminus \{r_i\}) \cup \{z^*\}]$ is contained in Π. But this graph is an induced subgraph of $G[S^{j^*}_{i-1}]$, thereby proving that the partition $\mathcal{S}_{i-1}$ that we started from is not valid since its j^*-th partite set induces a graph containing a member of Π; a contradiction. It follows that when we start from a valid partition $\mathcal{S}_{i-1}$, there is a partite set to which r_i can be added without creating forbidden subgraphs. This proves the lemma. □

Armed with this lemma we state the general kernelization theorem for partitioning problems.

Theorem 4.3. *If Π is a graph property such that:*

(i) Π *is characterized by* c_Π *adjacencies, and*
(ii) *there is a nondecreasing polynomial* $p\colon \mathbb{N} \to \mathbb{N}$ *such that all graphs* G *that are vertex-minimal with respect to* Π *satisfy* $|V(G)| \leq p(\mathrm{vc}(G))$*,*

then Partition into q Disjoint Π-Free Subgraphs [vc] *has a kernel with* $\mathcal{O}(p(k) \cdot k^{q \cdot c_\Pi})$ *vertices, where* $k := |X|$*.*

Proof. The kernelization reduces an instance (G, X) of Partition into q Disjoint Π-Free Subgraphs [vc] by executing Reduce $(G, X, q \cdot p(|X|), q \cdot c_\Pi)$ to obtain a graph G', and outputs the instance (G', X). As before, Observation 4.1 shows that the running time is polynomial for fixed q, and that the output instance has the appropriate size. Note that we absorb the constant factor q in the asymptotic notation. It remains to prove that the two instances are equivalent.

If $S_1 \cup \ldots \cup S_q$ is a partition of $V(G)$ such that $G[S_i]$ contains no induced subgraph in Π for all $i \in [q]$, then that partition can be safely restricted to the vertex set of G' to yield a solution to the output instance: since $G'[S_i \cap V(G')]$ is an induced subgraph of $G[S_i]$, the Π-freeness of the latter implies that no set of the restricted partition induces a graph in Π. Hence if the input is a yes-instance, then the output instance is as well. The reverse direction is given by Lemma 4.2, which concludes the proof. □

Table 4.3: Problems that admit polynomial kernels when parameterized by the size of a given vertex cover, by applying Theorem 4.3.

Partition into q	Forbidden property Π	c_Π
Independent Sets	$\{K_2\}$	1
Cliques	$2 \cdot K_1$	1
Bipartite Graphs	Graphs with an odd cycle	2
Chordal Graphs	Graphs with a chordless cycle	3
$\mathcal{F}$-Minor-Free Graphs	Graphs with an $H \in \mathcal{F}$-minor	$\max_{H \in \mathcal{F}} \Delta(H)$
Planar Graphs	Graphs with a K_5 or $K_{3,3}$ minor	4
Forests	Graphs with a cycle	2

The theorem has consequences for a multitude of graph partitioning problems; a sample is presented in Table 4.3. Observe that countless other problems such as Partition into q Distance-Hereditary Graphs can be captured by the theorem, by using Proposition 4.3 to find new graph properties characterized by few adjacencies.

Corollary 4.3. *All problems in Table 4.3 fit into the framework of Theorem 4.3 and admit polynomial kernels parameterized by the size of a given vertex cover.*

Proof. Since the graph properties Π needed to establish the claims in the table were also used in Corollary 4.1, and the preconditions for Theorem 4.1 are stronger than the preconditions to the current theorem, the proofs given there also apply to this case. The table already lists the relevant choice of Π and the resulting c_Π needed to apply Theorem 4.3. For completeness we state the corresponding choice of polynomial $p(n)$, and the resulting size bounds.

Partition into q Independent Sets. Since the forbidden family Π is finite and contains only a single graph on two vertices, $c_\Pi = 1$ and $p(n) = 2$ suffices. We obtain a kernel with $\mathcal{O}(|X|^q)$ vertices, which may also be seen as a kernel for q-Coloring parameterized by vertex cover.

Partition into q Cliques. As in the previous case, the forbidden family consists of a single two-vertex graph resulting in a kernel with $\mathcal{O}(|X|^q)$ vertices.

Partition into q Bipartite Graphs. The graphs with an odd cycle are characterized by $c_\Pi = 2$ adjacencies. The number of vertices in vertex-minimal graphs in this family is at most twice the vertex cover number, so $p(n) = 2n$ suffices. The resulting kernel size is $\mathcal{O}(|X|^{2q+1})$ vertices.

Partition into q Chordal Graphs. The forbidden family is characterized by $c_\Pi = 3$ adjacencies and the polynomial $p(n) = 2n$ suffices, resulting in a kernel with $\mathcal{O}(|X|^{3q+1})$ vertices.

Partition into q $\mathcal{F}$-Minor-Free Graphs. As shown in the proof of Corollary 4.1 the forbidden family is characterized by $c_\Pi = \max_{H \in \mathcal{F}} \Delta(H)$ adjacencies and the polynomial can be taken to be a linear function whose coefficient depends on $\mathcal{F}$. We obtain a kernel with $\mathcal{O}(|X|^{q \cdot \Delta + 1})$ vertices, where $\Delta = \max_{H \in \mathcal{F}} \Delta(H)$.

As the last two problems are special cases of the previous item (with $\mathcal{F} = \{K_5, K_{3,3}\}$ resp. $\mathcal{F} = \{K_3\}$), this directly shows that we obtain a kernel with $\mathcal{O}(|X|^{4q+1})$ and $\mathcal{O}(|X|^{2q+1})$ vertices for the planar and forest partitioning problems, respectively. $\square$

As mentioned in the introduction, Theorem 4.3 is a strong generalization of the known kernel with $\mathcal{O}(|X|^q)$ vertices for q-Coloring parameterized by vertex cover [147, Corollary 1]. Despite the generality of Theorem 4.3, the size of the q-Coloring kernel obtained through Theorem 4.3 matches that of the q-Coloring kernel given earlier up to constant factors. We will prove in Chapter 7 (Theorem 7.4) that for any $q \geq 4$ and $\varepsilon > 0$, q-Coloring parameterized by vertex cover does not have kernels of bitsize $\mathcal{O}(|X|^{q-1-\varepsilon})$ unless NP $\subseteq$ coNP/poly. This shows that in the kernel size bound of Theorem 4.3, the appearance of q in the exponent is unavoidable.

Other partitioning problems that were listed by Garey and Johnson include Partition into q Dominating Sets [117, GT3] (also known as Domatic Number), Partition into q Hamiltonian Subgraphs [117, GT13], and Partition into q Perfect Matchings [117, GT16]. These problems cannot be expressed in our framework. The last two have trivial polynomial-size kernels parameterized by vertex cover, as one may easily verify that the size of all yes-instances is bounded polynomially in their vertex cover number. A polynomial kernel can therefore be obtained by simply rejecting instances that are too large. The problem Partition into q Dominating Sets may be interesting for further study.

4.7 Concluding Remarks

We have studied the existence of polynomial kernels for graph problems parameterized by vertex cover. The general theorems we presented unify and extend known positive results for many problems, and the characterization in terms of forbidden or desired induced subgraphs from a class characterized by few adjacencies gives a common explanation for the results obtained earlier.

An obvious direction for further work is to find even more general kernelization theorems that can also encompass the known positive results for problems like Treewidth [vc] [35], Pathwidth [vc] [32], and Clique Minor Test [vc] [109]. There are also various problems for which the kernelization complexity parameterized by vertex cover is still open; among these are Perfect Deletion, Interval Deletion, Bandwidth, and Orientable Genus. One may also investigate whether Theorem 4.1 has an analogue for edge-deletion problems.

In light of the parameter ecology program discussed in Section 2.2 it is natural to ask whether the general kernelization theorems obtained in this chapter can be transferred to smaller parameters than the vertex cover number. As this parameter measures the vertex-deletion distance to a graph of treewidth zero, an obvious next step would be parameterization by the feedback vertex number — the vertex-deletion distance to a graph of treewidth one. Alas, this seems difficult. While Vertex Cover and Odd Cycle Transversal admit polynomial kernels for this parameter [146, 148], the kernelization schemes are rather involved and lack any similarity. For the Long Path problem, the existence of a polynomial kernel parameterized by feedback vertex number is still open. In the case of 3-Coloring [147] and Disjoint Paths [43] we even know that no polynomial kernel exists for the parameterization by feedback vertex number (unless NP $\subseteq$ coNP/poly). Hence it seems that a better understanding of polynomial kernelizability for parameterizations by feedback vertex number is needed before attempting to capture the phenomenon by general theorems.

Part III

Case Studies in the Parameter Hierarchy

Vertex cover is clearly the most popular parameterized problem of all time, and it would be extremely sad if fundamental investigation of its parameterized complexity should ever cease to produce new insights.

—Michael R. Fellows, 2005

5 Vertex Cover

This chapter gives the first of a series of case studies. In each study we focus on a single problem and try to pinpoint the boundaries of tractability in the parameter hierarchy of Section 2.3 by proving upper and lower bounds. To give the relevant background for our results, each chapter opens with a picture of the complexity landscape for the problem under consideration. After describing the properties of a problem under different parameterizations, we proceed by presenting our own contributions. We start with parameterized complexity's favorite problem, Vertex Cover.

In Section 5.1 we survey the advances in preprocessing and algorithmics for this problem. Section 5.2 is devoted to one of the first results in the area of kernelization for nonstandard Vertex Cover parameterizations: a cubic-vertex kernel parameterized by the feedback vertex number. The kernelization employs several novel reduction rules, of which the effectiveness is proven through an extremal argument from graph theory. The relevant parameter expresses the deletion distance of the input graph to a *forest*. In Section 5.3 we consider the distance to an *outerplanar* graph as the parameter. While outerplanar graphs can be considered a mild generalization of forests — they are sparse, and have treewidth two — we prove that the resulting parameterization does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.

The result of Section 5.2 originates from the paper "Vertex Cover Kernelization Revisited: Upper and Lower Bounds for a Refined Parameter" by Bart M. P. Jansen and Hans L. Bodlaender (28th International Symposium on Theoretical Aspects of Computer Science [146]). The revised presentation here is based on the journal version that appeared in Theory of Computing Systems [145]. The introduction contains parts of the survey by Michael R. Fellows, Bart M. P. Jansen, and Frances A. Rosamond [101]. The lower bound of Section 5.3 has not been published before.

5.1 Advances in Kernelizing Vertex Cover

The analysis of different facets of the Vertex Cover problem has yielded numerous important advances in parameterized algorithmics that have subsequently been applied successfully in other areas. In this section we describe the key results in vertex cover research that are relevant to this thesis. We begin by considering the relationships between the various parameterizations, in Section 5.1.1. These relationships are used in Section 5.1.2 to draw the lines of tractability implied by the current state of the art.

5.1.1 Interplay Between Parameterizations

The parameter hierarchy of Fig. 2.1 gives a clear picture of the relationships between the most popular structural parameterizations. There are some interesting graph parameters whose relevance appears to be limited to the Vertex Cover problem, which were therefore not discussed before. We introduce, motivate, and analyze these nonstandard parameterizations in this section, to find their correct place in the parameter hierarchy.

Some interesting parameterizations of Vertex Cover are based on the idea of parameterizing *above or below tight bounds*. Recall that for a graph G, we denote the cardinality of a maximum matching by $\mu(G)$ and the vertex cover number by $\textsc{vc}(G)$. If M is a matching, then a vertex cover contains at least one endpoint of every edge in M. Hence all graphs G satisfy $\textsc{vc}(G) \geq \mu(G)$, and it is easy to find graphs where equality holds. Rather than using the requested size of the vertex cover as the parameter, one may therefore attempt a parameterization by the excess of the size of a vertex cover over the matching number: one could ask whether G has a vertex cover of size $\mu(G) + k$, parameterized by k. To place the parameterization by this “above guarantee” value k in our hierarchy, we cannot rely on some input value k: we consider the corresponding structural measure $\textsc{vc}(G) - \mu(G)$, which will give us the right feeling for the strength of the parameterization.

We can also parameterize *below* a tight bound. Let $\gamma'(G)$ be the minimum cardinality[1] of a maximal matching in G. For any maximal matching $M \subseteq E(G)$, the $2|M|$ endpoints of the matching edges form a vertex cover — else the matching would not be maximal — hence $\textsc{vc}(G) \leq 2\gamma'(G) \leq 2|M|$. Given a maximal matching M in a graph G, one can ask whether G has a vertex cover of size $2|M| - k$, parameterizing below the guaranteed upper bound. To relate this parameterization to others, we again turn to a structural measure of graphs. In this case the relevant measure is $2\gamma'(G) - \textsc{vc}(G)$, and since $\gamma'(G) \leq \mu(G) \leq \textsc{vc}(G)$ it satisfies $2\gamma'(G) - \textsc{vc}(G) \leq \textsc{vc}(G)$. It is therefore a smaller measure than the vertex cover number.

An important class of Vertex Cover parameterizations arises from the vertex-deletion distance to graph classes in which the problem is polynomial-time

[1] This value coincides with the edge domination number, which explains the notation.

solvable. Cai [48] was the first to observe that, given a graph G, a vertex subset S, and an algorithm that solves Vertex Cover in polynomial time on $G - S$ and all its induced subgraphs, it is possible to solve Vertex Cover on G in FPT-time parameterized by $|S|$. As a minimum vertex cover of a perfect graph can be computed in polynomial time using the algorithm of Grötschel, Lovász and Schrijver [122, Chapter 9], and because perfect graphs are hereditary, this shows that Vertex Cover is FPT parameterized by the size of a given modulator to perfect graphs. This naturally implies the same FPT status for parameterizations by the size of modulators to subclasses of perfect graphs. For example, the problem is FPT when the parameter measures the size of a given vertex set whose deletion leaves a König graph. Since all bipartite graphs are König graphs, the vertex-deletion distance to a König graph does not exceed the odd cycle transversal number. It also has an interesting relationship to one of the "above guarantee" parameterizations discussed above.

Lemma 5.1 ([180, Lemma 12])**.** *Let G be a graph with deletion distance k to a König graph. Then $\textsc{vc}(G) - \mu(G) \leq k \leq 2(\textsc{vc}(G) - \mu(G))$.*

The next parameterization we consider — one of the smallest in the parameter hierarchy for which positive results are known — also takes the form of an "above lower bound" question. It revolves around the linear-programming relaxation of Vertex Cover: the linear program with non-negative real-valued variables w_v for $v \in V(G)$ that seeks to minimize $\sum_{v \in V(G)} w(v)$, with a constraint $w_u + w_v \geq 1$ for every edge $\{u, v\} \in E(G)$. Feasible solutions to this LP are also called *fractional vertex covers*, and the minimum value of a valid solution is denoted by $\textsc{lp}(G)$. By the nature of relaxation, all graphs satisfy $\textsc{lp}(G) \leq \textsc{vc}(G)$; the difference can be arbitrary. As before, we may ask whether G has a vertex cover of size $\textsc{lp}(G) + k$, parameterized by k. This boils down to the study of the corresponding structural measure $\textsc{vc}(G) - \textsc{lp}(G)$. It is related to several parameters we saw before, as was first proven by Kratsch and Wahlström in their recent work [167]. In particular [101, Lemma 3], any graph G satisfies $\textsc{vc}(G) - \mu(G) \leq 2(\textsc{vc}(G) - \textsc{lp}(G))$. This bound is best-possible, which is witnessed by graphs that are disjoint unions of triangles. In the other direction it is easy to see that $\textsc{vc}(G) - \textsc{lp}(G) \leq \textsc{vc}(G) - \mu(G)$, as the value of the LP-relaxation must be at least the size of a maximum matching: for every edge in the matching, the two endpoints of the edge together have value at least one.

Before we start the discussion of Vertex Cover results, there is one more "above guarantee" parameterization to consider. Let $\Delta(G)$ be the maximum degree of a vertex in G. For readability, we will write Δ instead of $\Delta(G)$, and m instead of $|E(G)|$ in the remainder. Since a single vertex can cover at most Δ edges, any vertex cover has size at least m/Δ, and there are graphs where this size is sufficient. So we may ask whether G has a vertex cover of size $m/\Delta + k$ parameterized by k, giving rise to the structural measure $\textsc{vc}(G) - m/\Delta$. It is not difficult to show how this measure relates to the excess of an integral vertex cover over a fractional vertex cover.

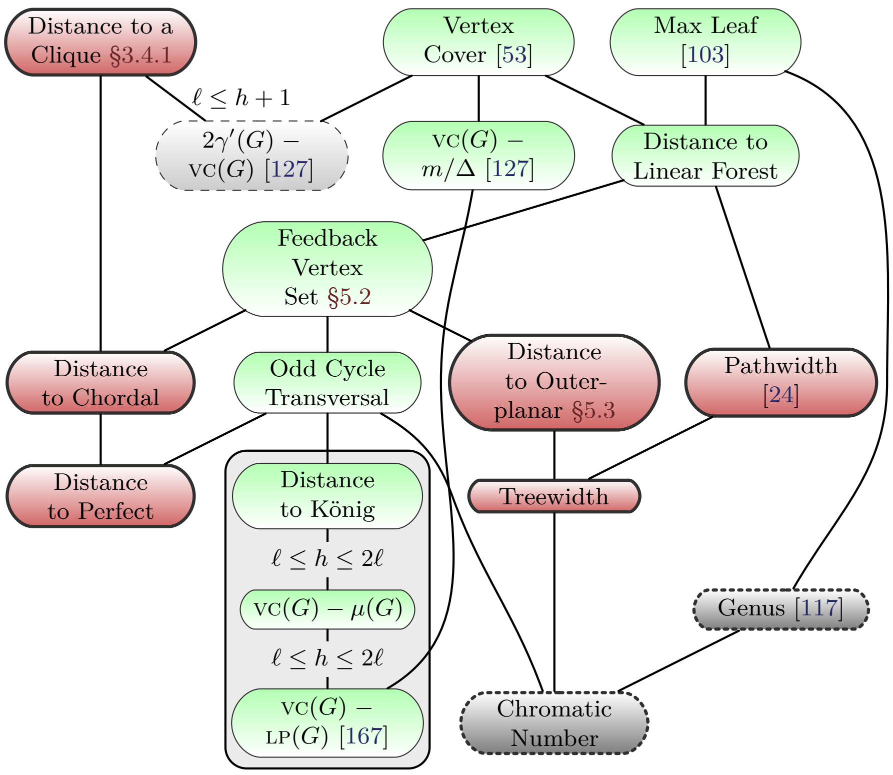


Figure 5.1: Complexity overview for various parameterizations of Vertex Cover, assuming suitable formalizations. For deletion distance parameters we assume that a modulator is given along with the input. The shading indicates that a parameterization is either para-NP-complete ▨, $W[1]$-hard but contained in XP ⬜, FPT but (conditionally) lacking a polynomial kernel 🟥, or FPT with a polynomial (randomized) kernel 🟩. When a line between parameters is labeled by a bound, the lower-drawn parameter is represented by ℓ and the higher-drawn parameter by h. Relationships between parameters present in Fig. 2.1 are not repeated. The three parameters grouped at the bottom are equivalent up to constant factors.

Proposition 5.1. *Every graph G satisfies* $\textsc{vc}(G) - \textsc{lp}(G) \leq \textsc{vc}(G) - m/\Delta$.

Proof. Fix some fractional vertex cover, and observe that $\Delta \cdot \textsc{lp}(G) = \Delta \cdot \sum_{v \in V(G)} w_v \geq \sum_{\{u,v\} \in E(G)} w_u + w_v \geq \sum_{\{u,v\} \in E(G)} 1 = m$. This implies the claim by simple formula manipulation. □

We have now established the relationships needed to organize the parameters relevant to this section into a hierarchy: it is given in Fig. 5.1. We proceed by interpreting recent results for Vertex Cover in the light of these interactions.

5.1.2 Complexity Boundaries for Vertex Cover

To make sense of the multitude of results for this thoroughly investigated problem, we describe the advances one parameter at a time. It has been known for decades that the natural parameterization Vertex Cover [sol] is FPT [46], the current-best run-time being $\mathcal{O}(1.2738^k + kn)$ which was obtained after a series of improvements [53, 54, 87, 190, 192]. Vertex Cover [sol] admits a kernel with $2k$ vertices, which can be obtained using the Nemhauser-Trotter theorem [53, 187], or by crown reductions [2, 58, 59]. The existence of smaller kernels has been the subject of repeated study. Chen et al. [52, Corollary 3.13] showed that for every $\varepsilon > 0$, no polynomial-time algorithm can reduce instances (G, k) of Vertex Cover [sol] to equivalent instances (G', k') on at most $(2 - \varepsilon)k$ vertices such that G' is a subgraph of G and $k' \leq k$, unless P = NP. A breakthrough result of Dell and van Melkebeek [80] gave evidence that the number of edges in the kernel (which is quadratic in the number of vertices) cannot be significantly improved either: unless NP $\subseteq$ coNP/poly, there is no kernel for Vertex Cover [sol] that reduces an instance to an equivalent one of bitsize $\mathcal{O}(k^{2-\varepsilon})$ for any $\varepsilon > 0$. It is possible to shave a little bit off the kernel size, though: Soleimanfallah and Yeo [205] showed that for every constant c there exists a kernel with $2k - c$ vertices. This is mostly of theoretical interest however, since the running time of the kernelization algorithm is exponential in c. Their approach was extended by Lampis [171], who obtained a kernel with $2k - c \log k$ vertices for any c by exploiting a connection to Almost 2-SAT.

The connection to Almost 2-SAT was first discovered by Mishra et al. [178–180] when studying the influence of the deletion distance to König graphs on the complexity of Vertex Cover: they established the FPT-equivalence of Vertex Cover Above $\mu(G)$ and Almost 2-SAT. Razgon and O'Sullivan later found an FPT-algorithm for the latter problem, thus placing Vertex Cover Above $\mu(G)$ in FPT. Successive improvements to the running time quickly followed each other [76, 195], the current record-holder being an algorithm by Cygan et al., which runs in $\mathcal{O}(4^k n^{\mathcal{O}(1)})$ time. Through their seminal use of matroid techniques for kernelization, Kratsch and Wahlström obtained a *randomized* polynomial kernel for Vertex Cover Above $\mu(G)$ [167]. By relating the measure $\text{vc}(G) - \text{lp}(G)$ to $\text{vc}(G) - \mu(G)$, they showed that Vertex Cover Above $\text{lp}(G)$ also admits a randomized polynomial kernel. This is currently one of the strongest parameterizations known to be tractable. The existence of a randomized polynomial kernel came as a surprise: we often find that the boundary for polynomial kernelizability lies higher in the hierarchy than the threshold for solvability in FPT-time.

A separation between the smallest parameterization admitting a polynomial kernel and the smallest parameterization admitting an FPT-algorithm is witnessed, for example, when considering more traditional measures of graph structure such as treewidth. As Vertex Cover can be formulated in Monadic Second Order Logic (even without using quantifications over edge sets), various forms of Courcelle's theorem [66] show that it is FPT parameterized by treewidth [37] and when parameterized by even smaller parameters such as cliquewidth. These *width measures* (parameters that do not increase when making multiple disjoint copies of a graph) generally do not yield parameterizations that admit polynomial kernels, as was indeed proven to be the case for Vertex Cover parameterized by treewidth [24]; the result is conditioned on NP $\not\subseteq$ coNP/poly, and also applies to pathwidth and cliquewidth.

Another branch of parameterizations can be interpreted as *parameterizing away from triviality* (cf. [48, 188, 189]). If the input graph is "close" to a graph class $\mathcal{F}$ in which Vertex Cover is polynomial-time solvable, then one expects to be able to exploit this connection to solve the problem more efficiently. Formalizing closeness as the vertex-deletion distance to $\mathcal{F}$, the proof technique of Cai [48] sketched earlier indeed shows that if $\mathcal{F}$ is hereditary and a modulator to $\mathcal{F}$ of size k is given, then the problem can be solved in $\mathcal{O}(2^k n^{\mathcal{O}(1)})$ time. As the feedback vertex number measures the deletion distance to a forest in which the problem is easily solvable, this implies Vertex Cover is FPT parameterized by the size of a feedback vertex set. The kernelization for the corresponding parameterization [145] that will be developed in Section 5.2 shows how this distance to triviality can be exploited in preprocessing schemes.

There is a remarkable gap between polynomial kernelizability and fixed-parameter tractability for such parameterizations away from triviality. If $\mathcal{F}$ contains all cliques, then Vertex Cover parameterized by distance from $\mathcal{F}$ does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly (Corollary 3.3, as originally proven by Bodlaender et al. [29]), while we saw before that even the distance to the much richer class of perfect graphs yields FPT-algorithms. The difficulty of data reduction for these parameters cannot just be explained by the density of the graphs in $\mathcal{F}$: considering sparser and simple classes $\mathcal{F}$ such as the graphs of treewidth two [73], or even outerplanar graphs (Section 5.3), the corresponding parameterizations admit no polynomial kernels unless NP $\subseteq$ coNP/poly .

With regards to parameterizations above and below tight bounds, Gutin et al. [127] proved that it is $W[1]$-hard to decide, given a maximal matching M in graph G and an integer k, whether G has a vertex cover of size $2|M| - k$, parameterized by k. In the same paper they posed the existence of an FPT-algorithm for the following parameterization as an open problem:

Vertex Cover Above m/Δ
Input: Graph G with m edges and maximum degree Δ, and an integer k.
Parameter: k.
Question: Does G have a vertex cover of size at most $m/\Delta + k$?

It follows [101, Theorem 3] from the relationships given in the previous section that this parameterization is FPT, and in fact admits a randomized polynomial kernel by applying the results of Kratsch and Wahlström [167]. This concludes our discussion of the key results in Vertex Cover research. By propagating the cited results through the hierarchy as described in Section 2.3, we arrive at the complexity picture of Fig. 5.1.

5.2 Cubic Kernel for Vertex Cover Parameterized by Feedback Vertex Set

In this section we present the details of a cubic-vertex kernel for a refined structural parameterization of VERTEX COVER: we use the feedback vertex number $\textsc{fvs}(G)$ as the parameter. Since every vertex cover is also a feedback vertex set we find that $\textsc{fvs}(G) \leq \textsc{vc}(G)$. This shows that the feedback vertex number of a graph is a *structurally smaller* parameter than the vertex cover number: there are trees with arbitrarily large values of $\textsc{vc}(G)$ for which $\textsc{fvs}(G) = 0$. Observe that for difficult instances of VERTEX COVER we have $\ell \in \Theta(\textsc{vc}(G))$ since the use of the 2-approximation algorithm immediately solves instances where $\ell > 2\,\textsc{vc}(G)$ or $\ell < \textsc{vc}(G)/2$. Therefore we call our parameter "refined" since it is structurally smaller than the standard parameter for the VERTEX COVER problem. Although Kratsch and Wahlström later obtained a randomized polynomial kernel for the — even smaller — parameterization above the value of the LP-relaxation, the kernelization presented here remains the strongest deterministic polynomial kernel known for the problem. For completeness we formally define the problem under consideration.

> VERTEX COVER [FVS]
> **Input:** A graph G, a feedback vertex set $X \subseteq V(G)$ such that $G - X$ is a forest, and an integer ℓ.
> **Parameter:** The size $k := |X|$ of the feedback vertex set.
> **Question:** Does G have a vertex cover of size at most ℓ?

We prove that VERTEX COVER [FVS] has a kernel in which the number of vertices is bounded by $\min(2\ell, 2|X|+28|X|^2+56|X|^3)$ that can be computed in $\mathcal{O}(\sqrt{n}m+n^{5/3})$ time. The kernel size is at least as small as the kernels based on the Nemhauser-Trotter theorem or crown reduction, but for graphs with small feedback vertex sets our bound can be expected to be significantly smaller.

We also consider the problem INDEPENDENT SET [FVS] that is similarly defined: the difference is that we now ask whether G has an independent set of the requested size, instead of a vertex cover. Throughout the chapter ℓ will always represent the total size of the set we are looking for; depending on the context this is either a vertex cover or an independent set. An instance (G, X, ℓ) of VERTEX COVER [FVS] is equivalent to an instance $(G, X, |V(G)| - \ell)$ of INDEPENDENT SET [FVS] that has *the same* parameter value $|X|$ and therefore the two problems are equivalent from a parameterized complexity and kernelization standpoint. For the ease of presentation, we first develop a kernel for INDEPENDENT SET [FVS]. Using the correspondence between the two problems, this will immediately yield a kernel for VERTEX COVER [FVS] of the same size.

The remainder of the section is organized as follows. We give some preliminaries relating to maximum independent sets in forests in Section 5.2.1. We then show in Section 5.2.2 that a single application of the Nemhauser-Trotter decomposition theorem [187], used for kernelization of the vertex cover problem by Chen et al. [53], allows us to restrict our attention to instances of Vertex Cover [fvs] where the forest $G - X$ has a perfect matching. This will greatly simplify the analysis of the kernel size as compared to the extended abstract [146] where we worked with arbitrary forests $G - X$. In Section 5.2.3 we introduce a set of reduction rules and prove they are correct. Afterwards we analyze the structure of the resulting reduced instances, in Section 5.2.4. This analysis focuses on *conflict structures*. An important ingredient in the proof of the bound on the kernel size is formed by a purely graph-theoretic extremal argument, which is developed in Section 5.2.5. It shows that many conflict structures exist in reduced instances. As the last step we discuss the running time of a possible implementation of the reduction rules, and tie all ingredients together into a kernelization algorithm in Section 5.2.6.

5.2.1 Independent Sets, Forests, and Matchings

Throughout this chapter we abbreviate maximum independent set as MIS and feedback vertex set as FVS. The following properties will be useful at various points in our proofs. Recall that a leaf in an arbitrary graph is a vertex of degree at most one.

König's Theorem. (cf. [203, Theorem 16.2]). *For every bipartite graph G, the size of a minimum vertex cover equals the number of edges in a maximum matching.*

The following observations can be made by applying König's theorem to forests with bipartite matchings.

Observation 5.1. *Let F be a forest with a perfect matching $M \subseteq E(F)$. The following hold:*

(i) $|V(F)| = 2|M|$ *and* $\textsc{vc}(F) = \alpha(F) = |M|$*, since F is bipartite.*
(ii) Every vertex of F is adjacent to at most one leaf.
(iii) If v is a leaf of F, then v has a unique neighbor $u \in V(F)$ and $\{u, v\} \in M$.

Observation 5.2. *If G' is a vertex-induced subgraph of graph G, then $\alpha(G) \geq \alpha(G')$.*

Observation 5.3. *If v is a leaf in G, then there is a MIS for G that contains v.*

5.2.2 Reducing to a Forest with a Perfect Matching

The first step in the kernelization is to reduce to instances in which the forest $G-X$ has a perfect matching. The restricted form of the graph $G - X$ will be convenient when performing our analysis. To facilitate the reduction we restate the Nemhauser-Trotter theorem here in terms of independent sets.

Proposition 5.2 ([53, Proposition 2.1])**.** *There is an $\mathcal{O}(\sqrt{n}m)$-time algorithm that, given a graph G with n vertices and m edges, constructs disjoint subsets $C_0, V_0 \subseteq V(G)$ such that:*

1. *if I is a maximum independent set in $G[V_0]$ then $I \cup J$ is a maximum independent set in G, with $J := V(G) \setminus (C_0 \cup V_0)$, and*
2. *$\alpha(G[V_0]) \leq |V_0|/2$.*

We use this decomposition to show that, after identifying a set of vertices that can be in any maximum independent set of G, there is a small (in terms of $|X|$) set $I \subseteq V(G) \setminus X$ that we can add to X, such that the forest $G - (X \cup I)$ has a perfect matching.

Lemma 5.2. *Let (G, X, ℓ) be an instance of* Independent Set [fvs]*. In $\mathcal{O}(\sqrt{n}m)$ time one can compute an equivalent instance (G', X', ℓ') such that:*

1. *$G' - X'$ has a perfect matching,*
2. *$|X'| \leq 2|X|$, and*
3. *$\ell' \leq \ell$.*

Proof. Given an instance (G, X, ℓ) of Independent Set [fvs], use the algorithm of Proposition 5.2 to compute the two sets $C_0, V_0 \subseteq V(G)$. Now set $G' := G[V_0]$, let $\hat{X} := X \cap V_0$, and $\ell' := \ell - (|V(G)| - |V_0| - |C_0|)$. The proposition ensures that the instances (G, X, ℓ) and $(G', \hat{X}, \ell')$ are equivalent, and it is easy to see that $G' - \hat{X}$ is a forest since it is a subgraph of $G - X$. The last property of the proposition ensures that $\alpha(G') \leq |V(G')|/2$.

Now, we compute a maximum matching M of the forest $G' - \hat{X}$, which can be done in $\mathcal{O}(\sqrt{n}m)$ time using the Hopcroft-Karp algorithm [138]. Note that $|V(G' - \hat{X})| = 2|M| + |I|$ where I is the set of vertices not covered by the matching. As $G' - \hat{X}$ is a forest, and hence bipartite, a minimum vertex cover for $G' - \hat{X}$ has size $|M|$ (by König's Theorem) and maximum independent sets have size $|V(G' - \hat{X})| - |M| = |M| + |I|$. Comparing $\alpha(G' - \hat{X}) = |M| + |I|$ with the upper bound of $\alpha(G' - \hat{X}) \leq \alpha(G') \leq \frac{1}{2}|V(G')|$ we get the following:

$$\begin{aligned} \alpha(G' - \hat{X}) &\leq |V(G')|/2 \\ |M| + |I| &\leq (|\hat{X}| + 2|M| + |I|)/2 \\ |I| &\leq |\hat{X}|. \end{aligned}$$

Thus, letting $X' := \hat{X} \cup I$, we know that $G' - X'$ is a forest, and that it has a perfect matching (namely M). Clearly $|X'| \leq 2|\hat{X}| \leq 2|X|$. We return the instance (G', X', ℓ'). □

The property that the forest $G - X$ of an instance of Independent Set [fvs] has a perfect matching is so useful that it warrants a special name.

Definition 5.1. An instance (G, X, ℓ) of Independent Set [fvs] is called *clean* if the forest $G - X$ has a perfect matching.

We apply Lemma 5.2 once at the start of our kernelization, and work on the resulting clean instance of the problem. The reduction rules we apply to shrink the instance maintain the property that the forest has a perfect matching.

5.2.3 Reduction Rules for Clean Instances

Consider a clean instance (G, X, ℓ) of Independent Set [fvs], which asks whether a graph G with the FVS X has an independent set of size ℓ. Throughout this section $F := G - X$ denotes the forest obtained by deleting the vertices in X from G. Recall that $G - X$ has a perfect matching by the assumption that the instance is clean. To formulate our reduction rules we use the following notion.

Definition 5.2. Let (G, X, ℓ) be an instance of Independent Set [fvs]. Define $\mathcal{X} := \{Y \subseteq X \mid Y \text{ is independent in } G \text{ and } 0 < |Y| \leq 2\}$ as the collection of *chunks* of X.

The chunks corresponding to an instance are the nonempty sets in $\binom{X}{\leq 2}$ that could be part of an independent set in G. Our first two reduction rules get rid of chunks when we can effectively determine that there is a MIS that does not contain them. We get rid of a chunk by either deleting it (when it is a single vertex) or by adding an edge (if a chunk consists of two nonadjacent vertices). Observe that after adding the edge $\{u, v\}$ for $u, v \in X$ the pair $\{u, v\}$ is no longer independent, and therefore no longer counts as a chunk.

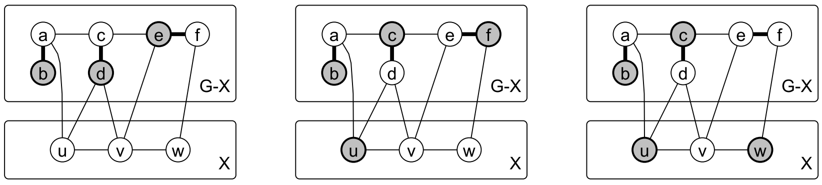
We rely on the fact that, when given an independent subset $X' \subseteq X$ of the feedback vertices, we can efficiently compute a largest independent set I in G that satisfies $I \cap X = X'$: since such a set intersects X exactly in X', and since it cannot use any neighbors of X' the maximum size is $|X'| + \alpha(F - N_G(X'))$ and this is polynomial-time computable since $F - N_G(X')$ is a forest. The following notion allows us to assess which chunks might occur in a MIS of G.

Definition 5.3. The number of *conflicts* $\textsc{Conf}_{F'}(X')$ induced by a subset $X' \subseteq X$ on a subforest $F' \subseteq F \subseteq G$ is defined as $\textsc{Conf}_{F'}(X') := \alpha(F') - \alpha(F' - N_G(X'))$.

The term $\textsc{Conf}_{F'}(X')$ can be interpreted as follows. Choosing vertices from X' in an independent set will prevent all their neighbors in the subforest F' from being part of the same independent set; hence if we fix some choice of vertices in X', then the number of vertices from F' we can add to this set (while maintaining independence) might be smaller than the independence number of F'. The term $\textsc{Conf}_{F'}(X')$ measures the difference between the two: informally it is the price we pay in the forest F' for choosing the vertices X' in the independent set (see Fig. 5.2). We can now state the first two reduction rules.

Reduction Rule 5.1. If there is a vertex $v \in X$ such that $\textsc{Conf}_F(\{v\}) \geq |X|$, then delete v from the graph G and from the set X.

Reduction Rule 5.2. If there are distinct vertices $u, v \in X$ with $\{u, v\} \notin E(G)$ for which $\textsc{Conf}_F(\{u, v\}) \geq |X|$, then add the edge $\{u, v\}$ to G.



(a) $\alpha(G - X) = 3$. (b) $\textsc{Conf}_F(\{u\}) = 0$. (c) $\textsc{Conf}_F(\{u, w\}) = 1$.

Figure 5.2: Illustration of the first three definitions (5.1–5.3). A clean instance (G, X, ℓ) is shown in three different states, with X visualized in the bottom container and the forest $F := G - X$ visualized in the top container. The perfect matching in F is indicated by thick edges. The set of *chunks* of this instance is $\mathcal{X} = \{\{u\}, \{v\}, \{w\}, \{u, w\}\}$. (a) MIS in F showing that $\alpha(F) = 3$. (b) The drawn independent set does not contain any neighbors of u and contains $3 = \alpha(F)$ vertices from F; hence $\alpha(F - N_G(u)) = \alpha(F) = 3$, implying that $\textsc{Conf}_F(\{u\}) = 0$. (c) Choosing vertices $\{u, w\}$ in an independent set prevents us from adding three vertices from F to the independent set; we can add only two, without violating independence. The difference $(3 - 2 = 1)$ is the number of conflicts induced by the pair: $\textsc{Conf}_F(\{u, w\}) = 1$.

Since these two rules only affect the graph induced by X, they do not change the fact that forest F has a perfect matching. Correctness of the rules can be established from the following lemma.

Lemma 5.3. *If $X' \subseteq X$ is a subset of feedback vertices with $\textsc{Conf}_F(X') \geq |X|$ then there is a MIS for G that does not contain all vertices of X'.*

Proof. Let $I \subseteq V(G)$ be an independent set containing all vertices of X'. We will prove that there is an independent set I' that is disjoint from X' with $|I'| \geq |I|$. Since $\textsc{Conf}_F(X') \geq |X|$ it follows by definition that $\alpha(F) - \alpha(F - N_G(X')) \geq |X|$; since I cannot contain any neighbors of vertices in X' we know that $|I \cap V(F)| \leq \alpha(F - N_G(X'))$, and since $V(G) = X \cup V(F)$ we have $|I| \leq |X| + \alpha(F - N_G(X')) \leq \alpha(F)$. Hence the maximum independent set for F, which does not contain any vertices of X', is at least as large as I; this proves that for every independent set containing X' there is another independent set that is at least as large and avoids the vertices of X'. Therefore there is a MIS for G avoiding at least one vertex of X'. □

The next rule is used to remove trees from the forest F when those trees do not interact with any of the chunks in X.

Reduction Rule 5.3. If F contains a connected component T (which must be a tree) such that for all chunks $Y \in \mathcal{X}$ it holds that $\textsc{Conf}_T(Y) = 0$, then delete T from graph G and decrease ℓ by $\alpha(T)$.

Since the rule deletes an entire tree from the forest F, it ensures that the remainder of the forest will have a perfect matching. To prove the correctness of Rule 5.3 we need the following lemma.

Lemma 5.4. *Let T be a connected component of F and let $X_I \subseteq X$ be an independent set in G. If $\textsc{Conf}_T(X_I) > 0$, then there is a set $X' \subseteq X_I$ with $|X'| \leq 2$ such that $\textsc{Conf}_T(X') > 0$.*

Proof. Assume the conditions in the lemma hold. Recall that throughout this section we work on a *clean* instance. Accordingly, let M be a perfect matching on T, which exists since the forest F has a perfect matching. We will try to construct a MIS I for T that does not use any vertices in $N_G(X_I)$; this must then also be a MIS for $T - N_G(X_I)$ of the same size. By the assumption that $\textsc{Conf}_T(X_I) > 0$ any independent set in T must use at least one vertex in $N_G(X_I)$ in order to be maximum, hence our construction procedure must fail somewhere; the place where it fails will provide us with a set X' as required by the statement of the lemma.

Construction of a MIS. It is easy to see that a MIS of a tree with a perfect matching contains exactly one vertex from each matching edge. We now start building our independent set I for T that avoids vertices in $N_G(X_I)$. To ensure I becomes a MIS for T, we need to add one endpoint of each edge in the matching M. If there is a vertex v in T such that $N_T(v) = \{u\}$ and $N_G(v) \cap X_I = \emptyset$, then the edge $\{v, u\}$ must be in the matching M (since M is a perfect matching and there are no other edges incident on v). Because we must choose one of $\{u, v\}$ in a MIS for T, and by Observation 5.3 choosing a degree-1 vertex will never conflict with choices that are made later on, we can add v to our independent set I while respecting the invariant that no vertex in I is adjacent in G to a vertex in X_I. Since we have then chosen one endpoint of the matching edge $\{u, v\}$ in I, we can delete u, v and their incident edges to obtain a smaller graph T' (which again contains a perfect submatching of M) in which we continue the process. As long as there is a vertex with degree one in T' that has no neighbors in X_I, we take it into I, delete it and its neighbor, and continue. If this process ends with an empty graph, then by our starting observation the set I must be a MIS for T, and since it does not use any vertices adjacent to X_I it must also be a MIS for $T - N_G(X_I)$; but this proves that $\alpha(T) = \alpha(T - N_G(X_I))$ which means $\textsc{Conf}_T(X_I) = 0$, which is a contradiction to the assumption at the start of the proof. So the process must end with a nonempty graph $T' \subseteq T$ such that vertices with degree one in T' are adjacent in G to a vertex in X_I and for which the matching M restricted to T' is a perfect matching on T'. We use this subgraph T' to obtain a set X' as desired.

Using T' to prove the claim. Consider a vertex v_0 in T' such that $\deg_{T'}(v_0) = 1$, and construct a path $P = (v_0, v_1, \ldots, v_{2p+1})$ by following edges of T' that are alternatingly in and out of the matching M, until arriving at a degree-1 vertex whose only neighbor was already visited. Since T' is acyclic, M restricted to T' is a perfect matching on T' and we start the process at a vertex of degree one, it is easy to verify that there is such a path P (there can be many; any arbitrary such path will suffice). Moreover, it easily follows that P contains an even number of vertices, that the first and last vertex on P have degree-1 in T' and that the edges $\{v_{2i}, v_{2i+1}\}$ must be in M for all $0 \leq i \leq p$. Since we assumed that all degree-1 vertices in T' are adjacent in G to X_I, there exist vertices $x_1, x_2 \in X$ such that $v_0 \in N_G(x_1)$ and $v_{2p+1} \in N_G(x_2)$. We now claim that $X' := \{x_1, x_2\}$ satisfies the requirements of the lemma, i.e., that $\textsc{Conf}_T(\{x_1, x_2\}) > 0$. The latter is witnessed by considering the path P in the original tree T. Any MIS for T that avoids $N_G(\{x_1, x_2\})$ must use one endpoint of the matched edge $\{v_0, v_1\}$, and since the choice of v_0 is blocked because v_0 is a neighbor to x_1, it must use v_1. But path P shows that v_1 is adjacent in T to v_2, and hence we cannot choose v_2 in the independent set. But since $\{v_2, v_3\}$ is again a matched edge, we must use one of its endpoints; hence we must use v_3. Repeating this argument shows that we must use vertex v_{2p+1} in a MIS for T if we cannot use v_0; but the use of v_{2p+1} is also not possible if we exclude $N_G(\{x_1, x_2\})$. Hence we cannot make a MIS for T without using vertices in $N_G(\{x_1, x_2\})$ which proves that $\alpha(T) > \alpha(T - N_G(\{x_1, x_2\}))$. By the definition of conflicts this proves that $\textsc{Conf}_T(X') > 0$ for $X' = \{x_1, x_2\}$, which concludes the proof. □

Using Lemma 5.4 we can prove the correctness of Rule 5.3. We remark that using a more involved argument based on a decomposition theorem describing independent sets in forests by Zito [213], it is possible to show that Lemma 5.4 holds even if F is a forest that does not admit a perfect matching. The argument can be found in an earlier presentation of this result [146, Lemma 4].

Lemma 5.5. *Rule 5.3 is correct: if T is a connected component in F such that for all chunks $Y \in \mathcal{X}$ it holds that $\textsc{Conf}_T(X') = 0$, then $\alpha(G) = \alpha(G - T) + \alpha(T)$.*

Proof. Assume the conditions in the lemma hold. Trivially $\alpha(G) \leq \alpha(G-T)+\alpha(T)$. To establish the lemma we only need to prove that $\alpha(G) \geq \alpha(G - T) + \alpha(T)$. We will do this by showing that any independent set I_{G-T} in $G - T$ can be transformed into an independent set of size at least $|I_{G-T}| + \alpha(T)$ in G. So consider such an independent set I_{G-T}, and let $X_I := I_{G-T} \cap X$ be the set of vertices that belong to both I_{G-T} and the feedback vertex set X. Suppose that $\alpha(T) > \alpha(T - N_G(X_I))$. Then by Lemma 5.4 there is a subset $Y \subseteq X_I$ with $|Y| \leq 2$ such that $\textsc{Conf}_T(Y) > 0$. Since X_I is an independent set, its subset Y would also be independent, and hence would be a chunk in $\mathcal{X}$. But by the preconditions to this lemma such a chunk Y does not exist. Therefore we have $\alpha(T) = \alpha(T - N_G(X_I))$.

Now we show how to transform I_{G-T} into an independent set for G of the required size. Let I_T be a MIS in $T - N_G(X_I)$, which has size $\alpha(T - N_G(X_I)) = \alpha(T)$. It is easy to verify that $I_{G-T} \cup I_T$ is an independent set in G because vertices of T are only adjacent to vertices of $G - T$ that are contained in X. Hence the set $I_{G-T} \cup I_T$ is independent in G and it has size $|I_{G-T}| + \alpha(T)$. Since this argument applies to any independent set I_{G-T} in graph $G - T$ it holds in particular for a MIS in $G - T$. This proves that $\alpha(G) \geq \alpha(G - T) + \alpha(T)$. □

We introduce the concept of *blockability* in order to state the last reduction rules.

Definition 5.4. The pair $x, y \in V(G) \setminus X$ is X*-blockable* if there is a chunk $Y \in \mathcal{X}$ such that $\{x, y\} \subseteq N_G(Y)$.

The definition can be interpreted as follows: any independent set in G containing the chunk Y cannot contain x nor y, so using the chunk Y in an independent set blocks both vertices of the pair x, y from being in the same independent set. It follows directly from the definition that if x, y is not X-blockable, then for any combination of $u \in N_G(x) \cap X$ and $v \in N_G(y) \cap X$ we have $u \neq v$ and $\{u, v\} \in E(G)$ — otherwise the singleton $\{u\}$ would block x and y, or the pair $\{u, v\}$ would be independent and would block x, y.

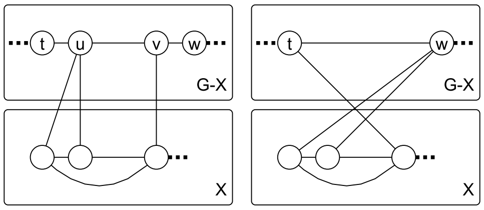
Fig. 5.3 illustrates the final two reduction rules, which are both meant to reduce the sizes of the trees in the forest F. Rule 5.3 deletes a tree T from the forest F when we can derive that for every independent set in $G - T$ we can obtain an independent set in G that is $\alpha(T)$ vertices larger. The last reduction rules act *locally* within one tree, but according to the same principle. Instead of working on an entire connected component of F, the rules reduce subtrees $T' \subseteq F$ in situations where we can derive that every independent set in X can be augmented with $\alpha(T')$ vertices from T'. In Rule 5.4 we reduce the subtree on vertices $\{u, v\}$, which has independence number one, and in Rule 5.5 we reduce the subtree on vertices $\{u, v, t, w\}$ with independence number two. Connections between the vertices adjacent to the reduced subtree are made to enforce that removal of the subtree does not affect the types of interaction between the neighboring vertices. We will see later in the analysis of the kernel size that these last two rules are needed to relate the size of the forest in a reduced instance, to the number of chunks in the instance and thereby to the size of the feedback vertex set.

Reduction Rule 5.4. If there are distinct vertices $u, v \in V(G) \setminus X$ that are adjacent in G and are not X-blockable and such that $\deg_F(u), \deg_F(v) \leq 2$, then reduce the graph as follows:

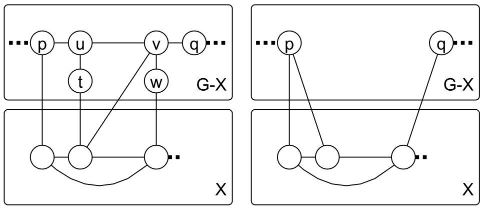
- Delete vertices u, v with their incident edges and decrease ℓ by one.
- If u has a neighbor t in F that is not v, make it adjacent to $N_G(v) \cap X$.
- If v has a neighbor w in F that is not u, make it adjacent to $N_G(u) \cap X$.
- If the vertices t, w exist then they are unique; add the edge $\{t, w\}$ to the graph.

It is not hard to see that this rule does not change the fact that F has a perfect matching: if the edge $\{u, v\}$ was contained in the perfect matching, then the matching restricted to the remaining vertices is a perfect matching for the remaining graph. If $\{u, v\}$ was not contained in the perfect matching then u was matched to t and v was matched to w; we obtain a perfect matching for the reduced graph by matching t to w, using the edge that is added to the graph by the reduction rule.

Lemma 5.6. *Let (G, X, ℓ) with $F := G - X$ be an instance to which Rule 5.4 is applicable at vertices u, v, and let $(G', X, \ell - 1)$ be the instance resulting from the reduction. Then it holds that $\alpha(G) \geq \ell \Leftrightarrow \alpha(G') \geq \ell - 1$.*



(a) Rule 5.4: Shrinking unblockable degree-2 paths in trees. ($\ell' := \ell - 1$)



(b) Rule 5.5: Removing unblockable leaves in trees. ($\ell' := \ell - 2$)

Figure 5.3: Illustrations of Rule 5.4 and Rule 5.5. The original structure is shown on the left, and the image on the right shows the structure after the reduction. Feedback vertices X are drawn in the bottom container, whereas the forest $G - X$ is visualized in the top container.

Proof. Assume the conditions in the lemma hold. We prove the two directions separately.

($\Rightarrow$) Let I_G be an independent set for graph G of size at least ℓ. We show how to obtain an independent set $I_{G'}$ for graph G' of size at least $|I_G| - 1 \geq \ell - 1$. Observe that no independent set in G can contain both $\{u, v\}$ since they are adjacent. If I_G does not contain any of the vertices $\{u, v\}$ then we show how to obtain I'_G that is at least as large and does contain one of $\{u, v\}$; so assume I_G avoids u and v. Since the pair u, v is not X-blockable by the preconditions for the reduction rule, we know that there is at least one vertex among u, v for which no neighbor in X is chosen in I_G. Assume without loss of generality (by symmetry) that this holds for u, such that $N_G(u) \cap X \cap I_G = \emptyset$. Since v is not in I_G by assumption, the only neighbor of u that can be in I_G is its neighbor t in F unequal to v (if such a t exists; see Fig. 5.3). If no such t exists then $I'_G := I_G \cup \{u\}$ is a bigger independent set in G; otherwise $I'_G := (I_G \setminus \{t\}) \cup \{u\}$ is an equally large independent set. So using this replacement argument and symmetry, we may assume that I_G is an independent set of size at least ℓ for G that contains u but not v.

We now claim that $I_{G'} := I_G \setminus \{u\}$ is an independent set of size $\geq \ell - 1$

in G'. Since it is easy to see that $I_{G'}$ has the desired size, it remains to show that it is an independent set in G'. To establish this we need to show that the transformation to G' does not add any edges between vertices of $I_{G'}$. This is ensured because all edges that are added by the transformation have at least one endpoint that is a neighbor of u: all added edges are either incident on w or a vertex in $N_G(u) \cap X$. Hence for each added edge one endpoint z is adjacent to u, and since we assumed $u \in I_G$ this implies that z cannot be in $I_{G'}$ since $I_{G'}$ is a subset of the independent set I_G in G and having adjacent vertices u and z in I_G would violate independence. Therefore $I_{G'}$ is indeed an independent set of the required size in G'.

($\Leftarrow$) Let $I_{G'}$ be an independent set for graph G' of size at least $\ell - 1$. We show how to obtain an independent set I_G for graph G of size at least $|I_{G'}| + 1 \geq \ell$. The structure of $I_{G'}$ determines how to augment to a larger independent set I_G. From the structure of the reverse transformation of G' to G it follows that $I_{G'}$ is an independent set in G; hence for each case we will only show that the new vertex we add to the set will not violate independence in graph G. We now do a case analysis based on whether or not the neighbors t of u and w of v are present.

- If vertex t exists and $t \in I_{G'}$, then define $I_G := I_{G'} \cup \{v\}$. To prove I_G is an independent set in G we show that $N_G(v) \cap I_{G'} = \emptyset$ by consecutively proving that $\{u, w\} \cap I_{G'} = \emptyset$ and $N_G(v) \cap X \cap I_{G'} = \emptyset$, which together suffice to establish our claim because $N_G(v) = \{u, w\} \cup (N_G(v) \cap X)$ (for as far as t exists). Since $u \notin V(G')$ we trivially have $u \notin I_{G'}$, and because the edge $\{t, w\}$ is added when forming G' and $t \in I_{G'}$ by the case distinction we have $w \notin I_{G'}$. To see that $N_G(v) \cap X \cap I_{G'} = \emptyset$ observe that $N_G(v) \cap X \subseteq N_{G'}(t)$ by the construction of G', and since $t \in I_{G'}$ and $I_{G'}$ is independent in G' this proves the claim and the correctness of this case.
- If vertex w exists and $w \in I_{G'}$, then define $I_G := I_{G'} \cup \{u\}$. The correctness argument is symmetric to that of the previous case.
- In the remaining case we know that $\{t, w\} \cap I_{G'} = \emptyset$. There must be some $z \in \{u, v\}$ such that $N_G(z) \cap X \cap I_{G'} = \emptyset$; because if there is no such z then by combining one vertex from $N_G(u) \cap X \cap I_{G'}$ and one from $N_G(v) \cap X \cap I_{G'}$ gives a pair which proves that $\{u, v\}$ is X-blockable in G, contradicting the precondition to the reduction rule. We now assign $I_G := I_{G'} \cup \{z\}$. Since $N_G(z) \cap F \subseteq \{t, u, v, w\}$ and these vertices either do not exist in G' or are not in $I_{G'}$ by the case distinction, we know $\{t, u, v, w\} \cap I_{G'} = \emptyset$. Since $N_G(z) \cap X \cap I_{G'} = \emptyset$ by our choice of z this proves that the addition of z to the independent set does not violate independence, because $N_G(z) \subseteq (N_G(z) \cap X) \cup \{t, u, v, w\}$.

Since the case distinction is exhaustive, it proves the lemma in the reverse direction, which concludes the proof. □

Reduction Rule 5.5. If there are distinct vertices t, u, v, w in $V(G) \setminus X$ that satisfy $\deg_F(u) = \deg_F(v) = 3$, $N_F(t) = \{u\}$, $N_F(w) = \{v\}$ and $\{u, v\} \in E(G)$

such that none of the pairs $\{u,t\}$, $\{v,w\}$, $\{t,w\}$ are X-blockable, then reduce as follows. Let $\{p\} = N_F(u) \setminus \{t,v\}$ and let $\{q\} = N_F(v) \setminus \{w,u\}$.

- Delete $\{t,u,v,w\}$ and their incident edges from G and decrease ℓ by two.
- Make p adjacent to all vertices of $N_G(t) \cap X$.
- Make q adjacent to all vertices of $N_G(w) \cap X$.

Once again it is not difficult to see that the rule preserves the fact that F has a perfect matching: since t and w have degree one in F, they must be matched to u and v in a perfect matching; hence the rule effectively deletes the endpoints of two matching edges from the graph.

Lemma 5.7. *Let (G,X,ℓ) with $F := G - X$ be an instance to which Rule 5.5 is applicable at vertices t,u,v,w, and let $(G',X,\ell-2)$ be the instance resulting from the reduction. Then it holds that $\alpha(G) \geq \ell \Leftrightarrow \alpha(G') \geq \ell - 2$.*

Proof. Assume the conditions in the lemma hold. We prove the two directions separately.

($\Rightarrow$) Let I_G be an independent set for graph G of size at least ℓ. We show how to obtain an independent set $I_{G'}$ for graph G' of size at least $|I_G| - 2 \geq \ell - 2$. We first show that without loss of generality we may assume that for one of the pairs $\{t,w\},\{t,v\},\{u,w\}$ both vertices of the pair belong to I_G. To see this, suppose that I_G avoids at least one vertex in each pair. We then obtain an alternative independent set I'_G that is at least as large, and contains both vertices of at least one pair, by distinguishing the following cases.

- If $I_G \cap X \cap N_G(t) = \emptyset$ and $I_G \cap X \cap N_G(w) = \emptyset$ then define $I'_G := (I_G \setminus \{u,v,t,w\}) \cup \{t,w\}$. This is easily seen to be an independent set. Since no independent set can contain three or more vertices from $\{u,v,t,w\}$ (because of the edges $\{u,t\}$ and $\{v,w\}$) we have $|I'_G| \geq |I_G|$.
- If $I_G \cap X \cap N_G(t) \neq \emptyset$ then we must have $I_G \cap X \cap N_G(w) = \emptyset$; for if both sets are nonempty, then taking one vertex from $I_G \cap X \cap N_G(t)$ and one vertex from $I_G \cap X \cap N_G(w)$ yields a pair that shows that $\{t,w\}$ is X-blockable, which contradicts the preconditions to Rule 5.5. Using the same argument we must have that $I_G \cap X \cap N_G(u) = \emptyset$, otherwise $\{t,u\}$ is X-blockable. Set $I'_G := (I_G \setminus \{p,u,t,v,w\}) \cup \{u,w\}$. The neighborhood conditions show that no neighbors of u,w in X are contained in I_G (and hence in $I_{G'}$). Because we explicitly delete any neighbors that u,w might have in F when forming I'_G, we see that I'_G is also an independent set in G. If $I_G \cap X \cap N_G(t) \neq \emptyset$ as specified by the precondition for this case, we cannot have $t \in I_G$ because then I_G would not be independent. The edges $\{p,u\}$ and $\{v,w\}$ in G show that of the set $\{p,u,v,w\}$ at most two vertices are in an independent set; hence in this situation I_G contains at most two vertices from $\{p,u,t,v,w\}$ and therefore we have $|I'_G| \geq |I_G|$.
- If $I_G \cap X \cap N_G(w) \neq \emptyset$ then we must have that $I_G \cap X \cap N_G(t) = I_G \cap X \cap N_G(v) = \emptyset$, and we set $I'_G := (I_G \setminus \{q,u,t,v,w\}) \cup \{t,v\}$. The correctness argument is symmetric to that of the previous case.

The argument above shows that we may assume without loss of generality that for one of the pairs $\{t, w\}, \{t, v\}, \{u, w\}$ the independent set I_G contains both vertices of the pair. Using this assumption we show how to obtain an independent $I_{G'}$ with $|I_{G'}| \geq |I_G| - 2$, again by case analysis.

- If $t, w \in I_G$ then define $I_{G'} := I_G \setminus \{t, w\}$. Since $t, w \in I_G$ implies that $u, v \notin I_G$ we know that all vertices in $I_{G'}$ still exist in G'. It remains to show that they form an independent set there. Because the reduction to G' only adds edges incident on p and q, it suffices to show that for all edges incident on p or q that are added by the reduction there is at least one endpoint not in $I_{G'}$. The transformation from G to G' adds edges from $N_G(t) \cap X$ to p, and edges from $N_G(w) \cap X$ to q. But since $t, w \in I_G$ we know that the independent set I_G contains no vertices of $N_G(t) \cap X$ or $N_G(w) \cap X$, and hence the defined set $I_{G'}$ is an independent set in G'.
- If $t, v \in I_G$ then define $I_{G'} := I_G \setminus \{t, v\}$. All vertices in $I_{G'}$ must exist in G' since u, w cannot be in I_G because their neighbors t, v are in I_G. The edges we add in the transformation to G' do not violate independence: because $t \in I_G$ we have $N_G(t) \cap I_G = \emptyset$, and similarly because $v \in I_G$ we have $N_G(v) \cap I_G = \emptyset$, which in particular means $q \notin I_G$. For all edges that we add, at least one endpoint is not in I_G and therefore not in $I_{G'}$; this proves that $I_{G'}$ is an independent set in G'.
- If $u, w \in I_G$ then define $I_{G'} := I_G \setminus \{u, w\}$. The proof of correctness is symmetric to that for the previous case.

Since one of these cases must apply, the listing is exhaustive and it concludes the forward direction of the equivalence.

($\Leftarrow$) Let $I_{G'}$ be an independent set for graph G' of size at least $\ell - 2$. We show how to obtain an independent set I_G for graph G of size at least $|I_{G'}| + 2 \geq \ell$. The structure of $I_{G'}$ determines how to augment to a larger independent set I_G by adding two vertices to $I_{G'}$. From the structure of the reverse transformation of G' to G it follows that $I_{G'}$ is an independent set in G; hence for each case distinguished below we will only show that the new vertices we add to the set will not violate independence in graph G.

- If $N_G(t) \cap X \cap I_{G'} = \emptyset$ and $N_G(w) \cap X \cap I_{G'} = \emptyset$ then assign $I_G := I_{G'} \cup \{t, w\}$. Since vertices t, w are clearly nonadjacent in G, and because the vertices in $I_{G'}$ form an independent set in G (as the transformation to G does not add edges between vertices in $I_{G'}$) we now have that I_G is an independent set in G of the required size.
- If $N_G(t) \cap X \cap I_{G'} \neq \emptyset$ then we must have $N_G(w) \cap X \cap I_{G'} = \emptyset$, otherwise taking one vertex from $N_G(t) \cap X \cap I_{G'}$ and one from $N_G(w) \cap X \cap I_{G'}$ would give a pair that shows that $\{t, w\}$ is X-blockable in the original graph G, which contradicts the preconditions for Rule 5.5. Similarly we must have $N_G(u) \cap X \cap I_{G'} = \emptyset$ by the assumption that $\{u, t\}$ is not X-blockable in G. Since vertex p is adjacent in G' to all vertices of $N_G(t) \cap X$, we know

that by independence of $I_{G'}$ if $N_G(t) \cap X \cap I_{G'} \neq \emptyset$ then $p \notin I_{G'}$. We now set $I_G := I_{G'} \cup \{u, w\}$ which must form an independent set in G because the established conditions show that none of the vertices of $N_G(\{u, w\})$ can be in $I_{G'}$. It is easy to see that $|I_G| \geq \ell$ in this case.

- If $N_G(w) \cap X \cap I_{G'} \neq \emptyset$ then we must have $N_G(t) \cap X \cap I_{G'} = N_G(v) \cap X \cap I_{G'} = \emptyset$ by the non-blockability of $\{w, t\}$ and $\{w, v\}$. We assign $I_G := I_{G'} \cup \{t, v\}$. The correctness proof is symmetric to that of the previous case.

Since the case distinction is exhaustive, it establishes the reverse direction of the equivalence, which concludes the proof. □

5.2.4 Structure of Reduced Instances

When no reduction rules can be applied to an instance, we call it (exhaustively) *reduced*. The main purpose of this section is to prove that in reduced clean instances, the number of vertices in the forest F is at most cubic in the size of the feedback vertex set. We describe the approach intuitively before presenting the details.

The analysis is based on the idea of identifying *conflict structures* in the forest $G - X$. Informally, one may think of a conflict structure S as a subgraph of the forest F with the following property: there is a chunk $Y \in \mathcal{X}$ such that any independent set in G that contains Y, contains fewer vertices from S than an optimal independent set in F would. Hence this conflict structure shows that by choosing Y to be a part of an independent set I, we pay for it inside the conflict structure S as $|I \cap S|$ will be smaller than $|I_F \cap S|$, where I_F is any optimal independent set in F. Since we trigger a reduction rule once there is a chunk $Y \in \mathcal{X}$ that induces at least $|X|$ conflicts (i.e., for which we have to pay at least $|X|$), there cannot be too many conflict structures in a reduced instance. The following notion is important for making these statements precise.

Definition 5.5. The number of *active conflicts* induced on the forest F by the set of chunks $\mathcal{X}$ is $\textsc{Active}_F(\mathcal{X}) := \sum_{Y \in \mathcal{X}} \textsc{Conf}_F(Y)$.

So the number of active conflicts is the number of conflicts induced on F summed over all chunks of the instance. For reduced instances, this value is cubic in $|X|$.

Observation 5.4. *Let (G, X, ℓ) be a reduced instance. By Rule 5.1 every $v \in X$ satisfies $\textsc{Conf}_F(\{v\}) < |X|$, and by Rule 5.2 every pair of distinct nonadjacent vertices $\{u, v\} \subseteq X$ satisfies $\textsc{Conf}_F(\{u, v\}) < |X|$. Hence $\textsc{Active}_F(\mathcal{X}) \leq |X|^2 + \binom{|X|}{2}|X|$.*

The global argument to bound the size of the kernel is therefore to show that in a reduced instance with forest F, the number of conflict structures that can be found is linear in the size of the forest. Since the total number of conflicts that are induced by the set of chunks $\mathcal{X}$ (the number of *active conflicts*) is bounded by $\mathcal{O}(|X|^3)$, this will prove that the number of vertices in F is $\mathcal{O}(|X|^3)$.

The proof of the kernel size bound is organized as follows. In the remainder of this section we will formally define conflict structures, and prove that the number of active conflicts induced on the forest F grows linearly with the number of conflict structures contained in F. We give an extremal graph-theoretic result showing that any forest with a perfect matching contains linearly many conflict structures, in Section 5.2.5. As the final step we will combine these results with Observation 5.4 to bound the size of the kernel in Section 5.2.6.

Definition 5.6 (Conflict Structures)**.** Let F be a forest with a perfect matching M.

- A *conflict structure of type A* in F is a pair of distinct vertices $\{v_1, v_2\}$ such that $\{v_1, v_2\} \in M$ and $\deg_F(v_1), \deg_F(v_2) \leq 2$.
- A *conflict structure of type B* in F is a path on four vertices (v_1, v_2, v_3, v_4) such that v_1 and v_4 are leaves of F, and $\deg_F(v_2) = \deg_F(v_3) = 3$.

Observe that in a conflict structure of type B, the edges $\{v_1, v_2\}$ and $\{v_3, v_4\}$ must be contained in the perfect matching M by Observation 5.1. Although conflict structures can be defined for arbitrary forests with a perfect matching, we are of course interested in the forests that occur in a reduced clean instance of Independent Set [fvs]. To capture the interaction between chunks of such an instance and conflict structures in the forest, we need the following definition.

Definition 5.7 (Hitting conflict structures)**.** Let (G, X, ℓ) be a clean instance of Independent Set [fvs], and consider the forest $F := G - X$ with a perfect matching M. Let $Y \in \mathcal{X}$ be a chunk.

- $Y \in \mathcal{X}$ *hits* a conflict structure $\{v_1, v_2\}$ of type A in F if $\{v_1, v_2\} \subseteq N_G(Y)$.
- $Y \in \mathcal{X}$ *hits* a conflict structure (v_1, v_2, v_3, v_4) of type B in F if one of the following holds:
  - $\{v_1, v_2\} \subseteq N_G(Y)$, or
  - $\{v_3, v_4\} \subseteq N_G(Y)$, or
  - $\{v_1, v_4\} \subseteq N_G(Y)$.

The importance of reduction rules 5.4 and 5.5 now becomes clear.

Observation 5.5. *If (G, X, ℓ) is a reduced clean instance of* Independent Set [fvs] *and S is a conflict structure in a tree T of the forest $F := G - X$, then S is hit by some chunk of $\mathcal{X}$. If a structure of type A is not hit, this triggers Rule 5.4, and if a structure of type B is not hit, this triggers Rule 5.5.*

The fact that each conflict structure is hit by at least one chunk in a reduced instance, allows us to relate the number of vertex-disjoint conflict structures to the number of active conflicts that must be induced by the chunks.

Lemma 5.8. *Let (G, X, ℓ) be a reduced clean instance of* Independent Set [fvs] *with forest $F := G - X$ such that M is a perfect matching in F, and let $\mathcal{S}$ be a set of vertex-disjoint conflict structures in F. Then* $\text{Active}_F(\mathcal{X}) \geq |\mathcal{S}|$.

Proof. Assume the conditions in the lemma hold. Consider some chunk $Y \in \mathcal{X}$, and let $\mathcal{S}_Y$ be the structures in $\mathcal{S}$ that are hit by Y according to Definition 5.7. We will first show that $\text{Conf}_F(Y) \geq |\mathcal{S}_Y|$. Later we will show how this implies the lemma.

So consider an arbitrary chunk $Y \in \mathcal{X}$ and the corresponding $\mathcal{S}_Y$. To prove that $\text{Conf}_F(Y) \geq |\mathcal{S}_Y|$ we prove that there is an induced subgraph $F' \subseteq F$ with $F - N_G(Y) \subseteq F' \subseteq F$ such that $\alpha(F) - \alpha(F') \geq |\mathcal{S}_Y|$. Since $\alpha(F) - N_G(Y) \leq \alpha(F')$ by Observation 5.2, this will show that $\text{Conf}_F(Y) \geq |\mathcal{S}_Y|$. To reason about the difference between the independence number of F and of the graph F' that we construct, we will ensure that F' has a perfect matching M' and compare the size of M' to M. The independence numbers can be compared by inspecting these matchings, since we know by Observation 5.1 that $\alpha(F') = |M'|$ and $\alpha(F) = |M|$ when M', M are perfect matchings for forests F', F respectively. Let us first show how to obtain F' and M' for a single arbitrary conflict structure $S \in \mathcal{S}_Y$:

1. If $S = \{v_1, v_2\}$ is a conflict structure of type A, then $\{v_1, v_2\} \subseteq N_G(Y)$ by Definition 5.7 since Y hits S, and edge $\{v_1, v_2\}$ is contained in M by Definition 5.6. Now obtain F' from F by deleting the vertices v_1 and v_2, and obtain M' from M by deleting the edge $\{v_1, v_2\}$.
2. If $S = (v_1, v_2, v_3, v_4)$ is a conflict structure of type B, then the edges $\{v_1, v_2\}$ and $\{v_3, v_4\}$ are contained in M by Observation 5.1. By Definition 5.7, using the fact that Y hits S, one of the following applies:
   - If $\{v_1, v_2\} \in N_G(Y)$ then delete vertices v_1, v_2 from F and delete the edge between them from M.
   - If $\{v_3, v_4\} \in N_G(Y)$ then delete vertices v_3, v_4 from F and delete the edge between them from M.
   - If $\{v_1, v_4\} \in N_G(Y)$ then delete vertices v_1, v_4 from F. Delete the edges $\{v_1, v_2\}$ and $\{v_3, v_4\}$ from M and replace them by the edge $\{v_2, v_3\}$.

   Let F' be the resulting graph, and M' the resulting matching.

Observe that in all cases the graph F' is a vertex-induced subgraph of F and has M' as a perfect matching. Since the perfect matching M' contains one edge less than M, we have $\alpha(F') = \alpha(F) - 1$ by Observation 5.1. Now it is not difficult to see that rather than doing the above step for just a single conflict structure in $\mathcal{S}_Y$, we can repeat this step for every conflict structure in the set. Since the conflict structures are vertex-disjoint, the changes we make for one operation do not affect the applicability of above-described operation for other conflict structures. Performing the update step for each conflict structure in $\mathcal{S}_Y$ results in a vertex-induced subgraph $F' \subseteq F$ with perfect matching M' such that $|M| - |M'| = |\mathcal{S}_Y|$, which shows that $\text{Conf}_F(Y) \geq |\mathcal{S}_Y|$ as argued before.

We have shown that for every chunk $Y \in \mathcal{X}$, it holds that $\textsc{Conf}_F(Y) \geq |\mathcal{S}_Y|$, where $\mathcal{S}_Y$ is the set of conflict structures hit by Y. The lemma now follows from the definition of active conflicts as the sum of the conflict values over all chunks, using that all conflict structures in $\mathcal{S}$ are hit by at least one chunk (Observation 5.5). This concludes the proof. □

The previous lemma shows that if F has many conflict structures, then the number of active conflicts must be large, and therefore the size of the feedback vertex set must be large. The extremal argument of the next section makes it possible to turn this relation into a kernel size bound.

5.2.5 Packing Conflict Structures

In this section we present an extremal result that shows that trees with a perfect matching contain linearly many conflict structures.

Theorem 5.1. *Let T be a tree with a perfect matching. There is a set $\mathcal{S}$ of mutually vertex-disjoint conflict structures in T with $|\mathcal{S}| \geq |V(T)|/14$.*

Proof. If T is the tree on two vertices then the statement follows trivially, since T contains exactly one conflict structure of type A (see Definition 5.6). In the remainder we therefore assume that $T \neq K_2$. This implies that T has at least four vertices: the number of vertices must be even, since T has a perfect matching. We use a proof by construction to find a set of conflict structures. The procedure grows a subtree $T' \subseteq T$ and set $\mathcal{S}$ incrementally, and during each augmentation step of the tree we enforce an incremental inequality that shows that the number of vertices of T which are contained in T', is proportional to the number of conflict structures found so far in the subtree T'. This proof strategy is inspired by the method of "amortized analysis by keeping track of dead leaves" that is used in extremal graph theory [120].

The proof revolves around a subtree $T' \subseteq T$ that is grown by successively adding vertices to it. We use the following characteristics of the subgraph T' in the analysis. The vertices $\textsc{Leaves}(T') \setminus \textsc{Leaves}(T)$ are the *open branches* of T'. The open branches are essentially the vertices on the boundary of the subgraph T', where we will eventually "grow" the subtree T' to make it larger, until it encompasses all of T. Observe that when we have grown T' until it equals T, the number of open branches is zero by definition. We use the letter O to denote the number of open branches of the current state of the subtree T'. While growing the subtree we construct a set $\mathcal{S}$ of vertex-disjoint conflict structures. We use C as an abbreviation for $|\mathcal{S}|$. It turns out that certain vertices of T play a special role in the amortized analysis implicit in the proof. We call these vertices *spikes*.

Definition 5.8. A *spike* in tree T is a vertex v such that $\deg_T(v) = 3$ and there is exactly one leaf of T adjacent to v. A vertex $v \in V(T)$ is a *live spike* with respect to the current subtree T' if v is a spike in T and an open branch of T'.

When an open branch vertex is a *spike*, this will allow us to find more conflict structures later on in the process, so that we may balance an increase in the number of vertices of the subtree T' against an increase in the number of live spikes. Overall, this means that we may justify an increase in the number of vertices that are contained in T' by increasing (a) the number of open branches, (b) the number of conflict structures that have been found, or (c) the number of live spikes. The number of live spikes in the subtree T' is denoted by S, and the total number of vertices of T' is denoted by N. The balancing process is captured by the following *incremental inequality* that we will satisfy while growing the subtree T':

$$8\Delta O + 14\Delta C + \Delta S \geq \Delta N. \tag{5.1}$$

The Δ-values in the incremental inequality refer to the changes in the values of O, C, S and N caused by augmenting the tree T'. For example, if T' has 5 open branches at a given moment and we perform an augmentation after which it has 4 open branches, then $\Delta O = -1$ for that step. We define the augmentations to the tree T' by adding vertices to it; it will be understood implicitly that the subtree T' we are considering is the subtree of T induced by all the vertices that were added at some point in the process.

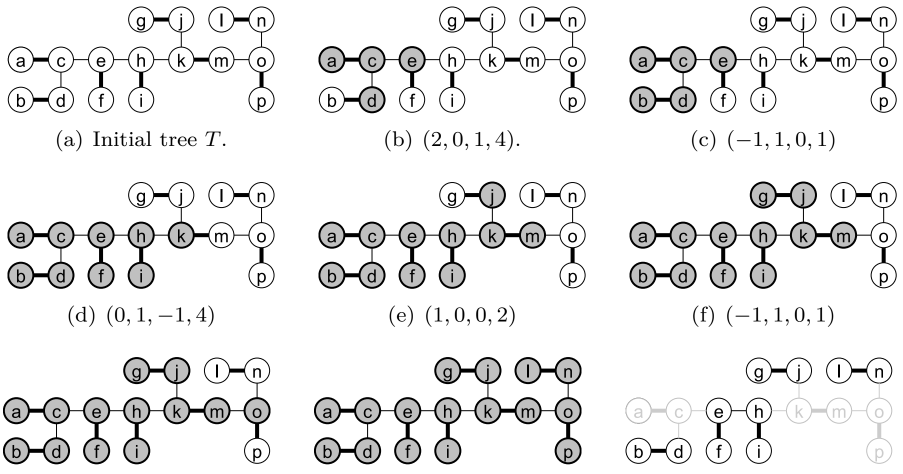
We will show that T' and the set $\mathcal{S}$ can be initialized and grown such that each augmentation satisfies the incremental inequality, until all vertices of T have been added to T' and the two graphs coincide. At that stage we will have $N = |V(T)|$, $O = 0$ and $S = 0$, for if $T' = T$ then T' contains exactly $|V(T)|$ vertices, and the set $\textsc{Leaves}(T') \setminus \textsc{Leaves}(T)$ is empty. By summing the incremental inequality over all augmentation steps we then find that the final state of the tree T' satisfies $8O + 14C + S \geq N$ which implies $C \geq N/14 = |V(T)|/14$ since $O = S = 0$ for this final state. Since C measures the number of conflict structures in the set $\mathcal{S}$ we construct, this shows that the process finds a set of at least $|V(T)|/14$ mutually vertex-disjoint conflict structures. Hence to establish the theorem, all that remains is to give the initialization and augmentation operations for T'. Fig. 5.4 illustrates the construction process.

We say that a vertex $u \in N_T(v) \setminus V(T')$ is a neighbor of u *outside* T', and a vertex $u \in N_T(v) \cap V(T')$ is a neighbor *inside* T'. The operations that augment the subtree T' will maintain the following invariants.

Invariant I. For all conflict structures $S \in \mathcal{S}$ it holds that $V(S) \subseteq V(T') \setminus (\textsc{Leaves}(T') \setminus \textsc{Leaves}(T))$, i.e., the vertices we use in conflict structures are contained in T' and are not open branches of T'.

Invariant II. All vertices of T' that have a neighbor outside T' are leaves of T', implying that when $|V(T')| \geq 2$ all vertices of T' that have a neighbor outside T' are open branches of T'.

The first invariant will ensure that the conflict structures we find are mutually vertex-disjoint. The second invariant is important because it implies that if T' has no open branch vertices, then T' coincides with T. It is trivial to see that



(a) Initial tree T. (b) $(2,0,1,4)$. (c) $(-1,1,0,1)$

(d) $(0,1,-1,4)$ (e) $(1,0,0,2)$ (f) $(-1,1,0,1)$

(g) $(0,0,1,1)$ (h) $(-1,1,-1,3)$ (i) Conflict structures.

Figure 5.4: Illustrations of some augmentation operations. Edges in the perfect matching of T are drawn with thick lines. Vertices in $V(T') \cap V(T)$ are visualized as shaded circles with thick borders. Unshaded vertices belong to $V(T) \setminus V(T')$. Each state of the subtree T' is labeled with the vector $(\Delta O, \Delta C, \Delta S, \Delta N)$ of the operation that yielded the state. (a) Tree T to which the theorem is applied. Vertices $\{c, e, h, o\}$ are *spikes* of T. (b) Result of applying Operation 5.1 with $v = a$. Vertices d and e become *open branches* of T', and since e is a spike, it becomes a *live spike*. (c) Applied Operation 5.4 to tree extending path (d, b), finding the conflict structure $\{d, b\}$ of type A. Vertex d is lost as an open branch. (d) Applied Operation 5.3 to tree extending path (e, h), finding the conflict structure (f, e, h, i) of type B. Since spike e is no longer an open branch after the operation, the number of live spikes decreases. The number of open branches does not change, as k becomes an open branch to replace e. (e) Applied Operation 5.5 to the singleton path (k). (f) Applied Operation 5.4 to the path (j, g), adding a conflict structure $\{j, g\}$ of type A. (g) Applied Operation 5.2 to vertex m, causing o to become a *live spike*. (h) Applied Operation 5.4 to the path (o, n, l). Vertex o is lost as an open branch and as a live spike vertex, which is compensated by finding the conflict structure $\{n, l\}$ of type A. (i) The conflict structures found by the process.

the invariants are initially satisfied for an empty tree T' and empty set of conflict structures $\mathcal{S}$. We proceed by describing the augmentation operations. Whenever we talk about the neighbors of a vertex v in this description, we mean v's neighbors in the graph T unless explicitly stated otherwise. Similarly, when we talk about a vertex being a leaf then we mean a leaf of the tree T, rather than T'.

Initialization

The first operation we describe shows how to initialize the subtree T'. Recall from the beginning of the proof that we could assume $|V(T)| \geq 4$.

Operation 5.1. Let v be a leaf of T and let u be its neighbor in the tree. Initialize T' as the tree on vertex set $N_T[u]$.

Claim 5.1. *Operation 5.1 satisfies the incremental inequality and maintains the invariants.*

Proof. For an empty tree we obviously have $O = S = C = N = 0$. Let us now consider how these values are affected by the tree initialization. Since T is connected and has at least four vertices, u has at least one neighbor other than v. We claim that all vertices $N_T(u) \setminus \{v\}$ are open branches of T' after the initialization. By Observation 5.1 vertex v is the only leaf adjacent to u, and since T is a tree, the subtree induced by vertex set $N_T[u]$ has the vertices $N_T(u)$ as leaves. Therefore the vertices $N_T(u) \setminus \{v\}$ are contained in $\textsc{Leaves}(T') \setminus \textsc{Leaves}(T)$ and are open branches of T' by definition, so $\Delta O = |N_T(u) - 1|$. The number of vertices added to the tree by the initialization is exactly $\Delta N = |N_T[u]|$. The number of live spikes cannot decrease by this operation (since it started at zero, and cannot become negative); hence $\Delta S \geq 0$. Since we do not add any conflict structures to $\mathcal{S}$ we find $\Delta C = 0$. It is easy to see that this combination of values satisfies the incremental inequality since $|N_T(u) - 1| \geq 1$. Since we do not add conflict structures, Invariant I is trivially maintained. Invariant II is maintained by adding all neighbors of u to the tree simultaneously. $\diamondsuit$

Observe that the initialization ensures that tree T' has at least three vertices, which will be used later on.

Augmentation

We will now describe the operations that are used to augment the tree once it is initialized. For each augmentation we prove that it satisfies the incremental inequality. After describing the remaining four operations, we prove that whenever the tree T' does not yet encompass all of T, then some augmentation is applicable. When describing the augmentation steps of the subtree T', we will use T'_a to refer to the status of the tree before the augmentation and T'_b to refer to its status after the augmentation. When the intended meaning is clear from the context we will just write T'.

Operation 5.2. If $|V(T')| \geq 3$ and there is a vertex $v_0 \in V(T')$ with $\deg_T(v_0) = 2$ such that $N_T(v) \setminus V(T')$ contains a spike vertex v_1, then add v_1 to T'.

Claim 5.2. *Operation 5.2 satisfies the incremental inequality and maintains the invariants.*

Proof. The number of vertices in T' increases by exactly one. As $\deg_T(v_0) = 2$, the vertex v_0 is not a spike. Therefore the number of live spikes increases by one through this operation ($\Delta S = 1$) since the spike v_1 becomes an open branch by this augmentation: v_1 will be a leaf of T', yet is not a leaf of T since $\deg_T(v_1) = 3$

by definition of a spike. The number of vertices increases by one ($\Delta N = 1$). The number of open branches does not change: vertex v_0 is lost as an open branch, but instead v_1 becomes an open branch ($\Delta O = 0$). Since the number of conflict structures does not change ($\Delta C = 0$) it is now trivial to see that these values satisfy the inequality. Since we do not add conflict structures we maintain Invariant I. Invariant II is maintained because prior to the augmentation, vertex v_1 is the only neighbor of v_0 that is not yet contained in T' which follows from the fact that v_0 must have a parent in the tree T' because $|V(T')| \geq 3$, and the degree of v_0 is only two. So the augmentation effectively adds all vertices $N_T[v_0]$ to T'. $\diamondsuit$

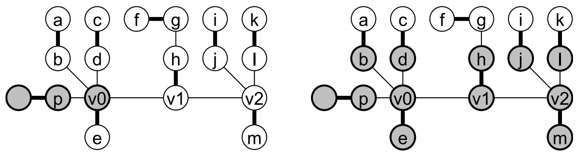
The remaining augmentation operations grow the subtree by extending it over a path.

Definition 5.9. A *tree extending path* is a path $P = (v_0, v_1, \ldots, v_q)$ in T such that $V(P) \cap V(T') = \{v_0\}$ and v_0 is an open branch vertex of T'.

Operation 5.3. If $|V(T')| \geq 3$ and there is a tree extending path $P = (v_0, v_1)$ such that v_0 and v_1 are adjacent to leaves l_0, l_1 of T respectively with $l_0, l_1 \notin V(T')$ and $\deg_T(v_0) = \deg_T(v_1) = 3$, then add the vertices $N_T[V(P)]$ to the tree T', and add the conflict structure of type B containing (l_0, v_0, v_1, l_1) to $\mathcal{S}$.

Claim 5.3. *Operation 5.3 satisfies the incremental inequality and maintains the invariants. The added conflict structure is disjoint from previously found structures.*

Proof. Before the operation, vertex v_0 is already contained in T' and has a unique neighbor p inside T' since v_0 is a leaf of the tree T' that has at least two vertices. Observe that p cannot be a leaf of T', since v_0 is a leaf of T' and $|V(T')| \geq 3$. Hence the neighbors of v_0 in T are exactly $\{p, l_0, v_0\}$. Similarly, the neighbors of v_1 in T are exactly $\{q, l_1, v_0\}$ for a vertex $q \notin V(T')$ that is not a leaf of T. Therefore the vertices that are added to T' by this operation, and that were not contained in T' already, are exactly $\{l_0, l_1, v_1, q\}$ which shows that $\Delta N = 4$. Now consider the effect of the augmentation on the number of live spike vertices. Vertex v_0 is a live spike in T' before the augmentation: it is an open branch vertex by definition of a tree extending path, and the degree and leaf requirements of Definition 5.8 are met. Vertex v_0 becomes an internal vertex of T' by adding its neighbors to the tree, and therefore it will no longer be a live spike after the augmentation. But no other live spikes can be lost by the augmentation, hence $\Delta S \geq -1$. Since we add a conflict structure in this operation, $\Delta C = 1$. Let us finally consider the effect of this operation on the number of open branches. Clearly vertex v_0 is no longer an open branch after the augmentation, and it was one before the augmentation. Vertices l_0 and l_1 are leaves of T and therefore do not become open branch vertices. But the vertex q cannot be a leaf of T by Observation 5.1, and it will be a leaf of T' after the augmentation. Hence the loss of v_0 as an open branch is compensated by q becoming an open branch, and $\Delta O = 0$. It is trivial to see that this combination of values satisfies the incremental inequality.



(a) Tree T with subtree T'. (b) State after extending T' over the path (v_0, v_1, v_2).

Figure 5.5: Illustrations of the vertex sets that are involved in the proofs of Claim 5.4 and 5.5. On the left is a tree T with a perfect matching (visualized by thick edges), with a subtree T' indicated by shaded vertices. Vertex v_0 is an open branch vertex for this state of the subtree. When considering the tree extending path $P = (v_0, v_1, v_2)$ the corresponding vertex sets S_i and L_i that are defined in Claim 5.4 are as follows. $L_0 = \{e\}$, $L_1 = \emptyset$, $L_2 = \{m\}$. $S_0 = \{b, d\}$, $S_1 = \{h\}$, $S_2 = \{j, l\}$. The state of T' after adding the vertices $N_T[V(P)]$ to the tree is shown on the right. Observe that all vertices $\bigcup_{i=0}^{2} S_i$ have become open branches by the augmentation, and that the vertices $\bigcup_{i=0}^{2} L_i$ are not open branches after augmentation.

Invariant II is maintained by adding the closed neighborhood of a path to the tree T', ensuring that afterwards no vertex on the path P can have neighbors outside T'. Adding $N_T[V(P)]$ to T' ensures that after the augmentation, none of the vertices of (l_0, v_0, v_1, l_1) can be open branches of T' while all those vertices are contained in T', which shows how Invariant I is maintained. By the same invariant, none of the vertices $\{l_0, v_0, v_1, l_1\}$ are contained in conflict structures in $\mathcal{S}$ prior to the augmentation, since the involved vertices are not in T' or open branches of T'. Hence the structure we add does not intersect any other structures in the set. $\diamondsuit$

Operation 5.4. If $|V(T')| \geq 3$ and there is a tree extending path $P = (v_0, \ldots, v_q)$ for $q \leq 2$ such that $\deg_T(v_{q-1}), \deg_T(v_q) \leq 2$, and the edge between v_{q-1} and v_q is contained in the perfect matching in T, then add the vertices $N_T[V(P)]$ to the tree T', and add the conflict structure $\{v_{q-1}, v_q\}$ to $\mathcal{S}$.

Claim 5.4. *Operation 5.4 satisfies the incremental inequality and maintains the invariants. The added conflict structure is disjoint from previously found structures.*

Proof. Let p be the unique neighbor of v_0 in T'_a, the subtree before the augmentation. Let L_i for $i \in \{0, \ldots, q\}$ be defined as $L_i := (N_T(v_i) \cap \textsc{Leaves}(T)) \setminus (V(T'_a) \cup \{v_0, \ldots, v_q\})$. By Observation 5.1 it follows that $|L_i| \leq 1$ for all i. Define S_i for $i \in \{0, \ldots, q\}$ as $S_i := N_T(v_i) \setminus (V(T'_a) \cup \textsc{Leaves}(T) \cup \{v_0, \ldots, v_q\})$. Refer to Fig. 5.5 for an illustration of these vertex sets, but note that the illustration does not show a path to which Operation 5.4 is applicable, as the illustration will also be used for the next operation.

It follows from these definitions that the vertices added to T' by the augmentation, which were not already in T', are exactly $\{v_1, \ldots, v_q\} \cup \bigcup_{i=0}^{q} (L_i \cup S_i)$ and that the sets involved in this expression are all vertex-disjoint. Hence $\Delta N = q + \sum_{i=0}^{q}(|L_i| + |S_i|)$. We have $\Delta S \geq -1$ since the only vertex that might be a live spike before the augmentation, but no longer after the augmentation, is v_0. The number of open branches is affected as follows: we lose the vertex v_0 as an open branch, but the vertices in $\bigcup_{i=0}^{q} S_i$ turn into open branches after the augmentation so $\Delta O \geq |\bigcup_{i=0}^{q} S_i| - 1$. Since we add one conflict structure in this operation, we have $\Delta C = 1$.

$$\begin{aligned}
&8\Delta O + 14\Delta C + \Delta S \\
&\geq 8(\sum_{i=0}^{q} |S_i| - 1) + 14 - 1 && \text{By bounds given above.} \\
&\geq 8\sum_{i=0}^{q} |S_i| + 5 && \text{Simplifying.} \\
&\geq \sum_{i=0}^{q} |S_i| + \sum_{i=0}^{q} |L_i| + q && \text{Since } \textstyle\sum_{i=0}^{q} |L_i| \leq 3 \text{ and } q \leq 2. \\
&= \Delta N.
\end{aligned}$$

Invariant II is maintained for the same reason as before, whereas I is maintained because we add the vertices involved in the conflict structure, and all their neighbors, to T'. Using this invariant it follows that the conflict structure we add must be disjoint from structures added to $\mathcal{S}$ earlier, since prior to the augmentation the vertices v_{q-1} and v_q were (a) not part of T', or (b) open branches of T'. $\diamondsuit$

Operation 5.5. If $|V(T')| \geq 3$ and there is a tree extending path $P = (v_0, \ldots, v_q)$ for $q \leq 2$ such that (a) $\deg_T(v_q) \geq 4$ or (b) $\deg_T(v_q) = 3$ and v_q is not adjacent to a leaf of T, then add the vertices $N_T[V(P)]$ to the tree T'.

Claim 5.5. *Operation 5.5 satisfies the incremental inequality.*

Proof. Let vertex p, sets L_i and S_i for $i \in \{0, 1, 2\}$ be defined as in the proof of Claim 5.4. By exactly the same reasoning as in that claim, the same bounds for ΔN, ΔO and ΔS hold for this augmentation and $\sum_{i=0}^{q} |L_i| \leq 3$. Since we do not add conflict structures in this operation we obviously have $\Delta C = 0$. Now observe that the precondition to the augmentation ensures that $|S_q| \geq 2$. We will use this with the bound $\Delta O \geq |\bigcup_{i=0}^{q} S_i| - 1$ that was derived in Claim 5.4:

$$\begin{aligned}
&8\Delta O + 14\Delta C + \Delta S \\
&\geq 8(\sum_{i=0}^{q} |S_i| - 1) + 14 \cdot 0 - 1 && \text{By bounds given above.} \\
&\geq 7\sum_{i=0}^{q} |S_i| + \sum_{i=0}^{q} |S_i| - 9 && \text{Rewriting.}
\end{aligned}$$

$$
\begin{aligned}
&\geq \sum_{i=0}^{q} |S_i| + 5 && \text{Since } |S_q| \geq 2. \\
&\geq \sum_{i=0}^{q} |S_i| + \sum_{i=0}^{q} |L_i| + q && \text{Since } \textstyle\sum_{i=0}^{q} |L_i| \leq 3 \text{ and } q \leq 2. \\
&= \Delta N.
\end{aligned}
$$

The invariants are maintained for the same reason as for the previous operation. ♢

This concludes the description of the augmentation operations. For the remainder of the proof, it suffices to show that the given set of augmentation operations can grow any subtree $T' \subseteq T$ that is initialized by Operation 5.1 until it encompasses all of T, while respecting the incremental inequality. Since T' contains at least three vertices after its initialization, Invariant II shows that if $T' \neq T$ then there is some open branch of T', i.e., there is a vertex $v \in \textsc{Leaves}(T') \setminus \textsc{Leaves}(T)$. The fact that an initialized tree T' has at least three vertices also implies that an open branch vertex v has exactly one neighbor u inside T', and that this vertex u cannot be a leaf of T: v is a leaf of T' by definition, and if u is a leaf of T then it is also a leaf of the subgraph $T' \subseteq T$, so the leaves u and v of T' would be adjacent; but then T' has only two vertices in total. Using this information about open branch vertices, we now show that for every open branch vertex v there is some applicable augmentation operation near this vertex. We use a case distinction depending on the local structure around v.

1. If (a) $\deg_T(v_0) \geq 4$ or (b) $\deg_T(v_0) = 3$ and v_0 is not adjacent to a leaf of T, then Operation 5.5 is applicable to the tree extending path (v_0).
2. If $\deg_T(v_0) = 3$ and v_0 is adjacent to a leaf of T, then consider some neighbor $v_1 \in N_T(v_0) \setminus V(T')$ that is not a leaf of T. Since v_0 has exactly one neighbor in T' and is adjacent to exactly one leaf of T (by Observation 5.1), such a vertex exists. Now consider the maximal path $P = (v_0, v_1, \ldots, v_q)$ obtained by starting with the edge $\{v_0, v_1\}$ and following vertices that have degree two in T, until arriving at the first vertex v_q that has $\deg_T(v_q) \neq 2$. If $\deg_T(v_1) \neq 2$ then this simply results in $P = (v_0, v_1)$. Observe that by this definition, vertices $v_1, \ldots, v_q$ are not contained in T'.
   (a) If (a) $\deg_T(v_1) \geq 4$ or (b) $\deg_T(v_1) = 3$ and v_1 is not adjacent to a leaf of T, we find that Operation 5.5 is applicable to the tree extending path (v_0, v_1).
   (b) If v_1 has degree three, then by the previous case it is adjacent to a leaf in T. Operation 5.3 is applicable to the tree extending path (v_0, v_1). Observe that the leaf of T adjacent to v_0 cannot be contained in T', as per the discussion above.
   (c) Since v_1 has degree at least two in T by our choice of v_1 as not being a leaf of T, in the remaining situations we have $\deg_T(v_1) = 2$ and

therefore there exists some v_2 on the path P we defined earlier. Now observe that T having a perfect matching implies that v_2 cannot be adjacent to a leaf of T: by definition of this case, v_0 is adjacent to a leaf of T. If v_2 is also adjacent to a leaf, a perfect matching must match v_0 and v_2 to their adjacent leaves. But then vertex v_1 with neighbors v_0 and v_2 cannot be matched. Hence v_2 is not adjacent to a leaf.

   i. If v_2 has degree at most two in T, then Operation 5.4 is applicable to the tree extending path (v_0, v_1, v_2). The edge $\{v_1, v_2\}$ is contained in the perfect matching of T: vertex v_0 can only be matched to its adjacent leaf, and since v_1 has degree two the only remaining edge incident on it that can be in a matching is indeed $\{v_1, v_2\}$.
   ii. If v_2 has degree at least three in T, then since we derived earlier that v_2 is not adjacent to a leaf we find that Operation 5.5 is applicable to the tree extending path (v_0, v_1, v_2).

3. If $\deg_T(v_0) = 2$, let v_1 be the unique neighbor of v_0 not contained in T', which exists by definition of an open branch vertex. Consider the maximal path $P = (v_0, v_1, \ldots, v_q)$ obtained by starting with the edge $\{v_0, v_1\}$ and following degree-2 vertices until arriving at the first vertex v_q that has degree unequal to two in T.

   (a) If (a) $\deg_T(v_1) \geq 4$ or (b) $\deg_T(v_1) = 3$ and v_1 is not adjacent to a leaf, then Operation 5.5 is applicable to the tree extending path (v_0, v_1).
   (b) If $\deg_T(v_1) = 3$ and v_1 is adjacent to a leaf, then v_1 is a spike vertex, which shows that Operation 5.2 is applicable.
   (c) If $\deg_T(v_1) = 1$ then Operation 5.4 is applicable to the extending path (v_0, v_1).
   (d) In the remainder we therefore have $\deg_T(v_1) = 2$, which implies by the definition of the path P we are considering that there is a vertex v_2.
      i. If $\deg_T(v_2) \leq 2$ then we claim Operation 5.4 is applicable. Since the degree of v_1 in T is two, either the edge $\{v_0, v_1\}$ or $\{v_1, v_2\}$ is contained in the perfect matching, which shows that the mentioned operation can be applied to the path (v_0, v_1) or (v_0, v_1, v_2) depending on which case holds.
      ii. In the remainder we therefore have $\deg_T(v_2) \geq 3$. If v_2 is adjacent to a leaf, then v_2 must be matched to this leaf in the perfect matching, which shows that v_1 is matched to v_0: hence Operation 5.4 is applicable to (v_0, v_1).
      iii. If v_2 is not adjacent to a leaf, then since its degree is at least three we find that Operation 5.5 is applicable to the tree extending path (v_0, v_1, v_2).

Observe that no open branch vertex can have degree one in T, by definition. Because the case distinction is exhaustive we have shown that whenever T' is not yet equal to T we can augment the tree T' while respecting the incremental inequality. By the argument given above this proves that the resulting set of conflict structures $\mathcal{S}$ satisfies $|\mathcal{S}| \geq |V(T)|/14$, which concludes the proof of Theorem 5.1. ☐

We remark that by using a more detailed case analysis, one could prove better bounds for the number of conflict structures in a tree with a perfect matching. An improvement in the bound immediately leads to a better theoretical upper bound on the kernel size. We have chosen not to pursue this bound further in the interest of space and readability.

5.2.6 The Kernelization Algorithm

Using the packing argument from the previous section, we can finally prove an upper bound on the size of reduced instances.

Lemma 5.9. *Let* (G, X, ℓ) *be a reduced* clean *instance of* Independent Set [fvs] *with forest* $F := G - X$*. Then* $|V(G)| \leq |X| + 14|X|(|X| + \binom{|X|}{2})$*.*

Proof. Consider a reduced instance (G, X, ℓ) as stated. As the instance is clean, the forest F has a perfect matching. By applying Theorem 5.1 to each tree in the forest F, we find obtain a set $\mathcal{S}$ of vertex-disjoint conflict structures in F with $|\mathcal{S}| \geq |V(F)|/14$. By Lemma 5.8 this shows that $\textsc{Active}_F(\mathcal{X}) \geq |V(F)|/14$. On the other hand, Observation 5.4 gives the bound $\textsc{Active}_F(\mathcal{X}) \leq |X|^2 + \binom{|X|}{2}|X|$. We therefore find that $|V(F)| \leq 14(|X|^2 + \binom{|X|}{2}|X|)$. Since $|V(G)| = |X| + |V(F)|$ we conclude that $|V(G)| \leq |X| + 14|X|(|X| + \binom{|X|}{2})$. ☐

The previous lemma gives a size bound for reduced instances. Before proving the existence of a kernel using this bound, let us consider how much time is needed to compute a reduced instance.

Lemma 5.10. *Given a clean instance* (G, X, ℓ) *of* Independent Set [fvs] *on* n *vertices we can exhaustively apply reduction rules 5.1–5.5 in* $\mathcal{O}(|X|^2 \cdot n)$ *time to output an equivalent reduced instance* (G', X', ℓ')*.*

Proof. The crucial idea is to apply the reduction rules in a suitable order, to prevent retriggering reduction rules that were already applied before. This will ensure that we need only a single pass over the instance to exhaustively reduce it, which improves the running time.

Phase one. Start by computing for each chunk $Y \in \mathcal{X}$ the value $\textsc{Conf}_F(Y)$. Since we can precompute the value $\alpha(F)$ once in linear-time, for each choice of Y we can compute $\textsc{Conf}_F(Y)$ in $\mathcal{O}(|V(F)|)$ time by marking which vertices of F are adjacent to Y, finding a MIS among the vertices of F that are not marked, and comparing its size to the precomputed value.

Now bucket-sort the chunks based on the number of conflicts they induce; since the number of conflicts is at most $|V(F)|$ we can bucket-sort in $|V(F)| + |\mathcal{X}|$

time. Then consider the chunks in decreasing value of the number of conflicts they induce and apply Rule 5.1 and Rule 5.2 where possible, using the current size of the feedback vertex set when testing for applicability.

Observe that an application of Rule 5.1 might decrease the size of the feedback vertex set X, which could cause a rule to become applicable for other chunks where it was not applicable before. By treating chunks in order of decreasing conflict value and testing for applicability of a rule when handling a chunk, we ensure that reduction rules do not become applicable to chunks we have already considered — observe that the number of conflicts induced by a chunk does not change when applying Rule 5.1 or Rule 5.2 elsewhere, except when deleting a vertex involved in some chunk (which can be handled easily).

After doing one such pass over the instance in $\mathcal{O}(|\mathcal{X}| \cdot |V(F)|) \subseteq \mathcal{O}(|X|^2 \cdot n)$ time, we end up with an equivalent instance (G_1, X_1, k_1) to which Rule 5.1 and Rule 5.2 do not apply.

Phase two. As the next phase we will exhaustively apply Rule 5.4 and Rule 5.5. The crucial fact we use here is that an application of one of these two rules does not change the number of conflicts that is induced by any chunk $Y \in \mathcal{X}$, which can be proven by arguments similar to those used to argue the correctness of the two reduction rules. Hence by applying Rule 5.4 and Rule 5.5 we do not change the fact that the instance is reduced with respect to Rule 5.1 and Rule 5.2.

It is not hard to see that a forest F can contain at most $\mathcal{O}(|V(F)|)$ structures that satisfy the degree constraints of Rule 5.4 and Rule 5.5, which follows from the fact that F is acyclic and the relevant substructures are subgraphs of constant degree. We may identify all these structures in $\mathcal{O}(|V(F)|)$ time by using a suitable depth-first search; we omit the straightforward details of such a procedure. For each substructure to which Rule 5.4 might be applied (an edge whose endpoints have degree at most two), or to which Rule 5.5 might be applied (four vertices on a path with degrees one, three, three, and one), we can test whether a rule is applicable in $\mathcal{O}(|X|^2)$ time: the effort here lies in testing whether a pair of vertices u, v from F is X-blockable. Using an adjacency-matrix for X we can test for each vertex in $N_G(u) \cap X$ whether it is adjacent to all vertices in $N_G(v) \cap X$; the pair is X-blockable if and only if this is false.

Once we determine that a rule is applicable, we modify the graph as needed. This involves modifying constant-degree constant-size substructures in F, which have arbitrary adjacencies to X. Using an appropriate data-structure such as an adjacency-list, we may perform these local modifications in $\mathcal{O}(|X|)$ time. By applying Rule 5.4 we might trigger Rule 5.5, or vice versa. Luckily, we can only trigger a rule that was not applicable before in the immediate neighborhood of the previous structure that was reduced, and we can test whether this happens in constant time. Since each reduction rule decreases the number of vertices in F, we apply the rules at most $|V(F)|$ times. Each application can be performed in $\mathcal{O}(|X|^2)$ time. By using a suitable depth-first search we can identify all structures that satisfy the degree constraints of the two rules in $\mathcal{O}(|V(F)|)$ time. In total we therefore find that from (G_1, X_1, k_1) we may compute an equivalent instance (G_2, X_2, k_2)

that is reduced with respect to rules 5.1, 5.2, 5.4 and 5.5 in $\mathcal{O}(|X|^2 \cdot |V(F)|)$ time.

Phase three. As the final phase, the algorithm needs to apply Rule 5.3. It is trivial to verify that this rule does not trigger any other reduction rules. We may apply this rule by computing for each chunk $Y \in \mathcal{X}$, for each remaining tree T in the forest, the number of conflicts induced on T by Y in a manner similar as described before. Afterwards we delete trees for which no chunks induce a conflict. This phase is easily implemented to run in $\mathcal{O}(|\mathcal{X}| \cdot |V(F)|)$ time. We output the resulting instance (G_3, X_3, k_3) of the problem, which was found in $\mathcal{O}(|X|^2 \cdot n)$ time overall. □

Theorem 5.2. Independent Set [fvs] *has a kernel with a cubic number of vertices: there is an algorithm that transforms an instance (G, X, ℓ) on n vertices and m edges into an equivalent instance (G', X', ℓ') in $\mathcal{O}(\sqrt{n}m + n^{5/3})$ time such that $|X'| \leq 2|X|$ and $|V(G')| \leq 2|X| + 28|X|^2 + 56|X|^3$.*

Proof. Given an input instance (G, X, ℓ) of Independent Set [fvs], we first apply Lemma 5.2 to obtain an equivalent *clean* instance (G_1, X_1, k_1) with $|X_1| \leq 2|X|$ in $\mathcal{O}(\sqrt{n}m)$ time. To optimize the running time of the kernelization algorithm, we do not further process the instance if $n \leq |X|^3$, but simply output (G_1, X_1, k_1) as the result of the procedure; this is suitably small since $|V(G_1)| \leq |V(G)| \leq |X|^3$ in this case.

In the remainder we may therefore assume that $n > |X|^3$, which implies $|X_1| \leq 2 \cdot n^{1/3}$ as $|X_1| \leq 2|X|$. We invoke Lemma 5.10 to obtain an equivalent reduced instance (G_2, X_2, k_2) in $\mathcal{O}(|X_1|^2 \cdot |V(G_1)|) \subseteq \mathcal{O}(n^{2/3} \cdot n)$ time. Since the reduction rules do not change the fact that the instance is clean, the reduced instance is also clean. By Lemma 5.9 the size of the resulting graph G_2 is bounded by $|V(G_2)| \leq |X_2| + 14|X_2|(|X_2| + \binom{|X_2|}{2})$. As the reduction rules do not increase the size of the feedback vertex set, we have $|X_2| \leq |X_1| \leq 2|X|$. We therefore obtain $|V(G_2)| \leq 2|X| + 28|X|^2 + 56|X|^3$ by plugging in the bound on X_2 and evaluating the binomial expression. We output the instance (G_2, X_2, k_2) as the result of the kernelization, or a trivial yes-instance if $k_2 \leq 0$. By the correctness of the reduction rules, this instance is equivalent to the input instance. The set X_2 is a feedback vertex set for G_2, since X_1 is a FVS for G_1 and the reduction rules preserve this. Observe that the original set X (or what is left of it in the final graph G_2) might not constitute a FVS for G_2, as edges may have been added between vertices that were added to the feedback vertex set in order to clean the instance. The running time of the procedure is $\mathcal{O}(\sqrt{n}m + n^{5/3})$. □

Using the previous theorem we easily obtain a corollary about kernelization for Vertex Cover from its relationship to Independent Set.

Corollary 5.1. Vertex Cover [fvs] *has a kernel with* $\min(2\ell, 2|X| + 28|X|^2 + 56|X|^3)$ *vertices, which can be computed in $\mathcal{O}(\sqrt{n}m + n^{5/3})$ time.*

Proof. Given an instance (G, X, ℓ) of Vertex Cover [fvs] we transform it into an instance $(G, X, |V(G)| - \ell)$ of Independent Set [fvs], which is an equivalent instance because the complement of an independent set is a vertex cover. We apply the kernelization algorithm from Theorem 5.2 to $(G, X, |V(G)| - \ell)$ to compute in $\mathcal{O}(\sqrt{n}m + n^{5/3})$ time an equivalent instance $(G', X', |V(G')| - \ell')$. By adjusting the target value we transform this back to an instance (G', X', ℓ') of Vertex Cover [fvs] and use it as the output, which shows that $|V(G')| \leq 2|X| + 28|X|^2 + 56|X|^3$. Since the kernelization for Independent Set [fvs] starts by applying the Nemhauser-Trotter [187] decomposition, which is known to yield a 2ℓ-vertex kernel [53], the number of vertices in the resulting graph G' is also bounded by 2ℓ, where ℓ is the size of the vertex cover that is asked for by the original input instance. □

We remark that the Vertex Cover kernelization with respect to the parameter fvs(G) can be combined with any existing Vertex Cover kernel that reduces the graph by only deleting vertices. Since the Buss rule [46], crown reductions [2, 58, 59], and the Nemhauser-Trotter reduction [53, 187], are of this type, our reduction rules can be combined with all of these. The kernelization by Kratsch and Wahlström [167], which compresses instances to small matroid representations, is incompatible with this approach.

5.2.7 Discussion of the Kernel

We have presented a cubic-vertex kernel for the Vertex Cover and Independent Set problems using the parameter fvs(G). The kernel we have presented for Vertex Cover [fvs] contains $\mathcal{O}(|X|^3)$ vertices. Since a graph G with feedback vertex set X has at most $\binom{|X|}{2} + |V(G) \setminus X| \cdot |X| + |V(G) \setminus X| - 1$ edges, a reduced instance can be encoded in $\mathcal{O}(|X|^4 \log |X|)$ bits using an adjacency-list since an adjacency-list encoding of a graph takes $\mathcal{O}(\log |V(G)| + |E(G)| \log |V(G)|)$ bits. The results of Dell and van Melkebeek [80] imply that it is unlikely that there exists a kernel that can be encoded in $\mathcal{O}(|X|^{2-\epsilon})$ bits for any $\epsilon > 0$. It might be possible to improve the size of the kernel to a quadratic or even a linear number of vertices, by employing new reduction rules. The current reduction rules can be seen as analogs of the traditional “high degree” rule for the Vertex Cover problem; it would be interesting to see whether it is possible to find analogs of crown reduction rules when using fvs(G) as the parameter.

Although we have assumed that a feedback vertex set is supplied with the input, we can drop this restriction by applying the known polynomial-time 2-approximation algorithm for FVS [11]. Observe that the reduction algorithm does not require that the supplied set X is a *minimum* feedback vertex set; the kernelization algorithm works if X is *any* feedback vertex set, and the size of the output instance depends on the size of the FVS that is supplied. Hence if we compute a 2-approximate FVS and use it in the kernelization algorithm, the bound on the number of vertices in the output instance is only a factor 8 worse than when running the kernelization using a *minimum* FVS.

Our presentation in this chapter focused on the decision version of the Vertex Cover problem, but the data reduction rules given here can also be translated to the optimization version to obtain the following result: given a graph G there is a polynomial-time algorithm that computes a graph G' and a non-negative integer c such that $\textsc{vc}(G) = \textsc{vc}(G') + c$ with $|V(G')| \leq 2\,\textsc{vc}(G)$ and $|V(G')| \in \mathcal{O}(\textsc{fvs}(G)^3)$; and a vertex cover S' for G' can be transformed back into a vertex cover of G of size $|S'| + c$ in polynomial time.

5.3 Kernel Lower Bound for Parameterization by Outerplanar Modulator

The purpose of this section is to give a superpolynomial kernel lower bound for the following parameterization of Vertex Cover.

> Vertex Cover [outerplanar mod]
> **Input:** A graph G, a modulator $X \subseteq V(G)$ such that $G - X$ is outerplanar, and an integer ℓ.
> **Parameter:** The size $k := |X|$ of the modulator.
> **Question:** Does G have a vertex cover of size at most ℓ?

While we have claimed the nonexistence of a polynomial kernel for this parameterized problem before (for example in our survey [101]), this is the first time that the lower bound appears in print. We provide it here to ensure that the complexity overview for Vertex Cover parameterizations as presented in Fig. 5.1 is fully justified by rigorous proofs.

To establish the lower bound we give a polynomial-parameter transformation from the cnf-sat problem parameterized by the number of variables (Proposition 3.1). As with the upper bound presented in the previous section, it is more intuitive to reason about the dual problem Independent Set [outerplanar mod], which is similarly defined except that we now ask for an independent set of size at least ℓ. As with the parameterization by fvs, the two dual problems are equivalent to each other from a parameterized point of view.

Theorem 5.3. Independent Set [outerplanar mod] *does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.*

Proof. We give a polynomial-parameter transformation (Definition 3.1) from cnf-sat parameterized by the number of variables n to Vertex Cover [outerplanar mod]. As the unparameterized versions of the two problems are NP-complete, this establishes the lower bound through Corollary 3.1 and Proposition 3.1. Our construction revolves around the following clause gadget.

Definition 5.10. A *clause gadget* of size t is the graph obtained from t disjoint triangles $\{a_1, b_1, c_1\}, \ldots, \{a_t, b_t, c_t\}$ by adding the edges $\{b_i, a_{i+1}\}$ for $i \in [t-1]$, and adding an extra vertex z adjacent to a_1 and b_t.

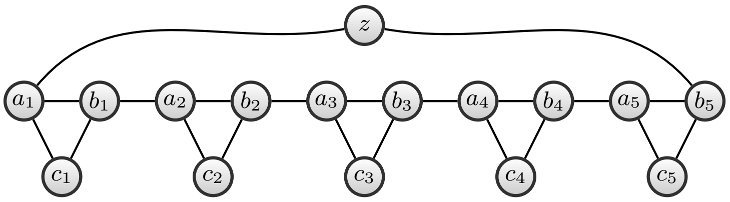


Figure 5.6: The clause gadget for the construction of Theorem 5.3. The vertices $\{c_1, \dots, c_5\}$ are terminals which will be connected to the remainder of the graph.

A clause gadget of size five is shown in Fig. 5.6. Before giving the construction we analyze the properties of such gadgets.

Claim. *A clause gadget H_t of size t has the following properties.*

(i) *The independence number $\alpha(H_t)$ is $t+1$.*
(ii) *Any independent set in H_t of size $t+1$ uses at least one vertex c_i for $i \in [t]$.*
(iii) *For any $i \in [t]$ there is an independent set in H_i of size $t+1$ that contains c_i and no other c-vertices.*
(iv) *The graph H_t is outerplanar.*

Proof. (i) An independent set contains at most one vertex of each clique. Since the vertex set of H_t can be partitioned into $t+1$ cliques (the t triangles together with the singleton $\{t\}$), the independence number of H_t is at most $t+1$. As the vertices $\{c_1, \dots, c_t\} \cup \{z\}$ form an independent set of size $t+1$, it follows that $\alpha(H_t) = t+1$.

(ii) Consider an independent set in H_t that avoids all c-vertices. Then it is also an independent set in the graph $H_t - \{c_1, \dots, c_t\}$, of the same size. But this graph is a cycle on $2t+1$ vertices, whose independence number is easily seen to be t. Hence all independent sets in H_t of size $t+1$ use at least one c-vertex.

(iii) Let $i \in [t]$ be an arbitrary index. Observe that the graph $H_t^i := H_t - (\{c_1, \dots, c_t\} \cup \{a_i, b_i\}$ is isomorphic to the path graph on $2(t-1)+1 = 2t-1$ vertices. If we relabel the vertices on H_t^i as $v_1, v_2, \dots, v_{2t-1}$ in the natural order of the path, then the t odd-numbered vertices from an independent set S in H_t^i of size t. Since the only neighbors of c_i in H_t are a_i and b_i, which are not present in H_t^i, none of the vertices in S are adjacent to c_i in H_t. Hence $S \cup \{c_i\}$ is an independent set in H_t of size $t+1$. Since no c-vertices are in H_t^i, it follows that the only c-vertex in this independent set is c_i.

(iv) Fig. 5.6 gives an outerplanar embedding of the gadget. $\diamondsuit$

Using the claim we give the construction and prove its correctness. Consider an input to CNF-SAT, which consists of clauses $C_1, \dots, C_m$, where each clause is a disjunction of literals of the form x_i or $\overline{x_i}$ for $i \in [n]$. We use $|C_j|$ to denote the number of literals in the j-th clause. Set ℓ to $n + \sum_{i=1}^{m}(|C_j| + 1)$. We build a graph G and a modulator $X \subseteq V(G)$ of size $2n$ such that $G - X$ is outerplanar. The

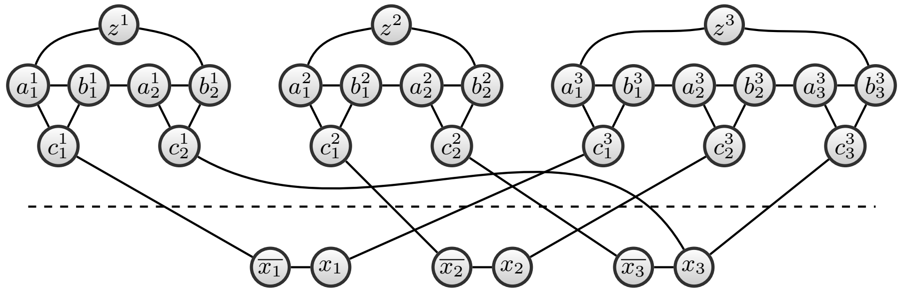

z^1
z^2
z^3
a_1^1 b_1^1 a_2^1 b_2^1
a_1^2 b_1^2 a_2^2 b_2^2
a_1^3 b_1^3 a_2^3 b_2^3 a_3^3 b_3^3
c_1^1 c_2^1 c_1^2 c_2^2 c_1^3 c_2^3 c_3^3
$\overline{x_1}$ x_1 $\overline{x_2}$ x_2 $\overline{x_3}$ x_3

Figure 5.7: Result of the construction of Theorem 5.3 for the CNF-SAT instance $(x_1 \vee \overline{x_3}) \wedge (x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$. The six vertices below the dashed line form the modulator; the remainder induces an outerplanar graph. The independent set $\{x_1, x_2, \overline{x_3}, c_1^1, a_2^1, z^1, c_1^2, a_2^2, z^2, c_3^3, b_2^3, b_1^3, z^3\}$ corresponds to the satisfying assignment that sets x_1 and x_2 to TRUE and x_3 to FALSE.

graph G will have independence number ℓ if and only if the formula is satisfiable. We construct G through the following steps, which are illustrated in Fig. 5.7.

1. For each variable $i \in [n]$ make two vertices x_i and $\overline{x_i}$ corresponding to its literals. Connect them by an edge.
2. For each clause C_j with $j \in [m]$, do the following. Add a clause gadget of size $|C_j|$ to the graph, and add the superscript j to the labels of these vertices such that, for example, the c-vertex of the fourth triangle of the gadget is c_4^j. For $k \in [|C_j|]$, consider the k-th literal in clause j.
   - If the k-th literal is positive (i.e., it is x_i for some $i \in [n]$) then make the k-th terminal vertex c_k^j adjacent to the vertex labeled $\overline{x_i}$.
   - If the k-th literal is negative, (i.e., it is $\overline{x_i}$ for some $i \in [n]$) then make the k-th terminal vertex c_k^j adjacent to the vertex labeled x_i.

The construction is concluded by setting $X := \bigcup_{i=1}^{n} \{x_i, \overline{x_i}\}$. Since the graph $G - X$ is a disjoint union of clause gadgets, the outerplanarity of a clause gadget established by Property (iv) shows that X is indeed a modulator to an outerplanar graph. Since $|X| = 2n$ the new parameter is clearly polynomial in the old parameter value. As the construction is easily performed in polynomial time, all that remains to be proven is that the two instances are equivalent.

Claim. *The* CNF-SAT *formula is satisfiable if and only if G has an independent set of size ℓ.*

Proof. ($\Rightarrow$) Suppose the formula is satisfiable, and consider a satisfying truth assignment to the variables. Construct an independent set $S_X \subseteq X$ in G by picking the vertices corresponding to the literals that are satisfied; this gives an independent set of size n. Now do the following for each clause C_j. Choose a value of k such that the k-th literal in C_j, say L, is satisfied by the truth assignment.

By construction of G, the k-th terminal vertex of the j-th clause gadget (c^j_k) was made adjacent to the vertex representing the *complement* of L. As L is satisfied, and each c-vertex has only one neighbor in X, it follows that c^j_k is not adjacent to any vertex in the independent set S_X corresponding to the truth assignment. By Property (iii) there is an independent set S_j of size $|C_j| + 1$ in the clause gadget that contains c^j_k and no other c-vertices. As the only edges between X and the clause gadgets are incident on the c-vertices, it follows that $S_j \cup S_X$ is independent in G. Since there are no edges between different clause gadgets, we may find such an independent set in each clause gadget and combine them together into an independent set in G. The resulting set $S_X \cup \bigcup_{j=1}^m S_j$ is independent in G and has size $n + \sum_{j=1}^m (|C_j| + 1)$, as required.

($\Leftarrow$) Suppose G has an independent set S of size ℓ. Observe that the vertex set of G can be partitioned into $n + \sum_{j=1}^m (|C_j| + 1)$ cliques: one clique for each adjacent pair of complementary literals, one clique for each triangle in a clause gadget, and singleton cliques for all the z-vertices in the clause gadgets. As S contains at most one vertex in each clique, the fact that $|S|$ equals the number of cliques in the partition implies that S contains exactly one vertex from each clique. Hence when considering the intersection of S with the literal pairs, the set S selects exactly one literal for each variable. We show that the corresponding truth assignment satisfies the input formula.

Since S contains n vertices in X, it contains $\sum_{j=1}^m (|C_j| + 1)$ vertices from the clause gadgets. As the independence number of the j-th clause gadget is $|C_j| + 1$ by Property (i) it follows that S contains exactly $|C_j| + 1$ vertices from each clause gadget C_j. By Property (ii) this implies that S contains some c-vertex c^j_k from each clause gadget. But then the k-th literal of the clause is satisfied by the truth assignment, as there is an edge between c^j_k and the vertex labeled by the complementary literal, while a literal corresponding to the variable is contained in S. Hence the assignment derived from $S \cap X$ is indeed a truth assignment for the formula, proving that the two instances are equivalent. $\diamondsuit$

The claim proves the correctness of the polynomial-parameter transformation. By Corollary 3.1 this establishes Theorem 5.3. $\square$

By the parameterized equivalence of Independent Set [outerplanar mod] and Vertex Cover [outerplanar mod] we immediately obtain the following corollary.

Corollary 5.2. Vertex Cover [outerplanar mod] *does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.*

5.4 Concluding Remarks

Our goal in this chapter was to analyze the kernelization complexity of Vertex Cover with respect to a hierarchy of parameters. We provided two complexity classifications: a kernelization upper bound in Section 5.2 and a superpolynomial lower bound in Section 5.3. Together with the lower bound parameterized by the vertex-deletion distance to a clique from Section 3.4.1, these results give a good idea of the limits of efficient instance compression for Vertex Cover. By relating our contributions to other results in the area, and propagating them throughout the hierarchy of parameters, we gave a complete complexity classification for the parameters in Fig. 5.1 in terms of fixed-parameter tractability and the (non)existence of polynomial kernels. The fact that polynomial kernels exist for refined parameterizations, which are typically much smaller than the solution size, may help to explain the empirically observed success of preprocessing strategies for Vertex Cover instances [1].

There are three incomparable, minimal parameters in Fig. 5.1 whose corresponding Vertex Cover parameterizations are FPT: the deletion distance to a perfect graph, the treewidth, and the excess of an optimal solution over the value of the LP-relaxation. The kernelization complexity of all these parameterizations has been settled through recent results [24, 29, 167], showing that only the last one admits a polynomial kernel (albeit randomized). It is an interesting challenge to derandomize this kernel and to implement it.

Are there relevant structural parameterizations of the Vertex Cover problem that are not included by Fig. 5.1? It would be interesting to find a parameterization that dominates the excess over the LP-relaxation, with respect to which the problem is still fixed-parameter tractable. Or is there a different type of parameterization that is incomparable to the three minimal elements named earlier? The success of recent investigations in classifying the kernelization complexity of "the usual suspects" forces us to be creative and invent novel parameterizations if we want to push the positive news to smaller parameters.

An alternative line of further research questions concerns the optimality of the obtained kernel bounds. Through the work of Dell and van Melkebeek [80], and the simplified construction of Dell and Marx [79], we know that the kernel of bitsize $\Theta(k^2)$ with respect to the natural parameterization cannot be improved to $\mathcal{O}(k^{2-\varepsilon})$ unless NP $\subseteq$ coNP/poly. For stronger parameterizations we do not have such optimal bounds yet. The kernel for the parameterization by feedback vertex size $|X|$ can be encoded in $\mathcal{O}(|X|^4 \log |X|)$ bits, and the degree of the polynomial in the randomized kernel is even higher. Can the techniques of Dell and Marx be used to give a superquadratic lower bound for a structural parameterization such as the distance from a König graph? To use their techniques to show that no kernel of bitsize $\mathcal{O}(k^{d-\varepsilon})$ exists for any $\varepsilon > 0$ unless NP $\subseteq$ coNP/poly, it suffices to embed the or of t distinct n-bit instances of an NP-hard problem into a single instance of Vertex Cover whose structural parameter k is bounded by $t^{1/d}n^{\mathcal{O}(1)}$. However, it seems difficult to create such embeddings for $d > 2$.

Introducing weights into the problem definition has a big effect on the kernelization complexity. In a different publication [145] we have shown that Weighted Vertex Cover parameterized by the cardinality of a vertex cover does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly. We omitted the result from this chapter due to space considerations. The kernelization complexity of weighted problems under structural parameterizations will be discussed in more detail in Chapter 9.

Can you beat treewidth?

—Dániel Marx, 2010

6 Treewidth

The treewidth of a graph expresses its structural likeness to a tree. The concept was discovered independently by several groups of researchers, during the nineteen eighties. It is of fundamental importance for some of the deepest results in (algorithmic) graph theory, including the Graph Minor theorem [197, 200] and Courcelle's theorem [65]. Many algorithms that exploit the treelike structure of a graph require a low-width tree decomposition as part of their input. The efficiency of such approaches therefore hinges on the method used to obtain tree decompositions. Experimental studies have shown that the use of preprocessing rules allows good tree decompositions to be found much more efficiently. Prior to our work, there was no theoretical justification for this observed success. In this chapter we give an explanation of this phenomenon in terms of polynomial kernels for structural parameterizations of TREEWIDTH. We also establish limits on the potential for efficient preprocessing.

This chapter is based on two papers: 1) "Preprocessing for Treewidth: A Combinatorial Analysis through Kernelization" by Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch (38th International Colloquium on Automata, Languages and Programming [35]), and 2) "Kernel Bounds for Structural Parameterizations of Pathwidth" by the same set of authors (13th Scandinavian Symposium and Workshops on Algorithm Theory [32]).

6.1 Rigorous Preprocessing for Treewidth Computations

In this chapter we study the decision problems related to (weighted) treewidth computations: given a (weighted) graph G and an integer ℓ, is the (weighted) treewidth of G at most ℓ? Preprocessing heuristics for Treewidth and Weighted Treewidth have been studied in a practical setting [38, 39, 91]. The experimental results reported in these papers show that there are simplification routines that give significant size reductions for many practical instances, making it more feasible to compute the treewidth of those graphs exactly or approximately. However, these heuristics do not give any guarantees on the effectiveness of the preprocessing: there is no provable bound on the size of the processed instances. The purpose of this chapter is to give a theoretical analysis of the potential of preprocessing for Treewidth. We study whether there are efficient preprocessing procedures whose effectiveness can be proven, and what the resulting size bounds look like. The investigation is formalized by analyzing kernelization for structural parameterizations of the problem.

From a theoretical point of view, the fact that Treewidth is fixed-parameter tractable [19, 170] implies by Lemma 2.1 that there is a kernel for the problem. However, the size of the implied kernel is exponential in ℓ^3 (where ℓ is the target treewidth). Bodlaender et al. [24] have shown that under a complexity-theoretic assumption (the AND-*distillation conjecture*) Treewidth with standard parameterization (i.e., parameterized by ℓ) does not admit a polynomial kernel. A recent breakthrough by Drucker [89] proved this conjecture under the assumption that NP $\not\subseteq$ coNP/poly. Hence it is unlikely that there is a polynomial-time algorithm that reduces the size of an instance (G, ℓ) of Treewidth to a polynomial in the desired treewidth ℓ.

We therefore turn to other parameters, such as the vertex cover number of the input graph, and determine whether we can efficiently shrink an input of Treewidth to a size that is polynomial in such a parameter. We consider different structural parameters of the input graph: these parameters measure the number of vertex deletions needed to transform the input into a member of some very simple graph class. All parameterized problems we consider fit the following template, where $\mathcal{F}$ is a class of graphs:

Treewidth [$\mathcal{F}$ modulator]
Input: A graph G, a positive integer ℓ, and a modulator $X \subseteq V(G)$ such that $G - X \in \mathcal{F}$.
Parameter: The size $k := |X|$ of the modulator.
Question: Is the treewidth of G at most ℓ?

The problem Weighted Treewidth [$\mathcal{F}$ modulator] is defined analogously, using the weighted variant of treewidth (see Appendix A.5.2).

Our results

We add positive theoretical results to the positive experimental findings described earlier. We first take the size of a vertex cover of G as the parameter, resulting in the problem Treewidth [vc] (which fits into the given template with $\mathcal{F}$ the class of empty graphs). We prove that this problem admits a polynomial kernel with $\mathcal{O}(k^3)$ vertices. If we are not given a vertex cover in the input, we can first compute a 2-approximation for the minimum vertex cover in polynomial time and then feed it to our kernelization algorithm. This implies that an instance (G, ℓ) of Treewidth on a graph with a minimum vertex cover of size $\textsc{vc}(G)$ can be shrunk in polynomial-time into an instance with $\mathcal{O}(\textsc{vc}(G)^3)$ vertices.

We then turn to the parameter "feedback vertex number", whose value is easily seen to be at most the vertex cover number. We extend our positive results by showing that Treewidth [fvs] (which fits the template when $\mathcal{F}$ is the class of forests) admits a kernel with $\mathcal{O}(k^4)$ vertices. By using a polynomial-time 2-approximation algorithm for Feedback Vertex Set [12], we can again drop the assumption that such a set is supplied in the input.

After these two examples it becomes an interesting question whether there is a parameter even smaller than the feedback vertex number, in which the size of an instance can be bounded polynomially after an efficient preprocessing phase. The relationships between structural graph parameters given in Fig. 6.1 show that the deletion distance (i.e., the minimum size of a modulator) to a chordal graph is a smaller parameter than the minimum size of a feedback vertex set. Since Treewidth is trivially solvable on chordal graphs, the deletion distance to a chordal graph measures the "distance from triviality" [189] of the instance. Accordingly one may hope that Treewidth [chordal mod] is FPT and admits a polynomial kernel. Unfortunately, the latter is very unlikely. We prove that even the larger parameterization by deletion distance to a single clique, fails to admit a polynomial kernel unless NP $\subseteq$ coNP/poly. We use an intricate cross-composition [29] to establish this result, starting from the NP-completeness result for Treewidth by Arnborg et al. [8]. Under the same assumption, our proof also shows that Pathwidth [clique mod] does not admit a polynomial kernel.

When it comes to the weighted version of the problem, incompressibility sets in at a much higher level in the parameter hierarchy. We prove that Weighted Treewidth does not even admit a polynomial kernel when parameterized by the size of a vertex cover, unless NP $\subseteq$ coNP/poly. Hence the presence of weights makes it difficult to preprocess instances of Treewidth, even when the graph has a very restricted structure. It is interesting to note the difference between Treewidth and Weighted Treewidth when parameterized by vertex cover.

Organization

After this introduction, we give preliminary results in Section 6.2. In Section 6.3 we show that Treewidth [vc] has a kernel with $\mathcal{O}(k^3)$ vertices. To do so, we

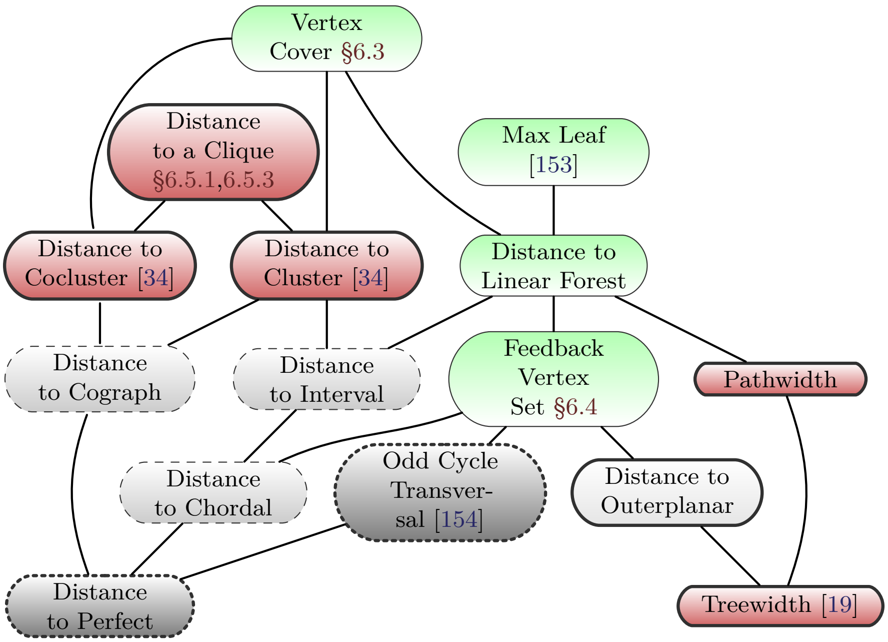


Figure 6.1: Complexity overview for various parameterizations of TREEWIDTH, assuming suitable formalizations. Larger parameters are drawn higher. For deletion distance parameters we assume that a modulator is given along with the input. The shading indicates that a parameterization is either FPT with currently unknown kernelization complexity ⬜, para-NP-complete ▨, FPT but (conditionally) lacking a polynomial kernel 🟥, FPT with a polynomial kernel 🟩, or of unknown complexity ⬚. Relationships between parameters are either given in Fig. 2.1, or follow trivially from graph class inclusions.

introduce a number of reduction rules: these are variants of rules from existing treewidth algorithms and preprocessing methods, including rules that remove simplicial vertices. In Section 6.4 we turn to TREEWIDTH [FVS] and show a kernel with $\mathcal{O}(k^4)$ vertices. In addition to variants of the rules for vertex cover, a key role will be played by *almost simplicial vertices*. We give a set of reduction rules that remove all such vertices.

In Section 6.5 we present our lower bound results. Our main lower bound, Theorem 6.7, shows that TREEWIDTH [CLIQUE MOD] does not admit a kernel of polynomial size unless NP $\subseteq$ coNP/poly. The theorem originates from our extended abstract on preprocessing for the PATHWIDTH problem [32]. The lower bound for WEIGHTED TREEWIDTH [VC] is given in Section 6.5.2. For completeness we establish in Section 6.5.3 that the problems for which we prove kernel lower bounds are fixed-parameter tractable, and therefore admit kernels of exponential size by Lemma 2.1. Some final remarks are made in Section 6.6.

6.2 Preliminaries

6.2.1 Properties of Tree Decompositions

We utilize a number of well-known facts to simplify our argumentation.

Proposition 6.1 (See e.g., [20, Lemma 16])**.** *Let H be a minor of G. Then the treewidth of H is at most the treewidth of G.*

Recall that a vertex v is *simplicial* in a graph G if $N_G(v)$ is a clique.

Proposition 6.2 ([82])**.** *Every chordal graph that is not complete contains two nonadjacent simplicial vertices.*

Lemma 6.1 (See e.g., [191])**.** *Let $S \subseteq V(G)$ be a clique in G. Let $V_1, \ldots, V_r$ be the vertex sets of the connected components of $G - S$. Then the treewidth of G equals the maximum treewidth of $G[S \cup V_i]$ for $i \in [r]$.*

Lemma 6.2 (Folklore)**.** *Let G be a graph, and let $(v_1, v_2, \ldots, v_r)$ be a path in G. Let $(T, \{\mathcal{X}_i \mid i \in V(T)\})$ be a tree decomposition of G. Suppose $i_1, i_2, i_3 \in V(T)$ and i_2 is on the path in T from i_1 to i_3. Suppose $v_1 \in \mathcal{X}_{i_1}$ and $v_r \in \mathcal{X}_{i_3}$. Then $\{v_1, v_2, \ldots, v_r\} \cap \mathcal{X}_{i_2} \neq \emptyset$.*

Lemma 6.3. *Let G be a graph, and let $W_1, \ldots, W_r$ be sets of vertices such that:*

- *For each $i \in [r]$ the set W_i induces a connected subgraph of G.*
- *For all $i, j \in [r]$ it holds that $W_i \cap W_j \neq \emptyset$ or W_i contains a vertex that is adjacent to a vertex in W_j.*

Then every tree decomposition of G has a bag containing at least one vertex of each set W_i for $i \in [r]$.

Proof. Since each W_i for $i \in [r]$ induces a connected subgraph, the bags containing a vertex in W_i form a subtree of T by Lemma 6.2. For each i and j in $[r]$, there is a bag containing a vertex of W_i and a vertex of W_j by the second condition. Using these observations the lemma follows from the Helly property for subtrees of a tree [45, Definition 1.3.4]. □

The lemma easily implies the following well-known facts.

Proposition 6.3. *If S is a clique in graph G, then any tree decomposition for G has a bag containing all vertices of S.*

Proposition 6.4. *The treewidth of a complete graph on n vertices is $n - 1$.*

Finally, we use a result of Bodlaender and Koster [38] that is based on the notion of a *minimal almost clique separator*. A set of vertices Q *separates* vertices v and w if each path from v to w uses at least one vertex in Q. A set of vertices is a *separator* if there exist vertices v and w such that Q separates v from w.

The set Q *minimally separates* v and w if it separates v and w but there is no proper subset of Q that separates v and w. It is a *minimal separator* if there is a pair of vertices v and w that are minimally separated by Q. A set of vertices Q is a *minimal almost clique separator*, if it is a minimal separator and there is a vertex $v \in Q$ such that $Q - \{v\}$ is a clique. Bodlaender and Koster showed that for the purpose of finding a minimum width tree decomposition, one may safely complete any minimal almost clique separator into a clique.

Theorem 6.1 ([38, Theorem 13 and Corollary 20]). *If S is a minimal almost clique separator in G, then $\text{TW}(G) = \text{TW}(G')$ where G' is obtained by completing S into a clique. The set of all minimal almost clique separators of a graph can be found in $\mathcal{O}(n^2m)$ time.*

6.2.2 Alternative Characterizations of Treewidth

The following theorem gives an alternative characterization of treewidth in terms of chordal supergraphs. Recall that $\omega(G)$ denotes the size of a maximum clique in G.

Theorem 6.2 ([20, Theorem 1]). *The treewidth of a graph G equals the minimum, over all chordal supergraphs H of G, of $\omega(H) - 1$.*

There is an analogous characterization for the weighted setting.

Theorem 6.3 ([42, Theorem 8]). *The weighted treewidth of a weighted graph (G, w) equals the minimum, over all chordal supergraphs H of G, of the maximum weight of a clique in H minus one.*

Treewidth can also be characterized in terms of *elimination orderings*. *Eliminating* a vertex v in a graph G is the operation of removing v while completing its open neighborhood into a clique. An elimination ordering of an n-vertex graph G is a permutation $\pi\colon V(G) \to [n]$ of its vertices. Given an elimination ordering π of G, we obtain a series of graphs by consecutively eliminating $\pi^{-1}(1), \dots, \pi^{-1}(n)$. The *fill edges of* π are the edges F_π that are added to the graph during this process when open neighborhoods are completed into cliques. The chordal supergraph of G on edge set $E(G) \cup F_\pi$ is the *filled graph* with respect to π. If F_π is empty, then the elimination ordering is *perfect*.

Theorem 6.4 ([20, Theorem 36]). *The treewidth of a graph G equals the minimum, over all elimination orderings of G, of the maximum degree of a vertex at the time it is eliminated.*

The weighted treewidth of a graph can be characterized by a weighted analogue of this statement. As the argument needed to establish this characterization is very similar to the proof of the unweighted case, we only give a sketch of the proof.

Definition 6.1. Consider a graph G weighted by function w, along with an elimination ordering π on the vertices of G. Eliminate the vertices of G in the order given by π. Then the *cost* of π is the maximum over all vertices $v \in V(G)$ of $\sum_{u \in N[v]} w(u)$ at the time that v is eliminated.

Proposition 6.5. *The weighted graph (G, w) has weighted treewidth at most ℓ if and only if there is an elimination ordering of G of cost at most $\ell + 1$.*

Proof sketch. In one direction, let G have a tree decomposition of weighted width at most ℓ. By completing the vertices of each bag into a clique we obtain a chordal graph H whose maximal cliques correspond to the bags of the decomposition. As the maximum weight of a bag is at most $\ell + 1$, the cliques in this chordal graph have weight at most $\ell + 1$. Now construct a perfect elimination ordering of G by repeatedly picking simplicial vertices and removing them from the graph; this ordering has cost at most $\ell + 1$ since for each vertex, its closed neighborhood when it is eliminated forms a clique in H.

In the reverse direction, suppose that G has an elimination ordering π of cost at most $\ell + 1$ under w. The filled chordal supergraph corresponding to the ordering has no clique of weight exceeding $\ell + 1$, as otherwise the vertex in this clique that is eliminated first would cause the cost to exceed $\ell + 1$ as in the proof of Lemma 6.13. As any chordal graph has a tree decomposition whose bags are the maximal cliques [45], no bag in such a tree decomposition has weight exceeding $\ell + 1$. Consequently such a decomposition has weighted width at most ℓ, proving the claim. □

6.3 Cubic Kernel for Treewidth Parameterized by Vertex Cover

The kernelization algorithms we present in this chapter consist of a number of reduction rules. In each case, the input to the rule is a graph G, an integer ℓ, and a modulator $X \subseteq V(G)$ such that $G - X$ is a member of the relevant graph class $\mathcal{F}$, and the output is an instance (G', ℓ', X'). A rule is correct if for all inputs (G, ℓ, X) that satisfy $G - X \in \mathcal{F}$ we have $\textsc{tw}(G) \leq \ell \Leftrightarrow \textsc{tw}(G') \leq \ell'$ and $G' - X' \in \mathcal{F}$. We will sometimes say that the algorithm answers YES or NO; this should be interpreted as outputting a constant-size YES or NO instance of the problem at hand, i.e., a clique on three vertices with $\ell = 2$, respectively the same clique with $\ell = 1$.

In this section we show our kernelization for Treewidth [vc] (i.e., parameterized by a modulator to an independent set). The kernelization focuses mostly on simplicial vertices; removing them (and possibly updating a bound for the treewidth) is a well-known and often used preprocessing rule for Treewidth; see the discussion in Bodlaender et al. [39]. Another rule, first used in the linear-time algorithm for computing treewidth [19] adds edges between vertices with many common neighbors. The rule was also used in lower bound heuristics for treewidth [22, 40, 63, 64].

Reduction Rule 6.3.1 (Low degree simplicial vertex)**.** If v is a simplicial vertex of degree at most ℓ, then remove v.

Reduction Rule 6.3.2 (High degree simplicial vertex)**.** If v is a simplicial vertex of degree greater than ℓ, then answer NO.

Standard theory on treewidth shows that Rules 6.3.1 and 6.3.2 are correct. By Proposition 6.3 the vertices of any clique in a graph G must occur together in at least one bag in any tree decomposition of G. If the clique given by the closed neighborhood of a simplicial vertex is too big (Rule 6.3.2) then we may reject; if it is sufficiently small, it is correct to add the vertex v later, since its neighbors occur in a shared bag.

Reduction Rule 6.3.3 (Common neighbors improvement)**.** Let v and w be distinct vertices such that $\{v, w\} \notin E(G)$ and $\{v, w\} \cap X \neq \emptyset$. If v and w have at least $\ell + 1$ common neighbors, then add the edge $\{v, w\}$.

Lemma 6.4. *Rule 6.3.3 is correct.*

Proof. Assume that the preconditions apply to the vertices $\{v, w\}$ of an instance (G, ℓ, X). Let G' be the graph after adding the edge $\{v, w\}$. As one of the endpoints of the added edge is contained in X, the set X is a vertex cover of G'. If G' has treewidth at most ℓ, then its minor G also has treewidth at most ℓ by Proposition 6.1.

For the reverse direction, if G has treewidth at most ℓ, then by Theorem 6.2 there is a chordal supergraph H of G with $\omega(H) \leq \ell + 1$. Consider such a chordal graph H. We prove that $\{v, w\}$ is an edge of H. Assume for a contradiction that $\{v, w\} \notin E(H)$. Then the set $N_G(v) \cap N_G(w)$ is a clique in H: if an edge, say $\{p, q\}$, between members of $N_G(v) \cap N_G(w)$ is missing, then (p, v, q, w) would form a chordless cycle in H, contradicting the fact that H is chordal. Hence $N_G(v) \cap N_G(w)$ is a clique in H of size at least $\ell + 1$. But this clique together with v forms a clique of size at least $\ell + 2$ in H, a contradiction to the assumption that $\omega(H) \leq \ell + 1$. Hence $\{v, w\}$ is indeed an edge of H. But then H is a chordal supergraph of G' whose maximum clique has size at most $\ell + 1$, proving that G' has treewidth at most ℓ by Theorem 6.2. □

Yet another simple rule is the following, using that X is a vertex cover of G.

Reduction Rule 6.3.4 (Trivial decision)**.** If $\ell \geq |X|$, then answer YES.

Correctness can be argued as follows. The treewidth of G is at most $|X|$: for each $v \in V(G) \setminus X$, take a bag with vertex set $X \cup \{v\}$, and connect these bags in any way. This gives a tree decomposition of G of width at most $|X|$.

It is not hard to argue that the exhaustive application of Rules 6.3.1–6.3.4 (i.e., until we answer NO or YES, or no application of one of these rules is possible) already gives a polynomial kernel for Treewidth [vc]. It is clear that this reduction can be performed in polynomial time; an algorithm with runtime $\mathcal{O}(nm)$ is easy to obtain.

Theorem 6.5. Treewidth [vc] *has a kernel with* $\mathcal{O}(k^3)$ *vertices.*

Proof. Let (G, ℓ, X) be an instance of Treewidth [vc]. Let (G', ℓ', X') be the instance obtained from exhaustive application of Rules 6.3.1–6.3.4. By correctness of the reduction rules (G', ℓ', X') is yes if and only if (G, ℓ, X) is yes.

The reduction rules guarantee that $X' \subseteq X$ is a vertex cover in G', with $|X'| \leq |X| = k$. Each vertex $v \in V(G') \setminus X'$ has at least one pair of distinct neighbors in X' that are not adjacent, otherwise v is simplicial and would have been handled by Rule 6.3.1 or Rule 6.3.2. Assign v to this pair. If we assign v to the pair $\{w, x\}$, then v is a common neighbor of w and x. Hence a pair cannot have more than ℓ vertices assigned to it, otherwise Rule 6.3.3 applies. As there are at most $\binom{k}{2}$ pairs of nonadjacent neighbors in X', we have $|V(G') \setminus X'| \leq \ell \cdot \binom{k}{2} \leq k \cdot \binom{k}{2} \in \mathcal{O}(k^3)$. The last step uses the fact that $\ell \leq k$ by Rule 6.3.4. □

By combining Theorem 6.5 with a polynomial-time 2-approximation algorithm for vertex cover, we obtain the following corollary.

Corollary 6.1. *There is a polynomial-time algorithm that, given an instance* (G, ℓ) *of* Treewidth, *computes an equivalent instance* (G', ℓ), *such that* $V(G') \subseteq V(G)$ *and* $|V(G')| \in \mathcal{O}(\text{vc}(G)^3)$, *where* vc$(G)$ *is the size of a minimum vertex cover of* G.

6.4 Quartic Kernel for Treewidth Parameterized by Feedback Vertex Set

In this section, we establish that Treewidth [fvs] has a kernel with $\mathcal{O}(k^4)$ vertices. The kernelization algorithm is again given by a set of reduction rules, which are applied while possible: three simple rules, three rules that remove all almost simplicial vertices (Section 6.4.1), two rules that reduce the graph when there is a clique-seeing path (defined in Section 6.4.2), and a cut-off rule that rejects when there are long clique-seeing paths left after exhaustive application of the earlier rules (Section 6.4.3). In Section 6.4.4, we show that graphs to which no rule applies have $\mathcal{O}(k^4)$ vertices, and thus arrive at our kernel bound.

The first new rule generalizes Rule 6.3.3; it was used in experiments by Clautiaux et al. [63] and its correctness is proven by Bodlaender [22, Lemma 5]. The rule can be implemented in polynomial time by using a maximum flow algorithm to find the disjoint paths.

Reduction Rule 6.4.1 (Disjoint paths improvement)**.** Let v and w be distinct vertices such that $\{v, w\} \notin E(G)$ and $\{v, w\} \cap X \neq \emptyset$. If there are at least $\ell + 1$ internally vertex-disjoint paths between v and w, then add the edge $\{v, w\}$.

The next rule also adds edges that do not affect the answer to the problem. It is based on the notion of minimal almost clique separators of Bodlaender and Koster [38] (see Section 6.2.1). They show that the treewidth is not changed when edges are added to complete a minimal almost clique separator into a clique. We use a version of this rule that ensures that X continues to be a feedback vertex set. Correctness of the following rule and an efficient implementation follow directly from Theorem 6.1.

Reduction Rule 6.4.2. Let Q be a minimal almost clique separator in G, and suppose there is at most one vertex $v \in Q$ with $v \notin X$. Then complete Q into a clique by adding edges between each pair of nonadjacent vertices in Q.

The third new rule is straightforward, and is correct because reasoning similar to that of Rule 6.3.4 shows that each graph with a feedback vertex set of size k has treewidth at most $k+1$ [20, Corollary 75].

Reduction Rule 6.4.3 (Trivial decision)**.** If $\ell \geq |X|+1$, then answer YES.

6.4.1 Almost Simplicial Vertices

Bodlaender et al. [39] introduced the notion of an *almost simplicial vertex*: a vertex v is *almost simplicial* in a graph G if v has a *special neighbor* w such that $N_G(v) - \{w\}$ is a clique. They gave a reduction rule that removes almost simplicial vertices whose degree is at most a known lower bound for the treewidth of the input graph. In this section we give a set of rules that can also remove almost simplicial vertices of higher degree. Rule 6.4.4 is a reformulation of their existing *Low Degree Almost Simplicial Vertex Rule*. Rule 6.4.5 gives a simple way to deal with almost simplicial vertices of degree larger than $\ell+1$. Our novel reduction rule is given as Rule 6.4.6; it deals with almost simplicial vertices of degree exactly $\ell+1$.

Reduction Rule 6.4.4 (Low Degree Almost Simplicial Vertex)**.** Let v be an almost simplicial vertex with special neighbor w. If the degree of v is at most ℓ, then contract the edge $\{v, w\}$ into w obtaining G'. If $v \in X$, then let $X' := X \setminus \{v\} \cup \{w\}$, else let $X' := X$.

Lemma 6.5. *Rule 6.4.4 is correct.*

Proof. It is clear that X' is a feedback vertex set of G'. Let G' be the graph resulting after the operation. If the treewidth of G is at most ℓ, then the treewidth of G' is at most ℓ as the treewidth cannot increase by contraction (Proposition 6.1).

Suppose the treewidth of G' is at most ℓ. Take a tree decomposition of G' of width at most ℓ. As $N_G(v)$ is a clique in G', by Proposition 6.3 there is a bag that contains all vertices of $N_G(v)$, say $N_G(v) \subseteq \mathcal{X}_i$. Add a new bag with vertex set $N_G[v]$ and make it adjacent in the tree decomposition to node i; we obtain a tree decomposition of G of width at most ℓ. □

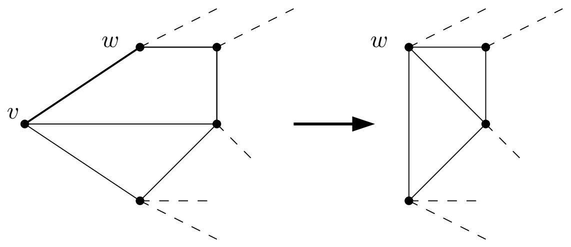


Figure 6.2: An example of an application of the *Degree* $\ell + 1$ *Almost Simplicial Vertex Rule*, second case ($\ell = 2$).

Reduction Rule 6.4.5 (High Degree Almost Simplicial Vertex)**.** Let v be an almost simplicial vertex. If the degree of v is at least $\ell + 2$, then answer NO.

Correctness is obvious: v with its neighbors except its special neighbor forms a clique with at least $\ell + 2$ vertices, so the treewidth is larger than ℓ. We introduce a new, more complex rule that deals with almost simplicial vertices of degree exactly $\ell + 1$.

Reduction Rule 6.4.6 (Degree $\ell + 1$ Almost Simplicial Vertex)**.** Let v be an almost simplicial vertex with special neighbor w, and let the degree of v be exactly $\ell + 1$.

- If for each vertex $x \in N_G(v) \setminus \{w\}$, there is an edge $\{x, w\} \in E(G)$ or a path in G from x to w that avoids $N_G[v] \setminus \{x, w\}$, answer NO.
- Otherwise, contract the edge $\{v, w\}$ to a new vertex x, obtaining G'. If $v \in X$ or $w \in X$, then let $X' := X \setminus \{v, w\} \cup \{x\}$, else let $X' := X$.

An example of an application of the second case of the *Degree* $\ell + 1$ *Almost Simplicial Vertex Rule* is given in Fig. 6.2. It is easy to see that X' is a feedback vertex set for G'. To argue correctness of Rule 6.4.6 we need two intermediate lemmata. The first one shows that deciding NO in the first case is indeed correct.

Lemma 6.6. *Let v be an almost simplicial vertex with special neighbor w, and let the degree of v be exactly $\ell + 1$. Suppose that for each vertex $x \in N_G(v) \setminus \{w\}$, there is an edge $\{x, w\} \in E(G)$ or a path in G from x to w that does not use any vertex in $N_G[v] \setminus \{x, w\}$. Then the treewidth of G is at least $\ell + 1$.*

Proof. We build a clique minor of G with $\ell + 2$ vertices. Consider the connected components of $G - N_G[v]$. Each connected component that has no vertex adjacent to w is removed. Each connected component with a vertex adjacent to w is contracted to w. We claim that this gives a clique on the vertex set $N_G[v]$, i.e., G has a clique with $\ell + 2$ vertices as a minor, and hence has treewidth at least $\ell + 1$ by Proposition 6.1 and Proposition 6.4.

Note that the only edges that are missing in $G[N_G[v]]$ are edges between vertices $x \in N_G(v) \setminus \{w\}$ and w. If $\{x, w\} \notin E(G)$, then there is a path from x

to w in G that avoids $N_G[v] \setminus \{x, w\}$. Hence, this path must belong to a connected component of $G - N_G[v]$ that is connected to w, and thus, the edge $\{x, w\}$ is present in the constructed minor. This shows the claim, and thus the lemma. □

For showing the correctness of the second step we use Lemma 6.1.

Lemma 6.7. *Let v be an almost simplicial vertex with special neighbor w. Suppose that there is a vertex $z \in N_G(v)$ with $z \neq w$ such that $N_G[v] \setminus \{z, w\}$ separates z from w. Then the treewidth of G is at most ℓ, if and only if the graph G' obtained by contracting the edge $\{v, w\}$ into w has treewidth at most ℓ.*

Proof. If the treewidth of G is at most ℓ, then the treewidth of its minor G' is at most ℓ (Proposition 6.1). It suffices to show the converse. Assume therefore that G' has treewidth at most ℓ.

Consider the set $Z := N_G[v] \setminus \{z, w\}$. By the preconditions to the lemma Z is a clique in G that separates z from w. Let $V_1, \ldots, V_r$ be the connected components of $G - Z$. Assume without loss of generality that V_1 contains the vertex w. By the separation property we know that $z \notin V_1$.

Applying Lemma 6.1 to the clique separator Z with respect to graph G we find that $\textsc{tw}(G) = \max_{i \in [r]} \textsc{tw}(G[V_i \cup Z])$. For $i \in \{2, \ldots, r\}$ it is easy to see that $G[V_i \cup Z]$ is a subgraph of G' and therefore has treewidth at most ℓ. To establish that G has treewidth at most ℓ, it remains to show that $G_1 := G[V_1 \cup Z]$ has treewidth at most ℓ.

Consider the graph $G'_1 := G'[V_1 \cup (Z \setminus \{v\}) \cup \{z\}]$. It contains a vertex labeled w (the result of the contraction), and it contains vertex z, but it does not have vertex v. As G'_1 is a subgraph of G', it has treewidth at most ℓ. We will show that G'_1 is a supergraph of G_1, resulting in the desired treewidth bound for G_1.

Observe that $N_{G'_1}(z) \supseteq N_{G_1}(v)$: in graph G, the vertex z is a neighbor of v different from the special neighbor; hence it is adjacent to all vertices in $N_G(v) \setminus \{z, w\}$, and this remains true when transforming into G'. Since $\{v, z\} \in E(G)$ the contraction of edge $\{v, w\}$ ensures that in G' (and therefore G'_1), vertex z is also adjacent to w. Thus it indeed holds that $N_{G'_1}(z) \supseteq N_{G_1}(v) = N_G(v) \setminus \{z\}$. As $z \notin V(G_1)$ one may now verify that relabeling vertex z to v in graph G'_1 yields a supergraph of G_1. This shows that $\textsc{tw}(G_1) \leq \textsc{tw}(G'_1) \leq \ell$ and concludes the proof. □

Correctness of Rule 6.4.6 now follows from Lemmata 6.6 and 6.7. Note that the rules for almost simplicial vertices (Rules 6.4.4, 6.4.5, and 6.4.6) can be easily seen to subsume the rules for simplicial vertices (Rules 6.3.1 and 6.3.2) used in the previous section. In particular, the first case of Rule 6.4.6 covers simplicial vertices of degree $\ell + 1$.

The following proposition follows from counting arguments similar to those in the proof of Theorem 6.5, by observing that after exhaustive application of the rules no leaf in the forest $G - X$ is almost simplicial, and hence every such leaf must have a pair of neighbors in X that are nonadjacent.

Proposition 6.6. *Suppose an instance (G, ℓ) of* Treewidth *is given together with a feedback vertex set X of size k. If we exhaustively apply Rules 6.4.1–6.4.6, then we obtain in polynomial time an equivalent instance (G', ℓ) with a feedback vertex set X' of size at most k, such that the forest $G' - X'$ has $\mathcal{O}(k^3)$ leaves.*

6.4.2 Clique-seeing Paths

The reduction rules presented so far ensure that the number of leaves in the forest $G - X$ is cubic in the size of the modulator X. Since the size of a forest can be bounded polynomially in the number of leaves and the maximum number of consecutive vertices of degree two, we can obtain a polynomial kernel by bounding the length of degree-two paths in the forest.

Rule 6.4.1 ensures that for each pair of nonadjacent vertices u and v in X, there are at most ℓ vertices in the forest that see both u and v. Hence the number of vertices that do not see a clique in X is $\mathcal{O}(\ell \cdot k^2) \leq \mathcal{O}(k^3)$, and it suffices to bound the length of paths in the forest $G - X$ that consist only of vertices seeing a clique in X. The following notion will therefore be the key to the reduction steps in this section.

Definition 6.2. We call a path $(v_0, v_1, \ldots, v_r, v_{r+1})$ in G a *clique-seeing path* that *sees* clique S if the following hold.

- $S = \bigcup_{i \in [r]} N_G(v_i) \setminus \{v_0, v_1, \ldots, v_{r+1}\}$ is a clique.
- For each $i \in [r]$ it holds that $N_G(v_i) \subseteq \{v_{i-1}, v_{i+1}\} \cup S$.

An example is given in Fig. 6.3. Note that v_0 and v_{r+1} play a special role. Each vertex v_i with $i \in [r]$ has exactly two neighbors outside S, namely the previous and next vertex on the path (v_{i-1} and v_{i+1}). Observe that even a path $(v_0, v_1, \ldots, v_r, v_{r+1})$ in which $N_G(v_i) \setminus \{v_{i-1}, v_{i+1}\}$ is a clique for each $i \in [r]$, might not be clique-seeing. For example, if these sets are disjoint for all $i \in [r]$ then even though the individual sets are cliques, the sets for different choices of i do not have to be adjacent to each other; but the definition of a clique-seeing path requires that the *union* of the vertices seen by the path forms a clique. Paths that fail to be clique-seeing for this reason are not a big obstacle in the kernelization: as each such path provides a connection between a pair of nonadjacent vertices, Rule 6.4.1 is triggered if there are many such paths.

In our analysis it is sufficient to consider clique-seeing paths in which all vertices on the path reside in the forest $G - X$, and all vertices in the seen clique belong to the feedback vertex set X.

Before presenting two reduction rules for clique-seeing paths, we show that the exhaustive application of Rule 6.4.2 has a useful effect on separators formed by the path and the seen clique.

Proposition 6.7. *Let $(v_0, v_1, \ldots, v_r, v_{r+1})$ be a clique-seeing path in G that sees clique $S \subseteq X$, such that $S \cup \{v_1, \ldots, v_r\}$ separates v_0 from v_{r+1}. If Rule 6.4.2*

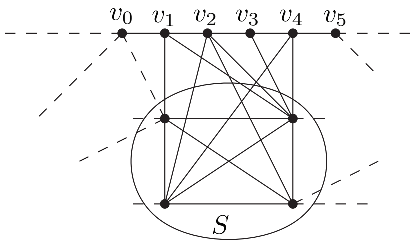


Figure 6.3: An example of a clique-seeing path.

cannot be applied, then for each $i \in [r]$ the set $\{v_i\} \cup (N_G(v_i) \cap S)$ is a clique separator that separates v_{i-1} from v_{i+1}.

Proof. Consider a clique-seeing path that satisfies the stated requirements, and let $i \in [r]$ be arbitrary. As S is a clique, the set $\{v_i\} \cup (N_G(v_i) \cap S)$ is a clique; it remains to prove that it separates v_{i-1} from v_{i+1} if the rule is not applicable.

From the structure of the clique-seeing path and the fact that $S \cup \{v_1, \ldots, v_r\}$ separates v_0 from v_{r+1}, it easily follows that $\{v_i\} \cup S$ separates v_{i-1} from v_{i+1}. Since S is a clique, the set $\{v_i\} \cup S$ is almost a clique. As we just established that it is a $v_{i-1} - v_{i+1}$ separator, it follows that it is an almost clique separator. Therefore the set $\{v_i\} \cup S$ contains a minimal almost clique separator Z that separates v_{i-1} from v_{i+1}, and Z must contain v_i because it is a common neighbor of the separated vertices. If Z is a clique, then $Z \subseteq \{v_i\} \cup (N_G(v_i) \cap S)$, which proves that the superset $\{v_i\} \cup (N_G(v_i) \cap S)$ is also a $v_{i-1} - v_{i+1}$ separator (and hence a clique separator) and we are done.

Now assume that Z is not a clique. Since the clique S is a subset of the feedback vertex set X, we know that the minimal almost clique separator Z contains at most one vertex that is not from X, and therefore Rule 6.4.2 is applicable. This proves the claim. □

Proposition 6.7 will be useful for proving the correctness of the rules we present below, and for analyzing their effects. The first rule in this section deals with clique-seeing paths that see a clique of size at most $\ell - 2$. The second considers the case that $\{v_1, \ldots, v_r\} \cup S$ separates v_0 and v_{r+1} in the graph. As the correctness proofs for these rules are fairly detailed, we postpone their proofs to the end of the section. We say that the vertices of a clique-seeing path $(v_0, v_1, \ldots, v_r, v_{r+1})$ avoid the set X if $\{v_0, v_1, \ldots, v_r, v_{r+1}\} \cap X = \emptyset$.

Reduction Rule 6.4.7. Suppose $(v_0, v_1, \ldots, v_r, v_{r+1})$ is a clique-seeing path in G, whose vertices avoid X, that sees a clique $S \subseteq X$ with $|S| \leq \ell - 2$. If

$$N_G(v_r) \cap S \subseteq \bigcup_{i \in [r-1]} N_G(v_i) \cap S$$

then contract the edge $\{v_r, v_{r+1}\}$ into the vertex v_{r+1}, obtaining G'.

The rule contracts vertex v_r, on a clique-seeing path that sees a small clique, into its neighbor v_{r+1}, when all clique-neighbors of v_r are also seen by an earlier vertex on the path. The requirement that $|S| \leq \ell - 2$ is crucial for the correctness of the rule. To see this, consider the setting where we have a clique S of size $\ell - 1$ and a path (v_0, v_1, v_2, v_3) such that v_0 sees no vertex of S, while $\{v_1, v_2, v_3\}$ see all vertices of S. In such situations, the existence of a path from v_0 to v_3 avoiding $S \cup \{v_1, v_2\}$ implies that the treewidth is at least $\ell + 1$: contracting the vertices on this path into v_3 results in a clique minor on the vertex set $\{v_1, v_2, v_3\} \cup S$ of size $|S| + 3 = \ell + 2$. Applying the rule to this structure would not be correct, as contracting the edge $\{v_r, v_{r+1}\}$ in this situation destroys the clique minor and can cause the treewidth to drop to ℓ.

This example shows that for clique-seeing paths, it is important whether the set $S \cup \{v_1, \ldots, v_r\}$ separates the extremal vertices on the path $\{v_0, v_{r+1}\}$ from each other. The next rule shows that if this is the case, we can reduce clique-seeing paths even if they see a large clique. Exhaustive application of the rule will provide us with a useful connectivity property that allows a rejection of instances with clique-seeing paths that remain long after application of Rules 6.4.1 to 6.4.8.

Reduction Rule 6.4.8. Suppose $(v_0, v_1, v_2, v_3, v_4)$ is a clique-seeing path in G, whose vertices avoid X, that sees clique $S \subseteq X$. Suppose $\{v_1, v_2, v_3\} \cup S$ separates v_0 from v_4 and suppose that Rule 6.4.2 cannot be applied. Compute the treewidth of $G[\{v_1, v_2, v_3\} \cup S]$. If it is larger than ℓ, then answer NO, otherwise remove v_2 from G.

It is not immediately obvious that given a path $(v_0, v_1, \ldots, v_r, v_{r+1})$ Rule 6.4.8 can be applied in polynomial time, when applicable. Since it is easy to verify whether the neighbors seen by the path form a clique, and whether v_0 and v_{r+1} are separated, the crucial part is to compute the treewidth of $G[\{v_1, v_2, v_3\} \cup S]$. The next lemma shows how to do this.

Lemma 6.8. *There is a polynomial-time algorithm that decides whether $G[\{v_1, v_2, v_3\} \cup S]$ has treewidth at most ℓ.*

Proof. Let us first observe that v_1 and v_3 are almost simplicial in $G[\{v_1, v_2, v_3\} \cup S]$. Thus, by two applications of the rules for almost simplicial vertices (Rules 6.4.4–6.4.6) we either get NO (since the treewidth exceeds ℓ) or a clique (with treewidth clique size minus one). In both cases we efficiently answer whether $\text{TW}(G[\{v_1, v_2, v_3\} \cup S]) \leq \ell$. □

Thus Rule 6.4.8 can be implemented in polynomial time. From exhaustive application of the rule we obtain a connectivity condition on the endpoints of clique-seeing paths, which is captured by the following proposition.

Proposition 6.8. *If Rule 6.4.2 and Rule 6.4.8 cannot be applied and there is a clique-seeing path $(v_0, v_1, \ldots, v_r, v_{r+1})$ with $r \geq 3$, whose vertices avoid X, which sees a clique $S \subseteq X$, then v_0 and v_{r+1} are connected via a path that avoids $\{v_1, \ldots, v_r\} \cup S$.*

Proof. Assume that $(v_0, v_1, \ldots, v_r, v_{r+1})$ with $r \geq 3$ avoids X and sees the clique $S \subseteq X$, and that the set $\{v_1, \ldots, v_r\} \cup S$ separates v_0 from v_{r+1}. Consider the clique-seeing subpath $(v_0, v_1, v_2, v_3, v_4)$ that sees a (not necessarily strict) subset S' of the clique S. By Proposition 6.7 the set $\{v_1\} \cup (N_G(v_1) \cap S)$ separates v_0 from v_2, and therefore separates v_0 from v_4. As $S' \supseteq N_G(v_1) \cap S$ the set $\{v_1, v_2, v_3\} \cup S'$ also separates v_0 from v_4; but then Rule 6.4.8 is applicable, a contradiction. □

Correctness of Rule 6.4.7

The intuition behind the correctness proof for Rule 6.4.7 is that a vertex v_r fulfilling the requirements can always be added at no cost in bag size or treewidth, by adding an additional bag. The detailed argumentation is unfortunately rather long, since the rule requires only a weak restriction of the graph structure; accordingly, there are many ways in which a tree decomposition for the resulting graph can be structured. Basically, we need to argue about the existence of certain bags containing the neighbors of v_r and draw conclusions that lead us to a good place for modifying the decomposition and appending bag(s) to place v_r.

Lemma 6.9. *Rule 6.4.7 is correct.*

Proof. For convenience, let us first recall the setting of Rule 6.4.7. It assumes that we have a clique-seeing path $(v_0, v_1, \ldots, v_r, v_{r+1})$ in G, whose vertices avoid X, that sees a clique $S \subseteq X$. Then, if $|S| \leq \ell - 2$ and

$$N_G(v_r) \cap S \subseteq \bigcup_{i \in [r-1]} N_G(v_i) \cap S,$$

we contract the edge $\{v_r, v_{r+1}\}$ into the vertex v_{r+1}, obtaining G'.

As the path avoids X, the contraction occurs in the forest $G - X$ and therefore the resulting graph $G' - X$ is still a forest; consequently, the set X is a feedback vertex set in G'. Let v_{r+1} be the name of the vertex resulting from the contraction. As G' is a minor of G, Proposition 6.1 shows that $\textsc{tw}(G) \leq \ell$ implies that $\textsc{tw}(G') \leq \ell$.

Now suppose that the treewidth of G' is at most ℓ. The remainder of this proof will show that the treewidth of G is at most ℓ. Consider a tree decomposition $(T, \{\mathcal{X}_i \mid i \in V(T)\})$ of G' of width at most ℓ. Recall from the definition of a clique-seeing path that S, the vertices that do not lie on path $(v_0, v_1, \ldots, v_r, v_{r+1})$ but are adjacent to an internal vertex of that path, form a clique. We show that the decomposition contains three special bags that will be useful to obtain a tree decomposition of G.

Using Lemma 6.3 (with one set for $\{v_1, \ldots, v_{r-1}\}$, and one set for each vertex in S) it follows that there is a bag $i_1 \in V(T)$ and an index $j \in [r-1]$ such that $\{v_j\} \cup S \subseteq \mathcal{X}_{i_1}$. As $\{v_{r+1}\} \cup (N_{G'}(v_{r+1}) \cap S)$ is a clique in G', there is a bag i_2 such that $\mathcal{X}_{i_2}$ contains $\{v_{r+1}\} \cup (N_{G'}(v_{r+1}) \cap S)$. From the definition of tree decomposition, and as $\{v_{r-1}, v_{r+1}\}$ is an edge in G', it follows that there must be a bag i_3 with $\{v_{r-1}, v_{r+1}\} \in \mathcal{X}_{i_3}$.

At least one of the following cases must hold:

1. Node i_3 is on the path in T from i_1 to i_2.
2. Node i_1 is on the path in T from i_2 to i_3.
3. Node i_2 is on the path in T from i_1 to i_3.
4. There is a node i_4 such that i_1, i_2, and i_3 are in different subtrees of $T - \{i_4\}$.

(The order of the cases is chosen in this way to make the proof easier to follow.)

Case 1: i_3 is on the path in T from i_1 to i_2. Note that $N_G(v_r) \cap S \subseteq (N_{G'}(v_{r+1}) \cap S) \subseteq \mathcal{X}_{i_1} \cap \mathcal{X}_{i_2}$. So $N_G(v_r) \cap S \subseteq \mathcal{X}_{i_3}$. Now add a new bag i' to the tree decomposition of G'. Make i' adjacent to i_3, and set $\mathcal{X}_{i'} := \{v_{r-1}, v_r, v_{r+1}\} \cup (N_G(v_r) \cap S)$. This is a tree decomposition of G. The new bag has size at most $3 + |S| \leq \ell + 1$ so the width is at most ℓ, which concludes this case.

This first case was convenient since it guaranteed the existence of a bag containing $N_G(v_r)$, making it possible to construct a tree decomposition for the original graph G by just appending another bag there containing $N_G(v_r) \cup \{v_r\}$. In the remaining cases we will have to restructure the tree decomposition more severely. The main idea behind each of the remaining cases will be to find a suffix $(v_{j'}, v_{j'+1}, \ldots, v_{r-1})$ of the clique-seeing path such that there exists a special bag containing $v_{j'}, v_{r+1}$, and all the neighbors that the vertices of $\{v_{j'}, \ldots, v_{r-1}\}$ have in the set S. We then build a new tree decomposition by taking the vertices of the path suffix out of the existing bags, making a path decomposition for this suffix to which we add all the path's neighbors in S, and attaching this path decomposition to the special bag. So let us now go into details.

Case 2: i_1 is on the path in T from i_2 to i_3. From the definition of a tree decomposition, it follows that $v_{r+1} \in \mathcal{X}_{i_1}$. We now modify the tree decomposition as follows:

- For j' with $j < j' \leq r-1$, remove $v_{j'}$ from all (old) bags that it appears in. (To clarify, if $j = r-1$ then we do not remove any vertices.)
- Add new bags i'_j, $i'_{j+1}, \ldots, i'_{r-1}$.
- Make bag i'_j adjacent in T to i_1.
- For j', with $j \leq j' < r-1$, make bag $i'_{j'}$ adjacent to bag $i'_{j'+1}$.
- For j', with $j \leq j' \leq r-1$, set $\mathcal{X}_{i_{j'}} := S \cup \{v_{j'}, v_{j'+1}, v_{r+1}\}$.

One can verify that this is indeed a tree decomposition of G. (E.g., the bag i'_{r-1} contains v_r and all its neighbors.) As the new bags have size bounded by $|S| + 3 \leq \ell + 1$, the width of the tree decomposition is at most ℓ.

Case 3: i_2 is on the path in T from i_1 to i_3. Consider the path from v_j to v_{r-1}. As $v_j \in \mathcal{X}_{i_1}$ and $v_{r-1} \in \mathcal{X}_{i_3}$, by Lemma 6.2 there is a j' with $j \le j' \le r-1$ such that $v_{j'} \in \mathcal{X}_{i_2}$. Consider the largest such index $j' := \max\{j^* \mid j \le j^* \le r-1 \wedge v_{j^*} \in \mathcal{X}_{i_2}\}$.

Consider a value j'' with $j' < j'' \le r-1$; we will show that any neighbors of $v_{j''}$ in the set S must be contained in $\mathcal{X}_{i_2}$. To this end, consider a vertex $w \in N_G(v_{j''}) \cap S$. There must be some bag $\mathcal{X}_{i_0}$ containing both $v_{j''}$ and w. Node i_0 must belong to the same subtree of $T - \{i_2\}$ as i_3: if this is not the case, then i_2 lies on the path from i_0 to i_3, which implies by Lemma 6.2 and the existence of the path $(v_{j''}, v_{j''+1}, \ldots, v_{r-1})$ that $\mathcal{X}_{i_2}$ contains a vertex of that path with index j'' or higher; but this contradicts our choice of j'.

Hence i_0 indeed belongs to the same subtree as i_3. Since $w \in S \subseteq \mathcal{X}_{i_1}$ and $w \in \mathcal{X}_{i_0}$, the properties of tree decompositions now ensure that $w \in \mathcal{X}_{i_2}$. As w and j'' were arbitrary this shows that $\bigcup_{j'<j''\le r-1} N_G(v_{j''}) \cap S \subseteq \mathcal{X}_{i_2}$.

Write $Z = \bigcup_{j'<j''\le r} N_G(v_{j''}) \cap S$, and observe carefully that in the definition of Z the union also includes v_r, which was excluded in the earlier formula. As $N_G(v_r) \cap S \subseteq N_{G'}(v_{r+1}) \cap S$ (as we obtained v_{r+1} from a contraction that involved v_r), we have that $Z \subseteq \mathcal{X}_{i_2}$. We now modify the tree decomposition, more or less similarly as in the previous case:

- For j'', with $j' < j'' \le r-1$, remove $v_{j''}$ from all (old) bags that it appears in.
- Add new bags $i'_{j'}, i'_{j'+1}, \ldots, i'_{r-1}$.
- Make bag $i'_{j'}$ adjacent in T to i_2.
- For j'' with $j' \le j'' < r-1$, make bag $i'_{j''}$ adjacent to bag $i'_{j''+1}$.
- For j'' with $j' \le j'' \le r-1$, set $\mathcal{X}_{i_{j''}} := Z \cup \{v_{j''}, v_{j''+1}, v_{r+1}\}$.

As in the previous case, this is a tree decomposition of width at most ℓ of G.

Case 4: There is a node i_4 such that i_1, i_2, and i_3 belong to different subtrees of $T - \{i_4\}$. The analysis is more or less similar as in Case 3. We have that $v_{r+1} \in \mathcal{X}_{i_4}$, and $N_G(v_r) \cap S \subseteq \mathcal{X}_{i_1} \cap \mathcal{X}_{i_2}$, hence $N_G(v_r) \cap S \subseteq \mathcal{X}_{i_4}$.

We have $v_j \in \mathcal{X}_{i_1}$ and $v_{r-1} \in \mathcal{X}_{i_3}$, so by Lemma 6.2 there is a j' with $j \le j' \le r-1$ such that $v_{j'} \in \mathcal{X}_{i_4}$. Assume j' is the maximum value with $j \le j' \le r-1$ and $v_{j'} \in \mathcal{X}_{i_4}$.

Consider a j'' with $j' < j'' \le r-1$. Since $v_{r-1} \in \mathcal{X}_{i_3}$, all bags containing $v_{j''}$ must belong to the same subtree of $T - \{i_4\}$ as i_3, as otherwise i_4 must contain a vertex from $\{v_{j'+1}, v_{j'+2}, \ldots, v_{r-1}\}$. All neighbors of $v_{j''}$ in S belong to some bag in the same subtree of $T - \{i_4\}$ as i_2. As $N_G(v_{j''}) \cap S \subseteq S \subseteq \mathcal{X}_{i_1}$, we have that $N_G(v_{j''}) \cap S \subseteq \mathcal{X}_{i_4}$. We can now use the same construction as in Case 3, except that we attach the new part of the tree decomposition to i_4.

Thus, in all cases, we obtain a tree decomposition of G of width at most ℓ, and conclude that the rule is correct. □

Correctness of Rule 6.4.8

Lemma 6.10. *Rule 6.4.8 is correct.*

Proof. Let us first recall the statement of Rule 6.4.8. It requires that we have a clique-seeing path $(v_0, v_1, v_2, v_3, v_4)$ in G, whose vertices avoid X, that sees a clique $S \subseteq X$. Furthermore, it requires that $\{v_1, v_2, v_3\} \cup S$ separates v_0 from v_4 and that Rule 6.4.2 cannot be applied. Then, we compute the treewidth of $G[\{v_1, v_2, v_3\} \cup S]$; if that is larger than ℓ we answer NO, else we remove v_2 from G. To prove the correctness of the rule, we establish two claims about the structure of the instance.

Claim. *For each vertex $u \notin \{v_1, v_2, v_3\} \cup S$ there is a clique separator $Z \subseteq \{v_1, v_3\} \cup S$ that separates u from v_2.*

Proof. We first identify a number of vertex sets.

- A: the vertices in the same connected component as v_0 in the graph $G - (\{v_1\} \cup (N_G(v_1) \cap S))$.
- B: the vertices in the same connected component as v_4 in the graph $G - (\{v_3\} \cup (N_G(v_3) \cap S))$.
- C: all vertices *not* in the same connected component as v_2 in the graph $G - S$.

We prove that each vertex $u \notin \{v_1, v_2, v_3\} \cup S$ is contained in at least one of A, B, C. We do this by showing that all such vertices u that are not contained in C, must be contained in A or B. So assume there is some vertex u that is not in set C because there exists a path P from u to v_2 in $G - S$. The first vertex of the clique-seeing path visited by P when starting in u, is either v_0 or v_4; there is no other way to enter the clique-seeing path. Hence $G - S$ has a subpath $P' \subseteq P$ from u to either v_0 or v_4, whose internal vertices do not belong to the clique-seeing path. As this path avoids $\{v_1, v_2, v_3\} \cup S$, which is a superset of the separators used when defining A and B, this path witnesses the fact that u belongs to A or B.

Now we are in position to prove the claim. It is easy to see that the sets $\{v_1\} \cup (N_G(v_1) \cap S)$, $\{v_3\} \cup (N_G(v_3) \cap S)$, and S, are all cliques, and subsets of $\{v_1, v_3\} \cup S$. As the preconditions to the reduction rule contain all the conditions of Proposition 6.7, the proposition shows that the set $\{v_1\} \cup (N_G(v_1) \cap S)$ separates v_0 from v_2. Consequently, it separates all vertices in A from v_2. Another application of the proposition shows that $\{v_3\} \cup (N_G(v_3) \cap S)$ separates v_2 from v_4, and therefore separates v_2 from B. Finally, the set S separates all vertices of C from v_2; this completes the proof of the claim. $\diamondsuit$

Claim. *The treewidth of G is the maximum of $\text{TW}(G - \{v_2\})$ and $\text{TW}(G[\{v_1, v_2, v_3\} \cup S])$.*

Proof. This follows from repeated application of Lemma 6.1, using the established claim that all vertices $u \notin \{v_1, v_2, v_3\} \cup S$ are separated from v_2 by clique separators that are subsets of $\{v_1, v_3\} \cup S$. $\diamondsuit$

Clearly, if the treewidth of $G[\{v_1, v_2, v_3\} \cup S]$ greater than ℓ, it is correct to answer NO. Otherwise, if its treewidth is at most ℓ, it follows from the claim that the treewidth of G is at most ℓ if and only if the treewidth of $G - \{v_2\}$ is at most ℓ. Hence the rule is correct. □

6.4.3 A Cut-off Rule

The rules presented so far shrink any YES-instance to $\mathcal{O}(k^4)$ vertices. The final rule needed to obtain the desired kernel is therefore the following cut-off rule. It rejects instances that have long clique-seeing paths despite the exhaustive application of the previous rules.

Reduction Rule 6.4.9. Suppose we have a clique-seeing path $(v_0, \ldots, v_{r+1})$ in G, whose vertices avoid X, that sees a clique $S \subseteq X$. If $r \geq 6\ell + 6$ and Rules 6.4.1–6.4.8 are not applicable, then answer NO.

A detailed counting argument, spread over the following two lemmata, shows the correctness of this rule.

Lemma 6.11. *Let $(v_0, \ldots, v_{r+1})$ be a clique-seeing path, seeing clique S. Suppose that v_0 and v_{r+1} belong to the same connected component of $G - (\{v_1, \ldots, v_r\} \cup S)$. Let Z be the vertex set of the connected component of $G - (\{v_1, \ldots, v_r\} \cup S)$ that contains v_0. If one of the following cases holds, then G has treewidth at least $\ell + 1$.*

1. *$|S| \geq \ell + 1$.*
2. *$|S| = \ell$ and each vertex in S has a neighbor in Z.*
3. *$|S| = \ell - 1$ and each vertex in S has a neighbor in Z, and there is an $s \in [r-1]$, such that each vertex in S is adjacent to at least one vertex in $\{v_1, \ldots, v_s\}$, and each vertex in S is adjacent to at least one vertex in $\{v_{s+1}, \ldots, v_r\}$.*

Proof. In each case G contains a clique of size $\ell + 2$ as a minor, and hence has treewidth at least $\ell + 1$, by Proposition 6.1. In the first case, contract the interior $\{v_1, \ldots, v_r\}$ of the path to a single vertex; it then forms a clique with the vertices in S. In the second case, contract the interior of the path to a single vertex, and contract Z to a single vertex; these form a clique with the vertices in S. In the last case, take S, and contract each of the following to a single vertex: all vertices in Z, $\{v_1, \ldots, v_s\}$, and $\{v_{s+1}, \ldots, v_r\}$. □

Lemma 6.11 sets up sufficient conditions for finding a clique minor of size $\ell + 2$, which rules out having treewidth at most ℓ. The following lemma uses this knowledge to bound the length of clique-seeing paths depending on the size of the corresponding clique seen by the path. If such a path is too long (in comparison to the size of its clique), then we can select an appropriate subpath and a connected component Z as in the assumption of Lemma 6.11 and may conclude that there is a $\ell + 2$-clique minor (which permits us to correctly answer NO).

Lemma 6.12. *Suppose that none of Rules 6.4.1–6.4.8 can be applied. Let* $(v_0, v_1, \ldots, v_{r+1})$ *be a clique-seeing path in* G*, whose vertices avoid* X*, that sees a clique* $S \subseteq X$*.*

1. *If* $|S| \leq \ell - 2$ *then* $r \leq |S| + 1 \leq \ell - 1$*.*
2. *If* $|S| = \ell - 1$ *and* $r \geq 3\ell + 3$ *then the treewidth of* G *is greater than* ℓ*.*
3. *If* $|S| = \ell$ *and* $r \geq 6\ell + 6$ *then the treewidth of* G *is greater than* ℓ*.*
4. *If* $|S| \geq \ell + 1$ *then the treewidth of* G *is greater than* ℓ*.*

Proof. We consider the cases consecutively.

1. Suppose $|S| \leq \ell - 2$. In particular, Rule 6.4.7 cannot be applied. Thus for each i with $1 < i \leq r$:
$$N_G(v_i) \cap S \not\subseteq \bigcup_{j=1}^{i-1} N_G(v_j) \cap S.$$
Hence, via induction it follows that $|\bigcup_{j=1}^{i} N_G(v_j) \cap S| \geq i + 1$, and therefore that $r \leq |\bigcup_{j=1}^{r} N_G(v_j) \cap S| \leq |S| + 1$.
2. Suppose $|S| = \ell - 1$ and $r \geq 3\ell + 3$. Let
$$S_1 = \bigcup_{j=1}^{\ell+1} N_G(v_j) \cap S, \quad S_2 = \bigcup_{j=\ell+2}^{2\ell+2} N_G(v_j) \cap S, \quad S_3 = \bigcup_{j=2\ell+3}^{3\ell+3} N_G(v_j) \cap S.$$
If $S_1 \neq S$, then $(v_0, v_1, \ldots, v_{\ell+1}, v_{\ell+2})$ is a clique-seeing path with $\ell + 1$ internal vertices and seeing clique S_1, with $|S_1| < \ell - 1$, contradicting Case 1. Similarly, we get a contradiction when $S_2 \neq S$ or $S_3 \neq S$.
Therefore, assume $S_1 = S_2 = S_3 = S$. Consider the path $(v_0, \ldots, v_{2\ell+3})$ seeing clique S. By Proposition 6.8 there is a path from v_0 to $v_{2\ell+3}$ that avoids $\{v_1, \ldots, v_{2\ell+2}\} \cup S$. Hence, the component Z containing v_0 in $G - (\{v_1, \ldots, v_{2\ell+2}\} \cup S)$ also contains $v_{2\ell+3}$, and hence $\{v_{2\ell+3}, \ldots, v_{3\ell+3}\} \subseteq Z$. Thus by $S_1 = S_2 = S_3 = S$ each vertex of S has a neighbor in each of $\{v_1, \ldots, v_{\ell+1}\}$, $\{v_{\ell+2}, \ldots, v_{2\ell+2}\}$, and $\{v_{2\ell+3}, \ldots, v_{3\ell+3}\} \subseteq Z$. Therefore, the conditions for the last case of Lemma 6.11 are fulfilled for $(v_0, \ldots, v_{2\ell+3})$ and $s = \ell + 1$. It follows that G has treewidth at least $\ell + 1$.
3. Suppose $|S| = \ell$ and $r \geq 6\ell + 6$. Let
$$S_1 = \bigcup_{j=1}^{3\ell+3} N_G(v_j) \cap S \text{ and } S_2 = \bigcup_{j=3\ell+4}^{6\ell+6} N_G(v_j) \cap S.$$
If $S_1 \neq S$, then considering the clique-seeing path $(v_0, v_1, \ldots, v_{3\ell+3}, v_{3\ell+4})$ we get a contradiction to Case 1 if $|S_1| \leq \ell - 2$ or the claim follows from Case 2 if $|S_1| = \ell - 1$. This can be argued similarly when $S_2 \neq S$.
Therefore, $S_1 = S_2 = S$. Now consider the path $(v_0, \ldots, v_{3\ell+4})$. By Proposition 6.8 there is a path from v_0 to $v_{3\ell+4}$ in $G - (\{v_1, \ldots, v_{3\ell+3}\} \cup S)$. It

follows that the connected component Z of v_0 in that graph also contains the vertices $v_{3\ell+4}, \dots, v_{6\ell+6}$. Thus the path $(v_0, \dots, v_{3\ell+4})$ fulfills the conditions of the second case of Lemma 6.11, and we conclude that the treewidth of G is at least $\ell + 1$.

4. This follows directly from the first case of Lemma 6.11. □

Correctness of Rule 6.4.9 follows immediately from the lemma: if the clique-seeing path has at least $6\ell + 6$ internal vertices, then (assuming that none of Rules 6.4.1–6.4.8 can be applied) it sees a clique of size at least $\ell - 1$ (by Case 1). Cases 2–4 then show that answering NO is correct.

6.4.4 The Kernelization Algorithm

Now we are ready to present the kernelization algorithm. It consists of the exhaustive application of Rules 6.4.1 through 6.4.9.

Theorem 6.6. TREEWIDTH [FVS] *has a kernel with* $\mathcal{O}(k^4)$ *vertices.*

Proof. Let (G, ℓ, X) be an instance of TREEWIDTH [FVS] and let (G', ℓ, X') be obtained from exhaustive application of Rules 6.4.1 through 6.4.9. Observe that Rules 6.4.7–6.4.9 are only tested for paths in $G - X$; as that is a forest, we need to test at most $\mathcal{O}(n^2)$ paths. Thus the test has to be done a polynomial number of times. We showed that all rules are correct and that they can be performed exhaustively in polynomial time. Hence the instances are equivalent, X' is a feedback vertex set of G', and $|X'| \leq |X| = k$.

Let us analyze the size of G'. The forest $G' - X'$ has $\mathcal{O}(k^3)$ leaves by Proposition 6.6, and hence $\mathcal{O}(k^3)$ vertices of degree at least three. There are $\mathcal{O}(k^3)$ paths in the forest whose endpoints are leaves or vertices of degree at least three. Each path of length at least $6\ell + 8$, i.e., with at least $6\ell + 6$ internal vertices is not clique-seeing by Rule 6.4.9. Thus, for the analysis, we split the paths into parts of size $6\ell + 8$ that are therefore not clique-seeing. At most $6\ell + 7$ vertices per path will not belong to such a part, but these are at most $\mathcal{O}(k^3 \cdot \ell)$ in total. We assign each part to a pair of nonadjacent vertices in X that are adjacent to internal vertices of the part. We can assign at most ℓ parts to a pair $\{u, v\}$: indeed, since the parts are disjoint they would otherwise give rise to more than ℓ disjoint $u-v$ paths, contradicting (G', ℓ, X') being reduced under Rule 6.4.1. Thus, we have $\mathcal{O}(\ell(6\ell + 8)k^2) + \mathcal{O}(k^3 \cdot \ell) = \mathcal{O}(k^4)$ vertices in the forest $G' - X'$, and thus $\mathcal{O}(k^4)$ vertices in G'. □

Using a polynomial-time 2-approximation algorithm for feedback vertex set [12] we obtain a corollary similar to the one given in the previous section.

Corollary 6.2. *There is a polynomial-time algorithm that given an instance* (G, ℓ) *of* TREEWIDTH *computes an equivalent instance* (G', ℓ) *such that* $V(G') \subseteq V(G)$ *and* $|V(G')| \in \mathcal{O}(\text{FVS}(G)^4)$*, where* $\text{FVS}(G)$ *is the size of a minimum feedback vertex set of* G*.*

6.5 Kernel Lower Bounds

In this section we present our lower bound results for TREEWIDTH and WEIGHTED TREEWIDTH. We employ the lower bound framework of Chapter 3. For completeness we show in Section 6.5.3 that the considered problems are all fixed-parameter tractable, and therefore admit (superpolynomial) kernelizations.

6.5.1 Kernel Lower Bound for Treewidth Parameterized by Clique Modulator

In this section we show that TREEWIDTH [CLIQUE MOD] does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly. The problem is covered by the general template given in the introduction, when using $\mathcal{F}$ as the class of all cliques. Observe that $\mathcal{F}$ only contains *connected* graphs and is therefore not closed under disjoint union.

To prove the lower bound we employ the technique of cross-composition: we show that CUTWIDTH ON SUBCUBIC GRAPHS cross-composes into TREEWIDTH [CLIQUE MOD]. Since the cutwidth problem is still NP-complete under the stated restriction [183, Corollary 2.10], this will give a lower bound using Corollary 3.2. The cutwidth problem we use is formally defined as follows:

CUTWIDTH ON SUBCUBIC GRAPHS (CUTWIDTH3)
Input: A graph G on n vertices, in which each vertex has degree at most three, and an integer $\ell \leq |E(G)|$.
Question: Is there a linear layout of G of cutwidth at most ℓ, i.e., a permutation $\pi \colon V(G) \to [n]$ of $V(G)$ such that $\max_{i \in [n]} |\{\{u,v\} \in E(G) \mid \pi(u) \leq i < \pi(v)\}| \leq \ell$?

To obtain a kernel lower bound through cross-composition, we have to embed the logical OR of a series of t input instances of CUTWIDTH3 on n vertices each into a single instance of the target problem for a parameter value polynomial in $n + \log t$. At the heart of our construction lies an idea of Arnborg et al. [8] employed in their NP-completeness proof for TREEWIDTH. They interpreted the treewidth of a graph as the minimum cost of an elimination ordering on its vertices, and showed how for a given graph G a cobipartite graph G^* can be created such that the cost of elimination orderings on G^* corresponds to the cutwidth of G under a related ordering. Our composition extends their construction significantly. We express it in terms of weighted treewidth, and afterwards use a simple transformation to reduce it to the unweighted setting.

Before giving the proof, we sketch the main ideas. By the degree bound, instances with n vertices have $\mathcal{O}(n^2)$ different degree sequences. The framework of cross-composition thus allows us to work on instances with the same degree sequence (and same ℓ). By enforcing that the structure of one side of the cobipartite graph G^* only has to depend on this sequence, all inputs can share the same "right hand side" of the cobipartite graph; this part will remain small and act as the modulator. By a careful balancing of weight values we then enforce that the costs of elimination orderings on the constructed graph G^* are dominated by eliminating the vertices corresponding to exactly one of the input instances. This ensures that a sufficiently low treewidth is already achieved when the answer to one of the input instances is YES. On the other hand, the use of a binary-encoding representation of instance numbers ensures that low-cost elimination orderings for G^* do not mix vertices corresponding to different input instances.

We recall some results for WEIGHTED TREEWIDTH that will be useful to prove the lower bound. The following lemma is a weighted analogue of a result proven by Bodlaender et al. [27]. They showed that for any clique B in a graph G, there is an optimal elimination ordering that ends with B. The lemma will be useful when proving the correctness of our cross-composition, because it allows us to assert the existence of well-behaved optimal elimination orderings for the constructed instance. A related statement in the restricted setting of cobipartite graphs was already employed by Arnborg et al. [8] in their NP-completeness proof.

Lemma 6.13 (Cf. [27, Lemma 4])**.** *Let G be a graph weighted by function w containing a clique $B \subseteq V(G)$, and let $A := V(G) \setminus B$. There is a minimum-cost elimination ordering π^* of G that eliminates all vertices of A before eliminating any vertex of B.*

Proof. Let π be a minimum-cost elimination ordering of G with cost ℓ. The filled graph of G with respect to π is a chordal supergraph H of G. We claim that the maximum weight of a clique in H is bounded by ℓ. To see this, assume that H contains a clique $C \subseteq V(H)$ of weight more than ℓ. Let v be the first vertex of the clique that is eliminated by ordering π. Then the ordering incurs cost $\sum_{u \in C} w(u)$ when eliminating v: the edges between v and the other members of C must already have been filled at that point, since they cannot become part of the filled graph after v is eliminated. Hence the existence of a clique of weight more than ℓ would imply that π has cost more than ℓ, a contradiction.

Construct a perfect elimination ordering π' of H as follows. As long as A is nonempty, there must be a vertex in A that is simplicial in H: if H is a clique then any vertex of A is simplicial, and if H is not a clique then by Proposition 6.2 there are two nonadjacent simplicial vertices in H. These cannot both belong to B, since B is a clique; hence at least one belongs to A. We pick such a simplicial vertex in A, remove it from the graph, and repeat. Once A is empty, end with the vertices in the clique B in arbitrary order; they are all simplicial. We end up with a perfect elimination ordering π' of H: since we only pick vertices that are simplicial at the time they are picked, they produce no fill edges in H. This property implies that when a vertex is eliminated under π' in H, its closed neighborhood forms a clique in H and therefore has weight at most ℓ.

Now consider the cost of using ordering π' on the weighted graph G. Since G is a subgraph of H, for each vertex v its neighborhood at the time of elimination is a subset of what would have been its neighborhood on elimination from the graph H. Since the closed neighborhood weight in H is at most ℓ, the weight of the neighborhood when eliminating it from G cannot be larger. Hence π' is an elimination ordering of G with cost at most ℓ under w. As it eliminates all vertices of A before eliminating any vertex in B, this concludes the proof. □

By considering the effect of updating an ordering such that it eliminates vertices with smaller neighborhoods earlier, we may observe the following.

Observation 6.1. *Let G be a graph with weight function w containing two adjacent vertices u, v such that $N_G[u] \subseteq N_G[v]$. Let π be an elimination ordering of G that eliminates v before u, and let the ordering π' be obtained by updating π such that it eliminates u just before v. Then the cost of π' is not higher than the cost of π.*

Finally, we will need the following transformation to reduce the weighted problem to the unweighted setting.

Proposition 6.9. *Let G be a graph with positive integral vertex weights given by function w. Let G' be the graph obtained from G by iterating the following procedure: as long as there is a vertex v with weight more than one, subtract one from the weight of v and add a new vertex v' of weight one and with open neighborhood $N[v]$. Then the treewidth of G' equals the weighted treewidth of G under w.*

Proof. Given a tree decomposition of G, replace the weighted vertices in each bag by the unweighted copies. A bag of total weight ℓ is transformed into a bag of size ℓ, and it is easy to verify that the decomposition satisfies all relevant conditions.

In the reverse direction, consider a tree decomposition of the unweighted graph G' of width ℓ. Make a tree decomposition of G on the same host tree, putting a weighted vertex v in a bag if all its unweighted copies are contained in that bag. As the total size of a bag in the unweighted decomposition is at most $\ell+1$, the maximum weight of the corresponding bag in the weighted decomposition is at most $\ell + 1$. The convexity property of the original decomposition ensures that all vertices induce connected subtrees. To see that each edge of the weighted graph is covered by this decomposition, note that for an edge $\{v, w\}$ in G the unweighted vertices corresponding to v and w together form a clique in G'. Hence by Proposition 6.3 there is a bag containing all unweighted vertices corresponding to v or w, and this bag will cover the edge $\{v, w\}$ in the resulting decomposition of the weighted graph. □

Theorem 6.7. TREEWIDTH [CLIQUE MOD] *does not admit a polynomial kernelization unless NP $\subseteq$ coNP/poly.*

Proof. We show that the NP-complete CUTWIDTH3 problem cross-composes into TREEWIDTH [CLIQUE MOD]. We start by defining a polynomial equivalence relation $\mathcal{R}$. Fix an encoding of instances of CUTWIDTH3, and choose $\mathcal{R}$ such that all strings that do not encode a valid instance are equivalent. For the strings that *do* encode a valid instance, define two instances (G_1, ℓ_1) and (G_2, ℓ_2) to be equivalent if all of the following hold: $\ell_1 = \ell_2$, $|V(G_1)| = |V(G_2)|, |E(G_1)| = |E(G_2)|$, and for each integer $i \in \{0, 1, 2, 3\}$ the number of degree-i vertices in G_1 and G_2 is the same. Since a set of valid instances on at most n vertices each is partitioned into at most $n \times n \times n^2 \times n^3$ equivalence classes, this constitutes a polynomial equivalence relation.

We now show how to cross-compose a set of instances of Cutwidth3 that belong to the same equivalence class of $\mathcal{R}$. If all instances are malformed, then this can be recognized in polynomial time and we simply output a constant-size NO-instance. So in the remainder we may assume that all input instances (G_1, ℓ_1), $\ldots, (G_t, \ell_t)$ are well-formed and belong to the same equivalence class; in particular $\ell_1 = \ldots = \ell_t = \ell$ and $|V(G_1)| = \ldots = |V(G_t)| = n$. Order the vertices within each graph by increasing degree, breaking ties arbitrarily. The choice of $\mathcal{R}$, together with the fact that each G_i has maximum degree three, guarantees that each graph has the same number of vertices of each degree.

Since Cutwidth on a graph on n vertices can be solved in $\mathcal{O}^*(2^n)$ time [26, Theorem 10], we may assume that $n \geq \log t$. For if $n < \log t$ then applying the algorithm by Bodlaender et al. [26] consecutively on each instance can be done in time polynomial in the total input size (which is at least t); we could then give a constant-size instance with the appropriate answer as the output of the cross-composition. For similar reasons we may assume $n \geq 2$. Finally, we may assume that the number of input instances t is a power of two, since we can duplicate some instances without changing the value of the OR, increasing the input size by at most a factor two.

To construct the instance of Treewidth [clique mod] that encodes the OR of the input instances, we use a two-stage process. We first show that the OR of the input instances can be encoded into an instance of Weighted Treewidth [clique mod] on a cobipartite graph with partite sets A and B, such that the total weight of the set B is polynomial in $n + \log t$. The set B will be the modulator, which is valid since removing the partite set B from a cobipartite graph leaves a clique. We then use Proposition 6.9 to obtain an equivalent instance of Treewidth [clique mod], and since the total weight of B is sufficiently small this produces an instance of Treewidth [clique mod] that encodes the OR of the input instances, and has a modulator to a single clique whose size is polynomial in $n + \log t$.

We now construct a graph G^* and weight function w^* such that computing the weighted treewidth of G^* corresponds to computing the OR of the instances of Cutwidth3. The construction is based on the NP-completeness proof for Treewidth by Arnborg et al. [8]. The graph G^* will be cobipartite with partite sets A and B, so $V(G^*) := A \cup B$ and A and B are cliques in G^*. The graph G^* is defined as follows:

- For each input graph G_i with $i \in [t]$, for each vertex $j \in V(G_i)$, we add a vertex $v_{i,j}$ of weight n^3 to A which corresponds to vertex j. For a given value of $j \in [n]$ we say that the vertices $v_{i,j}$ (for all relevant values of i) are *A-representatives* of node j. We also add a *dummy* vertex d_i for each instance $i \in [t]$ to A of weight n^6. We turn A into a clique.

- The vertex set B consists of three parts: the *instance selector vertices B_I*, the *node representatives B_N*, and the *edge representatives B_E*.
  - The instance selector vertices will be used to encode the binary representation of instance numbers. Since we assumed t to be a power of two, we need $\log t$ bits to encode an instance number and therefore $2 \log t$ vertices are used to represent all possible bit values for $\log t$ positions. So $B_I := \{a_q, b_q \mid q \in [\log t]\}$. Each vertex in B_I has weight n^5.
    We connect the vertices of B_I to the vertices of A as follows. We make a vertex $v_{i,j}$ in A (which corresponds to instance i) adjacent to the instance selector vertices of the bit values of the binary representation of i. So for $q \in [\log t]$, if the q-th bit of number i is 1 then we make $v_{i,j}$ adjacent to a_q, and if the bit is 0 then we make the vertex adjacent to b_q. The adjacency from dummy vertices d_i to the vertices of B_I is defined exactly the same through the binary representation of i.
  - The node representatives B_N contain a vertex for each node number in $[n]$. Recall that all input graphs have the same number of vertices of each degree, and that we sorted the vertices by degree. When we write $\deg(j)$ for $j \in [n]$ we will therefore take this to mean the value d such that in each input graph, the j-th vertex has degree d. For each $j \in [n]$ we add a vertex x_j to the set B_N and give it weight $n^3 - \deg(j)$. The vertex x_j is said to be the (unique) *B-representative* of node j.
    The adjacency between B_N and A is simple: for each $j \in [n]$ we make all A-representatives of j adjacent to the single B-representative of j, and we make all the nodes in B_N adjacent to all the dummy vertices d_i for $i \in [t]$.
  - The edge representatives B_E contain one vertex for each possible edge in an undirected n-vertex graph. So for $\{v, w\} \in \binom{[n]}{2}$ we have a vertex $e_{v,w}$ of weight two. Vertex $e_{v,w}$ is adjacent to an A-representative $v_{i,j}$ if instance G_i contains the edge $\{v, w\}$ and $j = v$ or $j = w$, i.e., the edge representative $e_{v,w}$ is adjacent to instance i's A-representatives of the endpoints of the edge, provided that instance i actually contains the edge. Additionally, all vertices of B_E are adjacent to all dummy vertices d_i for $i \in [t]$.

The construction is completed by turning $B := B_I \cup B_N \cup B_E$ into a clique. We set $\ell' := t \cdot (n^4 + n^6) + n^3 + n^5 \log t + \ell - 1$.

To complete the first stage, we need to prove that (G^*, w^*) has weighted treewidth at most ℓ' if and only if at least one of the input graphs G_i has cutwidth at most ℓ. Before proving this correspondence between the input and the constructed instance (G^*, w^*, ℓ'), we establish some properties of the latter.

Claim 6.1. *Let $S := \{v_{i,j} \mid j \in [n]\}$ for a given instance number $i \in [t]$ be the subset of the vertices in A corresponding to instance i. Let $\pi \colon S \to [n]$ be a permutation of S. Consider the process of eliminating the vertices in S from graph G^* in the order given by π, and let* E-WEIGHT$(\pi^{-1}(j))$ *be the total weight of $N[\pi^{-1}(j)]$ when eliminating the vertex $\pi^{-1}(j)$ for $j \in [n]$. Then* E-WEIGHT$(\pi^{-1}(j)) = t \cdot (n^4 + n^6) + n^3 + n^5 \log t + \ell$*, where $\ell := |\{\{u,v\} \in E(G_i) \mid \pi(u) \leq j < \pi(v)\}|$.*

Proof. The intuition behind the proof is that the elimination process has two effects on the weight of neighbors of some vertex $v \in A$: on the one hand, eliminated vertices in A are essentially replaced by the representatives in B_N in the neighborhood of v, which have slightly smaller weight than the originals; the difference is exactly equal to the degree of the corresponding vertex. On the other hand, the representative of any edge in B_E will be added to those neighborhoods, once one of the endpoints is eliminated; recall that those edges contribute a weight of two. Thus, when reaching the first endpoint of an edge, the weight increases by one (by the degree contribution); when reaching the second endpoint this increase is canceled. Together this leads to the contribution of ℓ in E-WEIGHT$(\pi^{-1}(j))$. This idea was used by Arnborg et al. [8].

Armed with this intuition, let us proceed with the proof. By definition of G^*, all vertices in S have the same set of neighbors in B_I so elimination of vertices from S does not affect the adjacency of other vertices in S to B_I. Consider a vertex $v_{i,j}$ in S. From the construction of G^* it follows that initially, the only vertex of S that is adjacent to the B-representative of j, is the vertex $v_{i,j}$. Since we only eliminate vertices from S, it follows that a vertex in S is only adjacent to the B-representative of a node number j if that vertex is itself the unique A-representative of j in S, or if the A-representative of j in S was eliminated earlier. We use these observations to prove the claim.

For an arbitrary value of $j \in [n]$ we consider the closed neighborhood of the vertex $\pi^{-1}(j)$ just before it is eliminated. We will study the neighborhood of $\pi^{-1}(j)$ in the sets A, B_I, B_N and B_E consecutively. For convenience, define E-WEIGHT$^Y(\pi^{-1}(j))$ for $Y \subseteq V(G^*)$ as the total weight of $N[\pi^{-1}(j)] \cap Y$ when $\pi^{-1}(j)$ is eliminated.

Neighbors in A. Since A is a clique and $S \subseteq A$, vertex $\pi^{-1}(j)$ is initially adjacent to all vertices of A. Since the only vertices that are eliminated are those in S corresponding to instance i, vertex $\pi^{-1}(j)$ will be adjacent to all vertices for other instances, i.e., to $v_{i',j}$ for $i' \neq i$ and $j \in [n]$, for a total weight of $(t-1)n \cdot n^3$. Vertex $\pi^{-1}(j)$ is also adjacent to all t dummy vertices for a weight of n^6 each. The remaining vertices of A are those in S, and $\pi^{-1}(j)$ is adjacent to those that are not already eliminated. Hence there are $n - j + 1$ vertices in S that are in the closed neighborhood of $\pi^{-1}(j)$ just before it is eliminated. These have weight $n^3(n - j + 1)$ so E-WEIGHT$^A(\pi^{-1}(j)) = (t-1)n^4 + t \cdot n^6 + n^3(n - j + 1)$.

Neighbors in B_I. Since the neighborhood of $\pi^{-1}(j)$ in B_I is not changed by the eliminations, vertex $\pi^{-1}(j)$ has exactly $\log t$ neighbors in B_I with weight n^5 each so E-WEIGHT$^{B_I}(\pi^{-1}(j)) = n^5 \log t$.

Neighbors in B_N. By construction of G^*, vertex $\pi^{-1}(j)$ is initially adjacent to the unique node in B_N that is the B-representative of $\pi^{-1}(j)$, and to no other vertices of B_N. For each vertex $1 \leq j' < j$ that was eliminated before j, vertex $\pi^{-1}(j)$ has become adjacent to the B-representative of $\pi^{-1}(j')$ in B_N. So E-WEIGHT$^{B_N}(\pi^{-1}(j)) = \sum_{j'=1}^{j}(n^3 - \deg(\pi^{-1}(j')))$.

Neighbors in B_E. Initially, vertex $\pi^{-1}(j)$ is adjacent to the edge-representative vertices in B_E for which $\pi^{-1}(j)$ represents an endpoint, so to $\deg(\pi^{-1}(j))$ vertices with weight two each. For each vertex $\pi^{-1}(j')$ with $1 \leq j' < j$ that is eliminated before $\pi^{-1}(j)$, vertex $\pi^{-1}(j)$ becomes adjacent to the edge-representatives in B_E for edges that are incident on $\pi^{-1}(j')$ in graph G_i. This shows that E-WEIGHT$^{B_E}(\pi^{-1}(j)) = 2|\bigcup_{1 \leq j' \leq j}\{e_{u,v} \mid \{u,v\} \in E(G_i) \wedge (j' = u \vee j' = v)\}|$.

We can now sum up the weights of the members of the closed neighborhood $N[\pi^{-1}(j)]$ in each of the respective subsets to establish that E-WEIGHT$(\pi^{-1}(j))$ equals:

$$\begin{aligned}
&\text{E-WEIGHT}^{A}(\pi^{-1}(j)) + \text{E-WEIGHT}^{B_I \cup B_N \cup B_E}(\pi^{-1}(j)) \\
&= [(t-1)n^4 + t \cdot n^6 + n^3(n-j+1)] + [n^5 \log t] + \left[\sum_{j'=1}^{j}(n^3 - \deg(\pi^{-1}(j')))\right] \\
&\quad + 2\left|\bigcup_{1 \leq j' \leq j}\{e_{u,v} \mid \{u,v\} \in E(G_i) \wedge (j' = u \vee j' = v)\}\right| \\
&= t \cdot (n^4 + n^6) + n^3 + n^5 \log t - \sum_{j'=1}^{j} \deg(\pi^{-1}(j')) \\
&\quad + 2\left|\bigcup_{1 \leq j' \leq j}\{e_{u,v} \mid \{u,v\} \in E(G_i) \wedge (j' = u \vee j' = v)\}\right| = \ldots
\end{aligned}$$

To simplify this further, we define E_1 as the set of edges of G_i that have one endpoint among the vertices represented by $\{\pi^{-1}(1), \ldots, \pi^{-1}(j)\}$, and E_2 as the edges of G_i with both endpoints represented by $\{\pi^{-1}(1), \ldots, \pi^{-1}(j)\}$. Observe that these definitions imply that $\sum_{j'=1}^{j} \deg(\pi^{-1}(j')) = |E_1| + 2|E_2|$ and $|\bigcup_{1 \leq j' \leq j}\{e_{u,v} \mid \{u,v\} \in E(G_i) \wedge (j' = u \vee j' = v)\}| = |E_1| + |E_2|$. We continue the derivation:

$$\begin{aligned}
\ldots &= t \cdot (n^4 + n^6) + n^3 + n^5 \log t - (|E_1| + 2|E_2|) + 2(|E_1| + |E_2|) \\
&= t \cdot (n^4 + n^6) + n^3 + n^5 \log t + |E_1|.
\end{aligned}$$

Now observe that by definition, $|E_1|$ is the number of edges that have exactly one endpoint among the first j vertices to be eliminated (and whose other endpoint is therefore eliminated later). Hence $|E_1|$ is exactly the value of ℓ as defined in the statement of the claim. This concludes the proof of Claim 6.1. $\Diamond$

The preceding claim relates the cost of the first n eliminations of an ordering of G^* to the cutwidth of an instance i, provided that the ordering starts by eliminating the A-representatives of instance i. The next claim shows that these first n eliminations essentially dominate the cost of elimination orderings with this structure.

Claim 6.2. *Let S, i, π, and* E-WEIGHT *be as defined in Claim 6.1. Consider an elimination ordering for G^* that starts by eliminating S in the order given by π, then eliminates the dummy d_i corresponding to instance i, and eliminates the remaining vertices in arbitrary order. The cost of π is* $\max_{j \in [n]} \text{E-WEIGHT}(\pi^{-1}(j))$.

Proof. By Claim 6.1, the maximum weight of a closed neighborhood when eliminating the vertices from S is exactly $\max_{j\in[n]} \text{E-WEIGHT}(\pi^{-1}(j)) \geq t \cdot (n^4 + n^6) + n^3 + n^5 \log t$. We show that after elimination of S, eliminating the dummy d_i and all remaining vertices does not incur a cost higher than this.

Consider the weight of the closed neighborhood of the dummy vertex d_i after the n vertices from S have been eliminated. At that stage, d_i is adjacent to all vertices that are left in A, to all vertices of B_N, some vertices of B_E, and to the $\log t$ vertices in B_I that correspond to the binary representation of the number i. Since the total weight of B_N is not more than $n \cdot n^3$, the weight of $N[d_i]$ when d_i is eliminated is bounded by $t(n^4+n^6) - n^4 + n \cdot n^3 + n^5 \log t + 2\binom{n}{2}$, which is at most $t(n^4 + n^6) + n^3 + n^5 \log t$ and so does not exceed the cost incurred for the first n vertices as $n \geq 2$.

After d_i and S have been eliminated from the graph, the total weight of the remaining vertices is at most $(t-1)(n^4 + n^6) + 2n^5 \log t + n^4 + 2\binom{n}{2}$, which is bounded by $t(n^4 + n^6) + n^3 + n^5 \log t$ as we assumed $n \geq \log t$ at the beginning of the proof. Hence the cost of this elimination ordering π is dominated by the cost of eliminating the first n vertices and is therefore $\max_{j\in[n]} \text{E-WEIGHT}(\pi^{-1}(j))$. $\Diamond$

Having shown how the cost of specific types of elimination orderings of G^* corresponds to the cutwidth of one particular input instance, we proceed to show that there is always an optimal ordering of this type.

Claim 6.3. *If there is an elimination ordering of (G^*, w^*) of cost at most ℓ', then there is such an ordering that starts by eliminating all vertices in the set $\{v_{i,j} \mid j \in [n]\}$ for some $i \in [t]$, i.e., there is an ordering that first eliminates all A-representatives corresponding to* one *particular input instance G_i.*

Proof. The proof contains of two parts. We first give a canonical elimination ordering of bounded cost, and then use a replacement argument that compares the cost of this canonical ordering to any ordering that does not match the form in the statement of the lemma.

Define a canonical elimination ordering π^* as follows. It starts with the sequence $v_{1,1}, v_{1,2}, \ldots, v_{1,n}$, then eliminates dummy d_1, and finally eliminates the rest of the vertices in arbitrary order. By Claim 6.2 the cost incurred by ordering π^* is $\max_{j \in [n]} \text{E-WEIGHT}(\pi^{-1}(j))$. Observe that the cutwidth of graph G_1 under any ordering does not exceed the number of its edges, which is at most $3n/2$ since G_1 has maximum degree at most three. Hence we find that the term ℓ in the expression for E-WEIGHT given by Claim 6.1 is bounded by $3n/2$. Using Claim 6.1 we therefore find that the cost of this canonical elimination ordering π^* is bounded by $t \cdot (n^4 + n^6) + n^3 + n^5 \log t + 3n/2$.

To complete the proof, we will show that any elimination ordering whose form does not match that in the statement of the claim, has cost as least as much as the canonical ordering. So consider any elimination ordering π of (G^*, w^*) of cost at most ℓ'. Since B is a clique in G^*, by Proposition 6.13 there is an optimal-cost elimination ordering that first eliminates all of A. So assume without loss of generality that π first eliminates all vertices of A. Since the construction of G^* guarantees that for all $i \in [t]$, all vertices $v_{i,j}$ of A satisfy $N_{G^*}[v_{i,j}] \subseteq N_{G^*}[d_i]$, Proposition 6.1 shows that we may assume without loss of generality that for all $i \in [t]$, the vertices $v_{i,j}$ for $j \in [n]$ are eliminated by π earlier than d_i; hence the first n vertices eliminated by π are not dummy vertices. If the first n vertices correspond to the same instance then we are done; hence in the remainder we may assume this is not the case. Consider the first index $1 < j \leq n$ such that all vertices $\pi^{-1}(j')$ for $1 \leq j' < j$ correspond to the same instance i (i.e., they are of the form $v_{i,j''}$ for $j'' \in [n]$) and $\pi^{-1}(j)$ corresponds to instance i' with $i \neq i'$. Let us consider the neighborhood of the vertex $\pi^{-1}(j)$ when it is eliminated.

By construction of G^*, vertex $\pi^{-1}(j)$ corresponding to instance i' is adjacent to the vertices in B_I that correspond to the binary representation of i'. Since vertex $\pi^{-1}(1)$ was eliminated before $\pi^{-1}(j)$, and since vertices $\pi^{-1}(1)$ and $\pi^{-1}(j)$ are adjacent in G^* because they are both members of the clique A, after elimination of $\pi^{-1}(1)$ the vertex $\pi^{-1}(j)$ has become adjacent to all neighbors of $\pi^{-1}(1)$. Since $\pi^{-1}(1)$ is adjacent to the vertices of B_I corresponding to the binary representation of i, and since the binary representations of i and i' must differ in at least one position, the number of neighbors of $\pi^{-1}(j)$ in B_I at the time it is eliminated is at least $1 + \log t$, and they have weight n^5 each. Since $\pi^{-1}(j)$ is also adjacent to all vertices of A except the $j - 1$ vertices of weight n^3 that were eliminated earlier, this shows that the weight of the closed neighborhood of $\pi^{-1}(j)$ at the time it is eliminated is at least $t(n^4 + n^6) - j \cdot n^3 + (1 + \log t)n^5$. Using that $j \leq n$ and $n \geq 2$ (which we assumed in the beginning the proof of the theorem), it now follows that the weight of $\pi^{-1}(j)$ at the time it is eliminated is at least as much as the cost of the canonical elimination ordering. Hence the canonical elimination ordering that we defined earlier has cost no more than π, and since the canonical ordering starts by eliminating $v_{1,1}, v_{1,2}, \ldots, v_{1,n}$ this concludes the proof of Claim 6.3. $\diamondsuit$

We are now finally ready to prove that the constructed instance acts as the OR of the inputs.

Claim 6.4. *The weighted graph (G^*, w^*) has weighted treewidth at most ℓ' if and only if at least one of the input graphs G_i has cutwidth at most ℓ.*

Proof. ($\Rightarrow$) First assume that (G^*, w^*) has weighted treewidth at most ℓ'. By Proposition 6.5 this implies that there is an elimination ordering π of G^* with cost at most $\ell'+1$. By Claim 6.3 we may assume that there is an instance number $i^* \in [t]$ such that π starts by eliminating all vertices in the set $S := \{v_{i^*,j} \mid j \in [n]\}$. As the cost of π is at most $\ell' + 1$, the weight of the closed neighborhood of a vertex in S at the time it is eliminated does not exceed $\ell' + 1$. By Claim 6.1 this proves that $\max_{j \in [n]} \text{E-WEIGHT}(\pi^{-1}(j)) \leq \ell' + 1$. Plugging in the value for ℓ' and the expression for E-WEIGHT obtained in the mentioned claim, and canceling terms on both sides, we find that $\max_{j \in [n]} |\{\{u, v\} \in E(G_{i^*}) \mid \pi(u) \leq j < \pi(v)\}| \leq \ell$, which proves that G_{i^*} has cutwidth at most ℓ, when using the ordering on S induced by π.

($\Leftarrow$) For the reverse direction, assume that G_{i^*} has cutwidth at most ℓ. Let π^* be an ordering that achieves this cutwidth. Build an elimination ordering for G^* by first eliminating the vertices of $S := \{v_{i^*,j} \mid j \in [n]\}$ in the order induced by π^*, then eliminating the dummy d_{i^*}, and then eliminating the remaining vertices in arbitrary order. By Claim 6.2 the cost of this ordering is dominated by the cost of eliminating the vertices of S, which is $\max_{j \in [n]} \text{E-WEIGHT}(\pi^{-1}(j))$. If ordering π^* achieves cutwidth at most ℓ on G_{i^*}, then evaluating the expression for E-WEIGHT given by Claim 6.1 proves that the cost of π is at most $\ell'+1$. Using Proposition 6.5 this proves that (G^*, w^*) has weighted treewidth at most ℓ'. ♢

To complete the cross-composition of CUTWIDTH3 into TREEWIDTH [CLIQUE MOD], we can transform the weighted graph (G^*, w^*) into the unweighted graph G' using the transformation of Proposition 6.9. Since this transformation duplicates the closed neighborhoods of vertices, it results in a cobipartite graph: the cliques A and B of G^* are just transformed into larger cliques in G'. Let B' be the clique in G' that results from the transformation of clique B in G^*. The size of B' is bounded by the maximum weight of a vertex in B (under w^*) times the size of B. Since both are polynomial in $n + \log t$, this shows that the size of B' is bounded polynomially in $n + \log t$. The set B' is a modulator to a single clique in G', as B' is one of the partite sets of the cobipartite graph. Now consider the instance of TREEWIDTH [CLIQUE MOD] that asks if G' with the modulator B' to a single clique has treewidth at most ℓ'; by the equivalence between the weighted treewidth of the original graph, and the unweighted treewidth of the result of the transformation, our constructed instance is equivalent to the OR of the input instances of CUTWIDTH3. The size of the modulator, which is the parameter of the TREEWIDTH [CLIQUE MOD] instance, is polynomial in $n+\log t$. This concludes the cross-composition; Theorem 6.7 follows by applying Corollary 3.2. □

Since the pathwidth of a cobipartite graph equals its treewidth [182] and the graph formed by the cross-composition is cobipartite, we obtain the following corollary.

Corollary 6.3. Pathwidth [clique mod] *does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.*

For completeness we remark that in the extended abstract [32] containing Theorem 6.7, we also proved that Pathwidth [vc] has a kernel with $\mathcal{O}(k^3)$ vertices. It is currently unknown whether Pathwidth [fvs] admits a polynomial kernelization.

6.5.2 Kernel Lower Bound for Weighted Treewidth Parameterized by Vertex Cover

The Weighted Treewidth problem asks, given a vertex-weighted graph G and integer ℓ, whether the weighted treewidth of G is at most ℓ. Preprocessing heuristics for the problem were considered by van den Eijkhof et al. [91]. We consider a parameterized version of this problem, where we choose the size of a vertex cover as the parameter:

> Weighted Treewidth [vc]
> **Input:** A graph G, a weight function $w\colon V(G) \to \mathbb{N}$ encoded in unary, an integer ℓ, and a vertex cover $X \subseteq V(G)$ of G.
> **Parameter:** The size $k := |X|$ of the vertex cover.
> **Question:** Does G have a tree decomposition of weighted width at most ℓ with respect to w?

We show that Weighted Treewidth [vc] does not admit a polynomial kernelization unless NP $\subseteq$ coNP/poly and the polynomial hierarchy collapses. Our lower bound proof is based on the effect of the join operation on the treewidth. We therefore use the following lemma due to Bodlaender and Möhring [41].

Lemma 6.14 ([41])**.** *For any two graphs G_1 and G_2 it holds that:*

$$\textsc{tw}(G_1 \otimes G_2) = \min(\textsc{tw}(G_1) + |V(G_2)|, \textsc{tw}(G_2) + |V(G_1)|).$$

From Lemma 6.14, we directly obtain the following corollary.

Corollary 6.4. *For any t graphs $G_1, \ldots, G_t$ on n vertices each, it holds that:*

$$\textsc{tw}(G_1 \otimes \ldots \otimes G_t) = (t-1) \cdot n + \min_{i \in [t]} \textsc{tw}(G_i).$$

We also use the following weighted variant of a lemma by Bodlaender [19].

Lemma 6.15. *Let G be a graph, let ℓ be an integer, and let u, v be two vertices of G whose shared neighbors have combined weight at least $\ell + 1$. Then in any tree decomposition of G of weighted width at most ℓ there is a bag containing both u and v.*

A proof of Lemma 6.15 follows easily by adapting the proof of Lemma 6.4: replace Theorem 6.2 in the argument by its weighted variant Theorem 6.3.

Theorem 6.8. Weighted Treewidth [vc] *does not admit a polynomial kernelization unless NP $\subseteq$ coNP/poly.*

Proof. We prove the theorem by giving a cross-composition from the classic (unweighted) Treewidth problem into Weighted Treewidth [vc].

By a suitable choice of polynomial equivalence relation $\mathcal{R}$, similar to that of Theorem 6.7, it suffices to show how to compose a series of t instances $(G_1, \ell), \ldots, (G_t, \ell)$ of the classical Treewidth problem that all ask for the same target value ℓ, and in which each input graph has the same number of n vertices and m edges. Additionally, we may assume without loss of generality that $t, n \geq 10$.

The cross-composition can be interpreted in terms of the graph $H := G_1 \otimes \ldots \otimes G_t$ obtained by joining all input graphs together. Let $\ell' := (t-1)n + \ell$. By Corollary 6.4 there is an input graph of treewidth at most ℓ if and only if $\textsc{tw}(H) \leq \ell'$. The joined graph H therefore encodes the logical or of the input instances. We want to exploit this fact, but the graph H does not have a small vertex cover. We therefore use a construction that simulates the edges in the graph H by adding a small number of high-weight vertices to the graph, effectively allowing these edges to be dropped from H at the expense of introducing $\mathcal{O}(n^2 \log t)$ new vertices that will form a vertex cover. Rather than replacing each edge by one high-weight vertex — which would cause the vertex cover to become too large — we introduce a constant number of vertices for each clique in an edge clique cover of H, and subsequently show that H has a small edge clique cover.

Claim 6.5. *Let (G, ℓ) be an instance of* Treewidth, *and let $\mathcal{C} = \{C_1, \ldots, C_r\}$ be an edge clique cover of G whose largest clique has size at most c. Let d be a positive integer, and let G' be the graph with weight function w' constructed as follows:*

- *Initialize G' as a graph without edges on the vertex set $V(G)$. Set the weight of all vertices to one.*
- *For each $i \in [r]$ add two vertices to G', of weight d, with open neighborhood C_i.*

If $2d > \ell$ and $c + d \leq \ell$, then the weighted treewidth of G' is at most ℓ if and only if $\textsc{tw}(G) \leq \ell$. Moreover, the $2r$ newly added vertices X' form a vertex cover of the graph G'.

Proof. As all edges from G are dropped, the $2r$ new vertices form a vertex cover of G'. We prove the correspondence between the treewidth values when $2d > \ell$ and $c + d \leq \ell$.

($\Leftarrow$) Let $\mathcal{T} = (T, \{\mathcal{X}_i \mid i \in V(T)\})$ be a tree decomposition of G of width ℓ. We show how to incorporate the newly added vertices in this decomposition without increasing its width. So consider a vertex v_i with open neighborhood C_i, which was added to G' on behalf of a clique C_i in the cover of G. As C_i is a clique in G, there is a bag $\mathcal{X}_i$ of $\mathcal{T}$ that contains all vertices of C_i by Proposition 6.3. Hence we may accommodate v_i by creating a new bag containing $C_i \cup \{v_i\}$, and making the bag adjacent to $\mathcal{X}_i$. Let $\mathcal{T}'$ be the result of consecutively adding all new vertices to $\mathcal{T}$ in this way. As each of the newly added vertices occurs in only one bag, we satisfy the convexity property and obtain a valid tree decomposition of G'. Each new bag that we introduce has weight at most $c + d \leq \ell$ since each clique has at most c vertices, all of weight one, and newly added vertices have weight d. Therefore the decomposition has weighted width no larger than ℓ. We conclude that the weighted treewidth of G' is at most ℓ.

($\Rightarrow$) Consider a tree decomposition $\mathcal{T}' = (T', \{\mathcal{X}'_i \mid i \in V(T)\})$ of G' having weighted width at most ℓ. We show that restricting each bag to the vertex set of G yields a valid tree decomposition of G. As the weighted width of $\mathcal{T}'$ is at most ℓ, and each vertex has weight at least one, this restriction has unweighted width at most ℓ. It is easy to see that the restriction operation satisfies the convexity property of a tree decomposition, and that each vertex of G is present in some bag. It therefore suffices to show that for each edge $\{u, v\}$ of G, the restriction has a bag containing both endpoints. To this end, let C_i be a clique in the edge clique cover containing u and v. The two vertices that were added to G' on behalf of clique C_i have combined weight $2w$, and are adjacent to both u and v. Therefore the common neighbors of u and v have weight at least $2d > \ell$, which shows by Lemma 6.15 that $\mathcal{T}'$ has a bag containing u and v. The restriction therefore also has such a bag, which shows that all edges are covered by the decomposition. $\diamondsuit$

A small edge clique cover of H can be constructed efficiently.

Claim 6.6. *Given t graphs $G_1, \ldots, G_t$ on n vertices and m edges each, we can construct an edge clique cover $\mathcal{C}$ of the graph $G_1 \otimes \ldots \otimes G_t$ consisting of $\mathcal{O}(n^2 \log t)$ cliques in polynomial time, such that each clique in the cover has at most $2t$ vertices.*

Proof. We first construct m cliques to cover all the edges contained in one of the input graphs; later we show how to cover the join edges that are added between the different graphs.

We number the edges in each input graph from 1 to m, and cover them one edge index at a time. For an edge index $j \in [m]$ let C_i contain the endpoints of the j-th edge in each input graph. Since vertices originating from different input graphs are made adjacent by the join, each C_i is a clique in the joined graph. Every clique created in this way has at most two vertices from each input graph, hence at most $2t$ vertices in total. It is easy to see that all edges of the input graphs are covered by this series of $m \leq n^2$ cliques.

To cover the join edges, we use a strategy based on binary expansions. For each $r \in [\lceil \log t \rceil]$ and each combination of $p, q \in [n]$ (including $p = q$) define a clique $C_{r,p,q}$ as follows. The clique contains exactly one vertex from each input graph G_i. If the r-th bit of the binary expansion of the number i is a zero, then $C_{r,p,q}$ contains the p-th vertex from G_i. If the bit is a one, the clique contains the q-th vertex from G_i. By the nature of the join operation, all sets defined in this way are cliques. The resulting family trivially consists of $\mathcal{O}(n^2 \log t)$ cliques.

It remains to show that all edges between different input graphs are covered by the family.

To see this, consider an edge between the p-th vertex of input G_i, and the q-th vertex of input $G_{i'}$. As $i \neq i'$ the binary expansions of i and i' differ in at least one bit position r. From the given construction it now follows easily that either $C_{r,p,q}$ or $C_{r,q,p}$ contains both vertices. Hence for each join edge there is a clique that covers them. It is easy to see that the construction can be performed in polynomial time. This concludes the proof of the claim. $\diamondsuit$

The final construction combines the two claims. Presented with the inputs $(G_1, \ell), \ldots, (G_t, \ell)$ the cross-composition builds the graph $H = G_1 \otimes \ldots \otimes G_t$, and uses Claim 6.6 to find an edge clique cover $\mathcal{C}$ of H consisting of $r \in \mathcal{O}(n^2 \log t)$ cliques. Let c be the size of a largest clique in the cover; Claim 6.6 guarantees that $c \leq 2t$. Recall that $\ell' = (t-1)n + \ell$, and let $d := \ell' - 2t$. Using $t, n \geq 10$ it is straightforward to verify that $d \geq 1$ and $2d > \ell'$. It is trivial that $c + d \leq \ell'$, which shows that we may apply Claim 6.5 to $G_1, \ldots, G_t$ and $\mathcal{C}$ with this choice of d. Applying the construction in the claim, the algorithm obtains a weighted graph (G', w') with a vertex cover X' of size $2r \in \mathcal{O}(n^2 \log t)$, which has weighted treewidth at most ℓ' if and only if H has treewidth at most ℓ'. By Lemma 6.14 the latter happens if and only if $\textsc{tw}(G_i) \leq \ell$ for some $i \in [t]$. Hence the instance (G', X', ℓ', w') is a valid output instance of Weighted Treewidth [vc] with $|X'| \in \mathcal{O}(n^2 \log t)$. As the construction can be carried out in polynomial time, and the output parameter is suitably bounded, this construction implies Theorem 6.8 using Corollary 3.2. $\square$

6.5.3 FPT Classifications for (Weighted) Treewidth with Structural Parameterizations

We justify the kernelization lower bounds given in this section by showing that the corresponding problems are fixed-parameter tractable. To this end we provide a short argument based on modular decomposition (cf. [45, §1.5]). A *nontrivial module* in a graph G is a nonempty vertex set $S \subsetneq V(G)$ such that for each vertex $u \in V(G) \setminus S$ it either holds that $N_G(u) \cap S = S$ or $N_G(u) \cap S = \emptyset$. A graph is *prime* if it has no nontrivial modules.

Proposition 6.10 (Compare [42, Corollary 13]). Treewidth *on an n-vertex graph G can be solved in polynomial time using an oracle for the* Weighted Independent Treewidth *problem on induced prime subgraphs of G whose vertices have weight at most n each.*

The statement is a variant of Corollary 13 in the work of Bodlaender and Rotics [42], tailored towards FPT classifications. The precise definition of Weighted Independent Treewidth is not important for our purposes; it suffices to know that it is a problem on weighted graphs that can be solved using a polynomial-time computation on each weighted chordal supergraph of the input.

Hence there is a computable function f such that WEIGHTED INDEPENDENT TREEWIDTH can be solved in $f(k)w^{\mathcal{O}(1)}$ time on graphs with k vertices, each having weight at most w. To leverage Proposition 6.10 to obtain FPT algorithms for structural parameterizations of TREEWIDTH, the following result relates the sizes of prime subgraphs and vertex covers.

Proposition 6.11. *If $X \subseteq V(G)$ is a vertex cover of G, then no prime subgraph of G has more than $|X| + 2^{|X|}$ vertices.*

Proof. If H is a prime subgraph of G with more than $|X| + 2^{|X|}$ vertices, then H contains at least $2^{|X|} + 1$ vertices from $V(G) \setminus X$. Since the number of possible neighborhoods of $V(G) \setminus X$ into X is bounded by $2^{|X|}$, such a subgraph H contains two vertices $u, v \notin X$ with the same neighborhood into X. As u and v are not contained in the vertex cover, all their neighbors belong to X. It follows that $\{u, v\}$ is a nontrivial module of H, contradicting the assumption that it is prime. □

Corollary 6.5. *The problems* TREEWIDTH [CLIQUE MOD] *and* WEIGHTED TREEWIDTH [VC] *are fixed-parameter tractable.*

Proof. We combine propositions 6.10 and 6.11, using the existence of an algorithm that solves WEIGHTED INDEPENDENT TREEWIDTH on k-vertex graphs with weights at most w in $f(k)w^{\mathcal{O}(1)}$ time.

Observe that a graph is prime if and only if its edge-complement is prime. Since a clique modulator X in G forms a vertex cover in $\overline{G}$, Proposition 6.11 gives a bound on the sizes of prime subgraphs of G in terms of G's deletion distance to a clique. For an instance (G, ℓ, X) of TREEWIDTH [CLIQUE MOD] we therefore know that all induced prime subgraphs of G have at most $|X| + 2^{|X|}$ vertices. Hence an oracle for computing WEIGHTED INDEPENDENT TREEWIDTH on induced prime subgraphs of G whose weights are at most n, can be implemented in $f(|X| + 2^{|X|})n^{\mathcal{O}(1)}$ time. Applying Proposition 6.10 therefore classifies TREEWIDTH [CLIQUE MOD] as FPT.

When it comes to the weighted variant of treewidth, observe that by Proposition 6.9 the weighted treewidth of G with weight function w equals the (unweighted) treewidth of the graph G' obtained by replacing each vertex $v \in V(G)$ by a clique on $w(v)$ vertices. Such a replacement step does not affect the size of prime subgraphs: no induced prime subgraph can contain two members of the same replacement clique, as they would form a nontrivial module. Proposition 6.11 therefore shows that if the weighted graph G has a vertex cover X, then the size of induced prime subgraphs of the corresponding unweighted graph G' is at most $|X| + 2^{|X|}$. By the definition of WEIGHTED TREEWIDTH the instances of that problem encode the vertex weights in unary, implying that G' can be obtained efficiently and that $|V(G')|$ is polynomial in the size of (G, w). We therefore obtain an FPT-algorithm for WEIGHTED TREEWIDTH [VC] by invoking Proposition 6.10 on G', using the fact that we can give an FPT-time oracle for WEIGHTED INDEPENDENT TREEWIDTH on the induced prime subgraphs of G' as they are guaranteed to be small. □

The approach used here cannot establish fixed-parameter tractability for Treewidth [cograph mod]. We can show that there are prime graphs G of arbitrary size that can be turned into cographs by just two vertex deletions. Hence there is no analogue of Proposition 6.11 in the cograph setting.

6.6 Concluding Remarks

We considered different parameterizations for the Treewidth problem and obtained both positive and negative results for the existence of polynomial kernels. Our first positive result, a cubic-vertex kernel for Treewidth [vc], is interesting as the algorithm largely consists of elements of existing preprocessing heuristics for Treewidth. Thus it also sheds new light on the experimentally observed success of these heuristics.

Our second positive result, a polynomial kernel for Treewidth [fvs], is not only interesting from a theoretical point of view but also has practical potential. We expect that some of the reduction rules — in particular, our new rule for almost simplicial vertices of degree $\ell + 1$ (Rule 6.4.6) — are effective when preprocessing inputs from applied problem domains such as probabilistic networks.

The lower bound for the parameterization by deletion distance from a clique rules out polynomial kernels for Treewidth for a number of possibly interesting parameters; among them are the distance from cographs, cluster, cocluster, chordal, interval, and split graphs. Furthermore, our lower bound for Weighted Treewidth [vc] rules out essentially all hope for useful kernelization results for the Weighted Treewidth problem, since neither almost-complete nor almost-independent graphs appear to admit a polynomial kernelization.

Apart from improving the kernel sizes (e.g., for parameterization by a feedback vertex set), or giving polynomial lower bounds (e.g., $\Omega(k^2)$ bits), it seems interesting to analyze the parameterized complexity of other structural parameterizations. The hierarchy of Fig. 6.1 points to a number of open problems in this direction. For example, is there a polynomial kernel with respect to the size of a modulator to an outerplanar graph? Several other problems such as Vertex Cover (Corollary 5.2), Hamiltonian Cycle (Theorem 8.6), and 3-Coloring (Theorem 7.3), do not admit polynomial kernels for this parameter unless NP $\subseteq$ coNP/poly. To gain a better understanding of the influence of graph structure on the difficulty of Treewidth instances, it would also be interesting to obtain complexity classifications for the parameterizations by distance from chordal graphs or cographs.

A student of mine asked me today to give him a reason for a fact which I did not know was a fact, and do not yet. He says, that if a figure be anyhow divided and the compartments differently colored so that figures with any portion of common boundary line are differently colored — four colors may be wanted but not more.

—Augustus De Morgan, 1852

7 Graph Coloring

In this chapter we study one of the oldest problems in graph theory, that of properly coloring the vertices of a graph with a limited number of colors. Since testing three-colorability is already NP-complete, the use of nonstandard parameterizations is especially relevant for this problem. It is therefore natural to consider structural parameterizations of coloring problems, as indeed has been done in the past [48, 177]. In this chapter we report on the first kernelization complexity results in this area. We study the vertex coloring problem parameterized by the vertex-deletion distance to graph classes in which the problem can be solved in polynomial time. Not only do we determine the existence of polynomial kernels for a large number of such parameterized coloring problems, we also establish upper and lower bounds on the *degrees* of the polynomials that bound the kernel size. Our results show that the kernel size for q-COLORING, parameterized by the vertex-deletion distance to a graph class $\mathcal{F}$, is strongly related to the asymptotic growth rate of the sizes of certain vertex-minimal NO-instances of the q-LIST COLORING problem, on graphs in $\mathcal{F}$.

This chapter is based on the paper "Data Reduction for Graph Coloring Problems" by Bart M. P. Jansen and Stefan Kratsch (18th International Symposium on Fundamentals of Computation Theory [147]). The presentation in this chapter is based on the revised version that will appear in Information and Computation. The result in Section 7.2 originates from the paper on cross-composition by Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch [29].

7.1 Coloring Problems Parameterized by Distance from Triviality

In his study of the parameterized complexity of vertex coloring problems, Cai [48] introduced a convenient notation to talk about structural parameterizations of graph problems. For a graph class $\mathcal{F}$ let $\mathcal{F} + k\mathrm{v}$ denote the graphs that can be built by adding at most k vertices to a graph in $\mathcal{F}$; the neighborhoods of these new vertices can be arbitrary. Equivalently, the class $\mathcal{F} + k\mathrm{v}$ contains those graphs that have a modulator $X \subseteq V(G)$ of size at most k such that $G - X \in \mathcal{F}$. Hence $\mathcal{F} + k\mathrm{v}$ contains the graphs whose vertex-deletion distance to $\mathcal{F}$ is at most k. For example, Forest $+ k\mathrm{v}$ is exactly the class of graphs that have a feedback vertex set of size at most k. Similarly one may define classes $\mathcal{F} + k\mathrm{e}$ and $\mathcal{F} - k\mathrm{e}$ where the modification distance to $\mathcal{F}$ is measured through the number of *edges* that were added or removed from a member of $\mathcal{F}$ to build the graph. Recall that $\chi(G)$ is the chromatic number of G: the minimum number of colors in a proper (vertex)coloring. Using this notation we can define, for every positive integer q, a class of parameterized coloring problems with structural parameters.

q-Coloring on $\mathcal{F} + k\mathrm{v}$ graphs
Input: A graph G and a modulator $X \subseteq V(G)$ such that $G - X \in \mathcal{F}$.
Parameter: The size $k := |X|$ of the modulator.
Question: Is $\chi(G) \leq q$?

To keep in line with the existing literature for the parameterized analysis of coloring problems, we adopt the $\mathcal{F} + k\mathrm{v}$ notation throughout this chapter to denote parameterized coloring problems. One should keep in mind that this gives different ways of expressing parameterizations that were already encountered earlier. For example, q-Coloring on Forest $+ k\mathrm{v}$ graphs is exactly the q-Coloring problem parameterized by a feedback vertex set, whereas the problem on Empty $+ k\mathrm{v}$ graphs corresponds to the parameterization by vertex cover. The Chromatic Number on $\mathcal{F} + k\mathrm{v}$ graphs problem is defined similarly as q-Coloring, with the important exception that the value q is not fixed, but part of the input. We give an informal description of the theorems in this chapter before presenting their technical details.

Our results for Chromatic Number

We start by showing the infeasibility of efficient preprocessing when the desired number of colors is part of the input. By cross-composing a restricted version of the 3-Coloring problem into Chromatic Number on Empty $+ k\mathrm{v}$ graphs (i.e., parameterized by vertex cover) we establish a superpolynomial lower bound on the size of a kernel under the assumption that NP $\nsubseteq$ coNP/poly. We show that even the compound parameterization by the vertex cover number *plus* the number of colors that is asked for, does not admit a polynomial kernel. Since

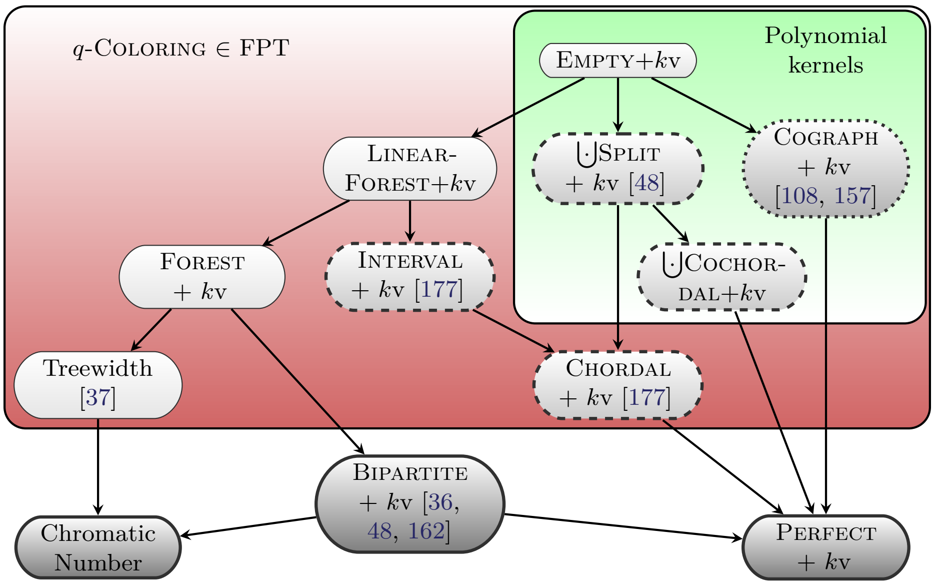


Figure 7.1: Complexity overview for various parameterizations of graph coloring problems, assuming suitable formalizations. Arrows point from larger parameters to smaller parameters: an arc $P \to P'$ signifies that every graph G satisfies $P(G) + 2 \geq P'(G)$. The complexity of the Chromatic Number problem for a given parameterization is expressed through the shading and border style of the parameters: the complexity status can be NP-complete for fixed k ▬, or contained in XP but not known to be W[1]-hard ⬚, or contained in XP and W[1]-hard ⬚, or FPT but without polynomial kernel unless NP $\subseteq$ coNP/poly ▭ . The complexity of q-Coloring for a given parameterization is expressed through the containers: the status is either FPT with a polynomial kernel, FPT but no polynomial kernel unless NP $\subseteq$ coNP/poly, or NP-complete for fixed k.

the vertex cover number is a maximal element in the hierarchy of Fig. 2.1, we cannot move to a larger parameterization to cope with the hardness of instance compression. This suggests that for all nontrivial structural parameterizations, the size of the kernel must depend superpolynomially on the number of colors provided that NP $\nsubseteq$ coNP/poly. In the remainder of the chapter we therefore focus on q-Coloring and consider how various structural parameterizations influence the complexity of the problem when keeping the number of colors q *fixed*.

The structural measures relevant to this chapter are shown in Fig. 7.1. It is known that there are several coloring problems, such as Precoloring Extension and Equitable Coloring, that are $W[1]$-hard when parameterized by treewidth, but fixed-parameter tractable parameterized by the vertex cover number [96, 106]. These parameters also yield differences in the *kernelization complexity* of q-Coloring. Our hierarchy includes these parameters, and several others that are sandwiched between them.

Our results for q-Coloring

In this chapter we pinpoint the boundary for polynomial kernelizability of q-Coloring in the given hierarchy, by exhibiting upper and lower bounds on kernel sizes. For all parameters in Fig. 7.1 for which q-Coloring is in FPT, we either give a polynomial kernel, or prove that the existence of such a kernel would imply NP $\subseteq$ coNP/poly and is therefore unlikely.

Upper bounds in the hierarchy. It follows from Theorem 4.3 that q-Coloring parameterized by vertex cover admits a kernel with $\mathcal{O}(k^q)$ vertices. Our aim is therefore to establish positive results with respect to smaller parameters. We derive a general theorem that associates the existence of polynomial kernels for q-Coloring on $\mathcal{F} + k$v graphs to properties of the q-List Coloring problem on graphs in $\mathcal{F}$: if the non-list-colorability of a graph in $\mathcal{F}$ is "local" in the sense that for any NO-instance there is a small subinstance on $g(q)$ vertices to which the answer is NO, then q-Coloring on $\mathcal{F} + k$v graphs admits a polynomial kernel with $\mathcal{O}(k^{q \cdot g(q)})$ vertices for every fixed q. This positive result even extends to the q-List Coloring problem with structural parameterizations. We apply the general theorem to give polynomial kernels for q-Coloring on the parameterized graph families Cograph $+ k$v and $\biguplus$Cochordal $+ k$v. Recall that $\biguplus$Cochordal is the class of graphs whose connected components are cochordal, i.e., complements of chordal graphs. The class $\biguplus$Split is a subclass of $\biguplus$Cochordal, containing the graphs where each connected component is a split graph.

Lower bounds in the hierarchy. In the seminal paper on lower bounds for kernelization, Bodlaender et al. [24, Theorem 2] prove that 3-Coloring parameterized by treewidth does not admit a polynomial kernel, unless their complexity-theoretic AND-conjecture fails (cf. [89]). We strengthen their result by showing that unless NP $\subseteq$ coNP/poly, the problem does not even admit a polynomial kernel parameterized by vertex-deletion distance to a single path: 3-Coloring on Path $+ k$v graphs does not admit a polynomial kernel. Under the same assumption, this immediately excludes polynomial kernels on, e.g., Forest $+ k$v or Interval $+ k$v graphs, since the latter are *smaller* parameters.

Besides investigating the *existence* of polynomial kernels, we also study the *degree* of the polynomial in the kernels that we obtain for q-Coloring on $\mathcal{F} + k$v graphs. Our general scheme yields kernels with $\mathcal{O}(k^q)$ vertices on Empty $+ k$v graphs, and with some effort we can encode these instances in $\mathcal{O}(k^q)$ bits. Using a connection to Not-All-Equal q-Satisfiability (q-nae-sat) we prove that for every $q \geq 4$ and $\varepsilon > 0$, the q-Coloring problem parameterized by vertex cover does not admit a kernel that can be encoded in $\mathcal{O}(k^{q-1-\varepsilon})$ bits, unless NP $\subseteq$ coNP/poly.

It turns out that the properties of NO-instances of q-List Coloring on graphs from $\mathcal{F}$ can also be used to obtain kernel *lower bounds* for q-Coloring on $\mathcal{F} + k$v graphs. We prove that if there is a NO-instance of $(q-2)$-List Coloring on t vertices that is irreducible in the sense that any vertex-deletion turns it into a YES-instance, then kernels for q-Coloring on $\mathcal{F} + k$v graphs cannot have bitsize $\mathcal{O}(k^{t-\varepsilon})$ for any $\varepsilon > 0$ unless NP $\subseteq$ coNP/poly. This theorem explains why the number of vertices in our general kernelization scheme must depend on the function $g(q)$ in its exponent.

Related work

Structural parameterizations of graph coloring problems were first studied by Cai [48]. He showed, amongst others, that Chromatic Number is in FPT on Chordal $-$ ke, Split $+$ ke, and Split $-$ ke graphs, but $W[1]$-hard on Split $+$ kv graphs. He also proved that 3-Coloring on Bipartite $+$ kv and Bipartite $+$ ke graphs is NP-complete for constant values of k. Marx [177] continued this line of research and showed, e.g., that Chromatic Number on Interval $+$ kv and Chordal $+$ kv graphs is in XP but $W[1]$-hard, and that Chromatic Number on Chordal $+$ ke graphs is in FPT. A summary of their results relevant to this chapter can be found in Fig. 7.1. Chor, Fellows, and Juedes [59] considered the problem of coloring a graph on n vertices with $n - k$ colors and obtained an FPT algorithm. They also showed that the problem of covering the vertices of a graph with $n - k$ cliques has a kernel with $3k$ vertices. Using the fact that Chromatic Number on Complete $+$ kv graphs is equivalent to partitioning the complement graph into cliques, parameterized by vertex cover, and that a graph G with a vertex cover of size k needs at least $n - k$ cliques to cover it, their result implies that Chromatic Number on Complete $+$ kv graphs has a linear-vertex kernel. Finally we observe that the q-Coloring problem on Interval $+$ kv and Chordal $+$ kv graphs is in FPT, using the fact that either the interval- or chordal graph $G - X$ contains a clique of size $q + 1$ and the answer is no, or the treewidth of G is at most $k + q$. An algorithm to solve Chromatic Number on $\biguplus$Cochordal $+$ kv graphs in XP-time can be obtained using the ideas of Lemma 7.4.

Organization

The chapter follows the description of our results. The lower bound for Chromatic Number parameterized by vertex cover is given in Section 7.2. Positive results related to the hierarchy are established in Section 7.3, by first deriving general theorems in Section 7.3.1 and then establishing upper bounds on the size of no-certificates for q-List Coloring in Section 7.3.2 to apply these general theorems to. In Section 7.4 we focus on negative results in the parameter hierarchy, giving lower bounds for important parameterizations in Section 7.4.1. We finish with a general negative theorem, which is applied using the lower bound constructions for sizes of no-certificates that are developed in Section 7.4.2.

7.2 Chromatic Number

In this section we give a kernelization lower bound for Chromatic Number parameterized by vertex cover, by cross-composing a restricted version of 3-Coloring into it. We consider the problem on the following class of graphs.

Definition 7.1. A graph G is a *triangle split graph* if $V(G)$ can be partitioned into sets X and Y such that $G[X]$ is an empty graph and $G[Y]$ is a disjoint union of triangles.

We will see that 3-Coloring instances on triangle split graphs lend themselves better to composition than general instances of that problem: if we take the disjoint union of a set of triangle split graphs and merge the induced triangles one by one, no adjacency information is lost. Before we give the composition, however, we must prove that 3-Coloring is still NP-complete with this restriction on the input graphs.

> 3-Coloring with Triangle Split Decomposition
> **Input:** A graph G with a partition of its vertex set into sets X and Y, such that $G[X]$ is an empty graph and $G[Y]$ is a union of vertex-disjoint triangles.
> **Question:** Is $\chi(G) \leq 3$?

We prove the NP-completeness of this problem by replacing edges in a general instance of 3-Coloring by triangles.

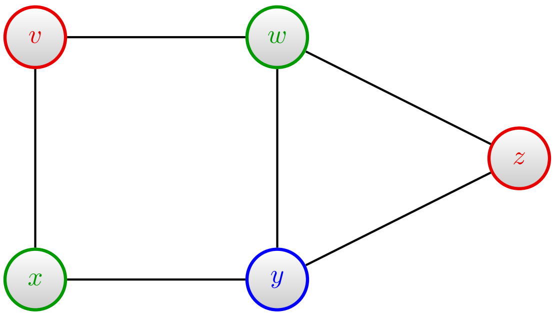
Lemma 7.1. 3-Coloring with Triangle Split Decomposition *is NP-complete.*

Proof. It is well-known [117, GT4] that 3-Coloring on general graphs is NP-complete, and it is trivial to see that the problem restricted to triangle split graphs is contained in NP. We show how to transform an instance G of 3-Coloring in polynomial time into an equivalent instance of 3-Coloring on a graph G' with a triangle split decomposition of $V(G')$ into sets X' and Y'. Number the edges in G as $e_1, e_2, \ldots, e_m$. Construct the graph G' as follows:

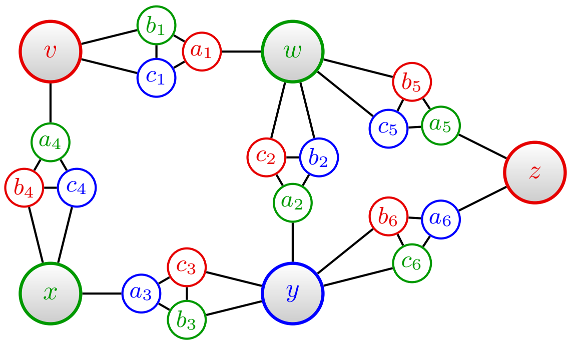
- Set $V(G') := V(G) \cup \{a_i, b_i, c_i \mid i \in [m]\}$ and $E(G') := \emptyset$.
- Add the edges $\{a_i, b_i\}, \{b_i, c_i\}, \{a_i, c_i\}$ to $E(G')$ for $i \in [m]$.
- For each edge $e_i = \{u_i, v_i\}$ $(i \in [m])$ of graph G, make vertex u_i adjacent in G' to a_i, and make v_i adjacent to b_i and c_i.
- Define $X' := V(G)$ and $Y' := \{a_i, b_i, c_i \mid i \in [m]\}$.

This concludes the description of G'. It is easy to see that G' is a triangle split graph with the partition X' and Y' since $G'[X']$ is an independent set and $G'[Y']$ is a disjoint union of triangles. We refer to Fig. 7.2 for an example.

Claim. $\chi(G') \leq 3$ *if and only if* $\chi(G) \leq 3$.



(a) Input graph G.



(b) Transformed graph G'. The vertices $\{v, w, x, y, z\}$ form the set X', the remainder is Y'.

Figure 7.2: An example of the reduction from 3-Coloring to 3-Coloring with Triangle Split Decomposition. A proper 3-coloring of the input is shown, along with the corresponding coloring of the output instance.

Proof. ($\Rightarrow$) Assume that $\chi(G') \leq 3$. Consider a 3-coloring of G'. For every edge $\{u_i, v_i\} \in E(G)$ we added a triangle on vertices $\{a_i, b_i, c_i\}$ to the graph G'. Hence $G'[N_{G'}(\{u_i, v_i\})]$ contains a triangle for all pairs of vertices $\{u_i, v_i\}$ that are adjacent in G. If some pair u_i and v_i receive the same color, then this leaves only two colors to use on the triangle in their open neighborhood. As it takes three colors to properly color a triangle, any proper 3-coloring of G' uses different colors for u_i and v_i, for all $\{u_i, v_i\} \in E(G)$. Therefore the 3-coloring of G' restricted to the vertex set of G is a proper 3-coloring of G.

($\Leftarrow$) Assume that G has a proper 3-coloring. We construct a 3-coloring for G' by coloring all vertices of $V(G') \cap V(G)$ the same as in G; now all that remains is to color the triangles we added to the graph. If there is a triangle $\{a_i, b_i, c_i\}$ for a pair $\{u_i, v_i\}$ then $\{u_i, v_i\}$ are adjacent in G and hence they receive different colors in the proper coloring. Now give a_i the color of v_i, give b_i the color of u_i and give c_i the remaining color. If we do this for every triangle then we obtain a proper 3-coloring of G', which proves that $\chi(G') \leq 3$. $\diamondsuit$

Since the instance (G', X', Y') can be built from G in polynomial time this proves that 3-Coloring with Triangle Split Decomposition is NP-complete. $\square$

Equipped with this lemma we can prove a kernel lower bound for the following problem.

Chromatic Number on Empty $+ k$v graphs
Input: A graph G, a vertex cover $X \subseteq V(G)$, and a positive integer q.
Parameter: The size $k := |X|$ of the vertex cover.
Question: Is $\chi(G) \leq q$?

The parameterization is easily seen to be fixed-parameter tractable by noting that the chromatic number of a graph exceeds its vertex cover number by at most one. Hence $q \leq |X| + 1$ for all relevant inputs. Thus an FPT algorithm is obtained by trying all q-colorings of $G[X]$ and testing whether they can be extended to the entire graph. This test comes down to checking whether for each vertex $v \in V(G) \setminus X$ there is a color that is not yet used on a neighbor in X. It results in a total runtime of $\mathcal{O}(q^{|X|} n^{\mathcal{O}(1)}) = \mathcal{O}((k+1)^k n^{\mathcal{O}(1)})$.

Theorem 7.1. Chromatic Number *on* Empty $+ k$v *graphs (i.e., parameterized by vertex cover) does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.*

Proof. To prove the theorem we show that 3-Coloring with Triangle Split Decomposition cross-composes into Chromatic Number parameterized by vertex cover. By a suitable choice of polynomial equivalence relation — in the same style as in Theorem 3.4 — we may assume that we are given t input instances that encode structures $(G_1, X_1, Y_1), \ldots, (G_t, X_t, Y_t)$ of 3-Coloring with Triangle Split Decomposition with $|X_i| = n$ and $|Y_i| = 3m$ for all $i \in [t]$ (i.e., m is the number of triangles in each instance). We will compose these instances into one instance (G', X', q', k') of Chromatic Number parameterized by vertex cover. By duplicating some instances we may assume that the number of inputs t is a power of 2; this does not affect the truth value of the or of the inputs, and a parameter value that is suitably bounded with respect to the longer sequence of inputs, is also suitably bounded with respect to the original list of inputs.

For each set Y_i, label the triangles in $G_i[Y_i]$ as $T_1, \ldots, T_m$ in some arbitrary way, and label the vertices in each triangle T_j for a set Y_i as a^i_j, b^i_j, c^i_j. We build a graph G' with a vertex cover of size $k' := 3 \log t + 4 + 3m \in \mathcal{O}(m + \log t)$ such that G' can be $q' := \log t + 4$-colored if and only if one of the input instances can be 3-colored.

- Initialize the graph G' as the disjoint union of the input graphs $G_1, \ldots, G_t$.
- For each $j \in [m]$, identify the vertices $\{a^i_j \mid i \in [t]\}$ into a new vertex a_j. Similarly identify all b^i_j into b_j, and all c^i_j into c_j, for each $j \in [m]$. The vertices resulting from these identification operations are the *triangle vertices* T'. As

we are effectively merging triangles one by one, $G'[T']$ is a disjoint union of triangles.

- Add a clique on vertices $\{p_i \mid i \in [\log t]\} \cup \{w, x, y, z\}$ to G'; it is called the *palette.*
- Make all vertices in T' adjacent to all vertices from the palette except x, y, and z.
- Make all the vertices in $\bigcup_{i=1}^{t} X_i$ adjacent to w.
- For $i \in [\log t]$ add a path on two new vertices $\{r_0^i, r_1^i\}$ to the graph, and make them adjacent to all vertices of the palette except p_i and w. These vertices form the *instance selector* vertices.
- For each instance number $i \in [t]$ we can write a binary representation of the value i in $\log t$ bits — we interpret the number t as the string of $\log t$ zeros. Consider each position $j \in [\log t]$ of this binary representation, where position 1 is most significant and $\log t$ is least significant. If bit number j of the representation of i is a 0 (resp. a 1) then make vertex r_0^j (resp. r_1^j) adjacent to all vertices of X_i.

This concludes the construction. The following claims about G' are easy to verify:

(I) In every proper $q' = \log t + 4$-coloring of G', the following holds:
   (a) each of the $\log t + 4$ vertices of the palette receives a unique color (since the palette is a clique),
   (b) for each $i \in [\log t]$ the vertices r_0^i and r_1^i receive different colors (since they are adjacent); one of them must take the color of w and the other of p_i (they are adjacent to all other vertices of the palette),
   (c) the triangle vertices T' are colored using the colors of x, y, z (they are adjacent to all other vertices of the palette),
   (d) the only colors that can occur on a vertex in X_i (for all $i \in [t]$) are the colors given to x, y, z and $\{p_j \mid j \in [\log t]\}$ (since the vertices in X_i are adjacent to w).

(II) For every $i \in [t]$, the graph $G'[X_i \cup T']$ is isomorphic to G_i.

(III) The set $X' := \{p_i \mid i \in [\log t]\} \cup \{w, x, y, z\} \cup T' \cup \{r_0^i, r_1^i \mid i \in [\log t]\}$ forms a vertex cover of G' of size $k' = |X'| = 3 \log t + 4 + 3m$. Hence G' has a vertex cover of size $\mathcal{O}(m + \log t)$.

The output of the cross-composition is the instance $(G', X', q' := \log t + 4, k')$. It is easy to verify that the construction can be performed in polynomial time. As k' is polynomial in $\log t$ plus the size of the largest input, it remains to prove that the output indeed acts as the logical OR of the input instances.

Claim. $\chi(G') \leq \log t + 4$ *if and only if there is an* $i \in [t]$ *such that* $\chi(G_i) \leq 3$.

Proof. ($\Rightarrow$) Suppose $\chi(G') \leq q'$ and consider some proper q'-coloring of G'. By (I.b) we know that for each $i \in [\log t]$ exactly one vertex of the pair $\{r_0^i, r_1^i\}$ receives the same color as p_i. Consider the string of $\log t$ bits where the i-th most significant bit is a 1 if and only if vertex r_1^i receives the same color as p_i. This bitstring encodes some integer $i^* \in [t]$. We focus on the instance with the number i^*. Let Q be the set of vertices that contains for each pair $\{r_0^i, r_1^i\}$ ($i \in [\log t]$) the unique vertex that is colored the same as p_i. By the definition of G' we know that all vertices of X_{i^*} are adjacent to all vertices of Q; hence in any proper coloring of G' the vertices of X_{i^*} cannot use any colors that are used on $\{p_i \mid i \in [\log t]\}$. By (I.d) this implies that the coloring for G' can only use the colors of x, y, z on the vertices of X_{i^*}. By (I.c) the triangle vertices T' are also colored using only the colors of x, y, z. The graph $G'[X_{i^*} \cup T']$ is isomorphic to the input graph G_{i^*} by (II), and since the coloring of G' only uses the colors of x, y, z on these vertices, this shows that the coloring of G' restricted to the induced subgraph $G'[X_{i^*} \cup T']$ is in fact a 3-coloring of graph G_{i^*}, which proves that $\chi(G_{i^*}) \leq 3$.

($\Leftarrow$) Suppose $\chi(G_{i^*}) \leq 3$ for some $i^* \in [t]$. We will construct a proper q'-coloring of G'. Start by giving all vertices of the palette different colors. By (II) the graph $G'[X_{i^*} \cup T']$ is isomorphic to G_{i^*}. Relabel the colors in the 3-coloring of G_{i^*} such that it uses the colors given to $\{x, y, z\}$ in our partial q'-coloring of G'. Give a vertex v in the induced subgraph $G'[X_{i^*} \cup T']$ the same color as the vertex in G_{i^*} that it is mapped to by the isomorphism. Afterwards we have a proper partial q'-coloring, where all vertices of the palette, all vertices of X_{i^*}, and all triangle vertices of G' are colored. It remains to color the sets X_i for $i \neq i^*$, and the pairs $\{r_0^i, r_1^i\}$. For each $i \in [\log t]$ we color the pair $\{r_0^i, r_1^i\}$ as follows: if the i-th most significant bit of the binary representation of the number i^* is a 1 then we color r_1^i the same color as p_i and we color r_0^i as w; if the bit is a 0 then we do it the other way around. It is straightforward to verify that we do not create any monochromatic edges in this way. As the final step we have to color the sets X_i for $i \neq i^*$; so consider some $i \in [t]$ with $i \neq i^*$. The binary representation of the number i^* must differ from the binary representation of i in at least one position; suppose they differ at position j. The vertex of $\{r_0^j, r_1^j\}$ that matches the bit value of i^* at position j was colored the same as p_j, hence the other vertex of the pair must have been colored the same as w. Since the bit values differ, by the definition of adjacencies in G' we find that the vertices X_i are adjacent to the vertex of $\{r_0^j, r_1^j\}$ that is colored as w. Therefore the vertices of X_i do not have any neighbors colored as p_j, and since X_i is an independent set we may color all vertices in it the same as p_j. If we color all sets X_i for $i \neq i^*$ in this way, we obtain a proper q'-coloring of G', which proves that $\chi(G') \leq q'$. ◇

As this establishes the correctness of the cross-composition, Theorem 7.1 now follows from Corollary 3.2 and Lemma 7.1. □

The instance of Chromatic Number that is constructed in the proof of the theorem has a vertex cover of size $\mathcal{O}(m + \log t)$, and asks for a coloring with $\log t + 4$ colors. Hence even the value of the compound parameter "vertex cover size plus number of colors" is bounded by $\mathcal{O}(m + \log t)$ in this construction. It therefore serves as a cross-composition of 3-Coloring with Triangle Split

Decomposition into the compound parameterization of Chromatic Number, yielding the following corollary.

Corollary 7.1. Chromatic Number *parameterized by the size of a vertex cover plus the allowed number of colors does not admit a polynomial kernel unless* $NP \subseteq coNP/poly$.

The lower bound shows that if we want to obtain polynomial kernels for nontrivial structural parameterizations of coloring problems, we should treat the number of desired colors q as a constant. In the remainder of the chapter we therefore focus on structural parameterizations of q-Coloring.

7.3 Positive Results for q-Coloring

7.3.1 Kernelization Upper Bounds

In this section we present several positive results (i.e., polynomial kernels) for parameterizations in the considered hierarchy. We begin by giving a general theorem that proves the existence of polynomial kernels for q-Coloring on $\mathcal{F}+k\text{v}$ graphs under certain conditions on the class $\mathcal{F}$. In Section 7.3.2 we investigate which classes $\mathcal{F}$ satisfy these conditions. We introduce some terminology to state the general theorem and its proof precisely. For a fixed value of q, the q-List Coloring problem is defined as follows.

> q-List Coloring
> **Input:** An undirected graph G and for each vertex $v \in V(G)$ a list $L(v) \subseteq [q]$ of allowed colors.
> **Question:** Is there a proper q-coloring $f\colon V(G) \to [q]$ such that $f(v) \in L(v)$ for each $v \in V(G)$?

Observe that in our definition of q-List Coloring, the *total* color universe has size q and the lists can have any size. The same problem name is sometimes used for the variant in which there can be arbitrarily many colors while each list has size q, but we do not consider this variant.

Let us define an instance (G', L') of q-List Coloring as a *subinstance* of (G, L) if G' is an induced subgraph of G and $L'(v) = L(v)$ for all $v \in V(G')$. If (G', L') is a no-instance we say it is a no-subinstance. The main condition on $\mathcal{F}$ that is needed to ensure the existence of polynomial kernels for q-Coloring is captured by the following definition.

Definition 7.2. Let $g\colon \mathbb{N} \to \mathbb{N}$ be a function. Graph class $\mathcal{F}$ has $g(q)$-size no-certificates for q-List Coloring if for all no-instances (G, L) of q-List Coloring with $G \in \mathcal{F}$ there is a no-subinstance (G', L') on at most $g(q)$ vertices.

We shall see later that a bounding function $g(q)$ as required in Definition 7.2 can be found for the graph classes Cograph, $\biguplus$Split, and $\biguplus$Cochordal, whereas no such bound exists for the class of all paths. The existence of small NO-certificates for q-List Coloring on $\mathcal{F}$ turns out to be intimately linked to the existence of polynomial kernels for q-Coloring on $\mathcal{F} + k$v graphs, as is shown in the following theorem.

Theorem 7.2. *Let $\mathcal{F}$ be a hereditary class of graphs with $g(q)$-size* NO*-certificates for q-*List Coloring*. The q-*Coloring *problem on $\mathcal{F} + k$*v *graphs admits a polynomial kernel with $\mathcal{O}(k^{q \cdot g(q)})$ vertices for every fixed q.*

Proof. Consider an instance (G, X) of q-Coloring on a graph class $\mathcal{F} + k$v that satisfies the stated requirements. We give an outline of the reduction algorithm.

```
1. For each undirected graph $H$ on $t \leq g(q)$ vertices $\{h_1, \ldots, h_t\}$, do:
       For each tuple $(S_1, \ldots, S_t) \in \binom{X}{\leq q}^t$, do:
           If there is an induced subgraph of $G - X$ on vertices $\{v_1, \ldots, v_t\}$
               that is isomorphic to $H$ by the mapping $h_i \mapsto v_i$, and $S_i \subseteq$
               $N_G(v_i)$ for $i \in [t]$, then mark the vertices $\{v_1, \ldots, v_t\}$ as important
               for one such subgraph, which can be chosen arbitrarily.
2. Let $Y$ contain all vertices of $G - X$ that were marked as important.
3. Output the instance $(G', X)$ with $G' := G[X \cup Y]$.
```

Let us verify that this procedure can be executed in polynomial time for fixed q, and leads to a reduced instance of the correct size. The number of undirected graphs on $g(q)$ vertices is constant for fixed q. The number of considered tuples is bounded by $\mathcal{O}\left((q|X|^q)^{g(q)}\right)$. For each graph H, for each tuple, we mark at most $g(q)$ vertices, which is a constant. These observations imply that the algorithm outputs an instance of the appropriate size, and that it can be made to run in polynomial time for fixed q because we can just try all possible isomorphisms by brute-force. Since $G' - X$ is an induced subgraph of $G - X$, it follows that $G' - X \in \mathcal{F}$ because $\mathcal{F}$ is hereditary. It remains to prove that the two instances are equivalent: $\chi(G) \leq q \Leftrightarrow \chi(G') \leq q$. The forward direction of this equivalence is trivial, since G' is a subgraph of G. We now prove the reverse direction.

Assume that $\chi(G') \leq q$ and let $f' \colon V(G') \to [q]$ be a proper q-coloring of G'. Obtain a partial q-coloring f of G by copying the coloring of f' on the vertices of X. Since $G'[X] = G[X]$ the function f is a proper partial q-coloring of G, which assigns all vertices of X a color. We will prove that f can be extended to a proper q-coloring of G, using an argument about list-coloring. Consider the graph $H := G - X$ that contains exactly the vertices of G which are not yet colored by f. For each vertex $v \in V(H)$ define a list of allowed colors as $L(v) := [q] \setminus \{f(u) \mid u \in N_G(v)\}$, i.e., for every vertex we allow the colors that are not yet used on a colored neighbor in G. From this construction it is easy to see that any proper q-list-coloring of the instance (H, L) gives a valid way to augment f to a proper q-coloring of all vertices of G. Hence it remains to prove that (H, L) is a YES-instance of q-List Coloring.

Assume for a contradiction that (H, L) is a NO-instance. Since the problem definition ensures that $H = G - X \in \mathcal{F}$, the assumptions on $\mathcal{F}$ imply there is a NO-subinstance (H', L') on $t \leq g(q)$ vertices.

Let the vertices of H' be $h_1, \ldots, h_t$. Since (H', L') is a subinstance of (H, L) we know by construction of the latter that for every vertex h_i with $i \in [t]$ and for every color $j \in [q] \setminus L'(h_i)$ there is a vertex of $N_G(h_i)$ that is colored with j. Now choose sets $S_1, \ldots, S_t$ such that for every $j \in [q] \setminus L'(h_i)$ set S_i contains exactly one neighbor $v \in N_G(h_i)$ with $f(v) = j$, which is possible by the previous observation. Since f only colors vertices from X we have $S_i \subseteq X$ for all $i \in [t]$.

Because H' is an induced subgraph on at most $g(q)$ vertices of $H = G - X$, we must have considered graph H' during the outer loop of the reduction algorithm. Since each S_i contains at most q vertices from X, we must have considered the tuple $(S_1, \ldots, S_t)$ during the inner loop of the reduction algorithm, and because the existence of H' shows that there is at least one induced subgraph of $G - X$ which satisfies the if-condition, we must have marked some vertices $\{v_1, \ldots, v_t\}$ of an induced subgraph H^* of $G - X$ isomorphic to H' by an isomorphism $v_i \mapsto h_i$ as *important*, and hence the induced subgraph H^* is contained in the graph G'. Recall that f' is a proper q-coloring of G', and that f and f' assign the same colors to vertices of X. By construction this shows that for each vertex h_i with $i \in [t]$ of the presumed NO-subinstance (H', L') of q-LIST COLORING, for each color $j \in [q] \setminus L'(h_i)$ that is not on the list of h_i, there is a neighbor of the corresponding vertex v_i (i.e., a vertex in $N_{G'}(v_i)$) that is colored j. Using the fact that H^* is isomorphic to H' we obtain a valid q-list-coloring of H' by using the colors assigned to H^* by f'. But this shows that (H', L') is in fact a YES-instance of q-LIST COLORING, which contradicts our initial assumption. This proves that the instance (H, L) of q-LIST COLORING that we created must be a YES-instance, and by our earlier observations this implies that $\chi(G) \leq q$, which concludes the proof of the equivalence of the input- and output instance.

Hence we have shown that for each fixed q there is a polynomial-time algorithm that transforms an input of q-COLORING on $\mathcal{F} + k$v graphs into an equivalent instance of size bounded polynomially in $k = |X|$, which concludes the proof. $\square$

As an example application of Theorem 7.2, observe that a NO-instance of q-LIST COLORING on an empty graph has an induced NO-subinstance on a single vertex: as there are no edges, a NO-instance must have a vertex with an empty list. Since empty graphs are hereditary, Theorem 7.2 implies the existence of a polynomial kernel with $\mathcal{O}(k^q)$ vertices for q-COLORING on EMPTY $+ k$v graphs, i.e., parameterized by vertex cover.[1] While most kernels with $\mathcal{O}(k^q)$ vertices require $\Omega(k^{q+1})$ bits to represent (as the number of edges can be quadratic in the number of vertices), we can prove that a kernel for q-COLORING on EMPTY $+ k$v graphs exists that can be encoded in $\mathcal{O}(k^q)$ bits.

[1] The same result also follows from Theorem 4.3.

Lemma 7.2. *For every fixed $q \geq 3$, q-*Coloring *on* Empty $+ k$v *graphs (i.e., parameterized by vertex cover) admits a kernel with $k + k^q$ vertices that can be encoded in $\mathcal{O}(k^q)$ bits.*

Proof. Let (G, X) be an instance of q-Coloring on Empty $+ k$v graphs, which implies that X of size k is a vertex cover of G. We create an equivalent instance (G', X) as follows. Start by setting $G' := G[X]$. For every $S \in \binom{X}{q}$, if $\bigcap_{v \in S} N_G(v) \setminus X \neq \emptyset$, add a new vertex v_S to G' with $N_{G'}(v_S) := S$. Using argumentation similar to that of Theorem 7.2 it can be proven that (G, X) is equivalent to the instance (G', X): from a proper q-coloring f' of G' we can extract a proper partial q-coloring f of $G[X]$, and if f cannot be extended to the rest of G then there is a vertex in $G - X$ whose neighborhood contains all q colors; but then such a vertex also exists in G', contradicting that f' is proper.

Observe that the graph G' consists of the vertex cover X and the independent set $G' - X$ of vertices with degree exactly q, no two vertices of which share the same open neighborhood; hence G' has at most $k + k^q$ vertices. To prove the lemma we give an efficient encoding for graphs with such a structure. We can represent the instance (G', X) in $\mathcal{O}(|X|^q) = \mathcal{O}(k^q)$ bits as follows. We store an adjacency matrix of $G'[X]$ in exactly k^2 bits. Then for every set $S \in \binom{X}{q}$ (of which there are less than k^q) we store exactly one bit, specifying whether or not there is a vertex with open neighborhood S. By fixing some ordering on the vertices of X for the adjacency matrix, and by fixing an ordering of the sets $\binom{X}{q}$, the instance can unambiguously be recovered from this encoding. Since the encoding uses less than $k^2 + k^q$ bits this concludes the proof. □

We will investigate richer graph classes that have small no-certificates for q-List Coloring in Section 7.3.2. In Section 7.4 we shall prove that the exponential dependency on the function $g(q)$ in the kernel size bound is necessary. But first we show that the result of Theorem 7.2 can be extended to q-List Coloring under structural parameterizations. For completeness we give a full definition of the generalized form of the problem.

> q-List Coloring on $\mathcal{F} + k$v graphs
> **Input:** An undirected graph G with for each vertex $v \in V(G)$ a list $L(v) \subseteq [q]$ of allowed colors, and a modulator $X \subseteq V(G)$ such that $G - X \in \mathcal{F}$.
> **Parameter:** The size $k := |X|$ of the modulator.
> **Question:** Is there a proper q-coloring $f \colon V(G) \to [q]$ such that $f(v) \in L(v)$ for each $v \in V(G)$?

The classical complexity of q-Coloring and q-List Coloring differs quite significantly when placing restrictions on the input graphs. For example, 3-Coloring is trivial on bipartite graphs whereas 3-List Coloring is NP-complete on such graphs [36, Theorem 1]. Nevertheless, both coloring problems admit polynomial kernels when parameterized by the vertex-deletion distance to graph classes with small NO-certificates.

Corollary 7.2. *Let $\mathcal{F}$ be a hereditary class of graphs with $g(q)$-size* NO*-certificates for q-*LIST COLORING*. The q-*LIST COLORING *problem on $\mathcal{F} + k\mathrm{v}$ graphs admits a polynomial kernel with $\mathcal{O}((k+q)^{q \cdot g(q)})$ vertices for every fixed q.*

Proof. Given an instance (G, L, X) of q-LIST COLORING on $\mathcal{F} + k\mathrm{v}$ graphs, we construct an equivalent instance of q-COLORING on $\mathcal{F} + (k+q)\mathrm{v}$ graphs as follows. We build G' by adding a clique on q vertices $p_1, \ldots, p_q$ to G, and connect vertex p_i for $i \in [q]$ in graph G' to all vertices v for which $i \notin L(v)$. Since a proper q-coloring of G' assigns unique colors to each of the vertices of the clique, the adjacency of the original vertices to this clique enforces the list requirements. By taking $X' := X \cup \{p_1, \ldots, p_q\}$ as the new modulator we find that $G - X = G' - X' \in \mathcal{F}$. Hence the q-COLORING on $\mathcal{F} + (k+q)\mathrm{v}$ graphs instance (G', X') is equivalent to the parameterized q-LIST COLORING on $\mathcal{F} + k\mathrm{v}$ graphs instance (G, L, X). Now apply the kernelization algorithm of Theorem 7.2 to (G', X'). Since the parameter is $|X'| = k + q$, the resulting reduced instance satisfies the claimed size bound. Afterwards we assign a full list $[q]$ to every vertex, to obtain the kernelized q-LIST COLORING instance. $\square$

For graph classes that are characterized by a finite set of forbidden induced subgraphs, such as cographs which are induced-P_4-free, we can even drop the requirement that a modulator X is given in the input. A 4-approximation to the minimum-size modulator to a cograph can be obtained by repeatedly finding an induced P_4 and adding all its vertices to the approximate solution. We may then obtain a polynomial kernel using this approximate modulator as the set X in Theorem 7.2 or Corollary 7.2.

7.3.2 Upper Bounds to Sizes of No-Certificates for q-List Coloring

Having proved that an upper bound on the size of NO-certificates for q-LIST COLORING on $\mathcal{F}$ yields polynomial kernels for coloring problems on $\mathcal{F} + k\mathrm{v}$ graphs, we set out to study specific graph classes for which such bounds exist. We start the section by giving an illustrative example for the unions of split graphs. We then consider the more general class containing the union of cochordal graphs, and conclude the section by studying cographs.

Lemma 7.3. *The class $\biguplus$*SPLIT *is hereditary and has $g(q) := q + 4^q$-size* NO*-certificates for q-*LIST COLORING*.*

Proof. Since split graphs are hereditary, so are their unions. To establish the desired bound on the size of NO-certificates, consider a NO-instance (G, L) of q-LIST COLORING with $G \in \biguplus$SPLIT. If G contains a clique on $q+1$ vertices, then the subinstance induced by this clique cannot be q-colored and is therefore a

NO-subinstance of list-colorability of size $q+1$. In the remainder we may therefore assume G contains no clique of size more than q. We show how to obtain a NO-subinstance of (G, L) on at most $q + 4^q$ vertices. We obtain our NO-subinstance by applying a cleaning rule to (G, L) that does not affect the list-colorability of the instance.

Claim (Cleaning Rule)**.** *If (G, L) is an instance of q-*LIST COLORING *and there are distinct nonadjacent vertices u, v such that $N_G(u) \subseteq N_G(v)$ and $L(u) \supseteq L(v)$ then (G, L) is a* YES-*instance if and only if $(G - \{u\}, L)$ is a* YES-*instance.*

Proof. Because $(G - \{u\}, L)$ is a subinstance of (G, L) we only have to show that a list-coloring f' for $(G - \{u\}, L)$ can be transformed into a list-coloring for (G, L). Since $N_G(u) \subseteq N_G(v)$, the color assigned to v by f' cannot occur on any neighbor of u, and by the condition $L(u) \supseteq L(v)$ this color is admissible for u. By nonadjacency of u and v it therefore follows that f' can be extended to a proper list-coloring of (G, L), by assigning u the same color as v. ◇

Now we use the cleaning rule to find a small NO-subinstance of our presumed NO-instance (G, L). Since the rule does not change the answer to the instance, we may exhaustively apply the rule to obtain an equivalent NO-instance (G', L'). We may assume without loss of generality that G' is connected, because a graph is list-colorable if and only if each connected component is list-colorable; hence in any disconnected NO-instance there is a connected NO-subinstance. We prove that $|V(G')| \leq q + 4^q$. Because $\biguplus$SPLIT is hereditary and G' is connected it follows that $G' \in$ SPLIT, so let W, Z be a partition of the vertex set of G' such that $G'[W]$ is a clique and $G'[Z]$ is an independent set; this partition exists by the definition of a split graph. By the assumption at the beginning of the proof we know that G' contains no clique of size more than q, therefore $|W| \leq q$. Since Z is an independent set, we know that $N_{G'}(v) \subseteq W$ for all $v \in Z$. Consider some $W' \subseteq W$ and the set of vertices $S_{W'} := \{v \in Z \mid N_{G'}(v) = W'\}$. Since all vertices in $S_{W'}$ are mutually nonadjacent and have the same open neighborhood, by the fact that the cleaning rule is not applicable to (G', L') it follows that for all vertices $u, v \in S_{W'}$ we have $L(u) \not\subseteq L(v)$. In particular this shows that every subset of q can occur at most once as the list of a vertex in $S_{W'}$, which proves that $|S_{W'}| \leq 2^q$. Since all vertices in Z occur in some set $S_{W'}$ for $W' \subseteq W$ it follows that $|Z| \leq 2^{|W|}2^q$, and since $|W| \leq q$ we find $|Z| \leq 2^{2q}$. As the vertex set of G' consists of W and Z we obtain $|V(G')| \leq q + 2^{2q} = q + 4^q$ which proves the lemma. □

Note the unusual use of the cleaning rule in the proof of Lemma 7.3: although it looks like a kernelization rule for q-LIST COLORING, we use it merely to prove the *existence* of small NO-certificates for the class $\biguplus$SPLIT. The actual kernelization of q-COLORING on $\biguplus$SPLIT $+ kv$ graphs implied by Theorem 7.2 relies only on the established property, and does not use the cleaning rule.

The next lemma shows that even the class $\biguplus$COCHORDAL, which is a strict superclass of $\biguplus$SPLIT, has bounded-size NO-certificates for q-LIST COLORING. As the class is more general, the bounding function $g(q)$ grows faster. We use the following simple fact about chordal graphs.

Proposition 7.1 ([45, Chapter 1.2]). *Every chordal graph has a simplicial vertex.*

Lemma 7.4. *The class $\biguplus$*Cochordal *is hereditary and has $(q+1)!$-size* no-*certificates for q-*List Coloring.

Proof. Since chordal graphs are hereditary, so are their complements and the unions of their complements. As in the previous proof we may restrict our attention to connected graphs without loss of generality. It therefore suffices to show that for every no-instance of q-List Coloring (G, L) with $G \in$ Cochordal, there is a no-subinstance with at most $g(q) := (q+1)!$ vertices.

We start by investigating the structure of an instance (G, L) of q-List Coloring on a cochordal graph. By Proposition 7.1 the chordal graph $\overline{G}$ has a simplicial vertex v, and from the definition of a simplicial vertex this shows that $N_{\overline{G}}(v)$ is a clique in $\overline{G}$ implying that $N_{\overline{G}}(v)$ is an independent set in G. Define the set $N^i_{\overline{G}}(v) := N_{\overline{G}}(v) \cap L^{-1}(i)$ containing the vertices that are non-neighbors of v in G and that may be assigned color i.

Claim. *If v is simplicial in $\overline{G}$ and there is a proper q-list-coloring of (G, L) that assigns v color i, then there is a proper q-list-coloring that gives all vertices $\{v\} \cup N^i_{\overline{G}}(v)$ color i.*

Proof. Consider a proper coloring f of G respecting the lists L with $f(v) = i$. If there is a vertex $u \in N^i_{\overline{G}}(v)$ with $f(u) \neq i$, consider the effect of changing u's color to i. Since $i \in L(u)$ by definition of $N^i_{\overline{G}}(v)$, we do not violate the requirement that all vertices receive a color from their list. So suppose that changing u's color to i results in an improper coloring, because a neighbor of u was already colored i. Since u is not a neighbor of v by definition of $N^i_{\overline{G}}(v) \subseteq N_{\overline{G}}(v)$, this means there is a vertex $w \neq v$ such that $\{u, w\} \in E(G)$ and $f(w) = i$. Now observe that since $N_{\overline{G}}(v)$ is an independent set in G and $u \in N^i_{\overline{G}}(v) \subseteq N_{\overline{G}}(v)$, the presence of the edge $\{u, w\}$ shows that $w \notin N_{\overline{G}}(v)$, i.e., that w is *not* in the set of v's non-neighbors. Together with $w \neq v$ this shows that w is a neighbor of v in G. But then f colors the adjacent vertices v and w with color i, which contradicts the assumption that f is a proper coloring. Hence changing u's color to i must result in a proper list coloring, which proves the claim: if v is colored i, then all vertices in $N^i_{\overline{G}}(v)$ can also receive color i. $\diamondsuit$

Using the claim, we describe an algorithm to solve q-List Coloring on cochordal graphs. The structure of this algorithm will prove the existence of a small no-subinstance. The algorithm proceeds as follows on an instance (G, L). If there is a vertex v with $L(v) = \emptyset$, then v shows that the answer is no and the algorithm is done. Otherwise we find a vertex v that is simplicial in $\overline{G}$, and branch on which color from $L(v)$ to assign to v. The claim shows that in the branch where we try color $i \in L(v)$ on v, we may also assign all vertices of $N^i_{\overline{G}}(v)$ color i without loss of correctness. The answer to an instance in which these vertices receive color i, is equivalent to the answer after removing the vertices $S_i := \{v\} \cup N^i_{\overline{G}}(v)$ from the graph and removing color i from the lists of the vertices $N_G(S_i)$. Observe that if v is simplicial in $\overline{G}$ then $N_G(v) = N_G(\{v\} \cup N_{\overline{G}}(v)) = N_G(S_i)$ and therefore we may equivalently simplify the instance in this branch by removing the vertices S_i and deleting color i from the lists of the vertices $N_G(v)$. The algorithm can therefore safely apply this transformation and call itself recursively on the remaining instance.

The crucial insight for the analysis of the algorithm is that in the resulting instance, color i no longer appears on any list. To see this, note that vertex v is removed, as are all non-neighbors of v that had i on their list. All vertices that are not non-neighbors of v, and are not v itself, must be neighbors of v; but then color i was explicitly removed from their lists. Hence in each branch we eliminate at least one color from the instance. Since the instance starts with at most q different colors occurring in the lists, after branching q levels deep we must either obtain a graph without vertices (and the answer is YES), or a graph containing a vertex with an empty list (and the answer is NO).

Using the structure of this algorithm we prove the lemma. Consider a NO-instance (G, L) of q-LIST COLORING where $G \in$ COCHORDAL. If we run the sketched algorithm on this instance, the answer will be NO since the algorithm is correct. Hence each branch ends with the detection of a vertex with an empty list. Suppose that in each call of the algorithm we mark the simplicial vertex v that is branched on, and we mark the detected vertex with an empty list when the algorithm outputs NO for a branch. Let $h(\ell)$ be the number of vertices marked in this process, for an instance whose lists contain ℓ different colors. From this definitions it follows that $h(0) = 1$. In an instance containing ℓ distinct colors, we mark the simplicial vertex that is branched on and then branch in at most ℓ ways, decreasing the total number of remaining colors in each branch. Hence $h(\ell) \leq 1 + \ell \cdot h(\ell - 1)$. This recurrence is bounded by $h(\ell) \leq (\ell + 1)!$ and since the input instance contains at most q distinct colors, the total number of vertices marked is at most $(q + 1)!$.

If we now consider the instance induced by the marked vertices and run the algorithm on this instance, branching on the simplicial vertices in the same order as before, then the algorithm will make exactly the same decisions as for the instance (G, L); thus the algorithm will output NO, which shows by correctness of the algorithm that the instance induced by the $(q + 1)!$ marked vertices is a NO-subinstance. Since this proves the existence of a NO-subinstance of the requested size, it completes the proof. □

By employing a similar argument, which shows how the structure of an algorithm to solve q-LIST COLORING on cographs can be used to find small NO-subinstances of the problem, we can prove that cographs have bounded-size NO-certificates. We use the following characterization of cographs in terms of cotree representations.

Definition 7.3. A *cotree* $\mathcal{T}$ is a rooted proper binary tree whose internal vertices are labeled as JOIN or UNION nodes. For $v \in V(\mathcal{T})$ the graph $\mathrm{CG}(\mathcal{T}, v)$ represented by the subtree of $\mathcal{T}$ rooted at v is defined as follows:

$$\mathrm{CG}(\mathcal{T}, v) := \begin{cases} \text{Graph } G := (\{v\}, \emptyset) & \text{If } v \text{ is a leaf.} \\ \mathrm{CG}(\mathcal{T}, \textsc{left}(v)) \cup \mathrm{CG}(\mathcal{T}, \textsc{right}(v)) & \text{If } v \text{ is a } \textsc{union} \text{ node.} \\ \mathrm{CG}(\mathcal{T}, \textsc{left}(v)) \otimes \mathrm{CG}(\mathcal{T}, \textsc{right}(v)) & \text{If } v \text{ is a } \textsc{join} \text{ node.} \end{cases}$$

where $\textsc{left}(v)$ and $\textsc{right}(v)$ are the left- and right child of node v in the tree $\mathcal{T}$, respectively. We define $\mathrm{CG}(\mathcal{T})$ as $\mathrm{CG}(\mathcal{T}, r)$, where r is the root of $\mathcal{T}$, and we say that $\mathcal{T}$ is a *cotree representation* of the graph $\mathrm{CG}(\mathcal{T})$. The JOIN-*height* $\textsc{JoinH}(\mathcal{T})$ of a cotree T is the maximum number of JOIN nodes on any path from the root to a leaf.

Recall that $\omega(G)$ denotes the size of the largest clique in G. Using that $\omega(G_1 \otimes G_2) = \omega(G_1) + \omega(G_2)$, it is not hard to verify the following proposition.

Proposition 7.2. *If $\mathcal{T}$ is a cotree representation of graph G, then $\omega(G) \geq \textsc{JoinH}(\mathcal{T}) + 1$.*

It is well-known that a graph is a cograph if and only if it has a cotree representation [45].

Lemma 7.5. *The class* COGRAPH *is hereditary and has 2^{q^2}-size* NO-*certificates for q-*LIST COLORING.

Proof. Since cographs can be characterized as not having an induced P_4 subgraph, it is easy to see they are hereditary. To give a bound on the size of NO-certificates we use a similar approach as in the proof of Lemma 7.4. We give a simple algorithm that correctly decides q-LIST COLORING on cographs. Then we argue that we can mark at most $g(q) := 2^{q^2}$ vertices such that the algorithm would also output NO on the instance induced by the marked vertices.

The algorithm for q-LIST COLORING uses dynamic programming on a cotree decomposition of the graph (Definition 7.3), similarly to the algorithm of (Klaus) Jansen and Scheffler [149, Theorem 4.7]. So let (G, L) be an instance of q-LIST COLORING with $G \in$ COGRAPH and consider a cotree representation $\mathcal{T}$ of the graph G. We create a dynamic programming table $T[v, S]$ whose first index ranges over the nodes of $\mathcal{T}$, and whose second index ranges over subsets of $[q]$. The interpretation of the table is that $T[v, S] =$ TRUE if and only if the cograph $\mathrm{CG}(\mathcal{T}, v)$ represented by the subtree of $\mathcal{T}$ rooted at v can be q-list-colored with respect to the list assignment L, such that only the colors from S are used. In this interpretation (G, L) is a YES-instance if and only if the root-node r of $\mathcal{T}$ satisfies $T[r, \{1, 2, \ldots, q\}] =$ TRUE.

The table values satisfy the following recurrence:

$$T[v,S] = \begin{cases} L(v) \cap S \neq \emptyset & \text{If } v \text{ is a leaf node.} \\ T[\textsc{left}(v), S] \wedge T[\textsc{right}(v), S] & \text{If } v \text{ is a } \textsc{union} \text{ node.} \\ \bigvee\limits_{S' \subseteq S} T[\textsc{left}(v), S'] \wedge T[\textsc{right}(v), S \setminus S'] & \text{If } v \text{ is a } \textsc{join} \text{ node.} \end{cases}$$

To justify that this recurrence indeed characterizes the behavior of the values $T[v,S]$, observe that the cograph represented by a leaf v of a cotree is just a singleton graph on the vertex v, which can be list-colored using colors from S if S contains at least one admissible color for v. The graph represented by a cotree whose root is a UNION node is simply the disjoint union of the graphs represented by the subtrees rooted at the children. Since no color conflicts can occur between vertices in different connected components of a graph this explains the second item of the recurrence. Finally for the JOIN node observe that in any proper coloring of a graph $G_1 \otimes G_2$ there can be no vertex in G_1 that obtains the same color as a vertex of G_2, since they are made adjacent by the JOIN operation. Hence any proper coloring of $G_1 \otimes G_2$ using only the color set S must use some subset $S' \subseteq S$ for the vertices of G_1 and the disjoint set $S \setminus S'$ for the vertices of G_2. This proves that a dynamic programming algorithm that computes the values of T bottom-up in the cotree is correct.

We now use this algorithm to prove that if (G, L) is a NO-instance of q-LIST COLORING with $G \in$ COGRAPH, then there exists a NO-subinstance on at most 2^{q^2} vertices. Consider a cotree representation $\mathcal{T}$ of the graph G. If G contains a clique on $q+1$ vertices, then this clique cannot be q-(list-)colored and hence the subinstance induced by this clique must be a NO-subinstance on $q+1$ vertices; then we are done. Hence in the remainder we may assume that $\omega(G) \leq q$, which implies by Proposition 7.2 that the join height of G is smaller than q. Using this fact we now describe a recursive procedure that, given a table entry $T[v,S]$ such that $T[v,S] =$ FALSE, marks a number of vertices to preserve the FALSE value of this entry:

- If v is a leaf of the cotree, then mark that single leaf.
- If v is a UNION node, then at least one of $T[\textsc{left}(v), S]$ and $T[\textsc{right}(v), S]$ is FALSE. Recursively mark vertices for a subexpression that yields FALSE.
- If v is a JOIN node, then for all $S' \subseteq S$ at least one of the terms $T[\textsc{left}(v), S']$ and $T[\textsc{right}(v), S \setminus S']$ yields FALSE. For each subset $S' \subseteq S$, recursively mark a "witness" for one subexpression that evaluates to FALSE.

Let M be the set of marked vertices. Observe that we may obtain a cotree representation of $G[M]$ by considering the leaves L of $\mathcal{T}$ corresponding to the marked vertices, taking the subcotree $\mathcal{T}'$ induced by the paths from L to the root, and splicing out the internal vertices of $\mathcal{T}'$ that have only a single child in $\mathcal{T}'$. For every node in $\mathcal{T}'$ there is a corresponding node in $\mathcal{T}$, and as we have marked witnesses for the FALSE values in the root of $\mathcal{T}$, it is not difficult to see that the witnesses ensure FALSE values in the root of $\mathcal{T}'$. This shows that the algorithm outputs NO on the subinstance induced by M, and, similarly as in the proof of Lemma 7.4, since the algorithm is correct this proves that the subinstance induced by M is a NO-instance. It remains to bound the number of marked vertices.

We can bound the number of vertices that were marked to preserve the NO value of a cell $T[v, S]$ based on the join height of the subtree $\mathcal{T}_v$ rooted at v. Let $h(x)$ be the maximum number of vertices that are marked to preserve the FALSE-value of a cell $T[v, S]$ with $\textsc{JoinH}(\mathcal{T}_v) \leq x$. If $\textsc{JoinH}(\mathcal{T}_v) = 0$ then the subtree rooted at v contains only UNION nodes and leaves, and the procedure to mark a vertex for $T[v, S]$ will trace a path of UNION nodes through the tree until it reaches a leaf, and it will mark that single leaf: hence $h(0) = 1$. Now consider what happens when $\textsc{JoinH}(\mathcal{T}_v) > 0$. The marking procedure will trace a path from the root of $\mathcal{T}_v$, following subexpressions that evaluate to FALSE, until it reaches a leaf or a JOIN node. Since we mark only one vertex for a leaf, we focus on the JOIN node. For this node the procedure will recursively mark vertices for each subset of S. Since $S \subseteq [q]$ there are at most 2^q subsets, and for each subset we mark vertices in a subtree rooted at LEFT(v) or RIGHT(v); but note that the join height of these subtrees is at least one lower than that of $\mathcal{T}_v$. Hence for $x \geq 1$ we have $h(x) \leq 2^q \cdot h(x-1)$. We can bound this recurrence as $h(x) \leq (2^q)^x$. Since we can assume that $\textsc{JoinH}(\mathcal{T}) < q$ it follows that the subinstance resulting from marking vertices of (G, L) contains less than $h(q) \leq 2^{q^2}$ vertices. This shows the existence of a small NO-subinstance of (G, L) and concludes the proof. □

We note that a more detailed analysis of the marking procedure of the previous theorem gives a better bound. By relating marked vertices to permutations of the members of a partition of the color set, it is possible to prove that the number of marked vertices does not exceed $q^{3q} = 2^{3q \log q}$.

By combining Theorem 7.2 and Corollary 7.2 with the bounds on sizes of NO-certificates for the hereditary graph classes considered in this section, we obtain the following corollary.

Corollary 7.3. *For every fixed integer q, the problems q-*COLORING *and q-*LIST COLORING *admit polynomial kernels on the parameterized graph classes* $\biguplus$SPLIT $+ k$V, $\biguplus$COCHORDAL $+ k$V, *and* COGRAPH $+ k$V.

7.4 Negative Results for q-Coloring

7.4.1 Kernelization Lower Bounds

We turn to the study of the limits of polynomial kernelizability in our parameter space, to complement the results of the previous section. We first show that 3-COLORING on PATH $+ k$V graphs does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly, thereby presenting a barrier to polynomial kernelizability in the hierarchy of Fig. 7.1. After establishing this theorem we consider parameterizations for which polynomial kernels were obtained in the previous section, and we give lower bounds on the degree of the polynomial bounding the bitsize of these kernels.

Our first negative result is a superpolynomial kernel lower bound for 3-Coloring on LinearForest + kv graphs, which we will later lift to the parameterized graph family Path + kv. We will need the following fact, which is easily verified.

Proposition 7.3. *In any proper* 2*-coloring of a graph* P_{2n}*, the first and last vertex on the path receive a different color.*

The gadget in the proof of the following theorem is inspired by a reduction of Lokshtanov et al. [175, Theorem 6.1]. We learned through a personal communication that Stefan Szeider independently found a similar result for Forest + kv graphs, which was not published.

Theorem 7.3. 3-Coloring *on* LinearForest + kv *graphs does not admit a polynomial kernel unless NP* $\subseteq$ *coNP/poly.*

Proof. We give a polynomial-parameter transformation (Definition 3.1) from cnf-sat parameterized by the number of variables n (Proposition 3.1) to 3-Coloring parameterized by deletion distance from a linear forest. Consider an input to cnf-sat, which consists of clauses $C_1, \ldots, C_m$ where each clause is a disjunction of literals of the form x_i or $\overline{x_i}$ for $i \in [n]$. We build a graph G and a modulator $X \subseteq V(G)$ such that $|X| = 2n + 3$ and $G - X \in$ LinearForest.

Construct a clique on three vertices p_1, p_2, p_3; this clique will serve as our palette of three colors, since in any proper 3-coloring all three colors must occur on this clique. For each variable x_i for $i \in [n]$ we make vertices T_i and F_i and add the edge $\{T_i, F_i\}$ to G. We make the vertices T_i, F_i adjacent to the palette vertex p_1. Now we create gadgets for the clauses of the satisfiability instance.

For each clause C_j with $j \in [m]$, let n_j be the number of literals in C_j and create a path $(a^1_j, b^1_j, a^2_j, b^2_j, \ldots, a^{n_j}_j, b^{n_j}_j)$ on $2n_j$ vertices. We call this the clause-path for C_j. Make the first and last vertices on the path a^1_j and $b^{n_j}_j$ adjacent to the palette vertex p_1. Make the b-vertices $b^1_j, b^2_j, \ldots, b^{n_j}_j$ adjacent to palette vertex p_3. As the last step we connect the vertices on the path to the vertices corresponding to literals. For $r \in [n_j]$ if the r-th literal of C_j is x_i (resp. $\overline{x_i}$) then make vertex a^r_j adjacent to T_i (resp. F_i). This concludes the construction of the graph G. We use the modulator $X := \{T_i, F_i \mid i \in [n]\} \cup \{p_1, p_2, p_3\}$. It is easy to verify that $|X| = 2n + 3$ and therefore that the parameter of the 3-Coloring instance is polynomial in the parameter of cnf-sat. Since vertices on a clause-path are not adjacent to other clause-paths, it follows that $G - X$ is a linear forest. It remains to prove that the two instances are equivalent.

Claim. *There is a satisfying assignment for the* cnf-sat *instance on clauses* C_1*,* $\ldots, C_m$ *if and only if the graph* G *is* 3*-colorable.*

Proof. ($\Rightarrow$) Assume that $v \colon [n] \to \{\text{TRUE}, \text{FALSE}\}$ is a satisfying assignment. We construct a proper 3-coloring $f \colon V(G) \to [3]$ of G as follows:

- For the palette vertices p_i with $i \in [3]$ define $f(p_i) := i$.
- For each $i \in [n]$ with $v(i) = \text{TRUE}$, set $f(T_i) := 2$ and $f(F_i) := 3$.
- For each $i \in [n]$ with $v(i) = \text{FALSE}$, set $f(T_i) := 3$ and $f(F_i) := 2$.

Using the definition of G it is easy to verify that this partial coloring f is proper; it remains to extend f to the clause-paths. Consider the clause-path P_j corresponding to a clause C_j. For each vertex a_j^r whose adjacent literal is TRUE under v, set $f(a_j^r) := 3$. Since v is a satisfying assignment we color at least one vertex on P_j with 3, and since literals that evaluate to TRUE were given color 2 in the previous step, we do not create any conflicts. We now show how to color the remainder of the clause-path P_j. If a_j^1 did not receive a color (i.e., its neighboring literal evaluates to FALSE and the literal-vertex is colored 3) then set $f(a_j^1) := 2$ and $f(b_j^1) = 1$. Now alternatingly color the successive a-vertices with 2 and b-vertices with 1, until arriving at an a-vertex that we already assigned color 3 (because its adjacent literal evaluates to TRUE). This assignment does not create any conflicts. Now start at the last vertex $b_j^{n_j}$ and color it with 2, and work backwards giving uncolored a-vertices color 1 and b-vertices color 2, again until we arrive at a 3-colored a-vertex. If there are any uncolored subpaths left after this procedure (which occurs if two or more literals of the clause are TRUE), then color the a-vertices on this subpath with 1 and the b-vertices with 2. Using the construction of G this color assignment is easily seen to be proper. Since the clause-paths are independent, we can perform this procedure independently on each clause-path to obtain a proper 3-coloring of G.

($\Leftarrow$) Let $f \colon V(G) \to [3]$ be a proper 3-coloring of G, and assume without loss of generality (by permuting the color set if needed) that $f(p_1) = 1, f(p_2) = 2$ and $f(p_3) = 3$. We show that the CNF-SAT instance has a satisfying assignment, using the following proposition. We first show that on every clause-path P_j corresponding to a clause C_j, there must be a vertex colored 3. So assume for a contradiction that some clause-path P_j is colored using only 1 and 2. Since the first and last vertices on the path are adjacent to palette vertex p_1, those vertices cannot be colored 1 and hence they must be colored 2. But since the path has an even number of vertices, Proposition 7.3 now shows that the clause-path cannot be 2-colored using only 1 and 2, which gives a contradiction. So in a proper 3-coloring of G all clause-paths contain a vertex colored 3. The b-vertices on a clause-path are adjacent to p_3 and hence cannot be colored 3; therefore some a-vertex a_j^r that is adjacent to a literal vertex T_i or F_i must be colored with 3. This implies that the corresponding literal-vertex must be colored 2. Now consider the valuation that makes all literals colored 2 TRUE, and all literals colored 3 FALSE. The previous argument shows that at least one literal of each clause is TRUE. Since T_i and F_i are adjacent to each other, and both are adjacent to p_1, we color exactly one of them with 2 in a proper 3-coloring of G and hence we obtain a valid satisfying assignment for the CNF-SAT instance. $\diamondsuit$

Since the construction can be carried out in polynomial time and guarantees that the parameter of the output instance is bounded polynomially in the parameter of the input instance, the given reduction is indeed a polynomial-parameter transformation. Theorem 7.3 now follows from Proposition 3.1 together with Corollary 3.1, since the classical versions of both problems are easily seen to be NP-complete. $\square$

By considering the proof of Theorem 7.3 we can obtain a corollary for a stronger parameterization. Consider the graph G and modulator X that are constructed in the proof: the remainder graph $G - X$ is a linear forest, a disjoint union of paths. By adding vertices of degree two to G we may connect all the paths in $G - X$ into a single path. Since degree-2 vertices do not affect the 3-colorability of a graph, this does not change the answer to the instance and ensures that $G - X$ is a single path. Hence we obtain:

Corollary 7.4. *3-Coloring on Path + kv graphs does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.*

The remainder of this section is devoted to the proof of various explicit lower bounds on the degree of the polynomial in the kernel size bounds. We use the machinery of Dell and van Melkebeek [80]. Although the result [80, Theorem 1] was originally phrased in terms of an oracle communication protocol, we only need the following weaker statement.

Proposition 7.4 ([80]). *If NP $\not\subseteq$ coNP/poly then for any $\varepsilon > 0$ and $q \geq 3$ there is no polynomial-time algorithm that transforms an instance x of q-CNF-SAT on n variables into an equivalent instance x' of a decidable problem with bitsize $|x'| \in \mathcal{O}(n^{q-\varepsilon})$.*

We first consider the q-Coloring problem on Empty + kv graphs, giving a lower bound that almost matches the upper bound we obtained earlier. Afterwards we present a general theorem that shows how to construct kernel size lower bounds from large NO-certificates to q-List Coloring.

Recall that an instance of q-NAE-SAT is a formula built from the conjunction of clauses containing at most q literals each, which is satisfied if at least one literal in each clause evaluates to FALSE and at least one evaluates to TRUE. By relating q-CNF-SAT to $(q + 1)$-NAE-SAT (both parameterized by the number of variables) through a folklore reduction, we obtain a compression lower bound for $(q + 1)$-NAE-SAT. By relating the latter to $(q + 1)$-Coloring on Empty + kv graphs we can obtain the following theorem.

Theorem 7.4. *For every $q \geq 4$ and $\varepsilon > 0$, q-Coloring on Empty + kv graphs does not have a kernel of bitsize $\mathcal{O}(k^{q-1-\varepsilon})$ unless NP $\subseteq$ coNP/poly.*

Proof. The proof consists of the following steps. We first show that there is a linear-parameter transformation (Definition 3.1) from q-CNF-SAT parameterized by the number of variables n to $(q + 1)$-NAE-SAT parameterized by n. As the second step we give a linear-parameter transformation from q-NAE-SAT parameterized by n to q-Coloring on Empty $+ k$v graphs. By combining these two transformations, we can use a kernelization algorithm for $(q+1)$-Coloring on Empty $+ k$v graphs to reduce the size of an instance of q-CNF-SAT in the following way. Start by transforming a given instance of q-CNF-SAT on n variables into an instance of $(q+1)$-NAE-SAT on $\mathcal{O}(n)$ variables, which in turn is transformed into $(q+1)$-Coloring on Empty $+ k$v graphs with $k \in \mathcal{O}(n)$, and finally apply the kernelization algorithm to this instance. Hence we see that a kernel with $\mathcal{O}(k^{(q+1)-1-\varepsilon})$ bits for $(q+1)$-Coloring would output an instance of size $\mathcal{O}(n^{q-\varepsilon})$, and by the result of Dell and van Melkebeek (Proposition 7.4) such a sparsification algorithm for any $q \geq 3$ would imply NP $\subseteq$ coNP/poly. Hence we find that for $q \geq 3$ the problem $(q+1)$-Coloring on Empty $+ k$v graphs cannot have kernels with bitsize $\mathcal{O}(k^{(q+1)-1-\varepsilon})$ unless NP $\subseteq$ coNP/poly. By a change of variables this shows that for $q \geq 4$ the q-Coloring on Empty $+ k$v graphs problem cannot have kernels with $\mathcal{O}(k^{q-1-\varepsilon})$ bits, which implies the theorem. To complete the proof it therefore suffices to give the two linear-parameter transformations.

Claim. *There is a linear-parameter transformation from q-CNF-SAT parameterized by n to $(q+1)$-NAE-SAT parameterized by n.*

Proof. This transformation can be considered folklore, and was used, e.g., by Knuth [156, §6]; we repeat it here for completeness. Let ϕ be a q-CNF-SAT-formula on n variables. Obtain the $(q+1)$-NAE-SAT formula ϕ' on $n+1$ variables by creating a single new variable z, and adding the positive literal z to each clause of ϕ. Any satisfying assignment of ϕ is transformed into a satisfying not-all-equal assignment of ϕ' by setting z to FALSE. In the other direction, any not-all-equal assignment that satisfies ϕ' and sets z to FALSE, is also a satisfying satisfiability assignment for ϕ. But if we have a not-all-equal assignment that sets z to TRUE, then we must also obtain a satisfying not-all-equal assignment if we flip the truth assignment of each variable, leading to a not-all-equal assignment that makes z FALSE and hence implies that ϕ is satisfiable. Thus the two instances are equivalent and, since $n' = n+1$, this constitutes a linear-parameter transformation. $\diamondsuit$

Claim. *There is a linear-parameter transformation from q-NAE-SAT parameterized by n to q-Coloring on Empty $+ k$v graphs.*

Proof. Consider an instance of q-NAE-SAT on n variables, and let the clauses be $C_1, \ldots, C_m$. We build a graph G and a modulator $X \subseteq V(G)$ such that $G - X \in$ Empty.

Again we construct a clique to act as our palette, this time using q colors and corresponding vertices $p_1, \ldots, p_q$; each vertex will take a different color in each q-coloring of G.

For each variable x_i for $i \in [n]$ we make $2q$ vertices $T_{i,1}, \ldots T_{i,q}, F_{i,1}, \ldots, F_{i,q}$. For all $j \in [q]$ we make $T_{i,j}$ adjacent to $F_{i,j}$. Then we make a cycle through successive vertices $T_{i,1}, \ldots, T_{i,q}$ and back to $T_{i,1}$. We now connect the vertices to

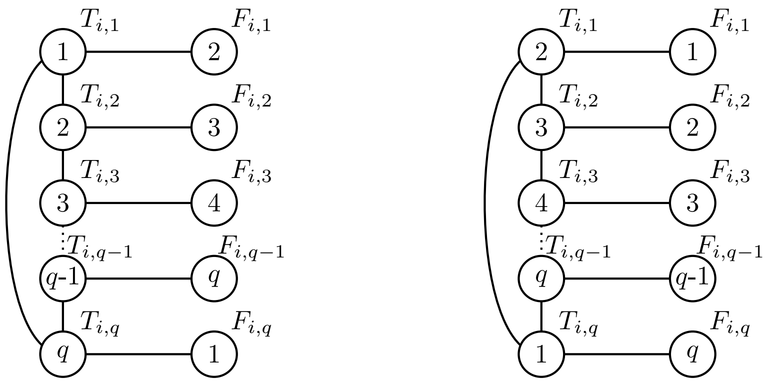


Figure 7.3: The two colorings for the variable gadget for x_i corresponding to assigning TRUE or FALSE respectively.

the palette, to ensure that there are only two different q-colorings for this gadget (modulo changing the permutation of the colors on the palette). When we discuss the effect of the gadget on potential q-colorings for G, we will indicate by color i ($i \in [q]$) the color that vertex p_i receives. Now do as follows. For all $j \in [q]$ we make $T_{i,j}$ and $F_{i,j}$ adjacent to all vertices of the palette except vertices p_i and p_{i+1}, ensuring that $T_{i,j}$ and $F_{i,j}$ can only take colors i or $i+1$. We evaluate these numbers modulo q, e.g., $T_{i,q}$ is adjacent to all palette vertices except p_q and p_1.

It is crucial to observe the following about these variable gadgets: each vertex $T_{i,j}$ can take only two different colors, but each possibility is also one of the only two choices for a neighbor on the cycle. For example, vertex $T_{i,1}$ can take color 1 or 2; if it has color 2 then vertex $T_{i,2}$ must take color 3, vertex $T_{i,3}$ must take color 4 and so on. Similarly, if $T_{i,1}$ has color 1 then we can make the same argumentation following the cycle in the other direction. Furthermore, since any vertices $T_{i,j}$ and $F_{i,j}$ have the same two possible colors j and $j+1$ (modulo q), one will take color j and the other must take color $j+1$. This means that effectively there are only two possible colorings for the gadget, which are shown in Fig. 7.3.

The first coloring, with $T_{i,1}$ colored 1, will be interpreted as assigning TRUE to x_i, the other one corresponds to assigning FALSE. We emphasize the key property: if x_i is TRUE, then each vertex $T_{i,j}$ will be colored j (and each $F_{i,j}$ will be colored $j+1$). If x_i is FALSE then each $F_{i,j}$ will be colored j (and each $T_{i,j}$ will be colored $j+1$).

We will now add one vertex c_k for each clause C_k. Let $C_k = (\ell_1 \vee \ldots \vee \ell_q)$. For each ℓ_j ($j \in [q]$) with $\ell_j = x_i$, we connect c_k to $T_{i,j}$. For each ℓ_j ($j \in [q]$) with $\ell_j = \neg x_i$, we connect c_k to $F_{i,j}$. It is easy to see that this has the desired effect, using the assignment corresponding to a given coloring: if all literals evaluate to TRUE, then c_k is connected to one vertex of each color in $[q]$. Then however, c_k cannot be properly colored. The same holds when all the literals evaluate to FALSE.

We briefly sketch the correctness of this reduction. If the graph G can be properly q-colored, then take the assignment corresponding to the coloring of the variable gadgets. The added vertices c_k ensure that there is no clause with literals all evaluating to TRUE or all FALSE. Conversely, let the formula be satisfiable and color the vertex gadgets according to a satisfying assignment. It follows easily, that among the q neighbors of any clause vertex c_k at most $q-1$ colors are used: indeed, consider two adjacent literals in the clause C_k, i.e., ℓ_j and ℓ_{j+1}, such that ℓ_j evaluates to FALSE and ℓ_{j+1} to TRUE. Using Fig. 7.3 it is easy to verify that the neighbor of c_k that represents ℓ_j is colored $j+1$ (since ℓ_j evaluates to FALSE), and the neighbor of c_k representing ℓ_{j+1} is also colored $j+1$ (as ℓ_{j+1} evaluates to TRUE). Hence out of the q neighbors of c_k there are two that share the same color, leaving at least one color free for c_k. This implies that the partial coloring can be extended to all clause vertices independently.

Clearly, the clause vertices c_k form an independent set. Thus we may define the modulator X as the vertices of the palette together with all vertices of the variable gadgets; the size of this set is $|X| = 2qn + q$, which is linear in n for fixed q. It is easy to see that $G - X \in$ EMPTY. Hence the instance (G, X) is a valid output of a linear-parameter transformation, which completes the proof. $\diamondsuit$

By the argument given at the beginning of the proof, the two claims together prove the theorem. $\square$

The proof of Theorem 7.4 shows that an improved compression lower bound for q-NAE-SAT also yields a better lower bound for q-COLORING on EMPTY + kV graphs. In particular, if it would be proven that for $q \geq 3$ q-NAE-SAT on n variables cannot be compressed in polynomial time into an equivalent instance on $\mathcal{O}(n^{q-\varepsilon})$ bits, then the kernel of Lemma 7.2 is optimal up to $k^{o(1)}$ factors.

The astute reader may notice that Theorem 7.4 actually implies Theorem 7.1. As the earlier theorem is a good example of the technique of cross-composition, gives a nice separation between the results for CHROMATIC NUMBER and q-COLORING, and was published in an earlier paper than Theorem 7.4, we have chosen to present it nonetheless. The remainder of this section revolves around the following notion.

Definition 7.4. Let $\mathcal{F}$ be a graph class, and let (G, L) be a NO-instance of q-LIST COLORING with $G \in \mathcal{F}$. Then (G, L) is an *irreducible* NO-*instance of* q-LIST COLORING *on* $\mathcal{F}$ if for all vertices $v \in V(G)$, the subinstance on graph $G - \{v\}$ is a YES-instance.

The following theorem shows how to turn irreducible NO-instances of q-LIST COLORING into explicit kernel size lower bounds.

Theorem 7.5. *If $\mathcal{F}$ is closed under disjoint union and there is an irreducible* NO-*instance of* $(q-2)$-LIST COLORING *on $\mathcal{F}$ containing $t \geq 3$ vertices, then q-*COLORING *on $\mathcal{F} + k$*V *graphs does not admit a kernel with* bitsize $\mathcal{O}(k^{t-\varepsilon})$ *for any $\varepsilon > 0$, unless NP $\subseteq$ coNP/poly.*

Proof. We proceed as in the proof of Theorem 7.4. We will show how the irreducible NO-instance on t vertices can be used to give a linear-parameter transformation from t-CNF-SAT parameterized by the number of variables n to q-COLORING on $\mathcal{F} + k\mathrm{v}$ graphs. So assume that (H, L) is a NO-instance of $(q-2)$-LIST COLORING with $|V(H)| = t$ such that any vertex deletion turns it into a YES-instance. Let the list function L for this instance assign admissible colors from the set $[q-2]$, and number the vertices of H in an arbitrary way as $h_1, \ldots, h_t$.

Now consider an input to t-CNF-SAT, which consists of clauses $C_1, \ldots, C_m$, where each clause contains exactly t literals. Each literal is of the form x_i or $\overline{x_i}$ for $i \in [n]$. We build a graph G and a modulator $X \subseteq V(G)$ such that $|X| = 2n + q$ and $G - X \in \mathcal{F}$. Construct a clique on q vertices $p_1, \ldots, p_q$ to act as a palette. For each variable x_i for $i \in [n]$ we make vertices T_i and F_i and add the edge $\{T_i, F_i\}$ to G. Make T_i and F_i adjacent to all palette vertices except p_{q-1} and p_q. We use the irreducible NO-instance (H, L) to create gadgets for the clauses. For each clause C_j with $j \in [m]$, add a disjoint copy of the graph H to G and denote it by H^j on vertices $h_1^j, \ldots, h_t^j$. For each $i \in [t]$ make vertex h_i^j adjacent to all palette vertices p_s with $s \in [q-2] \setminus L(h_i)$, and to the vertex p_q. If the i-th literal of C_j is x_s (resp. $\overline{x_s}$) then make h_i^j adjacent to T_s (resp. F_s). This concludes the description of the graph G. Use the set $X := \{T_i, F_i \mid i \in [n]\} \cup \{p_1, \ldots, p_q\}$ as the modulator. Since $G - X$ is a disjoint union of copies of $H \in \mathcal{F}$, and since $\mathcal{F}$ is closed under disjoint union by assumption, it follows that $G - X \in \mathcal{F}$ which proves that X is indeed a modulator to $\mathcal{F}$ of size linear in n. Let us now prove the correctness of the reduction.

Claim. *Graph G is q-colorable if and only if the t-*CNF-SAT *instance is satisfiable.*

Proof. $(\Rightarrow)$ Assume that function $f \colon V(G) \to [q]$ gives a proper q-coloring of G. Since a proper coloring gives unique colors to the vertices of a clique, we may assume without loss of generality that for the palette vertices p_i with $i \in [q]$ we have $f(p_i) = i$. Consider some variable index $i \in [n]$. By adjacency of the vertices T_i and F_i to the palette vertices, and by the edge $\{T_i, F_i\}$ it follows that one of T_i is colored $q-1$ and the other is colored q. Consider the truth assignment that sets a variable x_i to TRUE if and only if $f(T_i) = q$. We will show that this is a satisfying assignment for the input formula.

So consider a clause C_j of the input formula, which is represented by the graph H^j that is a copy of H. By adjacency of vertices $h_1^j, \ldots, h_t^j$ to the palette, it is easy to verify that h_i^j is assigned a color in $L(h_i) \cup \{q-1\}$ by the proper coloring f. For the next step, assume for a contradiction that no vertex of the graph H^j is colored with $q-1$. Then the coloring induced by f is a $(q-2)$-list-coloring of H^j that satisfies the list requirements from L, contradicting the assumption that (H, L) is a NO-instance. Hence there is at least one vertex h_i^j that is colored with $q-1$. But then the corresponding literal-vertex T_s or F_s that is adjacent to h_i^j cannot be colored $q-1$, so it is colored q. By our choice of valuation function, the literal evaluates to TRUE which shows that clause C_j is satisfied. Since this holds for every clause, the formula is satisfiable.

($\Leftarrow$) Assume that $v \colon [n] \to \{\textsc{true}, \textsc{false}\}$ is a satisfying assignment. We construct a proper q-coloring $f \colon V(G) \to [q]$ of G as follows:

- For the palette vertices p_i with $i \in [q]$ define $f(p_i) := i$.
- For each $i \in [n]$ with $v(i) = \textsc{true}$, set $f(T_i) := q$ and $f(F_i) := q - 1$.
- For each $i \in [n]$ with $v(i) = \textsc{false}$, set $f(T_i) := q - 1$ and $f(F_i) := q$.

It is easy to see that this partial assignment is proper; we show how to extend it to the remainder of the graph. The remainder consists of a disjoint copy H^j for each clause $j \in [m]$. For each such H^j function v satisfies at least one literal of clause j. Suppose that the i-th literal is satisfied, and look at the corresponding vertex h^j_i. By the assumption that (H, L) is an irreducible NO-instance of $(q-2)$-List Coloring, it follows that the subinstance on graph $H - \{h_i\}$ is a YES-instance. Take a $(q-2)$-list-coloring of $H - \{h_i\}$ and use it for the vertices of $H^j \setminus \{h^j_i\}$. By the correspondence between the list function and the adjacencies to the palette, and since the literal-vertices F_i and T_i only take colors $q - 1$ and q, which are not used in a $(q-2)$-list-coloring, this leads to a proper extension of the coloring f onto the graph $H^j \setminus h^j_i$. Observe that at this point, all neighbors of h^j_i in G have received a color, and that no such neighbor is colored with $q-1$: for the neighbors in $H^j \setminus h^j_i$ this follows from the $(q-2)$-list-coloring, for the palette vertices it follows by construction and for the literal-vertex adjacent to h^j_i this is ensured by the definition of f. Hence we can safely set $f(h^j_i) = q - 1$ to extend f onto the entire graph H^j_i. Since the various copies of H are mutually nonadjacent we can perform this color extension independently for all copies to obtain a proper coloring of G. $\diamondsuit$

The claim shows that the linear-parameter transformation is correct. It is easy to see that it can be computed in polynomial time. To establish the theorem, observe that if q-Coloring on $\mathcal{F} + k$v graphs would have a kernel with bitsize $\mathcal{O}(k^{t-\varepsilon})$, then we could compress an instance of t-CNF-SAT on n vertices by transforming it to the q-Coloring on $\mathcal{F} + k$v graphs instance with parameter value $k \in \mathcal{O}(n)$, and then applying the kernelization algorithm to obtain an equivalent instance of q-Coloring of bitsize $\mathcal{O}(n^{t-\varepsilon})$. Since q-Coloring is obviously decidable, by Proposition 7.4 a kernel of such bitsize does not exist unless NP $\subseteq$ coNP/poly. $\square$

As a graph on $\mathcal{O}(k^{t/2-\varepsilon})$ vertices can be encoded in $\mathcal{O}(k^{t/2-\varepsilon})^2 = \mathcal{O}(k^{t-2\varepsilon})$ bits using its adjacency matrix, Theorem 7.5 gives the following corollary.

Corollary 7.5. *If $\mathcal{F}$ is closed under disjoint union and there is an irreducible* NO-*instance of $(q-2)$*-List Coloring *on $\mathcal{F}$ containing $t \geq 3$ vertices, then q*-Coloring *on $\mathcal{F} + k$v graphs does not admit a kernel with* $\mathcal{O}(k^{t/2-\varepsilon})$ vertices *for any $\varepsilon > 0$ unless NP $\subseteq$ coNP/poly.*

In the next section we will prove the existence of large irreducible NO-instances of q-List Coloring to supply explicit kernel size lower bounds.

7.4.2 Lower Bounds to the Sizes of No-Certificates for q-List Coloring

In Section 7.3.2 we constructed functions that bound the sizes of NO-certificates for q-LIST COLORING on the graph classes $\biguplus$SPLIT, $\biguplus$COCHORDAL, and COGRAPH. The bounds we obtained were exponential in the number of colors q used in the instances. In this section we prove that the exponential dependence on q cannot be avoided, by giving explicit constructions of large irreducible NO-instances. We start by showing that for the class of paths, there is *no* function depending on q alone that can bound the size of NO-certificates, which explains why there is no polynomial kernel for the 3-COLORING problem on PATH + kv graphs.

Lemma 7.6. *For every $q \geq 2$ and every even integer $t \geq 2$, there is an irreducible* NO*-instance of q-*LIST COLORING *on a path graph containing t vertices.*

Proof. Let G be the path on successive vertices $v_1, \ldots, v_t$ with $t \geq 2$ an even integer. Define $L(v_1) = L(v_t) := \{1\}$, and let $L(v_i) := \{1, 2\}$ for $1 < i < t$. We claim that (G, L) is an irreducible NO-instance of q-LIST COLORING. To see that it is a NO-instance, observe that v_1 has only one color on its list and must therefore be colored 1, which forces v_2 to be colored 2, which forces v_3 to be colored 1, and so on: coloring v_1 with 1 forces all even-numbered vertices to be colored 2. But this prevents v_t from being assigned a color on its list, since t is even and $L(v_t) = \{1\}$. After a single vertex v_i is removed from the path, we may find a proper q-list-coloring by alternatingly assigning 1 and 2 from the beginning of the path until reaching v_i, and also alternating 1 and 2 backwards starting from v_t. Hence (G, L) is indeed an irreducible NO-instance. □

There is one unified construction that simultaneously provides a lower bound for the three considered graph classes $\biguplus$SPLIT, $\biguplus$COCHORDAL, and COGRAPH.

Lemma 7.7. *For every even integer $q \geq 2$ there is an irreducible* NO*-instance of q-*LIST COLORING *on a graph in the set* SPLIT $\cap$ COCHORDAL $\cap$ COGRAPH*, containing $q/2 + \binom{q}{q/2}$ vertices.*

Proof. Let q be even and consider the q-LIST COLORING instance constructed as follows. Initialize G as a clique X on $q/2$ vertices and set $L(x) := [q]$ for all $x \in X$. For every $S \in \binom{[q]}{q/2}$ add a vertex v_S to G with $N_G(v_S) := X$ and $L(v_S) := S$. Let the resulting instance be (G, L), and observe that G is a split graph on $q/2 + \binom{q}{q/2}$ vertices since it partitions into an independent set and a clique. Since split graphs are chordal and closed under complementation we have $G \in$ COCHORDAL. To see that G is a cograph observe that it is the join of a clique and an independent set, both of which are cographs, and that cographs are closed under taking joins. We will prove that (G, L) is irreducible: it is a NO-instance of q-LIST COLORING, but any vertex removal turns it into a YES-instance.

We first show that (G, L) is a NO-instance. Assume for a contradiction that $f\colon V(G) \to [q]$ is a proper list-coloring of G. Since X is a clique, all vertices of X receive different colors under f. Let $S := f^{-1}(X)$ be the $|X| = q/2$ colors assigned to X by f. During the construction of G we added a vertex v_S with neighborhood X and list $L(v_S) := S$ to the graph. But the choice of v_S ensures that all colors of $L(v_S)$ occur on the neighbors of v_S, and therefore f cannot be a proper list-coloring; contradiction. Hence (G, L) is indeed a NO-instance.

Now we prove that any vertex-deletion turns (G, L) into a YES-instance. So let $z \in V(G)$ be an arbitrary vertex, and consider the subinstance $(G - \{z\}, L')$ where L' is the restriction of L to the remaining vertices.

- If $z \in X$, then assign the $q/2-1$ remaining vertices of the clique $X' := X - \{z\}$ colors 1 up to $q/2 - 1$; this is compatible with their lists. Now consider the remaining vertices. For each $S \in \binom{[q]}{q/2}$ the vertex v_S has a list of $|S| = q/2$ admissible colors, and after removal of z vertex v_S has only $q/2-1$ neighbors left in $G - \{z\}$. Hence there must be one color on the list of v_S that is not yet used on a neighbor, and we use that color for v_S. Since the vertices not in X form an independent set, we may assign these colors independently to obtain a proper q-list-coloring of $(G - \{z\}, L')$.
- Now consider the remaining case that $z \notin X$, so $z = v_S$ for some $S \in \binom{[q]}{q/2}$. Give each vertex of X a unique color from the set S. For each set $S' \in \binom{[q]}{q/2} \setminus \{S\}$ it remains to choose a color for the vertex $v_{S'}$. By definition we have $L'(v_{S'}) = S'$. The neighbors of $v_{S'}$ in $G - \{v_S\}$ are exactly the vertices of the clique X and they use up all colors from S; so the colors in $S' \setminus S$ can be used on $v_{S'}$. Since $S' \neq S$ and $|S'| = |S| = q/2$ it follows that there is at least one color in $S' \setminus S$ that is available for use on $v_{S'}$, and as before we can use the fact that the vertices not in X form an independent set to argue that this leads to a proper q-list-coloring of the instance $(G - \{z\}, L')$.

Since this shows that (G, L) is an irreducible NO-instance with $G \in$ SPLIT $\cap$ COCHORDAL $\cap$ COGRAPH, this concludes the proof. $\square$

An overview of the various upper and lower bounds for NO-certificate and kernel sizes can be found in Table 7.1. We conclude the section with the following corollary, which results from combining the previous lemma with Corollary 7.5.

Corollary 7.6. *For every even integer $q \geq 4$ and $\varepsilon > 0$, the q-*COLORING *problem on parameterized* $\biguplus$SPLIT$+kv$, $\biguplus$COCHORDAL$+kv$, *or* COGRAPH$+kv$ *graphs does not admit a kernel with $\mathcal{O}\left(k^{\frac{1}{2}(q/2-1+\binom{q-2}{q/2-1})-\varepsilon}\right)$ vertices unless NP $\subseteq$ coNP/poly.*

7.5 Concluding Remarks

We studied the kernelizability of graph coloring problems within a hierarchy of structural parameterizations of the input graph, obtaining several positive and negative results. Our first result shows that effective preprocessing in terms of polynomial kernelizability is likely to be impossible when the number of colors is

Table 7.1: Kernel bounds for q-Coloring on parameterized graph families.

Parameterized graph family	NO-certificate bound		Kernel bitsize bound	
	upper	lower	upper	lower
Empty + kv	1	1	$\mathcal{O}(k^q)$	$\Omega(k^{q-1-\varepsilon})$
Path + kv	∞	∞	$\mathcal{O}(q^k)$	No $\mathcal{O}(k^{f(q)})$
$\dot\bigcup$Split + kv	$q + 4^q$	$q/2 + \binom{q}{q/2}$	$\mathcal{O}(k^{2q(q+4^q)})$	$\Omega(k^{q/2-1+\binom{q-2}{q/2-1}-\varepsilon})$
$\dot\bigcup$Cochordal + kv	$(q+1)!$	$q/2 + \binom{q}{q/2}$	$\mathcal{O}(k^{2q(q+1)!})$	$\Omega(k^{q/2-1+\binom{q-2}{q/2-1}-\varepsilon})$
Cograph + kv	2^{q^2}	$q/2 + \binom{q}{q/2}$	$\mathcal{O}\left(k^{2q\cdot 2^{q^2}}\right)$	$\Omega(k^{q/2-1+\binom{q-2}{q/2-1}-\varepsilon})$

The table shows upper- and lower bounds for the sizes of NO-certificates for q-List Coloring on graphs in $\mathcal{F}$, and upper- and lower bounds for the number of bits in kernels for q-Coloring on $\mathcal{F} + k$v graphs. In the order of the listed graph families, the upper bound on the number of bits needed for kernel representations follows a) from Lemma 7.2, b) from the existence of an $\mathcal{O}^*(q^k)$-time FPT algorithm [37], and c,d,e) by squaring the number of vertices that results from Theorem 7.2. The kernel size upper bounds holds for all values of q, whereas the lower bounds hold for all even $q \geq 4$; they are obtained using the size of a NO-certificate for $(q-2)$-List Coloring with Theorem 7.5.

part of the input. The results in the remainder of the chapter showed that when the number of colors is fixed as a constant, an interesting kernelization complexity landscape arises.

Ever since the first paper on parameterized coloring problems by Cai, it has been known that for a graph class $\mathcal{F}$ the parameterized complexity of Chromatic Number on $\mathcal{F} + k$v graphs is strongly related to the complexity of List Coloring on $\mathcal{F}$: if List Coloring on $\mathcal{F}$ is polynomial-time decidable then the corresponding parameterized problem lies in FPT [48, Theorem 3.3]. Marx [177] gave a refined view of the interaction between Precoloring Extension on $\mathcal{F}$ under parameterizations relating to the number of precolored vertices, and the complexity of Chromatic Number under structural parameterizations. The results in this chapter give another example of such a connection. We have shown that for graph classes $\mathcal{F}$ closed under disjoint union and vertex deletion, the existence of polynomial kernels for q-Coloring on $\mathcal{F} + k$v graphs is governed by the behavior of NO-certificates for q-List Coloring on graphs in $\mathcal{F}$. When there is an upper bound $g(q)$ on the size of NO-subinstances, then there is a kernel with $\mathcal{O}(k^{q \cdot g(q)})$ vertices. In the other direction, a lower bound on the size of NO-subinstances of $(q-2)$-List Coloring in the form of an explicit irreducible NO-instance on t vertices, shows that no kernel exists with $\mathcal{O}(k^{t/2-\varepsilon})$ vertices for any $\varepsilon > 0$, unless NP $\subseteq$ coNP/poly.

To apply our general theorems that provide upper and lower bounds on the size of kernelizations, we proved bounds on the size of NO-certificates for q-List Coloring on the graph classes Path, $\dot\bigcup$Split, $\dot\bigcup$Cochordal, and Cograph.

For the class of paths, no function of q can bound the size of NO-certificates. For other classes we were able to establish the existence of such functions, which grow exponentially with q. The lower bounds show that this exponential dependence on q is necessary, and that therefore the number of vertices in kernels for the considered parameterized coloring problems must grow very rapidly with the number of colors used: the degree of the polynomial that represents the kernel size must be exponential in q. Table 7.1 gives an overview of the bounds that were obtained for the various parameterized graph classes.

We conclude with some directions for further work. One could try to find larger classes of graphs with bounded-size NO-certificates for q-LIST COLORING. Since such a class cannot contain all paths, a natural candidate would be the class of P_5-free graphs, a superclass of cographs in which q-COLORING is known to be polynomial-time decidable for every fixed q [136]. Sandwiched between the cographs and the P_5-free graphs, one might also consider P_4-sparse graphs or related graph classes [45, §11.4].

The parameter hierarchy we considered uses vertex-deletion distance to well-studied graph classes as the parameter. The question of kernelizability can also be asked for the edge-deletion and edge-completion variants of these parameters [48, 177]. This will result in quite a different boundary between tractability and intractability: it is not hard to see that q-COLORING on LINEARFOREST $\pm k$e graphs admits a polynomial kernel by deleting vertices of degree at most two, whereas Theorem 7.3 shows that this is not the case on LINEARFOREST $+ k$v graphs. A final challenge for future work is to close the gap between the upper and lower bounds on kernel sizes for q-COLORING on EMPTY $+ k$v graphs.

The algorithm I would like to see
Is of polynomial degree
But it's elusive
Nobody has found conclusive
Evidence that we can find the longest path.
—*Dan Barrett, 1988*

8 Path and Cycle Problems

This chapter explores the existence of polynomial kernels for structural parameterizations of various path and cycle problems, such as LONG CYCLE, DISJOINT PATHS, and DISJOINT CYCLES. As the natural parameterizations of these problems do not admit polynomial kernels unless NP $\subseteq$ coNP/poly [24, 43], the study of structural parameterizations gives an alternative route to provably effective preprocessing procedures for these problems.

We show how to shrink a bipartite graph while preserving certain kinds of matchings, and use the technique to obtain quadratic-vertex kernels for several path and cycle problems parameterized by vertex cover. When it comes to parameterizations by max leaf number, we obtain polynomial kernels through a *win/win* approach (cf. [94]). A combination of these ideas leads to polynomial kernels by deletion distance to a cluster graph. We also consider the deletion distance to an outerplanar graph as the parameter, establishing superpolynomial lower bounds. Finally we explore the area of path problems with forbidden pairs, providing FPT-classifications, a $W[1]$-hardness proof, and kernel lower bounds.

This chapter is based on the paper "Kernel Bounds for Path and Cycle Problems" by Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch (6th International Symposium on Parameterized and Exact Computation [30]). The revised presentation here is based on the journal version that appeared in Theoretical Computer Science [31].

8.1 Preprocessing Path and Cycle Problems

Connectivity problems such as Long Path [sol] and Disjoint Paths [sol] play important theoretical and practical roles in the field of parameterized complexity. On the practical side, Long Path [sol] [117, ND29] has applications in computational biology [204] where the involved parameter is fairly small, thus giving an excellent opportunity to apply parameterized algorithms to find optimal solutions. On the theoretical side, these problems have triggered the development of very powerful algorithmic techniques. The Disjoint Paths [sol] problem [199] lies at the heart of the Graph Minors Algorithm, and is the source of the *irrelevant-vertex technique*. The *color coding* technique of Alon et al. [7] to solve Long Path [sol] has found a wide range of applications and extensions [55, 155], and new methods of solving Long Path [sol] are still developed [18]. Despite the success stories of parameterized algorithms for these problems, the quest for polynomial kernels has resulted in mostly negative results. Indeed, the failure to find a polynomial kernel for Long Path [sol] was one of the main motivations for the development of the kernel lower bound framework of Bodlaender et al. [24]. Using the framework it was shown that Long Path [sol] does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly, even when restricted to very specific graph classes such as planar cubic graphs. It did not take long before related connectivity problems such as Disjoint Paths [sol] [43], Disjoint Cycles [sol] [43], Connected Vertex Cover [sol] [83], and restricted variants of Connected Dominating Set [sol] [75] were also shown not to admit polynomial kernels unless NP $\subseteq$ coNP/poly.

Thus it seems that connectivity requirements in a problem form a barrier to polynomial kernelizability when it comes to the natural parameterization by solution size. In order to obtain useful preprocessing procedures for such problems, we may therefore investigate the kernelization complexity for nonstandard parameters. Early work by Fellows et al. [103] shows that such a different perspective can yield polynomial kernels: they proved that Hamiltonian Cycle parameterized by the max leaf number of the input graph (see Appendix A.5) admits a linear-vertex kernel. In this chapter we study the existence of polynomial kernels for various structural parameters such as the max leaf number, the size of a vertex cover, and the vertex-deletion distance to simple graph classes such as cluster graphs and outerplanar graphs.

Parameterization by vertex cover

The general kernelization tools developed in Chapter 4 can be applied to Long Path and Long Cycle parameterized by vertex cover. For a vertex cover of size k, this results in kernels with $\mathcal{O}(k^3)$ vertices. Our first goal in this chapter is to improve the bound on the number of vertices in the kernel to quadratic. Towards this end we prove a theorem about bipartite graphs. It roughly says the following. If we consider a bipartite graph G with partite sets X and Y, and are interested in the subsets of Y that can be covered by a matching, then we

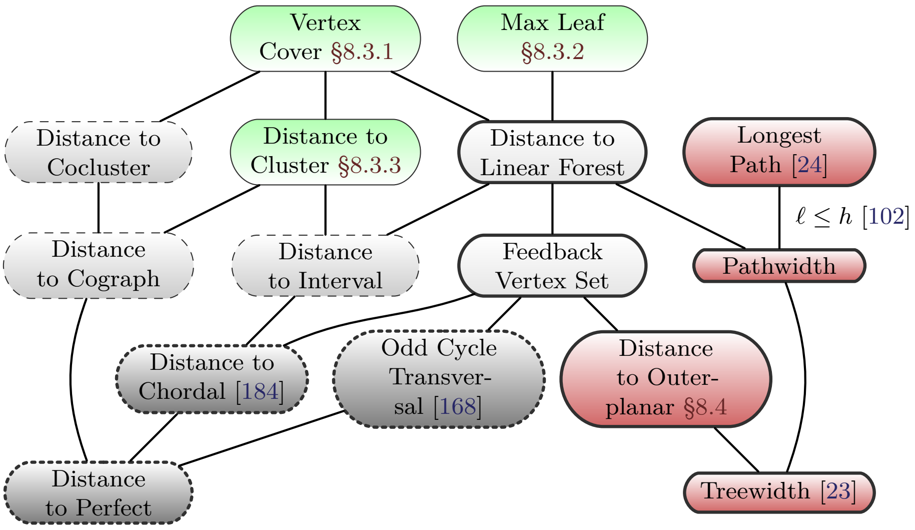


Figure 8.1: Complexity overview for various parameterizations of Long Path, assuming suitable formalizations. Larger parameters are drawn higher. For deletion distance parameters we assume that a modulator is given along with the input. The parameterization by longest path length is formalized as Long Path [sol]. The shading indicates that a parameterization is either para-NP-complete ▨, FPT with currently unknown kernelization complexity ▭, FPT but (conditionally) lacking a polynomial kernel 🟥, FPT with a polynomial kernel 🟩, or of unknown complexity ⬚. When a line between parameters is labeled by a bound, the lower-drawn parameter is represented by ℓ and the higher-drawn parameter by h. Unlabeled relationships are either given in Fig. 2.1, or follow trivially from graph class inclusions. The length of the longest path is measured in the number of edges.

may compute a maximum matching in G and remove the unmatched vertices of X. In the resulting smaller graph, exactly the same subsets of Y can be covered as in the original graph. To obtain quadratic-vertex kernels parameterized by vertex cover, we apply the theorem to the bipartite graph that represents possible connections between pairs of vertices in the cover, through a vertex outside the cover. We obtain kernels for Long Path [vc] and Long Cycle [vc] with $\mathcal{O}(k^2)$ vertices. The resulting kernelization scheme is widely applicable; we show how small modifications result in quadratic-vertex kernels for Disjoint Paths [vc] and Disjoint Cycles [vc].

Parameterization by max leaf number

We then consider the existence of polynomial kernels for other structural parameterizations. We parameterize by the max leaf number, aiming to lift the polynomial kernel for Hamiltonian Cycle by Fellows et al. [103] to the more general Long

Cycle problem and its variants. The main difficulty in this generalization lies in the fact that for the Hamiltonian Cycle problem, the length of paths is irrelevant and edges between adjacent degree-two vertices can therefore safely be contracted; any Hamiltonian cycle has to traverse the edge. When it comes to Long Cycle, such contractions could affect the answer to the problem by changing the lengths of such degree-two paths. To overcome this obstacle we exploit the fact that the structure guaranteed by having max leaf number k makes it possible to transform an instance of Long Cycle into a weighted cycle problem on $\mathcal{O}(k)$ vertices. We then use a win/win approach. If the number of vertices in the original graph is small, then the binary encoding of the weight values takes $\mathcal{O}(k)$ bits, resulting in a representation of the instance of total bitsize $k^{\mathcal{O}(1)}$. If the input graph is large with respect to k, then the running time of a Held-Karp [132] style dynamic program on the $\mathcal{O}(k)$-vertex weighted graph can be bounded by a polynomial in the input size. In both cases this results in a polynomial kernel by standard techniques. Having developed the win/win approach for Long Cycle parameterized by max leaf number, we show that it can be adapted to give polynomial kernels for Long Path, Disjoint Paths, and Disjoint Cycles, under the same parameterization.

Parameterization by cluster graph modulator

The win/win technique developed for the parameterization by max leaf number turns out to be applicable in other settings as well. Combining the general approach with structural insights, we obtain a polynomial kernel for Long Cycle parameterized by the distance to a cluster graph. This is an interesting parameter since it generalizes the vertex cover number, without restricting the density of the graph: while graphs with a vertex cover of size k have at most $\binom{k}{2} + kn$ edges, cluster graphs can have $\binom{n}{2}$ edges. Similarly as before, we can extend our positive results to the problems Long Path, Disjoint Paths, and Disjoint Cycles.

Parameterization by outerplanar graph modulator

Having established the existence of polynomial kernels for several structural parameterizations, one wonders how far down into the parameter hierarchy of Fig. 2.1 we can obtain polynomial kernels. For the case of Disjoint Paths and Disjoint Cycles, a limit is given by the results of Bodlaender, Thomassé, and Yeo [43]: their construction implies that these two problems do not admit polynomial kernels parameterized by the vertex-deletion distance to a path, unless NP $\subseteq$ coNP/poly. As this lower bound propagates to smaller parameters such as the feedback vertex number, it gives a good indication of the limits of efficient preprocessing for the disjoint paths and cycles problems.

When it comes to the variants of Hamiltonian Cycle, however, no such lower bounds were previously known. Although we have not been able to settle the kernelization complexity of Hamiltonian Cycle parameterized by feedback vertex set (deletion distance to treewidth one), we prove a superpolynomial kernel

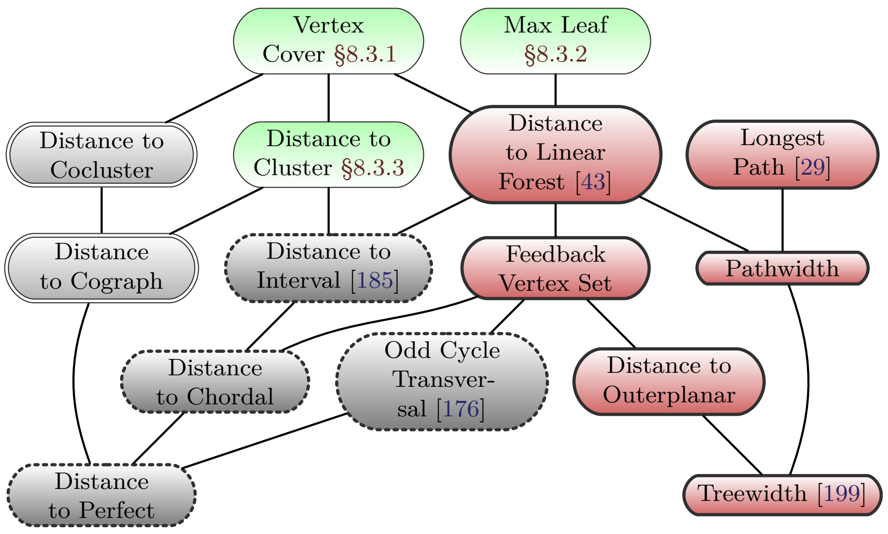


Figure 8.2: Complexity overview for various parameterizations of Disjoint Paths, assuming suitable formalizations. Larger parameters are drawn higher. For deletion distance parameters we assume that a modulator is given along with the input. The shading indicates that a parameterization is either para-NP-complete ▨, FPT but (conditionally) lacking a polynomial kernel 🟥, FPT with a polynomial kernel 🟩, or contained in XP but unknown to be FPT or $W[1]$-hard ▭. Relationships between parameters are given in previous figures or follow trivially from graph class inclusions. The XP-classification for distance to cographs [125] is based on the unpublished observation that if Disjoint Paths is polynomial-time solvable in graph class $\mathcal{F}$, then it is XP on $\mathcal{F} + k$v graphs.

lower bound parameterized by the distance to an outerplanar graph (which is at most the distance to treewidth two). The lower bound employs the cross-composition framework of Chapter 3. As Long Cycle is a generalization of Hamiltonian Cycle, the lower bound trivially extends to the former problem.

The implications of our results for the kernelization complexity of Long Path and Disjoint Paths, with respect to the hierarchy of parameters, are visualized in Fig. 8.1 and Fig. 8.2.

Path problems with forbidden pairs

While treating path and cycle problems, we also initiate the study of the parameterized complexity of path problems with forbidden pairs [117, GT54]. In this setting we are typically given a graph with specified terminals s and t and a list of forbidden vertex pairs, and the goal is to optimize the length of an $s - t$ path containing at most one vertex from each pair. Gabow et al. [115] introduced the problem in a directed setting, motivated by an application to the generation of test paths through the flow graphs of computer programs. They proved that testing the existence of an $s-t$ path respecting forbidden pairs is NP-complete in acyclic digraphs with in-degree and out-degree at most two.

Several path problems with forbidden pairs are even difficult to approximate. For the minimization and maximization problems that aim to optimize the length of a path that respects the forbidden pairs — regardless of its endpoints — the following is known [14, 150]: assuming P $\neq$ NP there is a positive real ε such that no polynomial-time algorithm can obtain a multiplicative factor $\mathcal{O}(n^{\varepsilon})$ approximation. Subsequent work by Kolman and Pangrác [158] and by Kovac [161] focused on the structure of the forbidden pairs and identified polynomial-time solvable cases of the problem.

We study structural parameterizations of path problems with forbidden pairs. Our results show that not just the structure of the input graph, but also the structure of the forbidden pairs, strongly influences the problem complexity. Besides obtaining FPT classifications and a $W[1]$-hardness proof, our main result in this direction is a superpolynomial kernel lower bound for a parameterization that simultaneously bounds the structure of the graph and the forbidden pairs.

Related work

Chen and Flum showed that Long Maximal Path [sol] is in FPT, and that Long Maximal Induced Path [sol] is $W[2]$-complete [56]. Chen, Flum, and Müller studied various forms of kernelization lower bounds, and showed amongst others that Long Pointed Path [sol] does not admit a parameter non-increasing polynomial kernelization unless P = NP, and that Long Path [sol] does not have a polynomial kernel on connected planar graphs unless NP $\subseteq$ coNP/poly [57]. For some recent algorithmic results on the other considered problems we refer to the following papers: Adler et al. [3] (Disjoint Paths in planar graphs), Björklund [17] (Hamiltonian Cycle), and Björklund et al. [18] (Long Path).

Organization

In Section 8.2 we show a useful property of bipartite matchings, which is needed to obtain our kernelization results. In Section 8.3 we give polynomial kernels for the mentioned path and cycle problems when parameterized by a vertex cover (Section 8.3.1), the max leaf number (Section 8.3.2), or a modulator to a cluster graph (Section 8.3.3). In Section 8.4 we prove kernelization lower bounds for Hamiltonian Cycle. Section 8.5 contains our results for path problems with forbidden pairs. We conclude in Section 8.6.

8.2 A Property of Maximum Matchings in Bipartite Graphs

The following theorem shows how to shrink a bipartite graph while preserving certain kinds of matchings. It will be useful for various reduction rules.

Theorem 8.1. *Let G be a bipartite graph with partite sets X and Y. Let $M \subseteq E(G)$ be a maximum matching in G. Let $X_M \subseteq X$ be the set of vertices in X that are endpoints of an edge in M. Then, for each $Y' \subseteq Y$, if there exists a matching M' in G that covers Y', then there exists a matching M'' in $G[X_M \cup Y]$ that covers Y'.*

Proof. Let G, M, and X_M be as stated in the theorem. Suppose the theorem does not hold for some $Y' \subseteq Y$, and let M' be a matching in G that covers Y'. Over all such matchings M', take one that covers the largest number of vertices in X_M. By assumption M' is not a matching in $G[X_M \cup Y]$, so there is a vertex $y_0 \in Y'$ that is matched in M' to a vertex in $X \setminus X_M$, say x_0. We use an iterative process to derive a contradiction, maintaining the following invariants:

- $x_0 \notin X_M$.
- $\{x_j, y_j\} \in M'$ for $0 \leq j \leq i$.
- $\{y_j, x_{j+1}\} \in M$ for $0 \leq j < i$.
- The vertices x_j for $0 \leq j \leq i$ are distinct members of X, and vertices y_j for $0 \leq j \leq i$ are distinct members of Y.

It is easy to verify that given our choice of x_0, y_0 these invariants are initially satisfied for $i = 0$. We now present a case distinction that either results in a contradiction, or finds x_{i+1} and y_{i+1} that satisfy the invariants.

- If y_i is not matched under M, then the sequence $(x_0, y_0, \ldots, x_i, y_i)$ is an M-augmenting path in G since x_0 and y_i are not matched under M, and all edges $\{y_j, x_{j+1}\}$ for $0 \leq j < i$ are contained in M. Hence $M'' := M \setminus \{\{y_j, x_{j+1}\} \mid 0 \leq j < i\} \cup \{\{x_j, y_j\} \mid 0 \leq j \leq i\}$ is a matching in G larger than M, contradicting that M is maximum. In the remaining cases we may therefore assume y_i is matched under M, say $\{y_i, x_{i+1}\} \in M$.
- If there is an index $0 \leq j \leq i$ such that $x_{i+1} = x_j$ then $j > 0$ (since $x_0 \notin X_M$) and the edges $\{y_i, x_{i+1}\}$ and $\{y_{j-1}, x_j\}$ are both contained in M and are distinct edges since $y_{j-1} \neq y_i$, contradicting the fact that M is a matching. Hence x_{i+1} is distinct from x_j for $0 \leq j \leq i$.
- If x_{i+1} is not covered by M' then the matching $M'' := M' \setminus \{\{x_j, y_j\} \mid 0 \leq j \leq i\} \cup \{\{y_j, x_{j+1}\} \mid 0 \leq j \leq i\}$ contains as many edges as M' but covers more vertices of X_M, contradicting the choice of M'. Hence x_{i+1} is covered by M', say $\{x_{i+1}, y_{i+1}\} \in M'$.
- If there is an index $0 \leq j \leq i$ such that $y_{i+1} = y_j$ then $\{x_{i+1}, y_{i+1}\}$ and $\{x_j, y_j\}$ are two distinct edges in M' incident on y_{i+1}, contradicting that M' is a matching. Hence y_{i+1} is distinct from y_j for $0 \leq j \leq i$.

- Now observe that with this choice of x_{i+1} and y_{i+1} the invariants hold for $i+1$, and we may proceed with the next step of the process.

By the last invariant the process must end. As it can only end by reaching a contradiction, the assumption that there is no matching in $G[X_M \cup Y]$ that covers Y' must be false. This proves the claim. $\square$

8.3 Polynomial Kernels for Path and Cycle Problems

This section provides polynomial kernels for various path and cycle problems when parameterized by a vertex cover (Section 8.3.1), the max leaf number (Section 8.3.2), or a modulator to a cluster graph (Section 8.3.3).

8.3.1 Parameterization by Vertex Cover

In this section we consider path and cycle problems when parameterized by the size k of a given vertex cover. We focus mainly on the Long Cycle problem, for which we present a kernel with $\mathcal{O}(k^2)$ vertices.

> Long Cycle [vc]
> **Input:** A graph G, an integer ℓ, and a vertex cover X of G.
> **Parameter:** The size $k := |X|$ of the vertex cover.
> **Question:** Does G contain a cycle of length at least ℓ?

We will need only one reduction rule to get a polynomial kernelization. The reduction rule will only work for instances that ask for a cycle of length five or more. Since instances with $\ell \leq 4$ can be solved trivially in polynomial time (for example, by testing for all ℓ-tuples over $V(G)$ whether they form a cycle), we can kernelize such instances by solving them and outputting a constant-size instance with the same answer. In the remainder we therefore assume that $\ell > 4$. The reduction rule is based on the following concept.

Definition 8.1. The *connection graph* H corresponding to an instance (G, ℓ, X) of Long Cycle [vc] is a bipartite graph. One partite set consists of the vertices in the independent set $I := V(G) \setminus X$, and the other consists of all (unordered) pairs of distinct vertices in X. There is an edge between a vertex $v \in I$ and a vertex representing the pair $\{p, q\} \in \binom{X}{2}$ if and only if v is adjacent to both p and q in G.

The following rule uses a maximum matching in the connection graph to reduce the instance size.

Reduction Rule 8.1. Given (G, ℓ, X) and $I := V(G) \setminus X$, construct the corresponding connection graph H and compute a maximum matching M in H. Let J be the vertices of I covered by M. Return the instance $(G[X \cup J], \ell, X)$.

Observation 8.1. *In Rule 8.1, $|J|$ is at most the number of pairs of distinct vertices in X. The resulting graph $G[X \cup J]$ has at most $k + \binom{k}{2} \in \mathcal{O}(k^2)$ vertices.*

Correctness of the rule follows from the following lemma.

Lemma 8.1. *Let (G, ℓ, X) be an instance of* Long Cycle [vc] *with $\ell > 4$, and let (G', ℓ, X) be the instance returned by Rule 8.1. Then G has a cycle of length at least ℓ if and only if G' has a cycle of length at least ℓ.*

Proof. Observe that $G' = G[X \cup J]$ is an induced subgraph of G so all cycles in G' also exist in G. It remains to prove the converse.

Let C be a cycle of length at least $\ell \geq 5$ in G. Clearly, as $I = V(G) \setminus X$ is an independent set, any vertices of I that are in C must be neighbored by vertices of X on C. Let $v_1, \ldots, v_r$ be all vertices of I contained in C and let p_i and q_i be the predecessor and successor of v_i on C, respectively. We know that $r \leq k$, but there might be only few vertices of I on C. Since C has length at least five, it follows that $\{p_i, q_i\} \neq \{p_j, q_j\}$ for all $i, j \in [r]$ with $i \neq j$ (else C would have length four). To show that G' contains a cycle of length at least ℓ, it suffices to find replacements for all vertices v_i that are not in J (and hence not in G'); for this we will use the matching.

The connection graph H with partite sets I and $\binom{X}{2}$ has a matching M covering $W := \{\{p_1, q_1\}, \ldots, \{p_r, q_r\}\}$: match each pair to the corresponding vertex v_i. By Rule 8.1 the set J contains the endpoints in I of some maximum matching of H. Hence by Theorem 8.1 there is a matching M' covering W in $H[J \cup \binom{X}{2}]$.

Let v_i' denote the vertex matched to $\{p_i, q_i\}$ by M', for $i \in [r]$. It is easy to see that we may replace each v_i on C by v_i', since v_i' is also adjacent to p_i and q_i in G. We obtain a cycle C' that intersects I only in vertices of J. Also, as all pairs $\{p_i, q_i\}$ are different, no vertex v_i' is required twice. Hence C' is also a cycle of G', and of length at least ℓ. □

The kernelization result now follows from Lemma 8.1 and Observation 8.1, noting that Rule 8.1 can easily be performed in polynomial time.

Theorem 8.2. Long Cycle [vc] *has a kernel with $\mathcal{O}(k^2)$ vertices.*

We remark that, in the obtained kernel, the number of edges may still be cubic in k. This results in an overall size bound of $\mathcal{O}(k^3)$ bits by using an adjacency matrix encoding on $G[X]$ and storing a k-bit vector for each $v \in V(G) \setminus X$ giving its neighbors in X. It would be interesting to know whether the number of edges can be reduced to $\mathcal{O}(k^2)$ and whether a size bound of $\mathcal{O}(k^2)$ bits could be shown to be tight, using recent results on polynomial lower bounds for kernelization [79, 80, 135].

Further problems parameterized by vertex cover

The same technique can be used for a number of additional problems, all parameterized by the size of a vertex cover. The basic argument is the same; the matching approach allows us to reroute any path or cycle such that it uses only matched vertices.

Corollary 8.1. LONG PATH [VC], DISJOINT PATHS [VC], *and* DISJOINT CYCLES [VC] *admit polynomial kernels with* $\mathcal{O}(k^2)$ *vertices.*

Proof. We briefly sketch how to modify the proof given for LONG CYCLE [VC] (Theorem 8.2).

For an instance (G, ℓ, X) of the LONG PATH [VC] problem, it is most convenient to introduce a universal vertex v (adjacent to all vertices of G) to obtain an equivalent instance $(G', \ell + 1, X \cup \{v\})$ of LONG CYCLE [VC]. Each ℓ-vertex path in G then corresponds to an $\ell + 1$-cycle in G' by adding v, and each $\ell + 1$-cycle in G' contains at least one ℓ-vertex path in G. We then apply the kernelization as for LONG CYCLE [VC], and finish up by removing v from the obtained instance.

For DISJOINT PATHS [VC] it is clear that each path fulfilling a request (s_i, t_i) must contain at least one vertex of X, hence YES-instances have at most $k = |X|$ request pairs. Thus we may assume that all vertices of request pairs are contained in X (else adding them will at most triple the size of X). Since the paths have to be vertex-disjoint, no two request pairs share any vertices (though the proof could be adapted to internally vertex-disjoint paths, and also to asking for multiple paths between certain pairs). It is hence clear that edges between vertices of different pairs cannot be part of any path. Therefore, the easiest way to argue correctness is to add all those edges to G, and observe that the kernelization for LONG CYCLE [VC] also preserves the existence of a cycle that traverses all request pairs in order, i.e., $(\ldots, s_1, \ldots, t_1, s_2, \ldots, t_2, s_3, \ldots)$. Such a cycle exists if and only if the ℓ requested disjoint paths exist and the answer to the instance is YES.

For DISJOINT CYCLES [VC] we may proceed essentially as for LONG CYCLE [VC], since each single cycle will be preserved (as the argument only comes down to providing the private shared neighbors). The only difference is that we also have to preserve cycles of length four, which may require two vertices of the independent set to be assigned (matched) to one pair of vertices of the vertex cover X. (We also must preserve cycles of length three of course, but only length four cycles may have the particular mentioned layout.) To do so, create the bipartite connection graph H corresponding to the instance, but duplicate all vertices corresponding to (unordered) pairs of vertices from X. This way, each pair can receive two private shared neighbors. $\square$

For HAMILTONIAN PATH [VC] and HAMILTONIAN CYCLE [VC] it is easy to see that any vertex cover of a YES-instance must have size at least $\lfloor \frac{|V(G)|}{2} \rfloor$, since vertices of the remaining independent set cannot be adjacent on a Hamiltonian path or cycle. Thus all nontrivial instances (G, X) satisfy $|V(G)| \leq 2|X| + 1 = \mathcal{O}(k)$.

8.3.2 Parameterization by Max Leaf Number

In this section we consider path and cycle problems parameterized by the max leaf number, i.e., the maximum number of leaves in any spanning tree of the graph. We take the max leaf number of a disconnected graph to be the sum of the max leaf numbers of its connected components (cf. Appendix A.5).

We will use LONG CYCLE as a running example but, as in Section 8.3.1, it is easy to generalize the arguments to further problems. As the max leaf number of a graph cannot be verified in polynomial time (unless P = NP), we consider the parameterization in the sense of a promise problem (see Section 2.2.1).

LONG CYCLE [ML]
Input: A graph G and an integer ℓ.
Parameter: An integer k that is promised to be the max leaf number of G.
Question: Is there a cycle in G of length at least ℓ?

The following reduction to a weighted setting is the crucial ingredient of the kernelization.

Lemma 8.2. *There is a polynomial-time algorithm that, given an n-vertex graph G and an integer ℓ, either decides that G has a cycle of length at least ℓ, or produces a graph H and an edge weight function $w\colon E(H) \to [n]$ such that:*

- *$V(H)$ is the set of vertices that have degree unequal to two in G.*
- *G has a cycle of length at least ℓ if and only if H has a cycle whose weight labels sum to ℓ or more.*

Proof. Given a graph G and integer ℓ, let B be the set of vertices of degree unequal to two. For each connected component of G consisting entirely of degree-two vertices, we test in polynomial time whether the cycle formed by the component has length ℓ or more. If so then we answer YES, and otherwise the component may safely be discarded. After this preprocessing step, all paths of degree-two vertices have well-defined endpoints in B.

Replace each such path connecting vertices $b, b' \in B$ by a direct edge between b and b', and set the weight of the edge equal to the total number of edges on the replaced degree-two path. Set the weight of existing edges between members of B to one. We obtain a multigraph on vertex set B, possibly with loops. A cycle of length ℓ in G corresponds to a cycle whose edge labels sum to ℓ in the multigraph. Clearly, a heaviest cycle will either consist of just two vertices, or use at most one of the edges between any two vertices. If there is a sufficiently long two-vertex cycle, then we answer that G has a cycle of length ℓ. Otherwise we may discard all but the heaviest edge connecting any two vertices in B. For each loop we may similarly answer YES, or remove it from the graph. The resulting simple graph H on vertex set B is given as the output, along with the weight function. The correspondence between cycles of length at least ℓ and heavy cycles in H follows from the discussion above. □

It is well known that a large graph having small max leaf number must contain long paths of degree-two vertices. The following bound was observed by Fellows et al. [103] based on work by Kleitman and West [153].

Lemma 8.3 (Cf. [103, Theorem 5])**.** *If a graph G has max leaf number at most k, then it is a subdivision of a graph on at most $4k-2$ vertices. In particular, G has at most $4k-2$ vertices of degree unequal to two.*

Combining this bound with Lemma 8.2, one of two good things must happen: either the weighted graph resulting from the lemma can be encoded in bitlength polynomial in the max leaf number, or the number of vertices in the original graph is large enough to allow the instance to be solved in polynomial time. For the latter purpose we require the following proposition. It follows from standard Held-Karp style dynamic programming [132].

Proposition 8.1. *There is an algorithm that, given a k-vertex graph G and weight function $w\colon E(G) \to [n]$, computes the weight of a heaviest cycle in G in time $\mathcal{O}(2^k k^3 n)$.*

The polynomial kernel for the considered parameterization of Long Cycle exploits the described win/win approach.

Theorem 8.3. Long Cycle [ml] *admits a polynomial kernel.*

Proof. Let (G, ℓ, k) be an instance of Long Cycle [ml] on n vertices. Let B denote those vertices that have degree unequal to two in G. If $|B| > 4k-2$ then by Lemma 8.3 the promise is not fulfilled, as the max leaf number of G is greater than k: output a constant-size no-instance. Otherwise we have $|B| \leq 4k-2$. If $\ell > |V(G)|$ then we may similarly output a no-instance. Apply Lemma 8.2 to (G, ℓ). If the algorithm concludes that G has a cycle of length at least ℓ, then output a constant-size yes-instance. Else let (H, w) be the resulting weighted graph on vertex set B, and observe that the weights assigned by w are at most n.

If $n > 2^{4k}$ then we apply the dynamic programming algorithm of Proposition 8.1 on (H, w) to decide the problem in time $\mathcal{O}(2^{|B|}|B|^3 n) \leq \mathcal{O}(2^{4k}k^3 n) \leq \mathcal{O}(n^2 k^3)$ time, and output a constant-size instance with the same answer. Otherwise, we know that $\log n \in \mathcal{O}(k)$, and therefore that the binary encoding of each weight assigned by w takes $\mathcal{O}(k)$ bits. Clearly ℓ can also be encoded in $\mathcal{O}(k)$ bits. Encoding H as an adjacency matrix, we obtain an equivalent instance of a weighted Long Cycle problem whose total bitlength is polynomial in k. This weighted cycle problem is easily seen to be contained in NP, which implies the existence of a Karp reduction back to the classical problem Long Cycle (cf. [43]). Applying this reduction to (H, w) we obtain an instance (G', ℓ') of Long Cycle that is equivalent to the input. Since the Karp reduction takes polynomial time, the resulting instance (G', ℓ') has bitsize polynomial in k. This will essentially be the output of the kernelization, but as a kernel is a reduction from a parameterized problem to itself, we need to output an instance of the parameterized (promise) problem LONG CYCLE [ML]. To define the output it therefore remains to choose a parameter value k'. We simply set k' to $|V(G')|$; this clearly forms an upper bound on the max leaf number of G', and is polynomial in k. The output is (G', ℓ', k'). □

Further problems parameterized by max leaf number

A polynomial kernel for HAMILTONIAN CYCLE [ML] was already found by Fellows et al. [103]. Kernels for HAMILTONIAN PATH [ML] as well as DISJOINT CYCLES [ML] can be obtained in a similar way, by observing that paths of consecutive degree-two vertices can be reduced to having only one internal vertex. For LONG PATH [ML] it is again necessary to use the binary encoding trick for the path lengths.

Corollary 8.2. LONG PATH [ML], HAMILTONIAN PATH [ML], *and* DISJOINT CYCLES [ML] *admit polynomial kernels.*

For DISJOINT PATHS, i.e., finding vertex-disjoint paths connecting ℓ terminal pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, some more work is necessary on the degree-two paths between vertices of degree unequal to two.

Theorem 8.4. DISJOINT PATHS [ML] *admits a polynomial kernel.*

Proof. Let $(G, \ell, \{(s_1, t_1), \ldots, (s_\ell, t_\ell)\}, k)$ be an instance of DISJOINT PATHS [ML]. Let B contain all vertices of degree unequal to two. If $|B| > 4k$ we return NO as, by Lemma 8.3, the max leaf number of G exceeds k. Similarly, we return NO if any vertex of B has degree greater than k. As in Lemma 8.2 we can get rid of all components of G that consist entirely of degree-two vertices: the existence of a solution inside these components can be determined in polynomial time, and they do not interact with the rest of the graph in any way. We can therefore either answer NO, or delete all components of degree-two vertices.

We now reduce the number of terminals on paths connecting vertices $b, b' \in B$ with internal vertices from $O := V(G) \setminus B$; the internal vertices of the paths have degree exactly two in G. Let us consider any such path P that contains at least five terminals; let $u_1, \ldots, u_r$ denote these terminals in the order of appearance (starting arbitrarily at one end of P), i.e., $\{u_1, \ldots, u_r\} \subseteq \{s_1, t_1, s_2, \ldots, t_\ell\}$. The key fact for the reduction is that a path connecting a terminal pair cannot contain further terminals as internal vertices. If there are terminals u_i, u_{i+1}, u_{i+2} such that $u_{i+1} = s_j$ but $t_j \notin \{u_i, u_{i+2}\}$ (or vice versa with s_j and t_j swapped) then the instance has no feasible solution and we return NO; this can be checked efficiently. If there is no such configuration of terminals, it follows that there is a sequence $u_i = s_j$, $u_{i+1} = t_j$, $u_{i+2} = s_{j'}$, and $u_{i+3} = t_{j'}$ among $u_1, \ldots, u_r$ (here the choice of $s.$ and $t.$ is without loss of generality). It is easy to see that the terminal pairs (s_j, t_j) and $(s_{j'}, t_{j'})$ must be connected by the obvious subpaths of P in any feasible solution. Furthermore, no other paths (connecting other terminal pairs) can use the subpath of P from u_i to u_{i+3}. Hence we may get an equivalent instance by replacing the subpath of P from u_i to u_{i+3} by a path (s_{j^*}, t_{j^*}) and by replacing the terminal pairs (s_j, t_j) and $(s_{j'}, t_{j'})$ by the new pair (s_{j^*}, t_{j^*}). This reduces the number of terminals on P by two; iteration of the argument reduces the number of terminals to at most four per path.

To see that the max leaf number does not increase, observe that the above modification is basically a contraction of some edges (terminals have no influence), and that it does not increase the number of connected components.

If there is a path consisting of three successive vertices of degree two, none of which are terminals, then we may safely contract one of the edges connecting them without changing the feasibility of the problem. Such operations do not increase the max leaf number, and do not create parallel edges or loops. Since the degree of vertices in B remains unchanged, the set B still contains the vertices of degree unequal to two in the resulting graph. Exhaustively applying this contraction rule has the following effect: each path between vertices $b, b' \in B$, whose internal vertices have degree two, contains at most 14 vertices: there are at most four terminals on this path, and each subsequence of three consecutive vertices contains a terminal. The resulting graph G' consists of the vertices B linked together by paths of no more than 14 vertices each. As each vertex in B has degree at most k, the number of vertices is bounded by $|B| + 14k|B| \in \mathcal{O}(k^2)$. We set $k' := k$, which is a valid upper bound on the max leaf number of G' if k bounds the max leaf number of G, and output the resulting instance. □

We remark that starting from a different structural claim about graphs of max leaf number at most k (cf. [142, Theorem 6]), the reduction process above can actually be shown to result in a linear-vertex kernel. In the interest of brevity we do not pursue this smaller bound here.

8.3.3 Parameterization by Cluster Graph Modulator

In this section, we consider path and cycle problems parameterized by vertex-deletion distance from cluster graphs. To this end, a cluster graph modulator X is provided along with the input. Again we consider the Long Cycle problem.

Long Cycle [cluster mod]
Input: A graph G, an integer ℓ, and a modulator $X \subseteq V(G)$ such that $G - X$ is a disjoint union of cliques.
Parameter: The size $k := |X|$ of the modulator.
Question: Does G contain a cycle of length at least ℓ?

We start with some simple tests to deal with trivial cases of the problem. Given an instance (G, ℓ, X), if there is a clique in $G - X$ with at least ℓ vertices, or a vertex in X with two neighbors in a clique of size $\ell - 1$, then G contains a cycle of length ℓ and we may output a constant-size yes-instance as the result of the kernelization. In the remainder we therefore assume that this does not occur. As a cycle must enter X between visits to different cliques of $G - X$, we observe the following.

Observation 8.2. *Cycles in G contain vertices of at most $|X|$ cliques of $G - X$.*

If a cycle uses at least one edge between vertices of a clique, then it can easily be extended to include all so far unused vertices of the clique. Hence, it can be seen that there is a preference for including the largest possible cliques and as many cliques as possible. This is used in the following reduction rule.

Reduction Rule 8.2. Let (G, ℓ, X) be an instance of LONG CYCLE [CLUSTER MOD] with $k := |X|$. For each pair $\{u, v\} \in \binom{X}{2}$, consider the cliques in $G - X$ and do the following:

- Mark $k + 1$ cliques that contain a shared neighbor of u and v.
- Mark the $k+1$ largest cliques that contain distinct vertices p, q with $p \in N_G(u)$ and $q \in N_G(v)$, breaking ties arbitrarily.

Obtain graph G' by deleting all unmarked cliques, and return (G', ℓ, X).

Clearly $G' - X$ is still a cluster graph, and if G' contains a cycle of length at least ℓ, then that cycle can also be found in its supergraph G. The following lemma completes the proof of correctness for Rule 8.2.

Lemma 8.4. *Let (G', ℓ, X) be obtained from an application of Rule 8.2 to (G, ℓ, X). If G has a cycle of length at least ℓ, then G' has a cycle of length at least ℓ.*

Proof. Assume that G contains a cycle of length at least ℓ. By our assumption on the structure of input instances, no clique in $G - X$ contains ℓ or more vertices, nor is there a clique of size $\ell - 1$ in which two vertices see the same member of X. Hence any cycle of length at least ℓ in G contains at least two vertices of X.

Let C be a cycle of length at least ℓ in G that contains as few vertices of unmarked cliques as possible. If C does not intersect unmarked cliques at all, then it is also a cycle in G' and we are done. Assume therefore that C contains vertices from a clique K in $G - X$ that was not marked, which therefore does not exist in G'. Let $P = (p_1, \ldots, p_r)$ denote one of the subpaths obtained by intersecting C with the vertex set of K. Further, let P_{uv} denote the extended subpath obtained by adding the vertices $u, v \in X$ that are adjacent to P on C. Without loss of generality we can assume that $P_{uv} = (u, p_1, \ldots, p_r, v)$. Note that $u \neq v$ since C intersects X in at least two vertices.

If $r = 1$ and $P_{uv} = (u, p_1, v)$, then by the marking process of Rule 8.2 and as K was not marked, there are $k + 1$ cliques in $G - X$ that contain a shared neighbor of u and v that were marked; consequently, these are present in G'. By Observation 8.2 at least one of those cliques, say K', is not used by C. We may replace p_1 by any shared neighbor of u and v in K'. We obtain a cycle C' of the same length as C, but containing fewer vertices of unmarked cliques; a contradiction to our choice of C.

Now, if $r > 1$, then $P_{uv} = (u, p_1, \ldots, p_r, v)$ with $p_1 \neq p_r$. Hence C visits a neighbor of u in K, and a different neighbor of v in K. As K is not marked, the reduction rule marked $k+1$ other cliques that are at least as large as K, which also contain such neighbors. Again, by Observation 8.2, there is a clique K' among them that is not used by C. The clique K' is at least as large as K and it contains vertices p and q with p adjacent to u and q adjacent to v. Hence, we may replace the subpath P_{uv} of C by a path from u to p, followed by a path from p to q using all vertices of K', and back to v; we call this P'_{uv}. Clearly, P'_{uv} is at least as long as P_{uv}, since P_{uv} could at most use all vertices of K (there might be other subpaths of C using K) and K' has at least that many vertices. Thus, replacing P_{uv} in C by P'_{uv} we obtain a cycle C' of at least the same length, which uses fewer vertices of unmarked cliques, contradicting the choice of C.

It follows that C contains only vertices of X and of marked cliques. Hence it exists also in G', completing the proof. □

For the remaining reduction we proceed as in Section 8.3.2. First we show that for each clique in $G-X$, we can identify $(k+1)^3$ vertices in the clique that we may safely take to be the entry- and exit points of long cycles on the clique; the remaining vertices in a clique can only be needed to extend the length of the cycle (i.e., they are visited after entering the clique and before leaving it). This allows for a straightforward FPT-algorithm of runtime $\mathcal{O}(k^{10k} \cdot n^c)$. Consequently, we may assume the number of vertices to be bounded by $\mathcal{O}(k^{10k})$, otherwise we can solve the whole instance in time $\mathcal{O}(n^{c+1})$. Thus the number of additional vertices in each clique may be encoded in binary, with coding length $\mathcal{O}(k \log k)$, giving a polynomial sized equivalent instance of a special version of our problem. Finally, we use a Karp reduction from this special version back to the basic problem, to obtain a polynomial kernel.

Lemma 8.5. *Given an instance (G, ℓ, X) of* Long Cycle [cluster mod]*, in polynomial time one can identify $(k+1)^3$ vertices in each clique of $G-X$ such that if (G, ℓ, X) is a* yes*-instance, then G contains a cycle of length at least ℓ that enters and leaves cliques of $G-X$ only through the identified vertices.*

Proof. Given (G, ℓ, X) mark for each pair of vertices $\{u, v\} \in \binom{X}{2}$ up to $2k+1$ shared neighbors in each clique. Furthermore, we mark for each vertex $v \in X$ up to $2k+1$ neighbors in each clique. We prove that cycles in G of length at least ℓ may be assumed to enter and leave cliques of $G-X$ through these marked vertices.

Assume that G contains a cycle of length at least ℓ, and let C be a cycle of length at least ℓ that enters and leaves cliques of $G-X$ through a minimum number of unmarked vertices (equivalently, C uses a minimum number of edges between the modulator and unmarked vertices). Without loss of generality C takes the form $(\ldots, u, p_1, \ldots, p_r, v, \ldots)$ where $u, v \in X$, all vertices p_i are from the same clique K of $G-X$, and at least p_1 is unmarked.

If $r=1$, then $C=(\ldots, u, p_1, v, \ldots)$. As p_1 is unmarked, the clique K contains $2k+1$ marked vertices that are shared neighbors of u and v, say $q_1, \ldots, q_{2k+1}$. If at least one of them is not used by C, say q_i, then we may replace p_1 by q_i, and thereby obtain a cycle of the same length as C that uses one less unmarked vertex to enter and leave cliques; a contradiction to the choice of C. So assume that C uses all vertices $q_1, \dots, q_{2k+1}$. By $k = |X|$ we know that at least one vertex, say q_j, is not adjacent to vertices of X on C, but is adjacent on C to other vertices of K. Therefore, we may swap the positions of p_1 and q_j in C: indeed, q_j is adjacent to u and v, and p_1 is adjacent to all other vertices of K (and clearly it is not adjacent to q_j on C). We obtain a cycle of the same length, which uses one less unmarked vertex to enter or leave cliques of $G - X$. This contradicts our choice of C: the new cycle now enters a clique at q_j instead of p_1.

Now we consider the case that $r > 1$, so $C = (\dots, u, p_1, \dots, p_r, v, \dots)$ with $p_1 \neq p_r$. Since p_1 is unmarked, there are $2k+1$ marked vertices in K that are adjacent to u, say $q_1, \dots, q_{2k+1}$. It follows that at least one of those vertices, say q_j, is not adjacent to X on C. We can now apply the same switching arguments as in the previous paragraph to obtain a cycle C' of the same length, which uses one less unmarked vertex to enter or leave cliques of $G - X$, contradicting our choice of C.

It follows that C enters and leaves cliques only through marked vertices. The marked vertices constitute the sets of vertices in the cliques as claimed by the lemma. There are at most $k \cdot (2k+1) + \binom{k}{2} \cdot (2k+1) \leq (k+1)^3$ marked vertices per clique of $G - X$, as claimed. □

Lemma 8.6. Long Cycle [cluster mod] *can be solved in* $\mathcal{O}(k^{10k} \cdot n^c) = 2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$ *time.*

Proof. Let (G, ℓ, X) be an instance of Long Cycle [cluster mod] and recall that $k := |X|$. We use Rule 8.2 to reduce the number of cliques in $G - X$ to $\mathcal{O}(k^3)$. Then we mark vertices according to Lemma 8.5. Assuming that (G, ℓ, X) is a yes-instance, it follows that there is a cycle of length at least ℓ in G that enters and leaves cliques only in marked vertices. The following brute force algorithm finds such a cycle if it exists:

1. If any clique in $G - X$ has size at least ℓ then return yes. Else at least one vertex of X must be used.
2. Try all ordered subsets $X' = \{x_1, \dots, x_t\}$ of X for the intersection of C with X.
3. For all pairs (x_i, x_{i+1}) (including (x_t, x_1)) try all cliques that contain neighbors of x_i and x_{i+1}, or, if possible, try the edge $\{x_i, x_{i+1}\}$ to connect x_i and x_{i+1} on C.
4. If a clique K has been chosen to connect x_i and x_{i+1} on C, then try all marked vertices of the clique for entering and leaving K on C, i.e., to find either vertices p and q with $C = (\dots, x_i, p, \dots, q, x_{i+1})$, or a single vertex p for $C = (\dots, x_i, p, x_{i+1}, \dots)$ (and check adjacency of p and possibly q to x_i and x_{i+1}).
5. Greedily connect the chosen vertices inside the cliques: for each clique where we have chosen the option $C = (\dots, x_i, p, \dots, q, x_{i+1}, \dots)$ we can

include all remaining vertices of the clique into our cycle. In case of $C = (\ldots, x_i, p, x_{i+1}, \ldots)$ this is not possible and no further vertices can be added.

It is easy to see that, given the existence of any cycle C' of length at least ℓ that enters and leaves cliques only through marked vertices, our simple algorithm finds a cycle of at least the same length: it will eventually try the same choice and order of vertices from X, and pick the same marked vertices to connect them to cliques. At that point it is easy to see that the greedy connection of the chosen marked vertices gives a cycle of at least the same length. Clearly, if our algorithm finds a cycle of length at least ℓ, then we have a YES-instance.

We close by giving an upper bound on the running time. It is easy to see that the dependence on the input size is bounded by (a low degree) polynomial. For each step of the algorithm, we bound the dependence on k in the runtime as follows:

1. This step can be performed in polynomial time.
2. There are less than $(k+1)^k = \mathcal{O}(k^k)$ ordered subsets X' of X.
3. We pick up to k cliques (possibly with repetition) out of the $\mathcal{O}(k^3)$ cliques of $G - X$, for a total of $\mathcal{O}(k^{3k})$ choices.
4. Up to k times, we pick two marked vertices among the $(k+1)^3$ vertices, i.e., we have $\mathcal{O}(k^{6k})$ choices.
5. This step can be performed in time polynomial in the input size.

This gives a total runtime of $\mathcal{O}(k^{10k} \cdot n^c) = 2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$. □

Theorem 8.5. LONG CYCLE [CLUSTER MOD] *admits a polynomial kernel.*

Proof. Let (G, ℓ, X) be an input instance and assume without loss of generality that it has the form described in the beginning of the section. Reduce it according to Rule 8.2 to ensure that G has at most $\mathcal{O}(k^3)$ cliques. If G has at least $2^{\mathcal{O}(k \log k)}$ vertices, then the algorithm of Lemma 8.6 can be used to solve the instance in time $n \cdot n^{\mathcal{O}(1)} = n^{\mathcal{O}(1)}$. Otherwise we use the fact that $n < 2^{\mathcal{O}(k \log k)}$ to create an equivalent instance of a different problem that can be encoded in bitlength polynomial in k. The new problem is defined as follows: given a graph whose vertices carry (positive) integer labels, is there a cycle such that the sum of its vertex labels is at least ℓ? The latter quantity is referred to as the length of the cycle. In the following, we will treat all vertices whose label we do not assign explicitly as having label one.

The weighted problem instance is created as follows. We mark vertices according to Lemma 8.5. In each clique of $G - X$, we replace all unmarked vertices by a single unmarked vertex with an integer label stating the number of unmarked vertices in that clique. In binary encoding, that label needs at most $\log n = \log(2^{\mathcal{O}(k \log k)}) = \mathcal{O}(k \log k)$ bits. For the so-encoded instance we ask for a cycle of length at least ℓ that enters and leaves cliques only through marked vertices, but we account the labeled vertices as subpaths with a number of vertices equal to their label. Clearly, this is equivalent to the original question.

In addition to the vertices of X, the instance has at most $\mathcal{O}(k^3)$ cliques each with at most $(k+1)^3+1$ vertices plus one integer label of bit size at most $\mathcal{O}(k \log k)$ per clique. Clearly, this gives a total size that is polynomial in k.

Finally, we observe that the question that we ask about this instance can be answered in NP. As Long Cycle is NP-complete, it remains so even when we append the distance to a cluster graph in unary as well as a reasonable encoding of a modulator X to the problem definition (both quantities can be encoded in n bits). Thus there is a Karp reduction from our alternative problem to Long Cycle [cluster mod]. By standard arguments (e.g., see Bodlaender et al. [43]) this implies that the latter problem admits a polynomial kernel: the result of the Karp reduction has bitsize polynomial in k, and is used as the output of the kernelization algorithm. □

Further problems parameterized by cluster graph modulator

The techniques introduced in this section can be used to obtain polynomial kernels for several other problems parameterized by a modulator to a cluster graph.

Corollary 8.3. Long Path [cluster mod]*,* Hamiltonian Cycle [cluster mod]*, and* Hamiltonian Path [cluster mod] *admit polynomial kernels.*

Proof. For an instance (G, ℓ, X) of Long Path [cluster mod], simply add a universal vertex v adjacent to all vertices of G and ask for a cycle of length $\ell + 1$. Furthermore, the vertex v is added to the modulator X, increasing the parameter value by one. Theorem 8.5 now creates an intermediate instance with $\mathcal{O}(|X|^6)$ vertices that reduces back to Long Path by a Karp reduction. (We may also remove the universal vertex to get an instance of Long Path with labeled vertices, resulting in a compression.)

It is straightforward to get similar reductions for Hamiltonian Cycle [cluster mod] and Hamiltonian Path [cluster mod] by asking for long cycles or paths respectively of length (at least) equal to the total number of vertices. However, it can easily be seen that, following the proof of Theorem 8.5 for an obtained instance of Long Cycle, one may replace each labeled vertex by a single unlabeled one and update the target length (such that it always matches the number of vertices). Indeed, any cycle using all vertices can be modified to contain all unmarked vertices as a single subpath. The number of those vertices (i.e., the single label in each clique) is then immaterial. Thus one obtains graphs with $\mathcal{O}(|X|^6)$ vertices. It is straightforward to make the modifications to get back to instances of Hamiltonian Cycle [cluster mod] and Hamiltonian Path [cluster mod] respectively. □

We now turn our attention to Disjoint Paths and Disjoint Cycles. In both problems the length of paths or cycles is not important, but there may be large numbers of paths and cycles inside each clique.

In Disjoint Paths [cluster mod] we can reduce to the setting where all requested pairs lie entirely in X, as follows. First, it is clear that request pairs (s_i, t_i) contained in the same clique of $G - X$, have the uniquely optimal path consisting only of s_i and t_i. Thus, all such vertex pairs may be deleted (both from the graph and the list of requests). To satisfy any other request pair the corresponding path needs to intersect X, bounding the maximum feasible number of requests by $|X|$; otherwise we reject the instance. Thus, including the vertices of all requests at most triples the size of X.

From this observation it is clear that a polynomial kernelization for Disjoint Paths [cluster mod] can be achieved by marking (or matching) enough vertices to allow for the necessary paths. The case for Disjoint Cycles [cluster mod] is similar: there are at most $|X|$ cycles that include vertices of X. All further cycles can be chosen as triangles inside single cliques. Again a marking procedure suffices, adding an additional step of removing triples of unmarked vertices from any single clique and each time reducing the target number of cycles by one (accounting for those trivial cycles not intersecting X). We obtain the following corollary.

Corollary 8.4. Disjoint Paths [cluster mod] *and* Disjoint Cycles [cluster mod] *admit polynomial kernels.*

8.4 Kernel Lower Bound for Hamiltonian Cycle

In this section we present a kernelization lower bound for Hamiltonian Cycle parameterized by the vertex-deletion distance to an outerplanar graph. Since this deletion distance linearly bounds the treewidth (Appendix A.5.2), the parameterized problem is easily seen to be fixed-parameter tractable using Courcelle's theorem. To establish a kernel lower bound we use the cross-composition technique of Chapter 3. As the source problem for the composition we use the following variant of Hamiltonian Path on bipartite graphs.

Hamiltonian $s - t$ Path in Bipartite Graphs
Input: A bipartite graph G with partite sets $A = \{a_1, \ldots, a_{n_A}\}$ and $B = \{b_1, \ldots, b_{n_B}\}$ with $n_B = n_A + 1$ such that b_1 and b_{n_B} have degree one in G.
Question: Does G contain a Hamiltonian path that starts in b_1 and ends in b_{n_B}?

Proposition 8.2. Hamiltonian $s - t$ Path in Bipartite Graphs *is NP-complete.*

Proof. Membership in NP is trivial. We prove hardness by a reduction from Hamiltonian $s - t$ Path, which is a classical NP-complete problem [117, GT39]. On input a graph G with distinguished vertices s and t, construct a graph G' on vertex set $V(G) \times \{1, 2, 3, 4\}$ with edges $\{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\} \mid v \in V(G)\}$

together with $\{\{v_1, u_4\}, \{v_4, u_1\} \mid \{u, v\} \in E(G)\}$. It is easy to verify that G has a Hamiltonian $s - t$ path if and only if G' has a Hamiltonian $s_1 - t_4$ path. The graph G' is bipartite since the vertices with odd subscripts and the vertices with even subscripts form two independent sets. To obtain an instance of HAMILTONIAN $s - t$ PATH IN BIPARTITE GRAPHS we add a new starting vertex s^* adjacent to s_1, a new vertex w adjacent to t_4, and a new ending vertex t^* adjacent to w. Taking the appropriate partite sets of the bipartition, labeling s^* as b_1 and t^* as b_{n_B}, we obtain an equivalent instance of HAMILTONIAN $s - t$ PATH IN BIPARTITE GRAPHS. □

The following property of the problem will be useful in our argumentation.

Proposition 8.3. *Let G be a bipartite graph with nonempty partite sets $A = \{a_1, \ldots, a_{n_A}\}$ and $B = \{b_1, \ldots, b_{n_B}\}$ with $n_B = n_A + 1$. If $C \subseteq E(G)$ is a set of edges such that:*

(I) C does not contain a cycle,
(II) all vertices $a \in A$ are incident on exactly two edges in C,
(III) no vertex $b \in B$ is incident on more than two edges in C,
(IV) the vertices b_1 and b_{n_B} are incident on at most one edge in C,

then C is a Hamiltonian path from b_1 to b_{n_B} in G.

Proof. Consider the subgraph G_C of G on edge set C and vertex set $V(G)$. As G is bipartite all edges have exactly one endpoint in each partite set. By (II) the total number of edges of G_C is therefore $2|A| = 2n_A$, while the number of vertices is $n_A + n_B = 2n_A + 1$. As G_C is acyclic (by (I)) and its vertex count exceeds its edge count by one, it is a tree.

We show that no vertex of $\{b_2, \ldots, b_{n_B-1}\}$ is a leaf of the tree G_C. Assume for a contradiction that b_i with $i \in [2 \ldots n_B - 1]$ has degree at most one in G_C. By (IV) the vertices b_1 and b_{n_B} have degree at most one, together accounting for two edges. The vertices with indices in the range $[2 \ldots n_B - 1] \setminus \{i\}$ have degree at most two by (III), together accounting for $2(n_B - 3)$ edges. Finally, vertex b_i has degree at most one, accounting for at most one extra edge. Hence the total number of edges is at most $2 + 2(n_B - 3) + 1 = 2n_A - 1$, contradicting the earlier observation that G_C has $2n_A$ edges.

Thus we may conclude that no vertex of $\{b_2, \ldots, b_{n_B-1}\}$ is a leaf in G_C. As each such vertex has degree at most two by (III), they must have degree exactly two. As the vertices in A also have degree exactly two by (II), we find that G_C is a tree in which all vertices except b_1 and b_{n_B} have degree two. It follows that b_1 and b_{n_B} are leaves of the tree, which shows that the tree is in fact a path between b_1 and b_{n_B} spanning the entire vertex set; hence it is a Hamiltonian path as desired. □

The problem for which we prove a lower bound is formally defined as follows.

Hamiltonian Cycle [outerplanar mod]
Input: A graph G and a modulator $X \subseteq V(G)$ such that $G - X$ is outerplanar.
Parameter: The size $k := |X|$ of the modulator.
Question: Does G have a Hamiltonian cycle?

Before presenting the construction we describe the main ideas. Suppose we combine a series of inputs $(G_1, A_1, B_1), \ldots, (G_r, A_r, B_r)$ to Hamiltonian $s-t$ Path in Bipartite Graphs by taking their disjoint union, and afterwards merging the B-sides of all graphs into a shared set B^* index-wise: taking the first vertex from each set B_i and identifying those into a single vertex b^*_1, identifying all second vertices into b^*_2, and so on. Assuming the sizes of the partite sets of all inputs coincide, a Hamiltonian path in one of the input graphs G_i corresponds to a path P_i in the resulting graph G^* that visits all the vertices in $A_i \cup B^*$. There are two problems with this crude construction: (i) we need to ensure that such a path P_i can be extended to a Hamiltonian cycle in G^*, and (ii) conversely we would like to extract from every Hamiltonian cycle in G^* a subpath that is Hamiltonian in one of the input graphs; the latter is only possible if the subpath between b^*_1 and $b^*_{n_B}$ restricts itself to vertices originating from a single input instance. Our cross-composition uses the "domino" gadget of Fig. 8.3 to resolve these two issues. The following property of the domino will be crucial.

Proposition 8.4. *Let G be a graph containing the* domino *(Fig. 8.3) as an induced subgraph, such that only the vertices labeled $\{a^+, a^-, \hat{a}^+, \hat{a}^-\}$ have neighbors outside the domino. A Hamiltonian cycle in G either contains:*

- *an a-*traversal *of the domino, which is a path between a^+ and a^- that visits all other vertices of the domino in between, or*
- *an $\hat{a}$-*traversal *of the domino, which is a path between $\hat{a}^+$ and $\hat{a}^-$ that visits all other vertices in between.*

This property of the domino gadget is used to remedy problems (i) and (ii) as follows. In our crude construction, we replace each vertex originating from a partite set A_i by a copy of the domino gadget, with suitable connections to the merged vertices B^*. We link these dominos together in a horizontal path, ensuring that a Hamiltonian path in an input graph G_i can be extended to a Hamiltonian cycle in G^* by traversing this horizontal path through the dominos that are not related to G_i; this solves issue (i).

To deal with issue (ii) we use the fact that a Hamiltonian cycle can only traverse a domino in two distinct ways, each with distinct entry- and exit points on the domino. In our construction, the entry- and exit vertices of the $\hat{a}$-traversal of the domino are only made adjacent to the merged vertices B^*. Hence each domino that uses an $\hat{a}$-traversal has vertices in B^* as its predecessor and successor on a Hamiltonian cycle.

The way we arrange the dominos on a path ensures that all dominos corresponding to the same input graph use the same type of traversal in a Hamiltonian

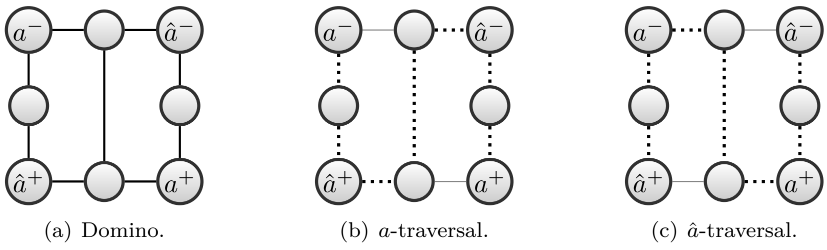


Figure 8.3: (a) The domino gadget with four terminal vertices $a^-, a^+, \hat{a}^-, \hat{a}^+$. If graph G contains the domino as a subgraph such that only these terminals are connected to the remainder of the graph, then any Hamiltonian cycle for G must use the a-traversal (b) or $\hat{a}$-traversal (c) of the domino.

cycle. If the dominos of two or more instances use $\hat{a}$-traversals, then there are not enough vertices in B^* to give each domino a unique predecessor and successor from that set. Hence there can be at most one instance whose dominos use $\hat{a}$-traversals. By restricting the connections to B^* we enforce that the dominos of at least one instance use the $\hat{a}$-traversal, resulting in soundness of the construction.

The graph G^* will be almost outerplanar: roughly speaking, removing the set B^* from the constructed graph will leave only a collection of dominos linked together into a path, and such graphs are easily seen to be outerplanar. Armed with this intuition we present the proof.

Theorem 8.6. Hamiltonian Cycle [outerplanar mod] *does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.*

Proof. By Corollary 3.2 and Proposition 8.2 it is sufficient to show that Hamiltonian $s-t$ Path in Bipartite Graphs cross-composes into Hamiltonian Cycle [outerplanar mod]. By choosing a suitable polynomial equivalence relation $\mathcal{R}$ it is sufficient to give an algorithm that, given[1] r well-formed instances $(G_1, A_1, B_1), \ldots, (G_r, A_r, B_r)$ of Hamiltonian $s-t$ Path in Bipartite Graphs such that $|A_i| = n_A$, $|B_i| = n_B$ with $n_B = n_A + 1$ for $i \in [r]$, outputs in polynomial time an instance (G^*, X^*) of Hamiltonian Cycle [outerplanar mod] that is a yes-instance if and only if the answer to one of the input instances is yes, and such that $|X^*|$ is polynomial in the size of the largest input instance plus $\log r$. Assume $A_i = \{a_{i,1}, \ldots, a_{i,n_A}\}$ and $B_i = \{b_{i,1}, \ldots, b_{i,n_B}\}$ for $i \in [r]$. We build a graph G^* as follows.

1. For $i \in [r]$, for $j \in [n_A]$ add a copy $O_{i,j}$ of the domino graph (Fig. 8.3) to G^* and label its terminals by $a^-_{i,j}, a^+_{i,j}, \hat{a}^-_{i,j}$ and $\hat{a}^+_{i,j}$.
2. As the next step we add edges to connect adjacent dominos. For $i \in [r]$ and $j \in [n_A - 1]$, add the edge $\{a^+_{i,j}, a^-_{i,j+1}\}$.

[1]Throughout this chapter we use r rather than t for the total number of inputs to a cross-composition algorithm, to prevent confusion with the t in $s-t$ path.

3. For each instance $i \in [r]$ add three special vertices x_i, y_i, and w_i, together with edges $\{w_i, x_i\}, \{x_i, a^-_{i,1}\}, \{a^+_{i,n_A}, y_i\}$. For $i \in [r-1]$ add the edges $\{y_i, w_{i+1}\}$.
4. Observe that at this stage, G^* is outerplanar. All vertices we add from this point on will go into the modulator X^* such that $G^* - X^*$ remains outerplanar.
5. We add three special vertices z^-, z, z^+ with edges with $\{z^-, z\}$ and $\{z, z^+\}$. Furthermore, we add edges $\{x_i, z^-\}$ and $\{z^+, y_i\}$ for $i \in [r]$.
6. For $j \in [n_B]$ add a vertex b^*_j to the graph G^*, and let B^* be the set of these vertices. Add edges $\{y_r, b^*_1\}$ and $\{b^*_{n_B}, w_1\}$.
7. As the last step of the construction we re-encode the behavior of the input graphs G_i into the instance. For $i \in [r]$, for each edge $\{a_{i,j}, b_{i,f}\}$ in $E(G_i)$ add the edges $\{\hat{a}^-_{i,j}, b^*_f\}$ and $\{\hat{a}^+_{i,j}, b^*_f\}$ to G^*. This concludes the description of G^*; see Fig. 8.4 for an example.

Now define $X^* := \{z^-, z, z^+\} \cup B^*$. The output of the cross-composition is the instance (G^*, X^*) of Hamiltonian Cycle [outerplanar mod]. It is easy to verify that $G^* - X^*$ is outerplanar, and that the construction can be carried out in polynomial time. The parameter $|X^*|$ is bounded by $1 + n_B$, which is sufficiently small. It remains to prove that G^* is a yes-instance if and only if the answer to one of the input instances is yes. We first establish some relevant properties of G^*.

Claim. *Let $C \subseteq E(G^*)$ be a Hamiltonian cycle in G^*. The following must hold.*

(i) If C uses an $\hat{a}$-traversal of some domino $O_{i,j}$ for $i \in [r]$ and $j \in [n_A]$, then there are distinct indices $f, f' \in [n_B]$ such that C consecutively visits b^*_f, the vertices of domino $O_{i,j}$, and $b^*_{f'}$.

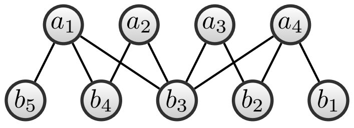
(ii) If $\{x_i, a^-_{i,1}\}$ is not *an edge on C for some $i \in [r]$, then none of the edges $\{a^+_{i,j}, a^-_{i,j+1}\}$ for $j \in [n_A - 1]$ are contained in C, nor is the edge $\{a^+_{i,n_A}, y_i\}$.*

(iii) There is an index $i^* \in [r]$ such that $\{x_{i^*}, a^-_{i^*,1}\}$ is not used on C.

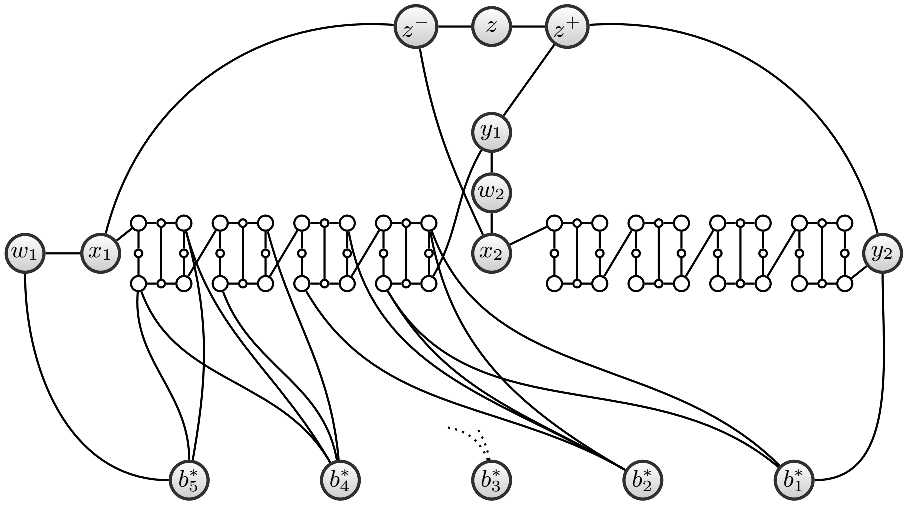
Proof. We prove the different components consecutively.

(i) Suppose C uses an $\hat{a}$-traversal of some domino $O_{i,j}$. By construction of G^* we know that the only neighbors of $\hat{a}^+_{i,j}$ and $\hat{a}^-_{i,j}$ that are not contained in the domino are of the form b^*_f for some $f \in [n_B]$. Since all vertices of the domino are used on the cycle C in an $\hat{a}$-traversal (see Fig. 8.3) and the vertices $\hat{a}^+_{i,j}$ and $\hat{a}^-_{i,j}$ each have only one neighbor on C within the domino, each of these vertices must have a neighbor on C that falls outside the domino; hence these must be of the form b^*_f and $b^*_{f'}$. We have $f \neq f'$ otherwise there would be a closed cycle containing the domino $O_{i,j}$ and a single vertex b^*_f: this cycle would not be Hamiltonian since it would not visit the vertex z.

(ii) Assume that $\{x_i, a^-_{i,1}\}$ is not on C, but at least one of the edges in the set $Z_i := \{\{a^+_{i,j}, a^-_{i,j+1}\} \mid j \in [n_A - 1]\} \cup \{\{a^+_{i,n_A}, y_i\}\}$ is used on C. Let j^* be smallest index such that the vertex a^+_{i,j^*} is the endpoint of an edge in $C \cap Z_i$.



(a) Input instance (G_1, A_1, B_1) of HAMILTONIAN $s-t$ PATH IN BIPARTITE GRAPHS.



(b) Output instance of HAMILTONIAN CYCLE [OUTERPLANAR MOD].

Figure 8.4: An example of the lower-bound construction of Theorem 8.6 when composing $r = 2$ inputs with $n_A = 4$ and $n_B = 5$. (a) The first input instance. (b) Resulting output instance. The edges between the right group of dominos and $\{b_1^*, \ldots, b_5^*\}$ have been omitted for readability, as have the edges between the left group of dominos and b_3^*. The labels of the dominos match the layout of Fig. 8.3; their indices increase from left to right.

Since a^+_{i,j^*} is endpoint of an edge in C and the other endpoint of this edge does not lie in the domino O_{i,j^*}, it follows that C contains at most one edge in the domino O_{i,j^*} that is incident on a^+_{i,j^*}. This rules out the possibility that C makes an $\hat{a}$-traversal of O_{i,j^*}, since that requires two edges within the domino incident on a^+_{i,j^*}. Because the construction of G^* together with Proposition 8.4 ensures there are only two possible traversals of the domino, we know that C must use an a-traversal of O_{i,j^*}. This traversal uses exactly one edge of the domino incident on a^-_{i,j^*}. Since a Hamiltonian cycle must contain exactly two edges incident on every vertex, this shows that C must contain some edge incident on a^-_{i,j^*} that is not in the domino O_{i,j^*}. But by construction of G^* there is only one such edge: if $j^* = 1$ then this edge is $\{x_i, a^-_{i,1}\}$ and otherwise this edge is $\{a^+_{i,j^*-1}, a^-_{i,j^*}\}$. But by the assumption at the start of the proof and our choice of j^*, this edge is not contained in C. Hence there is only one edge incident on a^-_{i,j^*} contained in C, contradicting the assumption that C is a cycle.

(iii) The Hamiltonian cycle C contains two edges incident on every vertex. By construction of G^* the vertex z^- is incident on the edge $\{z^-, z\}$ and on edges $\{x_i, z^-\}$ for $i \in [r]$. Hence if C contains two edges incident on z^- then at least one is of the form $\{x_{i^*}, z^-\}$ for $i^* \in [r]$. Now consider the vertex w_{i^*}. It has degree two by construction. Therefore both its incident edges are contained in C, and in particular $\{w_{i^*}, x_{i^*}\}$ is contained in C. So C contains two edges incident on x_{i^*} that are unequal to the edge $\{x_{i^*}, a^-_{i^*,1}\}$ and since only two edges incident to each vertex are contained in C, it follows that C does not contain $\{x_{i^*}, a^-_{i^*,1}\}$. $\diamondsuit$

Armed with the structural properties described by the claim, we prove that the constructed instance acts as the OR of the inputs.

Claim. *Graph G^* has a Hamiltonian cycle if and only if one of the input instances G_{i^*} has a Hamiltonian path starting in b_1 and ending in b_{n_B}.*

Proof. ($\Rightarrow$) Assume that G^* has a Hamiltonian cycle $C \subseteq E(G^*)$. By (iii) there is an index $i^* \in [r]$ such that $\{x_i, a^-_{i^*,1}\} \notin C$. By (ii) this implies there are no edges $\{a^+_{i^*,j}, a^-_{i^*,j+1}\}$ for $j \in [n_A - 1]$ contained in C, nor is the edge $\{a^+_{i^*,n_A}, y_i\}$. This shows that for each vertex $a^-_{i^*,j}$ the only edges incident on $a^-_{i^*,j}$ that can be contained in C are those of the domino $O_{i^*,j}$, which implies by Proposition 8.4 that C must do an $\hat{a}$-traversal of all dominos $O_{i^*,j}$ for $j \in [n_A]$. By (i) each traversal of a domino is preceded and succeeded by distinct vertices b^*_f and $b^*_{f'}$. By construction of G^* this can only occur if $\{a_{i^*,j}, b_{i^*,f}\}$ and $\{a_{i^*,j}, b_{i^*,f'}\}$ are edges of G_{i^*}. Now consider the following edge set:

$$C_{i^*} := \{\{a_{i^*,j}, b_{i^*,f}\} \mid b^*_f \text{ precedes or succeeds a domino } O_{i^*,j} \text{ on } C\}.$$

We prove using Proposition 8.3 that C_{i^*} is a Hamiltonian path in G_{i^*} between vertices $b_{i^*,1}$ and b_{i^*,n_B}. If C_{i^*} contains a cycle, then by definition of C_{i^*} we deduce that C contains a cycle that only visits B^* and the dominos; hence C contains a cycle avoiding z and is therefore not Hamiltonian. This establishes (I): C_{i^*} does not contain a cycle. From the definition of C_{i^*} and the choice of i^*, every vertex $a_{i^*,j}$ for $j \in [n_A]$ is incident on exactly two edges in C_{i^*}, establishing (II). Since the Hamiltonian cycle C induces a two-regular graph, condition (III) is also satisfied. To see that b^*_1 and $b^*_{n_B}$ are incident on at most one edge of C, observe that by definition of HAMILTONIAN $s - t$ PATH IN BIPARTITE GRAPHS the corresponding vertices have degree one in graph G_{i^*}: as C_{i^*} is an edge subset of $E(G_{i^*})$, there is at most one edge incident on each of b^*_1 and $b^*_{n_B}$, establishing (IV). As all conditions of Proposition 8.3 are satisfied, we conclude that the bipartite input graph G_{i^*} has a Hamiltonian path between $b_{i^*,1}$ and b_{i^*,n_B}.

($\Leftarrow$) Assume that there is an index $i^* \in [r]$ such that G_{i^*} has a Hamiltonian path C_{i^*} starting in $b_{i^*,1}$ and ending in b_{i^*,n_B}. Construct a Hamiltonian cycle C in G^* as follows.

- For $i \neq i^*$ add the edges $\{w_i, x_i\}$, $\{x_i, a^-_{i,1}\}$, $\{a^+_{i,n_A}, y_i\}$, and $\{a^+_{i,j}, a^-_{i,j+1}\}$ for $j \in [n_A - 1]$ to C.
- For $i \neq i^*$ add the edges on the a-traversals of all dominos $O_{i,j}$ with $j \in [n_A]$ to C.
- Add the edges $\{w_{i^*}, x_{i^*}\}$, $\{x_{i^*}, z^-\}$, $\{z^-, z\}$, $\{z, z^+\}$, and $\{z^+, y_{i^*}\}$ to C. Also add all edges on $\hat{a}$-traversals of the dominos $O_{i^*,j}$ for $j \in [n_A]$ to C.
- Add all edges $\{y_i, x_{i+1}\}$ for $i \in [r - 1]$ to C.
- Add the edges $\{b^*_{n_B}, w_1\}$ and $\{y_r, b^*_1\}$ to C.
- For $j \in [n_A]$, if the edges of G_{i^*} incident on $a_{i^*,j}$ in the Hamiltonian path C_{i^*} are $\{a_{i^*,j}, b_{i^*,f}\}$ and $\{a_{i^*,j}, b_{i^*,f'}\}$ then add the edges $\{\hat{a}^-_{i^*,j}, b^*_f\}$ and $\{\hat{a}^+_{i^*,j}, b^*_{f'}\}$ to C. The relative order of b^*_f and $b^*_{f'}$ is irrelevant since the domino can be traversed in either direction.

Using the construction of G^* is it straightforward to verify that C is a Hamiltonian cycle in G^*, which proves the reverse direction of the equivalence. $\diamondsuit$

As the claim shows that the cross-composition is correct, Theorem 8.6 now follows from Corollary 3.2 and Proposition 8.2. $\square$

8.5 Paths with Forbidden Pairs

In this section we study multiple parameterizations of path problems involving forbidden pairs. The first version we consider is defined as follows.

> $s - t$ Path with Forbidden Pairs parameterized by a vertex cover of G
> **Input:** A graph G, distinct vertices $s, t \in V(G)$, a set $H \subseteq \binom{V(G)}{2}$ of forbidden pairs, and a vertex cover X of G.
> **Parameter:** The size $k := |X|$ of the vertex cover.
> **Question:** Is there an $s - t$ path in G that contains at most one vertex of each pair $\{u, v\} \in H$?

Although the problem is easily seen to lie in XP — guess which vertices from the vertex cover are used in the path, and for each such vertex guess at most two edges incident on it that are used — we prove that it is not fixed-parameter tractable unless FPT $= W[1]$.

Theorem 8.7. *$s - t$ Path with Forbidden Pairs parameterized by a vertex cover of G is hard for $W[1]$.*

Proof. We give a parameterized reduction (Definition 2.5) from the $W[1]$-hard Multicolored Clique problem [99] parameterized by the desired size of the clique. Let (G, ℓ, ϕ) be an instance of Multicolored Clique, where $\phi \colon V(G) \to [\ell]$ assigns each vertex of G a color from 1 to ℓ and the task is to decide whether G

contains a clique with one vertex of each of the ℓ colors. The parameter equals the desired size of the clique, so $k := \ell$.

We construct a graph G' as follows. We first make an independent set on the vertices $V(G)$, and add a forbidden pair for each pair of vertices from $V(G)$ that are either not adjacent in G or that have the same color according to ϕ. Then we add $\ell + 1$ vertices $v_0, v_1, \ldots, v_\ell$ that will form the vertex cover. We connect vertex v_0 to all vertices of the independent set that have color one according to ϕ. For all $i \in [\ell - 1]$ we connect v_i to all vertices that have colors i or $i + 1$. Finally, we connect v_ℓ to all vertices of color ℓ. We set $s := v_0$ and $t := v_\ell$, and return the instance $(G', v_0, v_\ell, X' := \{v_0, \ldots, v_\ell\})$. We note that this can be easily performed in polynomial time, and that the parameter value to the new instance is $k' := |X'| = \ell + 1 = k + 1$, and is trivially bounded by a function of k.

It is easy to see that every $s - t$ path has $2\ell + 1$ vertices, and uses all vertices $v_0, \ldots, v_\ell$ as well as one vertex of each color. The latter vertices must also be adjacent in G, and hence they correspond to a multicolored clique of size ℓ. Similarly, given such a multicolored clique it is straightforward to find an $s - t$ path of length $2\ell + 1$ in G' that respects the forbidden pairs. Note that the colors are not part of the produced instance, but the forbidden pairs are used to encode this property.

Thus $W[1]$-hardness of $s - t$ Path with Forbidden Pairs parameterized by a vertex cover of G follows. □

Let us now consider some variations of the problem. The problems Short $s - t$ Path with Forbidden Pairs and Long $s - t$ Path with Forbidden Pairs are defined similarly as $s - t$ Path with Forbidden Pairs, with the difference that there is an extra integer ℓ in the input and we are asking for an $s - t$ path containing at most or at least ℓ vertices. In Long Path with Forbidden Pairs we omit the inputs s and t, and are looking for *any* sufficiently long path, regardless of its endpoints. The related problem Short Path with Forbidden Pairs is not interesting, since its solution always consists of a path containing a single vertex.

It can easily be verified that asking for a path on at least $2\ell + 1$ vertices, regardless of its endpoints, in the construction for the proof of Theorem 8.7 leads also to a path whose vertices from the independent set made of $V(G)$ correspond to a multicolored clique in G. Also, since testing the existence of a long or short $s - t$ path is as least as hard as testing the existence of any $s - t$ path, we obtain the following results as a corollary.

Corollary 8.5. *The problems* Short $s - t$ Path with Forbidden Pairs*,* Long $s - t$ Path with Forbidden Pairs*, and* Long Path with Forbidden Pairs *are hard for $W[1]$ when parameterized by a vertex cover of G.*

Clearly, the hardness of the path problems with forbidden pairs stems from the extra restrictions implied by the forbidden pairs H, which are not taken into account when considering structural parameters of G. In the following we consider the effect of parameterizing by the structure of the graph $G \cup H$, which is (in general) defined by $G \cup H := (V(G) \cup V(H), E(G) \cup E(H))$. In our case, using that $V(H) \subseteq V(G)$, this can be simplified to $(V(G), E(G) \cup E(H))$, adding an edge to G for each forbidden pair in H. As it is obvious how to alter the definition of $s-t$ Path with Forbidden Pairs parameterized by a vertex cover of G to obtain the definitions of the other forbidden path problems under structural parameterizations, we do not explicitly define each variant.

Using the optimization version of Courcelle's theorem applied to *structures* of bounded treewidth [9, 19, 44, 65, 67] (instead of graphs, as is more common) it is not difficult to establish fixed-parameter tractability of all mentioned forms of the forbidden path problems, parameterized by the treewidth of $G \cup H$. Take a structure on the base set $V(G) \cup E(G) \cup H$ that encodes an instance through unary predicates (that say whether an object is the vertex s, the vertex t, a different vertex of G, an edge of G, or a forbidden pair in H) and through binary predicates (that give the incidence between vertices and edges or pairs). The treewidth of the structure equals the treewidth of $G \cup H$. For such a representation it is well known how to devise an MSOL formula that asks whether there exists a set of edges Y that forms a path between s and t such that no two vertices incident on an edge in Y form a forbidden pair. Using standard extensions of MSOL we may also maximize or minimize the size of a set of edges that forms an $s-t$ path respecting forbidden pairs, obtaining the following result.

Proposition 8.5. *The problems* Short $s-t$ Path with Forbidden Pairs, Long $s-t$ Path with Forbidden Pairs, *and* Long Path with Forbidden Pairs *are fixed-parameter tractable parameterized by the treewidth of* $G \cup H$.

For the case of Short $s-t$ Path with Forbidden Pairs the structure of G is actually not so important for the complexity of the problem: it is sufficient to parameterize by a vertex cover of the graph on the edge set H to obtain fixed-parameter tractability, as demonstrated by the following theorem.

Theorem 8.8. Short $s-t$ Path with Forbidden Pairs parameterized by a vertex cover of H *is fixed-parameter tractable.*

Proof. Given a graph G with endpoints s, t and forbidden pairs H such that X is a vertex cover of H, we describe how to find the shortest $s-t$ path that avoids forbidden pairs. For all subsets $X' \subseteq X$ that do not contain a forbidden pair, we compute the shortest $s-t$ path that intersects X only in X' as follows. Let Y be the vertices that form a forbidden pair with a member of X': then the desired path is a shortest $s-t$ path in the graph $G-(X \setminus X')-Y$. Observe that no path in this restricted graph contains a forbidden pair by the structure of X and choice of Y. Since a shortest path in this graph can be found in linear time using breadth-first search, we solve the problem in $\mathcal{O}^*(2^{|X|})$ time by returning the shortest $s-t$ path found over all choices of $X' \subseteq X$. □

It is easy to see that the result of Theorem 8.8, namely membership in FPT parameterized by a vertex cover of H, cannot be extended to Long $s-t$ Path with Forbidden Pairs since the latter problem is already NP-complete when there are no forbidden pairs. We mention without proof that $s-t$ Path with Forbidden Pairs is NP-complete when the graph induced by H is a matching, which shows that we cannot improve the parameterization by a vertex cover of H to the treewidth of H.

As the final topic of this section we consider the kernelization complexity of forbidden path problems. We give a superpolynomial lower bound on the kernel size when parameterizing by a vertex cover of $G \cup H$.

Theorem 8.9. $s-t$ Path with Forbidden Pairs parameterized by a vertex cover of $G \cup H$ *does not admit a polynomial kernel unless NP ⊆ coNP/poly.*

Proof. We consider the classical problem $s-t$ Path with Forbidden Pairs where we simply drop the set X from the problem description. We show that $s-t$ Path with Forbidden Pairs cross-composes into $s-t$ Path with Forbidden Pairs parameterized by a vertex cover of $G \cup H$. This suffices to prove the claim by Corollary 3.2, since the construction of Theorem 8.7 shows that $s-t$ Path with Forbidden Pairs is NP-complete.

Let $\mathcal{R}$ be a polynomial equivalence relation under which two well-formed instances (G_1, s_1, t_1, H_1) and (G_2, s_2, t_2, H_2) are equivalent if $|V(G_1)| = |V(G_2)|$. We show how to compose a sequence of inputs $(G_1, s_1, t_1, H_1), \ldots, (G_r, s_r, t_r, H_r)$ on n vertices each. Assume the vertices of $V(G_i)$ are labeled $v_1, \ldots, v_n$ such that $v_1 = s_i$ and $v_n = t_i$. We build a graph G^* with a vertex cover X^*, and a set of forbidden pairs $H^* \subseteq \binom{V(G^*)}{2}$ such that all forbidden pairs have at least one member in X^*.

1. For $j \in [n]$ add a vertex v_j^* to G^*.
2. Add a special starting vertex w to G^*.
3. For $i \in [r]$ add a vertex z_i to G^* and make it adjacent to w and v_1^*.
4. For $1 \leq j < h \leq n$, add a *ladder* gadget $L_{j,h}$ with n *spokes* to G^*:
   - Add vertices $t_{j,h}^i$ and $f_{j,h}^i$ to G^* for $i \in [n]$. Each pair forms a spoke of the ladder.
   - Add the set of edges
   $$\left\{\{t_{j,h}^i, t_{j,h}^{i+1}\}, \{t_{j,h}^i, f_{j,h}^{i+1}\}, \{f_{j,h}^i, f_{j,h}^{i+1}\}, \{f_{j,h}^i, t_{j,h}^{i+1}\} \;\middle|\; i \in [n-1]\right\}$$
   to form the inside of the ladder.
   - Make v_j^* adjacent to $t_{j,h}^1$ and $f_{j,h}^1$, and make v_h^* adjacent to $t_{j,h}^n$ and $f_{j,h}^n$.
5. Repeat the following for each $i \in [r]$:
   - For $1 \leq j < h \leq n$, if $\{v_j, v_h\} \notin E(G_i)$ then add $\{\{z_i, x\} \mid x \in L_{j,h}\}$ to the set of forbidden pairs. Here $L_{j,h}$ denotes the set of $2n$ vertices on the ladder for $\{j, h\}$.
   - For $1 \leq j < h \leq n$, if $\{v_j, v_h\} \in E(G_i)$ then do as follows for all $q \in [n]$:
     - If $\{v_j, v_q\} \in H_i$ or $\{v_h, v_q\} \in H_i$ then add $\{z_i, f_{j,h}^q\}$ to H^*.

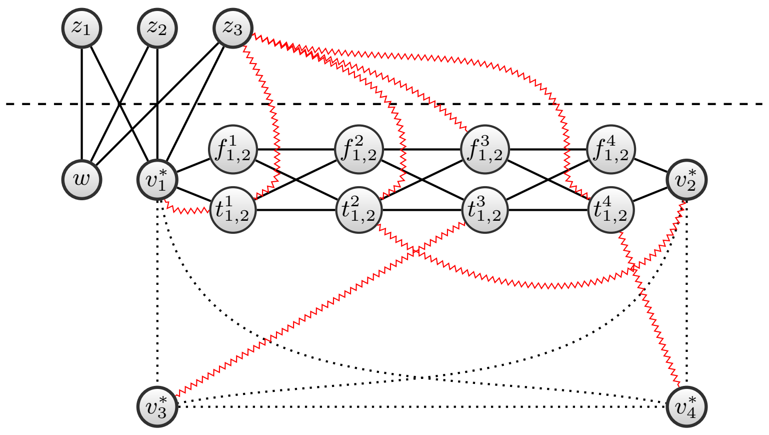


Figure 8.5: An example of the lower-bound construction of Theorem 8.9, cross-composing three instances with $n = 4$ into one. For clarity, only the vertices of the ladder $L_{1,2}$ are drawn; the other ladders are visualized by dotted paths. The forbidden pairs are drawn as zigzagged lines. Forbidden pairs with one element in $\{z_1, z_2\}$ are omitted for clarity. For this example, the third input (G_3, s_3, t_3, H_3) has a forbidden pair $\{v_1, v_3\} \in H_3$ causing the forbidden pair $\{z_3, f_{1,2}^3\} \in H^*$. The vertices below the horizontal dashed line form the vertex cover X*.

  – Else, if $\{v_j, v_q\} \notin H_i$ and $\{v_h, v_q\} \notin H_i$, then add $\{z_i, t_{j,h}^q\}$.

- For $1 \leq j < h \leq n$, for $q \in [n]$, add $\{t_{j,h}^q, v_q^*\}$ to H^*.

6. This concludes the description of G^* and H^*, which is illustrated in Fig. 8.5. Define $X^* := V(G^*) \setminus \{z_i \mid i \in [r]\}$.

It is easy to see that this construction can be carried out in polynomial time. The set X^* is a vertex cover for $G^* \cup H^*$ since all edges and forbidden pairs of G^* have at least one element in X^*. The size of X^* is $1 + n + 2n\binom{n}{2}$, which is polynomial in n. Therefore the parameter $k := |X^*|$ of the constructed instance is polynomial in the size of the largest input instance. We set $s^* := w$ and $t^* := v_n^*$. Let us establish some properties of the constructed instance.

Claim. *Consider an $s^* - t^*$ path P in G^* that respects the forbidden pairs H^*, and orient the path such that it starts at s^*. The following must hold.*

(i) There is exactly one index $i^* \in [r]$ such that $z_{i^*} \in P$, and the path P has the form $(s^* = w, z_{i^*}, v_1^*, \ldots, v_n^* = t^*)$.
(ii) If P contains v_j^* and v_h^* then $\{v_j, v_h\} \notin H_{i^*}$, where i^* is as defined above.
(iii) If v_j^* and v_h^* are vertices on P, and no vertices of the form v_f^* are visited by P between visiting v_j^* and v_h^*, then $\{v_j, v_h\} \in E(G_{i^*})$, where i^* is as defined above.

Proof. (i) Since all vertices z_i for $i \in [r]$ have the same neighborhood consisting of w and v_1^*, if a path contains at least two such vertices then at least one of them is an endpoint of the path, which is not possible since P is an $s^* - t^*$ path. Any $s^* - t^*$ path must use at least one vertex z_{i^*} since all neighbors of $s^* = w$ have this form. The successor of z_{i^*} is v_1^*, since z_{i^*} has only two neighbors in G^*.

(ii) Suppose P contains v_j^* and v_h^*, and assume for a contradiction that $\{v_j, v_h\} \in H_{i^*}$. Assume without loss of generality that $j < h$ and therefore $v_j^* \neq v_n^*$. Since v_j^* is not an endpoint of P, vertex v_j^* must have two neighbors on P. By construction of G^* it easily follows that at least one of these neighbors must lie on a ladder. Assume that v_j^* has a neighbor on a ladder $L_{j,p}$; the case that the neighbor lies on a ladder $L_{p,j}$ is symmetric. The forbidden pairs involving z_{i^*} and the vertices of the ladder $L_{j,p}$ ensure that for each spoke of the ladder on vertices $\{t_{j,p}^q, f_{j,p}^q\}$ there is a forbidden pair containing z_{i^*} and exactly one vertex of the pair. Since P contains z_{i^*} it must avoid exactly one vertex of each spoke of the ladder, which implies by construction of G^* that all vertices on the ladder have only one valid option along which to continue the path, implying that P must traverse the entire ladder to the other endpoint v_p^*. Since $\{v_j, v_h\} \in H_{i^*}$ the definition of G^* shows that $\{z_{i^*}, f_{j,p}^h\} \in H^*$, and therefore path P must visit the other vertex $t_{j,p}^h$ of the spoke when traversing the ladder. But by the last step of the construction we know that $\{t_{j,p}^h, v_h^*\} \in H^*$ is a forbidden pair. So P contains both vertices of a forbidden pair; a contradiction.

(iii) Suppose that P successively visits the v^*-vertices v_j^* and v_h^* with $j < h$. By construction of G^* it is easy to see that P must traverse the vertices of the ladder $L_{j,h}$. Since P contains z_{i^*}, and z_{i^*} forms a forbidden pair with every vertex on the ladder $L_{j,h}$ if $\{v_j, v_h\} \notin E(G_{i^*})$, it follows that if P can traverse the ladder $L_{j,h}$ without hitting a forbidden pair then the corresponding edge must be contained in G_{i^*}. ♢

With the structural properties of the constructed instance, as given by the claim, we can prove the correctness of the cross-composition.

Claim. *There is an $s^* - t^*$ path in G^* that avoids the forbidden pairs in H^* if and only if there is an index $i^* \in [r]$ such that G_{i^*} has a $v_1 - v_n$ path that avoids the forbidden pairs in H_i.*

Proof. ($\Rightarrow$) Let P be an $s^* - t^*$ path in G^* that avoids the forbidden pairs. By (i) path P has the form $(w, z_{i^*}, v_1^*, \ldots, v_n^*)$. Orient P from s^* to t^*, and consider the vertices $(v_{i_1}^*, \ldots, v_{i_m}^*)$ in the set $\{v_i^* \mid i \in [n]\}$ that are successively visited on this path. By (ii) it follows that there are no indices i_j, i_f such that $\{v_{i_j}, v_{i_f}\} \in H_{i^*}$, and by (iii) the set $E(G_{i^*})$ contains all edges $\{v_{i_j}, v_{i_{j+1}}\}$ for $j \in [m-1]$. Hence the vertices $(v_{i_1}, \ldots, v_{i_m})$ form a path in G_{i^*} that does not contain a forbidden pair. By the form of P described in (i) it follows that $v_{i_1}^* = v_1^*$, and since P ends with v_n^* it follows that $v_{i_m}^* = v_n^*$. Hence this suggested path is a $v_1 - v_n$ path in G_{i^*} that avoids forbidden pairs, which proves that the answer to instance number i^* is YES.

Table 8.1: Complexity overview of path problems with forbidden pairs.

Problem to be solved	Parameterization			
	VC(G)	VC(H)	TW(H)	TW$(G \cup H)$ VC$(G \cup H)$
$s - t$ PATH F.P.	$W[1]$-hard	FPT, no poly	NP-C$_{k=1}$	FPT, no poly
SHORT $s - t$ PATH F.P.	$W[1]$-hard	FPT, no poly	NP-C$_{k=1}$	FPT, no poly
LONG $s - t$ PATH F.P.	$W[1]$-hard	NP-C$_{k=0}$	NP-C$_{k=0}$	FPT, no poly
LONG PATH F.P.	$W[1]$-hard	NP-C$_{k=0}$	NP-C$_{k=0}$	FPT, no poly

Each column represents a different parameterization; VC(G) denotes the minimum vertex cover size of G, and TW(G) denotes its treewidth. F.P. abbreviates "WITH FORBIDDEN PAIRS". The classification "No poly" means "no polynomial kernel unless NP $\subseteq$ coNP/poly", and "NP-C$_{k=1}$" means "NP-complete for fixed parameter value 1". SHORT PATH F.P. is trivially in P.

($\Leftarrow$) Suppose that G_{i^*} contains a path $(v_{i_1}, \ldots, v_{i_m})$ with $i_1 = 1$ and $i_m = n$ that does not contain a forbidden pair. Now construct a path in G^* that starts at $s^* = w$ and successively visits z_{i^*} and v^*_1. It then traverses the ladder to $v^*_{i_2}$, and by construction of G^* there is exactly one vertex on each spoke of the ladder that is not in a forbidden pair with z_{i^*}; the path uses these vertices. We can continue this, alternatingly traversing a vertex $v^*_{i_j}$ and a ladder, until reaching $v^*_{i_m} = t^*$. It is not difficult to verify that the constructed path does not contain any forbidden pairs of H^*, which concludes the proof. $\diamondsuit$

As the claim shows that the cross-composition is correct, Theorem 8.9 now follows from Corollary 3.2. $\square$

It is easy to modify the construction of Theorem 8.9 to prove a kernel lower bound for LONG PATH WITH FORBIDDEN PAIRS PARAMETERIZED BY A VERTEX COVER OF $G \cup H$. The definition of the latter problem does not allow us to specify the endpoints s and t of the path, but by creating two suitably long paths and connecting these only to s and t we can ensure that a sufficiently long path in the resulting graph is actually an $s - t$ path. Also, similarly as before the hardness results for the existence of $s - t$ paths immediately carry over to long and short $s - t$ paths. We obtain the following results as a corollary.

Corollary 8.6. *The problems* SHORT $s - t$ PATH WITH FORBIDDEN PAIRS, LONG $s - t$ PATH WITH FORBIDDEN PAIRS, *and* LONG PATH WITH FORBIDDEN PAIRS *do not admit polynomial kernels parameterized by a vertex cover of $G \cup H$ unless NP $\subseteq$ coNP/poly.*

Table 8.1 contains a summary of the results of this section.

8.6 Concluding Remarks

In this chapter we have shown that for sufficiently restricting structural parameterizations, many path and cycle problems admit polynomial kernels even though their natural parameterizations do not. The marking technique using bipartite matching yields quadratic-vertex kernels for several problems parameterized by the size of a vertex cover. We introduced a win/win approach using a binary encoding trick, which gives polynomial kernels for problems parameterized by the max leaf number or by a modulator to a cluster graph. On the negative side, we also exhibited that the smaller structural parameter "distance to outerplanarity" provably does not lead to a polynomial kernel for HAMILTONIAN CYCLE unless NP $\subseteq$ coNP/poly.

The complexity overviews given in Fig. 8.1 and Fig. 8.2 give directions for further research; we highlight those that we consider the most interesting.

First of all, we expect that the problems considered in this chapter admit polynomial kernels parameterized by the deletion distance to a cograph. It would be interesting to see whether this is true. Can the use of a binary encoding scheme and Karp reductions be avoided? A direct kernelization based on step-by-step reduction rules seems to be more useful for practical applications.

When it comes to LONG PATH, there are two interesting challenges. Is the problem contained in XP, or even FPT, parameterized by the deletion distance to an interval graph? Since researchers only recently managed to prove that LONG PATH is polynomial-time solvable on interval graphs [140], this may be a difficult question. In terms of kernelization complexity, the existence of a polynomial kernel parameterized by feedback vertex set is wide open. We know that there is a polynomial kernel by deletion distance to a graph of treewidth zero (Theorem 8.2), but not by distance to a graph of treewidth two (Theorem 8.6). The parameterization by feedback vertex number — distance to treewidth one — fits exactly in the gap between these classifications. When it comes to proving kernelization upper bounds, it might be easier to start with the HAMILTONIAN CYCLE problem, or the parameterization by deletion distance to a path.

Finally, the kernelization complexity of compound parameterizations remains largely unexplored. For example, how does the LONG CYCLE problem behave when parameterized by the solution size plus the vertex-deletion distance to an outerplanar graph? It follows from the work of Bodlaender, Thomassé, and Yeo [43] that DISJOINT PATHS and DISJOINT CYCLES do not admit polynomial kernels parameterized by the target value ℓ plus the deletion distance to a path. We hope that a search for polynomial kernels of structural parameterizations leads to reduction rules for these problems that are useful in practice.

Part IV

Conclusion

Todo tiene su final.

—Héctor Lavoe, 1973

9 Conclusion

While Parts II and III were devoted to technical transformations, reduction rules, and lower bound constructions, we conclude by focusing on the story that these results tell. We also take the opportunity to reflect on recent progress in the area of problem kernelization, aiming to extract patterns that are emerging in problem behavior under varying parameterizations.

As we ended each chapter with open problems that are specific to its subject matter, we restrict ourselves here to abstract challenges for future research and concrete open problems not directly relating to the results of this thesis.

Some of the material in Section 9.5 originates from “Towards Fully Multivariate Algorithmics: Parameter Ecology and the Deconstruction of Computational Complexity” by Michael R. Fellows, Bart M. P. Jansen, and Frances A. Rosamond (European Journal of Combinatorics [101].)

9.1 Main Contributions

Motivated by Karp's challenge to explain the "unreasonable" success of heuristic algorithms, which are often based on preprocessing and simplification rules, our work in this thesis began in Chapter 2 by formalizing efficient preprocessing as kernelization. We described the parameter ecology program, aimed at finding the limits of tractability with respect to a hierarchy of parameters. Its relevance for the study of preprocessing was motivated by the hypothesis that a thorough understanding of kernelization complexity under varying parameterizations guides us towards effective reduction schemes, and helps us to explain the empirically observed success of preprocessing in existing scenarios.

To facilitate the analysis of kernelization complexity, we developed some technical tools in Part II. Adding "syntactic sugar" to the kernelization lower bound framework of Bodlaender et al. [24] resulted in the concept of cross-composition, presented in Chapter 3. We showed that the resulting techniques are particularly convenient to prove kernelization lower bounds for structural problem parameterizations. Chapter 4, born out of the desire to obtain kernelization meta-theorems for problems parameterized by vertex cover, gave a tool that establishes kernelization upper bounds for wide classes of problems under this parameterization.

The four case studies in Part III formed the heart of the thesis. We devoted Chapter 5 to the study of Vertex Cover, a problem for which there have been dramatic improvements over the course of its investigation. We surveyed the current state of the art in kernelization for this problem, and presented a cubic-vertex kernel for the parameterization by a feedback vertex set. Although subsequent improvements have resulted in randomized polynomial kernels for smaller parameters such as the deletion distance to a König graph, to this date there is no deterministic kernel known for smaller parameters than the feedback vertex number. The resulting complexity picture for Vertex Cover, visualized in Fig. 5.1, shows that we currently know a lot of parameterizations of the problem that admit polynomial kernels.

Chapter 6 focused on the decision problems related to treewidth computations. We showed that combinations of existing heuristic preprocessing rules yield a cubic-vertex kernel for Treewidth [vc]. This fact alone can already be seen as an explanation for the success of the heuristics. Utilizing more sophisticated rules from prior work, and adding reduction rules relating to clique-seeing paths specifically tailored to instances with large induced forests, we derived a polynomial kernel for the smaller parameterization by feedback vertex set. Our quest for a polynomial kernel for the latter parameterization also led to the discovery of a new *Almost Simplicial Vertex Rule* that can get rid of all such vertices in an instance, strengthening earlier rules that were only applicable if favorable bounds on the target treewidth were known. We presented a kernelization lower bound for the parameter that measures the deletion distance to a clique. It implies the infeasibility of obtaining polynomial kernels for the parameterizations that measure the vertex-deletion distance to graph classes such as cographs and chordal graphs, in which Treewidth is nevertheless polynomial-time solvable.

The classic problem of properly coloring the vertices of a graph was analyzed from various angles in Chapter 7. We proved that the potential for effective preprocessing strongly depends on the number of colors allowed in the coloring. We performed an investigation of coloring problems on graphs that "almost" belong to well-understood graph classes, modeled after Cai's [48] study of the parameterized complexity of the resulting parameterized problems. We identified a characteristic of the graph class that governs the kernelization complexity of q-Coloring under such parameterizations, which was established by giving lower and upper bounds with respect to this characteristic.

Finally we attacked path and cycle problems in Chapter 8. We introduced two widely applicable reduction techniques. The first, based on a property of matchings in bipartite graphs, was used to give quadratic-vertex kernels for path and cycle problems, improving on the cubic-vertex bounds resulting from the general theorems developed in Part II. The second technique, revolving around a win/win for weighted problems encoded in binary, yielded polynomial kernels for parameterizations by the max leaf number and by the vertex-deletion distance to a cluster graph. In the same chapter we also obtained kernel lower bounds for a structural parameterization of Hamiltonian Cycle, and studied the behavior of path problems with forbidden pairs.

9.2 Discussion

Our contributions give theoretical bounds on the effectiveness of efficient preprocessing. Figures 5.1, 6.1, 7.1, 8.1, and 8.2 stand testimony to the fact that the results of the four case studies in the core of this thesis paint fairly accurate pictures of the kernelization complexity of several fundamental graph problems.

Recall that the narrative started with the desire to explain observed successes that defy our pessimistic worst-case bounds. To leverage the theoretical findings into an explanation of the effectiveness of preprocessing, we need to take our theorems back to the realm of real-world data. The properties of practical instances should be analyzed to see whether the parameters corresponding to currently known polynomial kernels are indeed small on those inputs, and whether the reduction rules resulting from our kernels are as effective as, or functioning similar to, the existing heuristics. As already indicated in Section 2.5, a theoretical study alone cannot explain phenomena on practical data sets; but our work succeeded in providing the theoretical underpinnings of a potential explanation.

In the case of Treewidth, the fact that existing reduction rules combine into a polynomial kernel for the parameterization by vertex cover already gives a first theoretical justification for the success of those rules. But even for this problem we need to go back to the data to see whether this is really the right explanation. Are the instances where the rules are effective large with respect to their vertex cover number? If this is the case, then the obtained kernel shows why the reduction rules are effective. If not, then there are additional effects of the reduction rules that are not captured in the parameterization by vertex cover. The latter outcome would prompt us to perform further theoretical analysis, aimed at finding the parameterization that captures the hidden structure in the instances that is implicitly being exploited.

Although the introduction in Chapter 1 focused on the desire to explain the power of preprocessing, the subsequent study of kernelization complexity that we have performed resulted in more than just explanations. Our polynomial kernelization algorithms contain novel and efficient reduction strategies that can be applied in real-world situations.

The reduction scheme based on vertex marking, presented in Chapter 4, is simple to implement and experiment with. The forest-shrinking kernel for Vertex Cover [fvs] contains several new reduction rules, and can be computed by an algorithm with a low-degree polynomial as its running time (Lemma 5.10). Our reduction rules for Treewidth handle situations that were irreducible by existing approaches. The kernels for q-Coloring given in Chapter 7, and the kernels of Chapter 8 for parameterization by vertex cover, use easy to implement marking- and matching arguments to shrink the instance. Thus the work in this thesis has uncovered an array of new data reduction techniques with good claims to practicality.

In some cases, however, the polynomial kernels obtained from our theorems seem to be mostly of theoretical interest. The kernels of Chapter 8 based on Karp reductions to obtain a kernelized instance out of a small-bitsize encoding of a different NP-problem, are unlikely to be practical in their current form. But the knowledge that these parameterizations do admit polynomial kernels, paves the way for more viable reduction schemes. It is hard to estimate the extent to which the randomized matroid-based kernels for Vertex Cover, described in Section 5.1, are practical. These new theoretical advances carry interesting implementation challenges, waiting to be met.

9.3 Trends in Kernelization Complexity

In the introduction to Chapter 8, we already remarked that connectivity requirements in a problem seem to form a barrier to polynomial kernelizability when it comes to the parameterizations by solution size. As part of the reflection process of this conclusion, we consider this trend in more detail, while also extracting other trends from recent advances in the area.

9.3.1 Natural Parameterizations of Connectivity Problems

As the simplest reduction rules are local in nature, the presence of global connectivity constraints often poses problems for kernelization. Although Vertex Cover [sol] has a very simple polynomial kernel based on the local “high-degree” rule [46], the situation is different for its closely related cousin Connected Vertex Cover [sol]. While trying to lift the efficient linear-vertex kernels for Vertex Cover [sol] to Connected Vertex Cover [sol], Guo et al. [124] already noted the difficulty of finding a small kernel for the latter problem. As their study was conducted before the development of the lower bound framework, they were not able to prove that no such kernel exists.

When Bodlaender et al. [24] later introduced their framework, Long Path [sol] was the first connectivity-type problem for which the nonexistence of a polynomial kernel (subject to $\mathrm{NP} \not\subseteq \mathrm{coNP/poly}$) was proven. Using transformations, Bodlaender, Thomassé, and Yeo [43], proved superpolynomial kernel lower bounds for the connectivity problems Disjoint Paths [sol] and Disjoint Cycles [sol]. Sophisticated constructions by Dom, Lokshtanov, and Saurabh [83], based on colors and ID's, later proved the same lower bound for Connected Vertex Cover [sol]. They also proved that for the connectivity problem Steiner Tree, not even the compound parameterization by solution size plus the number of terminals admits a polynomial kernel. Superpolynomial kernel lower bounds were subsequently proven for Connected Feedback Vertex Set [sol], Connected Odd Cycle Transversal [sol], and Connected Dominating Set [sol], in d-degenerate[1] graphs [75, 181].

Observing the trend in this series of results, it seems that the natural parameterizations of problems whose solutions demand certain forms of connectivity tend not to admit polynomial kernels. A very recent result of Wahlström [208] shows that the problem of finding a cycle through a set of k specified vertices might form an exception to this rule. Although he does not obtain a polynomial kernel for the problem, he shows that an instance can be compressed into a $k^{\mathcal{O}(1)}$-bitlength encoding of a different problem (that presumably lies outside NP). It would be interesting to determine more precisely how connectivity requirements stand in the way of polynomial kernels.

9.3.2 Parameterizations by Width Measures

The complexity of graph problems parameterized by width measures such as treewidth, branchwidth, and cliquewidth, has been studied for decades. Through evolved forms of Courcelle's theorem and its variants [9, 19, 44, 65–67], we know that many problems admit linear-time solutions for fixed values of such parameters. Such parameterizations tend not to admit polynomial kernels, however. In their seminal work on kernel lower bounds, Bodlaender et al. [24] proved that subject to various complexity assumptions, problems like 3-Coloring, Independent Set, Clique, and Dominating Set do not admit polynomial kernels when parameterized by the treewidth of the graph.

If the answer to the problem is the logical or (respectively and) of the answers to the connected components (as for Clique and 3-Coloring, respectively) then

[1]In the case of Connected Dominating Set, the restriction is necessary to make the problem fixed-parameter tractable at all.

one easily obtains a composition algorithm for the same choice of logical operator. In the case of optimization problems like Independent Set and Dominating Set, where the solution sizes inside different connected components have to be summed, the situation is more involved. Nevertheless, the use of an *improvement* or *refinement* version for the problem (that asks whether a given solution can be improved) often allows composition algorithms to be built. While this originally required the corresponding improvement version to be proven NP-complete, a sometimes arduous task, the use of co-nondeterminism [165] allowed by the latest versions of the lower bound frameworks [80], can replace the need for such NP-completeness proofs in some cases.

The main message is that problems parameterized by width measures tend not to admit polynomial kernels; but proving this fact may involve some technical machinery. Moving beyond traditional width measures, more generally it seems to hold that parameterizations by the maximum value that a parameter attains within one connected component, do not admit polynomial kernels.[2]

9.3.3 Parameterizations by Vertex Cover

In this thesis we have seen many examples of problems that are not effectively reducible with respect to the solution size, but do allow polynomial kernels parameterized by vertex cover. However, there seem to be several types of problems that do not admit polynomial kernels even for the latter parameterization. As the vertex cover number bounds the treewidth, the corresponding parameterizations are often easily seen to be fixed-parameter tractable by applying variants of Courcelle's theorem. We identify three types of problems whose parameterization by vertex cover does not admit polynomial kernels.

Induced Subgraph Tests Parameterized by Vertex Cover

The first type is formed by induced subgraph testing problems. Conditional superpolynomial kernel lower bounds have been obtained for finding induced paths, matchings, or bicliques, parameterized by vertex cover [109], while Clique was one of the first problems known not to admit a polynomial kernel parameterized by vertex cover [29].

Let us give some intuition as to why the general reduction schemes of Chapter 4 fail for the mentioned problems. The reduction routine in Section 4.3 shrinks the part of a graph G outside a vertex cover X by marking a small number of relevant vertices outside of X, and removing the unmarked vertices. The correctness of this approach is based on the fact that in a solution, a vertex may safely be replaced by

[2]There are trivial exceptions: consider for example the parameterization by the value 2^k, where k is the size of the largest connected component. The parameterization allows a polynomial kernel for any problem that can be solved by optimizing, in each connected component, a polynomial-time verifiable property of a vertex subset — simply because the problem is polynomial-time solvable if the number of vertices exceeds the parameter.

one that has the same adjacencies to a relevant constant-size subset of X. When it comes to induced subgraph testing problems such as Long Induced Path, the latter fails: the inducedness constraint in the problem definition means that even the slightest change in the open neighborhood of a vertex in the solution can violate a constraint. Therefore the replacement arguments behind the marking schemes do not go through, and consequently the discriminatory power of such problem settings allows superpolynomial kernel lower bounds to be proven.

Vertex-Domination Problems Parameterized by Vertex Cover

A second class of problems that fail to admit polynomial kernels parameterized by vertex cover, is formed by vertex-domination problems. Dominating Set is of course the canonical example, but the same bad news holds for Connected Dominating Set and its dual problem Max Leaf. All of these lower bounds assume NP $\not\subseteq$ coNP/poly, and follow from the work of Dom et al. [83]. Although there are no published results on other variants of domination problems (such as $[\sigma, \rho]$-domination) parameterized by vertex cover, we expect that they do not admit polynomial kernels either.

Weighted Problems Parameterized by Vertex Cover

Although we have intentionally omitted several results for weighted problems from this thesis (cf. [33, 145, 148]) in the interest of saving space, there are a number of notable results in this area. We feel that they are worth mentioning as they follow a common theme: when parameterizing by the cardinality of a given vertex cover, many weighted problems fail to admit a polynomial kernel. These weighted problems are generally defined by allowing a positive integral weight for each vertex in the input, asking for a solution that satisfies the usual constraints, but has bounded total weight. Assuming NP $\not\subseteq$ coNP/poly the weighted versions of Vertex Cover [145], Feedback Vertex Set [33], Odd Cycle Transversal [148], and Treewidth [35], do not admit polynomial kernels parameterized by the cardinality of a vertex cover.

9.3.4 Parameterizations by Structures that are Weaker than Solutions

An interesting trend is emerging when it comes to parameterizations that measure a smaller deletion distance than what is measured by the solution itself. Recall that Vertex Cover can be interpreted as asking for a given graph G and integer ℓ, whether the treewidth of G can be reduced to zero by at most ℓ vertex deletions. Feedback Vertex Set asks whether the treewidth can be reduced to one (or less) by such a set of deletions, and more generally the Treewidth-η Transversal problem asks whether there is a set of at most ℓ vertices whose deletion results in a graph of treewidth at most η.

In Chapter 5 we gave a polynomial kernel for Vertex Cover [fvs], that is, for Treewidth-0 Transversal parameterized by the size of a 1-transversal. In the first publication on the material, we asked whether this can be pushed further: does Vertex Cover have a polynomial kernel parameterized by a 2-transversal? Are there any integers $i \geq 1$ such that Treewidth-i Transversal parameterized by the size of a $(i+1)$-transversal has a polynomial kernel? These questions were soon resolved in the negative by Cygan et al. [73], under the assumption that NP $\not\subseteq$ coNP/poly. They proved that the choice $i = 0$ is the only case where Treewidth-i Transversal has a polynomial kernel parameterized by a $(i+1)$-transversal, and that Vertex Cover parameterized by a 2-transversal does not have a polynomial kernel.

This range of negative results is nicely complemented by the recent work of Fomin et al. [111], who showed that the natural parameterization of $\mathcal{F}$-Minor-Free Deletion has a polynomial kernel if $\mathcal{F}$ contains a planar graph. Since every graph class of constant treewidth excludes a planar graph as a minor [198], their results imply that for any $i \in \mathbb{N}$ the problem Treewidth-i Transversal has a polynomial kernel parameterized by the size of a (suboptimal) i-transversal. Thus we find that Vertex Cover is an exception to the following general trend: for $i \geq 1$ the Treewidth-i Transversal problem parameterized by the size of a j-transversal has a polynomial kernel if and only if $j \geq i$ (assuming NP $\not\subseteq$ coNP/poly).

Besides Vertex Cover, there are several other problems that admit polynomial kernels by feedback vertex set, but not by deletion distance to treewidth two: these include Odd Cycle Transversal [148] and Feedback Vertex Set [207]. Hamiltonian Cycle (Theorem 8.6) forms another candidate, but the existence of a polynomial kernel by feedback vertex set is still open.

9.4 Challenges

We conclude with a discussion of some directions for future research.

Tight Bounds for Structural Parameterizations

The results of this thesis show that many problems admit nontrivial polynomial kernels for structural parameterizations of the instance. Using the techniques of Dell and van Melkebeek [80], later refined by Dell and Marx [79], it has become possible to obtain polynomial lower bounds on kernel sizes. In some cases, for example for Vertex Cover [sol], this makes it possible to obtain tight kernel bounds (up to $k^{o(1)}$ factors). It would be interesting to apply these techniques to structural parameterizations in an attempt to give tight kernel bounds. We expect that the search for tight bounds results in powerful new reduction rules for such parameterizations.

The Structure of Set Systems and Multisets of Integers

The parameter hierarchy of Fig. 2.1 is tailored towards graph problems, using the myriad of ways in which the complexity of a graph can be measured. Some feel that parameterized complexity has an "unhealthy" focus on graph problems, and call for wider research horizons — pursued, for example, in the workshop "Parameterized Complexity: Not About Graphs!" and its follow-up "FPT Algorithms and Algorithm Engineering: Preferably Not About Graphs" [193, 194]. To transfer the ideas advocated in Chapter 2 to such areas, parameters are needed that measure the complexity of instances not involving graphs.

Since many problems are defined on set systems (Hitting Set, Set Cover, Set Packing, and Set Basis, to name a few) a hierarchy of complexity measures for set systems might make for an interesting first target. The interpretation of a set system as a hypergraph allows some of the existing graph measures to be adapted to the new setting. More original parameterizations of set systems can be found in the area of learning theory, such as the Vapnis-Chervonenkis dimension (see [85, Chapter 11] for an FPT-oriented description of this topic). Langerman and Morin [173] defined a notion of combinatorial dimension that also proved to be an interesting parameter for set systems.

Moving beyond set systems, one may consider ways to measure the structure of multisets of integers; these appear in inputs to Subset Sum and related knapsack variants. Besides parameterizations by the number of numbers (taking the cardinality of either the multiset [130, 186], or the underlying simple set [97]), or by the total number of distinct sums of subsets of the input integers [151], not much is known for these types of inputs. Possibilities for interesting parameterizations seem to be abundant. Can the algebraic structure of the numbers (for example, a bound on the number of distinct prime factors) be used to obtain interesting parameterizations?

Experimentation

We already discussed implementation- and experimentation challenges originating from this thesis in Section 9.2. Here we introduce a puzzle regarding one of our favorite problems, Vertex Cover, that should be solved by a combination of experimentation and theoretical analysis. Besides the reduction rules introduced in Chapter 5, there are a multitude of preprocessing strategies ranging from the Nemhauser-Trotter decomposition [53, 187], Buss' high-degree rule [46], to crown rules [2, 59]. Experiments have shown that combinations of these strategies are very effective to reduce real-world inputs [2].

The observed behavior in these experiments is that the target value ℓ is often decreased significantly in the preprocessing phase; exhaustive application of the rules then shrinks the number of vertices to at most twice the new bound ℓ'. Although the theoretical analysis of these kernels explains why the number of vertices in the resulting graph G' is at most $2\ell'$, the parameterization by the solution size cannot account for the observed decrease in ℓ. Can structural insights based on smaller parameterizations be used to explain why these rules are so effective in decreasing the target bound, and therefore the overall size of the resulting output? The kernelization for the parameterization by feedback vertex set proves that the target value ℓ is reduced to a cubic polynomial in the size of a feedback vertex set, but requires additional reduction rules to do so; it does not show why the mentioned rules have similar effects. An experimental study of Vertex Cover that pays attention to structural properties of the input, as it is being processed, might yield interesting insights.

Algorithm Engineering

Moving beyond mere experimentation, the algorithm engineering methodology offers great potential to the field of kernelization and multivariate analysis. Due to the increasing pressure towards science that has clear-cut applications, it is becoming increasingly important to bridge the gap between theory and practice. For example, a recent essay by Stege [206] treats such issues in the context of biology and cognitive science. Algorithm engineering is the way to bring life into our theoretical results, by paying attention to the implementation details that make or break industrial-strength systems. Although there are some projects aimed at practical implementations (such as AGAPE [15]), there are still many potentially interesting FPT algorithms waiting to be implemented in robust problem-solving packages.

The following quote regarding algorithm engineering is interesting in the context of this thesis.

> It might generally be a good idea to implement the preprocessing first, to look at the results, and to invest time in the design of the core routine only afterwards, because only then are the characteristics of the input to the core routine definitely known.
>
> —Karsten Weihe, 1998

Weihe's remarks implicitly assume that it is "obvious" how one should preprocess. Indeed, in his case, the most obvious reduction rules already dealt with the majority of the input. But in other scenarios the preferred choice of preprocessing routine might be not as clear, allowing a choice of different reduction rules corresponding to different parameterizations. The right strategy could be to develop a preprocessing algorithm based on relevant parameters of the data. After one round of implementation and experimentation, one could analyze whether the resulting instances have sufficient remaining structure to be successfully attacked by another round of preprocessing, targeted at different parameters. The problem-solving routine could then be developed based on the properties of the final output distribution.

9.5 Concrete Open Problems

We conclude with a number of open problems in the area of problem kernelization. Some are seasoned veterans of open problem sessions, while others originate from recent investigations and might have accessible solutions.

Open Problem 1. Do the natural parameterizations of the problems

- Chordal Deletion,
- Interval Deletion,
- Directed Feedback Vertex Set, and
- $\mathcal{F}$-Minor-Free Deletion (when $\mathcal{F}$ contains no planar graphs),

admit polynomial kernels?

Open Problem 2. Does the natural parameterization of the Orientable Genus problem admit a polynomial kernel?

Open Problem 3. Does Bandwidth admit a polynomial kernel when parameterized by the size of a vertex cover?

Open Problem 4. Is the Independent Set problem (or equivalently, Vertex Cover) in FPT when parameterized by a (promised) bound on the vertex-deletion distance to a perfect graph, without giving a modulator in the input?

Open Problem 5. Is 3-Coloring in FPT when parameterized by a promised bound on the domination number of the graph?

Open Problem 6. Is Bandwidth in FPT parameterized by the size of a modulator to a linear forest?

Open Problem 7. Is there a structural parameter smaller than feedback vertex number with respect to which Feedback Vertex Set admits a polynomial kernel?

Open Problem 8. Find a deterministic, polynomial kernel for Vertex Cover Above $\mu(G)$ (Section 5.1).

Open Problem 9. Find a parameterization that dominates the parameter $\text{vc}(G) - \text{lp}(G)$ (in the sense of Section 2.4) with respect to which Vertex Cover is FPT.

Open Problem 10. Find a meta-kernelization theorem (cf. [28, 110, 112]) that is applicable to problems on *weighted graphs*.

Open Problem 11. Does Subset Sum admit a deterministic[3] polynomial kernel parameterized by the cardinality of the multiset of input numbers?

[3] A randomized kernel follows from the work of Harnik and Naor [130, Claim 2.7].

Part V

Appendix

The origins of graph theory are humble, even frivolous.
—Robin J. Wilson, 1999

A Graph Theory

The language of graph theory is indispensable for this thesis. All the relevant definitions are collected here for easy reference. We begin our discussion by defining graphs and the concepts needed to analyze their substructures. Section A.2 introduces special notation for some graphs that occur particularly often. We then define modification operations on graphs in Section A.3, leading to the definition of graph minors. Excluding specified graphs as a minor will be useful when describing well-known graph classes in Section A.4. Finally, we can use distance measures of a graph to specified graph classes as parameters that describe the graph structure (Section A.5).

A.1 Graphs

Graphs are elegant mathematical structures that express binary relations between entities. All graphs we consider are finite, undirected, and simple, unless explicitly stated otherwise. A *finite simple graph* G consists of a non-empty finite set $V(G)$ of elements called *vertices* (or *nodes*), and a finite set $E(G) \subseteq \binom{V(G)}{2}$ of elements called *edges*. We call $V(G)$ the *vertex set* and $E(G)$ the *edge set* of G. An edge of G is an unordered pair $\{u, v\} \in E(G)$ for distinct vertices $u, v \in V(G)$, which form the *endpoints* of the edge. A graph H is a *subgraph* of graph G, denoted $H \subseteq G$, if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. For $X \subseteq V(G)$ the subgraph of G *induced* by X is denoted by $G[X]$. Its vertex set is $V(G) \cap X$ and its edge set is $E(G) \cap \binom{X}{2}$.

The *open neighborhood* of vertex v in graph G is the set $\{u \in V(G) \mid \{u, v\} \in E(G)\}$, and is denoted by $N_G(v)$. The *closed neighborhood* of v is $N_G[v] :=$

$N_G(v) \cup \{v\}$. The notation extends naturally to sets of vertices S. The open neighborhood is $N_G(S) := \bigcup_{v \in S} N_G(v) \setminus S$, whereas the closed neighborhood is $N_G[S] := \bigcup_{v \in S} N_G(v) \cup S$. The *degree* of a vertex v in graph G is $\deg_G(v) := |N_G(v)|$. The maximum degree of a vertex in G is denoted by $\Delta(G)$.

A *path* in G is a sequence of distinct vertices $(v_0, v_1, \ldots, v_k)$ with $\{v_{i-1}, v_i\} \in E(G)$ for $i \in [k]$. The *length* of the path is the number k of edges on it, whereas the *size* is the number of vertices $(k+1)$. The vertices v_0 and v_k are the *endpoints* of the path. We may also identify a path by its edges, rather than its vertices. A *cycle* is a sequence of vertices $(v_0, v_1, \ldots, v_k)$ for $k \geq 3$ such that the elements $\{v_1, \ldots, v_k\}$ are pairwise distinct and $v_0 = v_k$, with $\{v_{i-1}, v_i\} \in E(G)$ for $i \in [k]$. The size and length of a cycle coincide, and are equal to the number of edges (or alternatively, the number of vertices) on the cycle. An *odd cycle* is a cycle of odd length. A *chord* in a cycle is an edge between two vertices that are not successive on the cycle. A cycle without chords is *chordless*. A graph is *Hamiltonian* if there is a cycle that meets all its vertices, which is called a *Hamiltonian cycle*. A *Hamiltonian path* is a path that meets all vertices of the graph.

Two vertices u and v are *connected* in graph G if there is a path in G that has u and v as its endpoints. A *connected component* of a graph is an inclusion-maximal set of vertices with the property that any pair of vertices in the component is connected. It is easy to see that the connected components of a graph partition its vertex set. We will also use the term connected component to refer to the subgraph that it induces. A graph is *connected* if it has exactly one connected component.

A graph is *complete* if all unordered pairs of vertices occur as edges. A *clique* is a vertex set that induces a complete subgraph, while a set of vertices is *independent* if there is no edge between any pair of vertices in the set. A vertex v in a graph G is *simplicial* if $N_G(v)$ is a clique. A vertex v is *almost simplicial* if v has a neighbor w such that $N_G(v) \setminus \{w\}$ is a clique. We call such a vertex w a *special neighbor* of v. An *edge clique cover* of a graph G is a collection of vertex subsets, each inducing a clique, such that for every edge $\{u, v\} \in E(G)$ there is a clique in the collection containing u and v. A *matching* in graph G is a set of edges M such that no two distinct edges in M contain the same vertex. If no matching M' has more edges than matching M, then M is a *maximum matching*. If M is a matching but no proper superset is a matching, then M is a *maximal* matching. The same distinction between maximum and maximal is applied to other subsets in the natural way. A matching M *covers* a set of vertices U, if each vertex in U is an endpoint of an edge in M. If $Y \subseteq E(G)$ is a subset of edges then $V(Y)$ denotes the endpoints of the edges in Y. In particular, we will use $V(M)$ for a matching M to represent the endpoints of the matching edges. A matching is *perfect* if every vertex of the graph is incident on exactly one edge in the matching. Given a matching M in a graph G, an M-*augmenting path* in G is a path $(v_0, v_1, \ldots, v_k)$ in G with $k \geq 1$ such that (i) vertices v_0 and v_k are not covered by M and (ii) for every $i \in [k]$ the edge $\{v_{i-1}, v_i\}$ is contained in M if and only if i is even.

A q-coloring of a graph G is a function $f \colon V(G) \to [q]$. A coloring is *proper* if the endpoints of each edge receive distinct colors. A graph admitting a proper q-coloring is called *q-colorable*. An *isomorphism* between two graphs G and H is a bijection $f\colon V(G) \to V(H)$ such that $\forall\{u,v\} \in \binom{V(G)}{2} : \{u,v\} \in E(G) \Leftrightarrow \{f(u), f(v)\} \in E(H)$. Two graphs are *isomorphic* if there is an isomorphism between them. For graphs G and H, an *H-packing* in G is a set of vertex-disjoint subgraphs of G, each of which is isomorphic to H. An H-packing is *perfect* if the subgraphs cover the entire vertex set. We refer to Diestel [81] for a more extensive treatment of graph theory.

A.2 Special Graphs

By P_n we denote the path on n vertices, i.e., the graph with vertex set $[n]$ and edge set $\{\{i, i+1\} \mid i \in [n-1]\}$. The cycle graph on $n \geq 3$ vertices C_n is obtained from P_n by adding the edge $\{1, n\}$. The complete graph on n vertices is K_n. The graph K_3 is also called a *triangle*.

A *biclique* $K_{s,t}$ (complete bipartite graph) whose partite sets have sizes s and t, is a graph whose vertex set can be partitioned as $V(G) = X \uplus Y$ such that $|X| = s$, $|Y| = t$ and $E(G) = \{\{x, y\} \mid x \in X \wedge y \in Y\}$.

A *hole* is a chordless cycle of length at least four. An *anti-hole* is the edge-complement of a hole. An odd (anti-)hole is an (anti-)hole of odd length.

A.3 Operations on Graphs

For a vertex set $X \subseteq V(G)$ we denote by $G - X$ the graph obtained from G by deleting all vertices of X and their incident edges. Deleting the set of edges $Y \subseteq E(G)$ from graph G results in the graph on the same vertex set, with the edge set $E(G) \setminus Y$.

We may also modify a graph G by *contracting an edge* $\{u, v\} \in E(G)$ *into vertex* v. This results in the graph G' obtained from G by removing u and its incident edges, while adding the edges between v and $N_G(\{u, v\})$ that were not already present. Observe that this definition of edge contraction results in a simple graph. When the identity of the vertices in the resulting graph is not important, we will simply speak of *contracting edge* $\{u, v\}$. *Subdividing an edge* $\{u, v\}$ in graph G is the operation of removing the edge from the graph and replacing it by a new vertex x with edges $\{u, x\}$ and $\{x, v\}$. It effectively transforms the direct edge into a path of length two. If graph H can be made from G by repeated edge subdivisions, then H is a *subdivision* of G.

Any graph H that can be obtained from G by a (possibly empty) sequence of vertex deletion, edge deletion, and edge contraction operations, is a *minor* of G. If $H \neq G$ then H is a *proper minor* of G. A *minor model* of a graph H in a graph G is a mapping ϕ from $V(H)$ to subsets of $V(G)$ (called *branch sets*) that satisfies the following conditions: (a) $\phi(u) \cap \phi(v) = \emptyset$ for distinct $u, v \in V(H)$, (b)

$G[\phi(v)]$ is connected for $v \in V(H)$, and (c) there is an edge between a vertex in $\phi(u)$ and a vertex in $\phi(v)$ for all $\{u, v\} \in E(H)$. It is easy to verify that G has a minor model of G if and only if H is a minor of G.

The *edge-complement* $\overline{G}$ of a graph G has the same vertex set as G, with an edge between two vertices if and only if they are not adjacent in G. The *join* of two graphs G_1 and G_2 is the graph $G_1 \otimes G_2$ on the vertex set $V(G_1) \uplus V(G_2)$ and edge set $E(G_1) \uplus E(G_2) \uplus \{\{x, y\} \mid x \in V(G_1) \wedge y \in V(G_2)\}$. The *union* of G_1 and G_2 is the graph $G_1 \cup G_2$ with vertex set $V(G_1) \cup V(G_2)$ and edge set $E(G_1) \cup E(G_2)$. The *disjoint union* of G_1 and G_2 is the graph $G_1 \uplus G_2$ with vertex set $V(G_1) \uplus V(G_2)$ and edge set $E(G_1) \uplus E(G_2)$. Observe that two graphs on the same vertex set are merged by the union operation, while the disjoint union operation produces two distinct copies of each vertex. The disjoint union of t copies of a graph G is represented by $t \cdot G$. By *completing a vertex set $S \subseteq V(G)$ of graph G into a clique* we mean the operation that adds edges between all pairs in S that were previously nonadjacent.

A.4 Graph Classes

A *graph class* Π is a (possibly infinite) set of graphs. A restricted class of graphs is also referred to as a *graph property.* A graph class is *hereditary* if the class is closed under taking vertex-induced subgraphs. We say that a graph G is *vertex-minimal* with respect to Π if $G \in \Pi$ and for all $S \subsetneq V(G)$ the graph $G[S]$ is not contained in Π. A graph G with $E(G) = \emptyset$ is called an *empty* graph. A graph is *acyclic* if it does not contain a cycle. Acyclic graphs are *forests.* A *linear forest* is a forest with maximum degree two or less; it can be interpreted as a disjoint union of paths. A connected acyclic graph is a (free) *tree.* A *rooted tree* is a tree T together with a distinguished vertex $r \in V(T)$ called the *root.* In a rooted tree (T, r), the *children* of a vertex v are the neighbors of v that do not lie on the unique path from v to r. A *binary tree* is a tree of maximum degree at most three. A rooted binary tree is *proper* if each vertex has either zero or two children. A *spanning tree* of G is a tree subgraph that includes all vertices of G. A vertex of degree at most one is called a *leaf*. If v is a vertex in a tree and v is not a leaf, then it is an *internal node* of the tree. The *leaf set* of a graph G is $\textsc{Leaves}(G) := \{v \in V(G) \mid \deg_G(v) \leq 1\}$.

A graph is *bipartite* if it does not contain an odd cycle as a subgraph, or equivalently, if it has a proper 2-coloring. The vertex set of such graphs can be partitioned into two *partite sets* such that all edges have exactly one endpoint in each set. All forests are bipartite. The edge-complement $\overline{G}$ of a bipartite graph G is a *cobipartite* graph; the partite sets of G form cliques in $\overline{G}$. Graphs whose vertex cover number (see Section A.5) equals the number of edges in a maximum matching are König(-Egerváry) graphs; Kőnig [160] and Egerváry [90] independently proved that bipartite graphs have this property. Unlike many other graph classes considered in this work, König graphs are not hereditary.

The *intersection graph* of a family of sets $\mathcal{F}$ is the graph on vertex set $\mathcal{F}$, with an edge between $u, v \in \mathcal{F}$ if the corresponding sets have a nonempty intersection. An *interval graph* is an intersection graph of a set of closed intervals on the real line. A graph is *chordal* if every cycle of length at least four has a chord. Equivalently, a graph is chordal if it does not contain a hole as an induced subgraph. A *cochordal* graph is the edge-complement of a chordal graph. We denote by $\biguplus$Cochordal the class of graphs in which each connected component is cochordal. A graph is a *split graph* if there is a partition of $V(G)$ into sets $X \uplus Y$ such that X is a clique and Y is an independent set. The class $\biguplus$Split contains those graphs in which each connected component is a split graph. A graph is a *cograph* if it does not contain P_4 as an induced subgraph. A graph is *perfect* if for all its induced subgraphs the chromatic number equals the size of the largest clique. Equivalently, a graph is perfect if and only if it does not contain any odd hole or odd anti-hole as an induced subgraph. This equivalence was conjectured a long time ago [13], but proven only recently [60]. All interval graphs are chordal. All chordal graphs, cographs, and split graphs are perfect [45].

Planar graphs are those graphs that can be drawn in the plane without crossing edges. By Kuratowski's theorem [169] (cf. [81, §4.4]), these are exactly the graphs that do not contain $K_{3,3}$ or K_5 as a minor. *Outerplanar graphs* are those graphs that can be drawn in the plane without crossings such that all the vertices lie on the outer face; these are exactly the $\{K_4, K_{2,3}\}$-minor-free graphs [81, Exercise 4.22]. *Cluster graphs* are disjoint unions of cliques; their edge-complements are *cocluster graphs*. For a graph class $\mathcal{F}$ and a vertex set $X \subseteq V(G)$ of a graph G such that $G - X \in \mathcal{F}$, we say that X is a *modulator* to the class $\mathcal{F}$. Such a set X is also called an $\mathcal{F}$ modulator.

The book by Brandstädt, Le, and Spinrad [45] contains more information about the graph classes used in this work.

A.5 Graph Parameters

The structural complexity of a graph can be measured in terms of various graph parameters. A *vertex cover* of a graph G is a subset of the vertices meeting all the edges. The *vertex cover number* $\text{vc}(G)$ is the cardinality of a minimum-size vertex cover in G. The *independence number* $\alpha(G)$ is the cardinality of a maximum independent set (MIS), while the *clique number* $\omega(G)$ is defined as the maximum number of vertices in a complete subgraph of G. A *feedback vertex set* (FVS) is a vertex subset meeting all the cycles; the removal of such a set results in a forest. The *feedback vertex number* $\text{fvs}(G)$ is the cardinality of a minimum FVS in G. Similarly, the *linear forest number* is the minimum size of a vertex subset whose removal results in a linear forest. An *odd cycle transversal* is a vertex set meeting all the odd cycles: its removal leaves the graph bipartite. The *chromatic number* $\chi(G)$ of a graph is the smallest number q for which it has a proper q-coloring. The *matching number* $\mu(G)$ is the cardinality of a maximum matching.

The *orientable genus* of a graph G is a measure of the minimum complexity of a surface in which G can be drawn without crossings. Informally, the genus of an orientable surface is the number of handles that have to be added to a sphere to form the surface. The genus of a planar graph is zero. The *max leaf number* (cf. [103]) of a connected graph is the maximum number of leaves in any of its spanning trees. For disconnected graphs we define the max leaf number to be the sum of the max leaf numbers of the connected components. The linear forest number of a connected graph is at most its max leaf number minus one [78].

We may measure the complexity of a graph by its modification distance to specified graph classes. The *vertex-deletion distance* of G to a graph class $\mathcal{F}$ is the cardinality of a smallest vertex set $X \subseteq V(G)$ such that $G - X \in \mathcal{F}$, or ∞ if no such vertex set exists. In our applications, however, the graph classes we use always contain the singleton graph K_1, which means there is a well-defined modulator to $\mathcal{F}$. The *edge-deletion (completion) distance* of G to $\mathcal{F}$ is the cardinality of a smallest edge set whose removal (addition) results in a graph in $\mathcal{F}$. Finally, the *edge modification distance* of G to $\mathcal{F}$ is the smallest set $Y \subseteq \binom{V(G)}{2}$ such that the graph on vertex set $V(G)$ and whose edge set is the symmetric difference of $E(G)$ and Y, is contained in $\mathcal{F}$.

A.5.1 Linear Layouts

Several graph width metrics are defined in terms of optimization problems over linear orderings of the vertex set. A *linear layout* of a graph G on n vertices is a permutation $\pi\colon V(G) \to [n]$. The *cutwidth* of graph G under a layout π is $\textsc{cutw}_\pi(G) := \max_{i=1}^{n} |\{\{u,v\} \in E(G) \mid \pi(u) \leq i < \pi(v)\}|$. Informally, one may think of the cutwidth under π as the maximum number of edges cut by a vertical line placed between vertices, when arranging the graph on a horizontal line in the order specified by π. The *modified cutwidth* of G under layout π is $\textsc{cutw}^*_\pi(G) := \max_{i=1}^{n} |\{\{u,v\} \in E(G) \mid \pi(u) < i < \pi(v)\}|$. It corresponds to the setting when restricting the vertical line to go through a vertex, while counting the number of edges whose endpoints lie on different sides of the line. The *bandwidth* of a layout is $\textsc{bandw}_\pi(G) := \max_{\{u,v\} \in E(G)} |\pi(u) - \pi(v)|$, i.e., the maximum distance that adjacent vertices are placed apart by the ordering.

The *cutwidth of a graph* is the minimum cutwidth of any of its layouts. The modified cutwidth and bandwidth of a graph are defined similarly. Note that the bandwidth of a graph can decrease when edges are subdivided. The *topological bandwidth* of a graph G is the minimum bandwidth over all subdivisions of G.

A.5.2 Treewidth and Pathwidth

The concepts of treewidth and pathwidth have been rediscovered several times and feature prominently in Robertson and Seymour's proof of Wagner's conjecture [200]. They measure the structural similarity of a graph to a tree or path, respectively.

Definition A.1. A *tree decomposition* of a graph G is a pair $(T, \mathcal{X})$, where T is a tree and $\mathcal{X} = \{\mathcal{X}_i \mid i \in V(T)\}$ a collection of subsets of $V(G)$, called *bags*, such that:

1. $\bigcup_{i \in V(T)} \mathcal{X}_i = V(G)$.
2. For each edge $\{u, v\} \in E(G)$ there is a node $i \in V(T)$ with $\{u, v\} \subseteq \mathcal{X}_i$.
3. For each $v \in V(G)$ the nodes $\{i \mid v \in \mathcal{X}_i\}$ induce a connected subtree of T.

The *width* of the tree decomposition is $\max_{i \in V(T)} |\mathcal{X}_i| - 1$. The *treewidth* of a graph G, denoted $\textsc{tw}(G)$, is the minimum width over all tree decompositions of G.

The definition implies that if $(T, \mathcal{X})$ is a tree decomposition of a graph G and v is a vertex of G occurring in bags $\mathcal{X}_i$ and $\mathcal{X}_j$, then v is also contained in all bags corresponding to nodes of T that lie on the unique path between i and j in T. This is sometimes referred to as the *convexity property* of tree decompositions.

Suppose we have a graph G with a weight function $w \colon V(G) \to \mathbb{N}$. The *weighted width* of a tree decomposition $(T, \{\mathcal{X}_i \mid i \in V(T)\})$ of G is $\max_{i \in V(T)} \sum_{v \in \mathcal{X}_i} w(v)$ minus one, and the *weighted treewidth* of G is the minimum weighted width of a tree decomposition of G.[1]

A *path decomposition* of a graph can be defined as a tree decomposition with the extra restriction that T is a path ($\Delta(T) \leq 2$). However, we prefer to use the following simpler, but equivalent, definition.

Definition A.2. A *path decomposition* of a graph G is a sequence $(\mathcal{X}_1, \ldots, \mathcal{X}_r)$ of subsets of $V(G)$, called *bags*, such that:

1. $\bigcup_{i \in [r]} \mathcal{X}_i = V(G)$.
2. For each edge $\{u, v\} \in E(G)$ there is a bag $\mathcal{X}_i$ containing v and w.
3. For each $v \in V(G)$, the bags containing v are consecutive in the sequence.

The *width* of a path decomposition is $\max_{i \in [r]} |\mathcal{X}_i| - 1$. The *pathwidth* of a graph G, denoted $\textsc{pw}(G)$, is the minimum width over all path decompositions of G.

The following facts help to assess the relationships between parameters. If a graph class $\mathcal{F}$ has treewidth (pathwidth) bounded by a constant c, then the treewidth (pathwidth) of a graph G with vertex-deletion distance k to $\mathcal{F}$ is at most $k + c$. Hence $\textsc{tw}(G) \leq \textsc{pw}(G) \leq \textsc{vc}(G)$, and $\textsc{tw}(G) \leq \textsc{fvs}(G) + 1$. The treewidth or pathwidth of a graph does not increase when taking a minor [20, Lemma 16]. Outerplanar graphs have treewidth at most two [20, Lemma 78]. The treewidth (pathwidth) of a graph is exactly one less than the maximum clique size over all its chordal (interval) supergraphs. These facts, and many more, can be found in Bodlaender's survey of the graph-structural aspects of treewidth [20].

If the maximum number of edges in a simple path in G is k, then the pathwidth of G is at most k. This connection was first exploited by Fellows and Langston [102]

[1]Observe that we deviate from the definition of weighted width employed by van den Eijkhof et al. [91], who do not subtract one from the maximum. Our definition ensures that the treewidth of a graph equals its weighted treewidth when all vertices have weight one.

(see also [85, Theorem 8.2]), who claimed a bound on the treewidth in terms of the length of a longest cycle. It is straightforward to adapt their proof to show the pathwidth bound in terms of the length of a longest path.

A.5.3 Cliquewidth

The concept of *cliquewidth* was introduced by Courcelle, Makowsky and Rotics [66] in an attempt to generalize the algorithmic properties of bounded-treewidth graphs to structurally simple, yet dense graphs. We do not need the formal definition of cliquewidth in this thesis, but due to the existence of algorithmic meta-theorems for graphs of bounded cliquewidth it is important to know how cliquewidth relates to other graph parameters. The cliquewidth of a graph G is denoted $\textsc{cw}(G)$.

Proposition A.1 ([68])**.** *Cographs have cliquewidth at most two.*

Proposition A.2 ([5])**.** *If a graph G is obtained from a graph H by deleting k vertices, then $\textsc{cw}(G) \leq \textsc{cw}(H) \leq 2^k(\textsc{cw}(G) + 1)$.*

The preceding propositions show, similarly to the case of treewidth exhibited above, that having bounded deletion distance to a graph family of bounded cliquewidth, gives a bound on the overall cliquewidth. In particular, if the vertex-deletion distance of G to the class of cographs is k then its cliquewidth is at most $3 \cdot 2^k$. Combined with the extended version of Courcelle's theorem for cliquewidth [66], this fact can be useful when classifying the complexity of parameterizations by deletion distance to cographs or its subclasses.

Most controversies would soon be ended, if those engaged in them would first accurately define their terms, and then adhere to their definitions.

—Tryon Edwards, 1809–1894

B Compendium of Classical Problems

This compendium provides the definitions of the problems under consideration. As we study a range of different parameterizations for each problem, we define the *classical* variants of the problems, thereby allowing the parameterization to depend on the context.

In the following definitions we use the convention that variables occurring in problem names (such as the q in q-Coloring) are treated as fixed constants in complexity analysis.

Bandwidth
Input: A graph G and an integer ℓ.
Question: Is there a linear layout of G of bandwidth at most ℓ, i.e., a permutation $\pi\colon V(G) \to [n]$ such that $\max_{\{u,v\}\in E(G)} |\pi(u) - \pi(v)| \leq \ell$?

Chordal Deletion
Input: A graph G and an integer ℓ.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ that intersects all chordless cycles?

Clique
Input: A graph G and an integer ℓ.
Question: Does G contain a complete subgraph on ℓ vertices?

cnf-sat
Input: A set U of n variables and a collection C of clauses. Each clause is a subsets of the literals over U.

Question: Is there a truth assignment to the variables such that each clause contains at least one true literal?

Connected Dominating Set
Input: A graph G and an integer ℓ.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ that forms a dominating set of G and induces a connected subgraph?

Connected Feedback Vertex Set
Input: A graph G and an integer ℓ.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ that forms a feedback vertex set of G and induces a connected subgraph?

Connected Odd Cycle Transversal
Input: A graph G and an integer ℓ.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ that forms an odd cycle transversal of G and induces a connected subgraph?

Connected Vertex Cover
Input: A graph G and an integer ℓ.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ that forms a vertex cover of G and induces a connected subgraph?

Disjoint Cycles
Input: A graph G and an integer ℓ.
Question: Does G contain a set of ℓ vertex-disjoint cycles?

Disjoint Paths
Input: A graph G, an integer ℓ, and a set of ℓ vertex pairs $\{(s_1, t_1), \ldots, (s_\ell, t_\ell)\}$.
Question: Is there a set of ℓ vertex-disjoint paths, such that each pair (s_i, t_i) is connected by exactly one of the paths?

Dominating Set
Input: A graph G and an integer ℓ.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ such that each vertex not in S, has a neighbor in S?

Equitable Coloring
Input: A graph G and an integer ℓ.
Question: Is there a proper coloring of G with at most ℓ colors, in which the sizes of any two color classes differ by at most one?

$\mathcal{F}$-Minor-Free Deletion
Input: A graph G and an integer ℓ.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ such that $G - S$ does not contain any graph in $\mathcal{F}$ as a minor?
($\mathcal{F}$ can be any fixed finite set of graphs.)

Feedback Vertex Set
Input: A graph G and an integer ℓ.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ that intersects all cycles?

H-Packing
Input: A graph G and an integer ℓ.
Question: Does G contain ℓ vertex-disjoint subgraphs isomorphic to H?
(H can be any fixed graph.)

Hamiltonian Cycle
Input: A graph G.
Question: Is there a cycle in G that visits all vertices of G?

Hamiltonian Path
Input: A graph G.
Question: Is there a path in G that visits all vertices of G?

Hitting Set
Input: A family of sets $\mathcal{F}$ over a finite universe U, and an integer ℓ.
Question: Is there a set $S \subseteq U$ of size at most ℓ that intersects all sets in $\mathcal{F}$?

Independent Set
Input: A graph G and an integer ℓ.
Question: Is there a set of ℓ vertices in G that are pairwise nonadjacent?

Interval Deletion
Input: A graph G and an integer ℓ.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ such that $G - S$ is an interval graph?

List Coloring
Input: A graph G and a specification of allowed colors $L(v) \subseteq \mathbb{N}$ for each vertex $v \in V(G)$.
Question: Is there a proper coloring of G that assigns each vertex v a color from its list $L(v)$?

Long Colored Path
Input: A graph G, an integer ℓ, and a (not necessarily proper) coloring $\phi\colon V(G) \to [\ell]$.
Question: Is there a path in G containing exactly one vertex of each color?

Long Cycle
Input: A graph G and an integer ℓ.
Question: Is there a cycle in G on at least ℓ vertices?

Long Induced Path
Input: A graph G and an integer ℓ.
Question: Is there a set $S \subseteq V(G)$ of size at least ℓ that induces a path in G?

Long Maximal Induced Path
Input: A graph G and an integer ℓ.
Question: Is there a set $S \subseteq V(G)$ of size exactly ℓ that induces a path in G, and is maximal with this property?

Long Maximal Path
Input: A graph G and an integer ℓ.
Question: Is there a path in G consisting of ℓ edges $Y \subseteq E(G)$ such that no strict superset of Y forms a path in G?

Long Path
Input: A graph G and an integer ℓ.
Question: Is there a path in G on at least ℓ vertices?

Long Pointed Path
Input: A graph G, a starting point $v \in V(G)$, and an integer ℓ.
Question: Is there a path in G on at least ℓ vertices that starts in v?

Max Cut
Input: A graph G and an integer ℓ.
Question: Is there a partition of the vertex set into $X \uplus Y$ such that at least ℓ edges have one endpoint in each set?

Max Leaf
Input: A connected graph G and an integer ℓ.
Question: Is there a spanning subtree of G with at least ℓ leaves?

ML Type Checking
Input: A set of type declarations in ML.
Question: Are the type declarations consistent?
(See [133, 174] for details.)

Multicolored Clique
Input: A graph G, an integer ℓ, and a (not necessarily proper) coloring $\phi\colon V(G) \to [\ell]$.
Question: Is there a clique in G containing one vertex of each color?

nae-sat
Input: A set U of n variables and a collection C of clauses. Each clause is a subsets of the literals over U.
Question: Is there a truth assignment to the variables such that each clause contains at least one true literal and one false literal?

Odd Cycle Transversal
Input: A graph G and an integer ℓ.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ that intersects all odd cycles?

Orientable Genus
Input: A graph G and an integer ℓ.
Question: Can G be embedded in an orientable surface of genus ℓ?

Partition into q $\mathcal{F}$-Minor-Free Graphs
Input: A graph G.
Question: Is there a partition of the vertex set into q sets $S_1 \cup S_2 \cup \ldots \cup S_q$ such that for each $i \in [q]$ the graph $G[S_i]$ does not contain a graph from $\mathcal{F}$ as a minor? ($\mathcal{F}$ can be any fixed finite set of graphs, q can be any positive integer.)

Partition into q Forests
Input: A graph G.
Question: Is there a partition of the vertex set into q sets $S_1 \cup S_2 \cup \ldots \cup S_q$ such that for each $i \in [q]$ the graph $G[S_i]$ is a forest?

Pathwidth
Input: A graph G and an integer ℓ.
Question: Is the pathwidth of G at most ℓ?

Perfect Deletion
Input: A graph G and an integer ℓ.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ that intersects all odd holes and odd anti-holes?

Planarization
Input: A graph G and an integer ℓ.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ such that $G - S$ is planar?

Precoloring Extension
Input: A graph G, an integer ℓ, and a partial function $f\colon V(G) \to [\ell]$.
Question: Can f be extended to a proper ℓ-coloring of G?

q-List Coloring
Input: A graph G and a specification of allowed colors $L(v) \subseteq [q]$ for each vertex $v \in V(G)$.
Question: Is there a proper coloring of G that assigns each vertex v a color from its list $L(v)$?

$s - t$ Path with Forbidden Pairs
Input: A graph G, distinct vertices $s, t \in V(G)$, and a set $H \subseteq \binom{V(G)}{2}$ of forbidden pairs.
Question: Is there an $s - t$ path in G that contains at most one vertex of each pair $\{u, v\} \in H$?

Set Basis
Input: A family of sets $\mathcal{F}$ over a finite universe U, and an integer ℓ.
Question: Is there a subfamily $\mathcal{F}'$ of $\mathcal{F}$, containing exactly ℓ sets, such that for each $S \in \mathcal{F}$ there is a subfamily of $\mathcal{F}'$ whose union is S?

Set Cover
Input: A family of sets $\mathcal{F}$ over a finite universe U, and an integer ℓ.
Question: Is there a subfamily of $\mathcal{F}$, containing at most ℓ sets, whose union is U?

Set Packing
Input: A family of sets $\mathcal{F}$ over a finite universe U, and an integer ℓ.
Question: Does $\mathcal{F}$ contain ℓ pairwise disjoint sets?

Short Single-Tape Turing Machine Acceptance
Input: A description of a single-tape Turing machine M of unbounded alphabet size and nondeterminism, and an integer ℓ.
Question: Does M accept the empty string in at most ℓ steps?

Steiner Tree
Input: A graph G, an integer ℓ, and a set $T \subseteq V(G)$ of terminals.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ such that $G[S \cup T]$ is connected?

Subset Sum
Input: A multiset S of positive integers, and a positive integer ℓ.
Question: Is there a subset of S whose elements sum to exactly ℓ?

Treewidth
Input: A graph G and an integer ℓ.
Question: Is the treewidth of G at most ℓ?

Treewidth-η Transversal
Input: A graph G and an integer ℓ.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ such that $G - S$ has treewidth at most η?
(η can be any positive integer.)

Vertex Cover
Input: A graph G and an integer ℓ.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ that contains at least one endpoint of every edge?

Weighted Treewidth
Input: A graph G, a weight function $w\colon V(G) \to \mathbb{N}$ encoded in unary, and an integer ℓ.
Question: Is the weighted treewidth of G under w at most ℓ?
(See Appendix A.5.2 for details.)

Bibliography

[1] Faisal N. Abu-Khzam, Rebecca L. Collins, Michael R. Fellows, Michael A. Langston, W. Henry Suters, and Christopher T. Symons. "Kernelization algorithms for the vertex cover problem: Theory and experiments". In: *Proceedings of the 6th Workshop on Algorithm Engineering and Experiments and the First Workshop on Analytic Algorithmics and Combinatorics, ALENEX/ANALC 2004*. 2004, pages 62–69.

[2] Faisal N. Abu-Khzam, Michael R. Fellows, Michael A. Langston, and W. Henry Suters. "Crown structures for vertex cover kernelization". In: *Theory Comput. Syst.* 41(3), 2007, pages 411–430. DOI: 10.1007/s00224-007-1328-0.

[3] Isolde Adler, Stavros G. Kolliopoulos, Philipp Klaus Krause, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. "Tight bounds for linkages in planar graphs". In: *Proceedings of the 38th International Colloquium on Automata, Languages and Programming, ICALP 2011, Part I.* Volume 6755 of LNCS. 2011, pages 110–121. DOI: 10.1007/978-3-642-22006-7_10.

[4] Jochen Alber, Michael R. Fellows, and Rolf Niedermeier. "Polynomial-time data reduction for dominating set". In: *J. ACM* 51(3), 2004, pages 363–384. DOI: 10.1145/990308.990309.

[5] Peter Allen, Vadim V. Lozin, and Michaël Rao. "Clique-width and the speed of hereditary properties". In: *Electron. J. Combin.* 16(1), 2009. EJC ID: v16i1r35.

[6] Noga Alon, Gregory Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. "Solving MAX-r-SAT above a tight lower bound". In: *Algorithmica* 61(3), 2011, pages 638–655. DOI: 10.1007/s00453-010-9428-7.

[7] Noga Alon, Raphael Yuster, and Uri Zwick. "Color-coding". In: *J. ACM* 42(4), 1995, pages 844–856. DOI: 10.1145/210332.210337.

[8] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. "Complexity of finding embeddings in a k-tree". In: *SIAM J. Algebra. Discr.* 8(2), 1987, pages 277–284. DOI: 10.1137/0608024.

[9] Stefan Arnborg, Jens Lagergren, and Detlef Seese. "Easy problems for tree-decomposable graphs". In: *J. Algorithms* 12(2), 1991, pages 308–340. DOI: 10.1016/0196-6774(91)90006-K.

[10] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009, pages 1–579.

[11] Vineet Bafna, Piotr Berman, and Toshihiro Fujito. “A 2-approximation algorithm for the undirected feedback vertex set problem”. In: *SIAM J. Discrete Math.* 12(3), 1999, pages 289–297. DOI: 10.1137/S0895480196305124.

[12] Ann Becker and Dan Geiger. “Optimization of Pearl’s method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem”. In: *Artif. Intell.* 83(1), 1996, pages 167–188. DOI: 10.1016/0004-3702(95)00004-6.

[13] Claude Berge. “Färbung von graphen, deren sämtliche bzw. deren ungerade kreise starr sind”. In: *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe* 10, 1961, pages 114–115.

[14] Piotr Berman and Georg Schnitger. “On the complexity of approximating the independent set problem”. In: *Inf. Comput.* 96(1), 1992, pages 77–94. DOI: 10.1016/0890-5401(92)90056-L.

[15] Pascal Berthomé, Jean-François Lalande, and Vincent Levorato. *Implementation of exponential and parametrized algorithms in the AGAPE project*. 2012. arXiv:1201.5985.

[16] Daniel Binkele-Raible, Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. “Kernel(s) for problems with no kernel: On out-trees with many leaves”. In: *ACM Trans. Algorithms* 8(4), 2012, page 38. DOI: 10.1145/2344422.2344428.

[17] Andreas Björklund. “Determinant sums for undirected Hamiltonicity”. In: *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010*. 2010, pages 173–182. DOI: 10.1109/FOCS.2010.24.

[18] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. *Narrow sieves for parameterized paths and packings*. 2010. arXiv:1007.1161.

[19] Hans L. Bodlaender. “A linear-time algorithm for finding tree-decompositions of small treewidth”. In: *SIAM J. Comput.* 25(6), 1996, pages 1305–1317. DOI: 10.1145/167088.167161.

[20] Hans L. Bodlaender. “A partial k-arboretum of graphs with bounded treewidth”. In: *Theor. Comput. Sci.* 209(1-2), 1998, pages 1–45. DOI: 10.1016/S0304-3975(97)00228-4.

[21] Hans L. Bodlaender. “Kernelization: New upper and lower bound techniques”. In: *Proceedings of the 4th International Workshop on Parameterized and Exact Computation, IWPEC 2009*. Volume 5917 of LNCS. 2009, pages 17–37. DOI: 10.1007/978-3-642-11269-0_2.

[22] Hans L. Bodlaender. "Necessary edges in k-chordalizations of graphs". In: *J. Comb. Optim.* 7(3), 2003, pages 283–290. DOI: 10.1023/A:1027320705349.

[23] Hans L. Bodlaender. "On linear time minor tests with depth-first search". In: *J. Algorithms* 14(1), 1993, pages 1–23. DOI: 10.1006/jagm.1993.1001.

[24] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. "On problems without polynomial kernels". In: *J. Comput. Syst. Sci.* 75(8), 2009, pages 423–434. DOI: 10.1016/j.jcss.2009.04.001.

[25] Hans L. Bodlaender, Michael R. Fellows, and Michael T. Hallett. "Beyond NP-completeness for problems of bounded width (extended abstract): Hardness for the W hierarchy". In: *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing, STOC 1994*. 1994, pages 449–458. DOI: 10.1145/195058.195229.

[26] Hans L. Bodlaender, Fedor V. Fomin, Arie M. C. A. Koster, Dieter Kratsch, and Dimitrios M. Thilikos. "A note on exact algorithms for vertex ordering problems on graphs". In: *Theory Comput. Syst.* 50(3), 2012, pages 420–432. DOI: 10.1007/s00224-011-9312-0.

[27] Hans L. Bodlaender, Fedor V. Fomin, Arie M. C. A. Koster, Dieter Kratsch, and Dimitrios M. Thilikos. "On exact algorithms for treewidth". In: *Proceedings of the 14th Annual European Symposium on Algorithms, ESA 2006*. Volume 4168 of LNCS. 2006, pages 672–683. DOI: 10.1007/11841036_60.

[28] Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. "(Meta) Kernelization". In: *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009*. 2009, pages 629–638. DOI: 10.1109/FOCS.2009.46.

[29] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. "Cross-composition: A new technique for kernelization lower bounds". In: *Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011*. Volume 9 of LIPIcs. 2011, pages 165–176. DOI: 10.4230/LIPIcs.STACS.2011.165.

[30] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. "Kernel bounds for path and cycle problems". In: *Proceedings of the 6th International Symposium on Parameterized and Exact Computation, IPEC 2011*. Volume 7112 of LNCS. 2011, pages 145–158. DOI: 10.1007/978-3-642-28050-4_12.

[31] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. "Kernel bounds for path and cycle problems". In: *Theor. Comput. Sci.* 2012. Online First. DOI: 10.1016/j.tcs.2012.09.006.

[32] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. "Kernel bounds for structural parameterizations of pathwidth". In: *Proceedings of the 13th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2012*. Volume 7357 of LNCS. 2012, pages 352–363. DOI: 10.1007/978-3-642-31155-0_31.

[33] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. *Kernelization Lower Bounds By Cross-Composition*. 2012. arXiv:1206.5941.

[34] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. *Preprocessing for Treewidth: A Combinatorial Analysis through Kernelization*. 2011. arXiv:1104.4217v1.

[35] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. "Preprocessing for treewidth: A combinatorial analysis through kernelization". In: *Proceedings of the 38th International Colloquium on Automata, Languages and Programming, ICALP 2011, Part I*. Volume 6755 of LNCS. 2011, pages 437–448. DOI: 10.1007/978-3-642-22006-7_37.

[36] Hans L. Bodlaender, Klaus Jansen, and Gerhard J. Woeginger. "Scheduling with incompatible jobs". In: *Discrete Appl. Math.* 55(3), 1994, pages 219–232. DOI: 10.1016/0166-218X(94)90009-4.

[37] Hans L. Bodlaender and Arie M. C. A. Koster. "Combinatorial optimization on graphs of bounded treewidth". In: *Comput. J.* 51(3), 2008, pages 255–269. DOI: 10.1093/comjnl/bxm037.

[38] Hans L. Bodlaender and Arie M. C. A. Koster. "Safe separators for treewidth". In: *Discrete Math.* 306(3), 2006, pages 337–350. DOI: 10.1016/j.disc.2005.12.017.

[39] Hans L. Bodlaender, Arie M. C. A. Koster, and Frank van den Eijkhof. "Preprocessing rules for triangulation of probabilistic networks". In: *Comput. Intell.* 21(3), 2005, pages 286–305. DOI: 10.1111/j.1467-8640.2005.00274.x.

[40] Hans L. Bodlaender, Arie M. C. A. Koster, and Thomas Wolle. "Contraction and treewidth lower bounds". In: *J. Graph Algorithms Appl.* 10(1), 2006, pages 5–49. DOI: 10.7155/jgaa.00117.

[41] Hans L. Bodlaender and Rolf H. Möhring. "The pathwidth and treewidth of cographs". In: *SIAM J. Discrete Math.* 6(2), 1993, pages 181–188. DOI: 10.1137/0406014.

[42] Hans L. Bodlaender and Udi Rotics. "Computing the treewidth and the minimum fill-in with the modular decomposition". In: *Algorithmica* 36(4), 2003, pages 375–408. DOI: 10.1007/s00453-003-1026-5.

[43] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. "Kernel bounds for disjoint cycles and disjoint paths". In: *Theor. Comput. Sci.* 412(35), 2011, pages 4570–4578. DOI: 10.1016/j.tcs.2011.04.039.

[44] Richard B. Borie, R. Gary Parker, and Craig A. Tovey. "Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families". In: *Algorithmica* 7(5&6), 1992, pages 555–581. DOI: 10.1007/BF01758777.

[45] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, 1999.

[46] Jonathan F. Buss and Judy Goldsmith. "Nondeterminism within P". In: *SIAM J. Comput.* 22(3), 1993, pages 560–572. DOI: 10.1137/0222038.

[47] Jin-yi Cai, Venkatesan T. Chakaravarthy, Lane A. Hemaspaandra, and Mitsunori Ogihara. "Competing provers yield improved Karp-Lipton collapse results". In: *Inf. Comput.* 198(1), 2005, pages 1–23. DOI: 10.1016/j.ic.2005.01.002.

[48] Leizhen Cai. "Parameterized complexity of vertex colouring". In: *Discrete Appl. Math.* 127(3), 2003, pages 415–429. DOI: 10.1016/S0166-218X(02)00242-1.

[49] Liming Cai, Jianer Chen, Rodney G. Downey, and Michael R. Fellows. "On the parameterized complexity of short computation and factorization". In: *Arch. Math. Log.* 36(4-5), 1997, pages 321–337. DOI: 10.1007/s001530050069.

[50] Liming Cai and David W. Juedes. "On the existence of subexponential parameterized algorithms". In: *J. Comput. Syst. Sci.* 67(4), 2003, pages 789–807. DOI: 10.1016/S0022-0000(03)00074-6.

[51] Marco Cesati and Miriam Di Ianni. "Computation models for parameterized complexity". In: *Math. Log. Q.* 43, 1997, pages 179–202. DOI: 10.1002/malq.19970430204.

[52] Jianer Chen, Henning Fernau, Iyad A. Kanj, and Ge Xia. "Parametric duality and kernelization: Lower bounds and upper bounds on kernel size". In: *SIAM J. Comput.* 37(4), 2007, pages 1077–1106. DOI: 10.1137/050646354.

[53] Jianer Chen, Iyad A. Kanj, and Weijia Jia. "Vertex cover: Further observations and further improvements". In: *J. Algorithms* 41(2), 2001, pages 280–301. DOI: 10.1006/jagm.2001.1186.

[54] Jianer Chen, Iyad A. Kanj, and Ge Xia. "Improved upper bounds for vertex cover". In: *Theor. Comput. Sci.* 411(40-42), 2010, pages 3736 –3756. DOI: 10.1016/j.tcs.2010.06.026.

[55] Jianer Chen, Songjian Lu, Sing-Hoi Sze, and Fenghui Zhang. "Improved algorithms for path, matching, and packing problems". In: *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007*. 2007, pages 298–307. ACM ID: 1283415.

[56] Yijia Chen and Jörg Flum. “On parameterized path and chordless path problems”. In: *Proceedings of the 22nd Annual IEEE Conference on Computational Complexity, CCC 2007*. 2007, pages 250–263. DOI: 10.1109/CCC.2007.21.

[57] Yijia Chen, Jörg Flum, and Moritz Müller. “Lower bounds for kernelizations and other preprocessing procedures”. In: *Theory Comput. Syst.* 48(4), 2011, pages 803–839. DOI: 10.1007/s00224-010-9270-y.

[58] Miroslav Chlebík and Janka Chlebíková. “Crown reductions for the minimum weighted vertex cover problem”. In: *Discrete Appl. Math.* 156(3), 2008, pages 292–312. DOI: 10.1016/j.dam.2007.03.026.

[59] Benny Chor, Michael R. Fellows, and David W. Juedes. “Linear kernels in linear time, or how to save k colors in $\mathcal{O}(n^2)$ steps”. In: *Proceedings of the 30th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2004*. Volume 3353 of LNCS. 2004, pages 257–269. DOI: 10.1007/978-3-540-30559-0_22.

[60] Maria Chudnovsky, Neil Robertson, Paul D. Seymour, and Robin Thomas. “The strong perfect graph theorem”. In: *Ann. Math.* 164, 2006, pages 51–229. DOI: 10.4007/annals.2006.164.51.

[61] Fan R. K. Chung. “On the cutwidth and topological bandwidth of a tree”. In: *SIAM J. Algebra. Discr.* 6, 1985, pages 268–277. DOI: 10.1137/0606026.

[62] Fan R. K. Chung and Paul D. Seymour. “Graphs with small bandwidth and cutwidth”. In: *Discrete Math.* 75(1-3), 1989, pages 113–119. DOI: 10.1016/0012-365X(89)90083-6.

[63] François Clautiaux, Jacques Carlier, Aziz Moukrim, and Stéphane Nègre. “New lower and upper bounds for graph treewidth”. In: *Proceedings of the Second International Workshop on Experimental and Efficient Algorithms, WEA 2003*. Volume 2647 of LNCS. 2003, pages 70–80. DOI: 10.1007/3-540-44867-5_6.

[64] François Clautiaux, Aziz Moukrim, Stéphane Nègre, and Jacques Carlier. “Heuristic and meta-heuristic methods for computing graph treewidth”. In: *RAIRO Rech. Opér.* 38(1), 2004, pages 13–26. DOI: 10.1051/ro:2004011.

[65] Bruno Courcelle. “The monadic second-order logic of graphs I: Recognizable sets of finite graphs”. In: *Inf. Comput.* 85(1), 1990, pages 12–75. DOI: 10.1016/0890-5401(90)90043-H.

[66] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. “Linear time solvable optimization problems on graphs of bounded clique-width”. In: *Theory Comput. Syst.* 33(2), 2000, pages 125–150. DOI: 10.1007/s002249910009.

[67] Bruno Courcelle and Mohamed Mosbah. “Monadic second-order evaluations on tree-decomposable graphs”. In: *Theor. Comput. Sci.* 109(1&2), 1993, pages 49–82. DOI: 10.1016/0304-3975(93)90064-Z.

[68] Bruno Courcelle and Stephan Olariu. "Upper bounds to the clique width of graphs". In: *Discrete Appl. Math.* 101(1-3), 2000, pages 77–114. DOI: 10.1016/S0166-218X(99)00184-5.

[69] Robert Crowston, Michael R. Fellows, Gregory Gutin, Mark Jones, Frances A. Rosamond, Stéphan Thomassé, and Anders Yeo. "Simultaneously satisfying linear equations over F_2: MaxLin2 and max-r-lin2 parameterized above average". In: *Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2011.* Volume 13 of LIPIcs. 2011, pages 229–240. DOI: 10.4230/LIPIcs.FSTTCS.2011.229.

[70] Marek Cygan, Fedor V. Fomin, and Erik Jan van Leeuwen. "Parameterized complexity of firefighting revisited". In: *Proceedings of the 6th International Symposium on Parameterized and Exact Computation, IPEC 2011.* Volume 7112 of LNCS. 2011, pages 13–26. DOI: 10.1007/978-3-642-28050-4_2.

[71] Marek Cygan, Stefan Kratsch, Marcin Pilipczuk, Michał Pilipczuk, and Magnus Wahlström. "Clique cover and graph separation: New incompressibility results". In: *Proceedings of the 39th International Colloquium on Automata, Languages, and Programming, ICALP 2012, Part I.* Volume 7391 of LNCS. 2012, pages 254–265. DOI: 10.1007/978-3-642-31594-7_22.

[72] Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. "On cutwidth parameterized by vertex cover". In: *Proceedings of the 6th International Symposium on Parameterized and Exact Computation, IPEC 2011.* Volume 7112 of LNCS. 2011, pages 246–258. DOI: 10.1007/978-3-642-28050-4_20.

[73] Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. "On the hardness of losing width". In: *Proceedings of the 6th International Symposium on Parameterized and Exact Computation, IPEC 2011.* Volume 7112 of LNCS. 2011, pages 159–168. DOI: 10.1007/978-3-642-28050-4_13.

[74] Marek Cygan, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Ildikó Schlotter. "Parameterized complexity of Eulerian deletion problems". In: *Proceedings of the 37th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2011.* Volume 6986 of LNCS. 2011, pages 131–142. DOI: 10.1007/978-3-642-25870-1_13.

[75] Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. "Kernelization hardness of connectivity problems in d-degenerate graphs". In: *Discrete Appl. Math.* 160(15), 2012, pages 2131–2141. DOI: 10.1016/j.dam.2012.05.016.

[76] Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. "On multiway cut parameterized above lower bounds". In: *Proceedings of the 6th International Symposium on Parameterized and Exact Computation, IPEC 2011*. Volume 7112 of LNCS. 2011, pages 1–12. DOI: 10.1007/978-3-642-28050-4_1.

[77] Jean-Paul Delahaye. "The science behind Sudoku". In: *Sci. Am.* 294, 2006, pages 80–87. DOI: 10.1038/scientificamerican0606-80.

[78] Ermelinda DeLaViña and Bill Waller. "A note on a conjecture of Hansen et al." Manuscript. 2009. URL: http://cms.dt.uh.edu/faculty/delavinae/research/DelavinaWaller2009.pdf.

[79] Holger Dell and Dániel Marx. "Kernelization of packing problems". In: *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012*. 2012, pages 68–81. ACM ID: 2095122.

[80] Holger Dell and Dieter van Melkebeek. "Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses". In: *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010*. 2010, pages 251–260. DOI: 10.1145/1806689.1806725.

[81] Reinhard Diestel. *Graph Theory*. 4th edition. Springer-Verlag, Heidelberg, 2010.

[82] Gabriel Andrew Dirac. "On rigid circuit graphs". In: *Abh. Math. Sem. Hamburg* 25(1-2), 1961, pages 71–76. DOI: 10.1007/BF02992776.

[83] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. "Incompressibility through colors and IDs". In: *Proceedings of the 36th International Colloquium on Automata, Languages and Programming, ICALP 2009, Part I*. Volume 5555 of LNCS. 2009, pages 378–389. DOI: 10.1007/978-3-642-02927-1_32.

[84] Rodney G. Downey, Vladimir Estivill-Castro, Michael R. Fellows, Elena Prieto, and Frances A. Rosamond. "Cutting up is hard to do: The parameterized complexity of k-cut and related problems". In: *Electr. Notes Theor. Comput. Sci.* 78, 2003, pages 209–222. DOI: 10.1016/S1571-0661(04)81014-4.

[85] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.

[86] Rodney G. Downey, Michael R. Fellows, and Michael A. Langston, editors. *The Computer Journal: Special Issue on Parameterized Complexity*. Volume 51. 2008. DOI: 10.1093/comjnl/bxm111.

[87] Rodney G. Downey, Michael R. Fellows, and Ulrike Stege. "Parameterized complexity: A framework for systematically confronting computational intractability". In: *Contemp. Trends in Discrete Math.* Volume 49. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. 1999, pages 49–99. CiteSeer ID: 10.1.1.12.4079.

[88] Rodney G. Downey and Dimitrios M. Thilikos. “Confronting intractability via parameters”. In: *Computer Sci. Rev.* 5(4), 2011, pages 279–317. DOI: 10.1016/j.cosrev.2011.09.002.

[89] Andrew Drucker. “New limits to classical and quantum instance compression”. In: *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012*. 2012, pages 609–618. DOI: 10.1109/FOCS.2012.71.

[90] Jenő Egerváry. “Matrixok kombinatorius tulajdonságairól (On combinatorial properties of matrices)”. Hungarian. In: *Matematikai és Fizikai Lapok* 38, 1931. English translation by Harold William Kuhn, pages 16–28.

[91] Frank van den Eijkhof, Hans L. Bodlaender, and Arie M. C. A. Koster. “Safe reduction rules for weighted treewidth”. In: *Algorithmica* 47(2), 2007, pages 139–158. DOI: 10.1007/s00453-006-1226-x.

[92] Friedrich Eisenbrand and Fabrizio Grandoni. “On the complexity of fixed parameter clique and dominating set”. In: *Theor. Comput. Sci.* 326(1-3), 2004, pages 57–67. DOI: 10.1016/j.tcs.2004.05.009.

[93] Vladimir Estivill-Castro, Michael R. Fellows, Michael A. Langston, and Frances A. Rosamond. “FPT is P-Time extremal structure I”. In: *Proceedings of the First Workshop on Algorithms and Complexity in Durham, ACiD 2005*. Volume 4 of Texts in Algorithmics. 2005, pages 1–41. CiteSeer ID: 10.1.1.89.6749.

[94] Michael R. Fellows. “Blow-ups, win/win’s, and crown rules: Some new directions in FPT”. In: *Proceedings of the 29th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2003*. Volume 2880 of LNCS. 2003, pages 1–12. DOI: 10.1007/978-3-540-39890-5_1.

[95] Michael R. Fellows. “The lost continent of polynomial time: Preprocessing and kernelization”. In: *Proceedings of the Second International Workshop on Parameterized and Exact Computation, IWPEC 2006*. Volume 4169 of LNCS. 2006, pages 276–277. DOI: 10.1007/11847250_25.

[96] Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances A. Rosamond, Saket Saurabh, Stefan Szeider, and Carsten Thomassen. “On the complexity of some colorful problems parameterized by treewidth”. In: *Inf. Comput.* 209(2), 2011, pages 143–153. DOI: 10.1016/j.ic.2010.11.026.

[97] Michael R. Fellows, Serge Gaspers, and Frances A. Rosamond. “Parameterizing by the number of numbers”. In: *Theory Comput. Syst.* 50(4), 2012, pages 675–693. DOI: 10.1007/s00224-011-9367-y.

[98] Michael R. Fellows, Danny Hermelin, and Frances A. Rosamond. “Well quasi orders in subclasses of bounded treewidth graphs and their algorithmic applications”. In: *Algorithmica* 64(1), 2012, pages 3–18. DOI: 10.1007/s00453-011-9545-y.

[99] Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. “On the parameterized complexity of multiple-interval graph problems”. In: *Theor. Comput. Sci.* 410(1), 2009, pages 53–61. DOI: 10.1016/j.tcs.2008.09.065.

[100] Michael R. Fellows, Bart M. P. Jansen, Daniel Lokshtanov, Frances A. Rosamond, and Saket Saurabh. “Determining the winner of a Dodgson election is hard”. In: *Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010*. Volume 8 of LIPIcs. 2010, pages 459–468. DOI: 10.4230/LIPIcs.FSTTCS.2010.459.

[101] Michael R. Fellows, Bart M. P. Jansen, and Frances A. Rosamond. “Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity”. In: *European J. Combin.* 34(3), 2013, pages 541–566. DOI: 10.1016/j.ejc.2012.04.008.

[102] Michael R. Fellows and Michael A. Langston. “On search, decision and the efficiency of polynomial-time algorithms (extended abstract)”. In: *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*. 1989, pages 501–512. DOI: 10.1145/73007.73055.

[103] 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”. In: *Theory Comput. Syst.* 45(4), 2009, pages 822–848. DOI: 10.1007/s00224-009-9167-9.

[104] Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Frances A. Rosamond, and Saket Saurabh. “Graph layout problems parameterized by vertex cover”. In: *Proceedings of the 19th International Symposium on Algorithms and Computation, ISAAC 2008*. Volume 5369 of LNCS. 2008, pages 294–305. DOI: 10.1007/978-3-540-92182-0_28.

[105] Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Daniel Raible, Saket Saurabh, and Yngve Villanger. “Kernel(s) for problems with no kernel: On out-trees with many leaves”. In: *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009*. Volume 3 of LIPIcs. 2009, pages 421–432. DOI: 10.4230/LIPIcs.STACS.2009.1843.

[106] Jirí Fiala, Petr A. Golovach, and Jan Kratochvíl. “Parameterized complexity of coloring problems: Treewidth versus vertex cover”. In: *Theor. Comput. Sci.* 412(23), 2011, pages 2513–2523. DOI: 10.1016/j.tcs.2010.10.043.

[107] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer-Verlag New York, Inc., 2006.

[108] Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. "Intractability of clique-width parameterizations". In: *SIAM J. Comput.* 39(5), 2010, pages 1941–1956. DOI: 10.1137/080742270.

[109] Fedor V. Fomin, Bart M. P. Jansen, and Michał Pilipczuk. "Preprocessing subgraph and minor problems: When does a small vertex cover help?" In: *Proceedings of the 7th International Symposium on Parameterized and Exact Computation, IPEC 2012.* Volume 7535 of LNCS. 2012, pages 97–108. DOI: 10.1007/978-3-642-33293-7_11.

[110] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, Geevarghese Philip, and Saket Saurabh. "Hitting forbidden minors: Approximation and kernelization". In: *Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011.* Volume 9 of LIPIcs. 2011, pages 189–200. DOI: 10.4230/LIPIcs.STACS.2011.189.

[111] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. "Planar $\mathcal{F}$-Deletion: Approximation, kernelization and optimal FPT algorithms". In: *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012.* 2012, pages 470–479. DOI: 10.1109/FOCS.2012.62.

[112] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. "Bidimensionality and kernels". In: *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010.* 2010, pages 503–510. ACM ID: 1873644.

[113] Lance Fortnow and Rahul Santhanam. "Infeasibility of instance compression and succinct PCPs for NP". In: *J. Comput. Syst. Sci.* 77(1), 2011, pages 91–106. DOI: 10.1016/j.jcss.2010.06.007.

[114] Markus Frick and Martin Grohe. "The complexity of first-order and monadic second-order logic revisited". In: *Ann. Pure Appl. Logic* 130(1-3), 2004, pages 3–31. DOI: 10.1016/j.apal.2004.01.007.

[115] Harold N. Gabow, Shachindra N. Maheswari, and Leon J. Osterweil. "On two problems in the generation of program test paths". In: *IEEE Trans. Software Eng.* 2(3), 1976, pages 227–231. DOI: 10.1109/TSE.1976.233819.

[116] Robert Ganian. "Twin-cover: Beyond vertex cover in parameterized algorithmics". In: *Proceedings of the 6th International Symposium on Parameterized and Exact Computation, IPEC 2011.* Volume 7112 of LNCS. 2011, pages 259–271. DOI: 10.1007/978-3-642-28050-4_21.

[117] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, 1979.

[118] Oded Goldreich. *Computational Complexity - A Conceptual Perspective.* Cambridge University Press, 2008, pages 1–606.

[119] Oded Goldreich. “On promise problems: A survey”. In: *Theoretical Computer Science, Essays in Memory of Shimon Even.* Volume 3895 of LNCS. 2006, pages 254–290. DOI: 10.1007/11685654_12.

[120] Jerrold R. Griggs, Daniel J. Kleitman, and Aditya Shastri. “Spanning trees with many leaves in cubic graphs”. In: *J. Graph Theory* 13, 1989, pages 669–695. DOI: 10.1002/jgt.3190130604.

[121] Martin Grohe. “Logic, graphs, and algorithms”. In: *Logic and Automata: History and Perspectives.* 2007, pages 357–422. CiteSeer ID: 10.1.1.113.9457.

[122] Martin Grötschel, Lászlo Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Volume 2. Algorithms and Combinatorics. Springer, 1988.

[123] Sylvain Guillemot, Frédéric Havet, Christophe Paul, and Anthony Perez. “On the (non-)existence of polynomial kernels for P_ℓ-free edge modification problems”. In: *Algorithmica*, 2012. Online First. DOI: 10.1007/s00453-012-9619-5.

[124] Jiong Guo, Rolf Niedermeier, and Sebastian Wernicke. “Parameterized complexity of vertex cover variants”. In: *Theory Comput. Syst.* 41(3), 2007, pages 501–520. DOI: 10.1007/s00224-007-1309-3.

[125] Frank Gurski and Egon Wanke. “Vertex disjoint paths on clique-width bounded graphs”. In: *Theor. Comput. Sci.* 359(1-3), 2006, pages 188–199. DOI: 10.1016/j.tcs.2006.02.026.

[126] Gregory Gutin, Leo van Iersel, Matthias Mnich, and Anders Yeo. “Every ternary permutation constraint satisfaction problem parameterized above average has a kernel with a quadratic number of variables”. In: *J. Comput. Syst. Sci.* 78(1), 2012, pages 151–163. DOI: 10.1016/j.jcss.2011.01.004.

[127] Gregory Gutin, Eun Jung Kim, Michael Lampis, and Valia Mitsou. “Vertex cover problem parameterized above and below tight bounds”. In: *Theory Comput. Syst.* 48(2), 2011, pages 402–410. DOI: 10.1007/s00224-010-9262-y.

[128] Paul R. Halmos. “How to write mathematics”. In: *L’Enseignement Mathématique* 16, 1970, pages 123–152. CiteSeer ID: 10.1.1.188.8910.

[129] Frank Harary. *Graph Theory.* Addison-Wesley series in mathematics. Perseus Books, 1994.

[130] Danny Harnik and Moni Naor. “On the compressibility of NP instances and cryptographic applications”. In: *SIAM J. Comput.* 39(5), 2010, pages 1667–1713. DOI: 10.1137/060668092.

[131] Pinar Heggernes, Pim van ’t Hof, Bart M. P. Jansen, Stefan Kratsch, and Yngve Villanger. “Parameterized complexity of vertex deletion into perfect graph classes”. In: *Theor. Comput. Sci.* 2012. Online First. DOI: 10.1007/978-3-642-22953-4_21.

[132] Michael Held and Richard M. Karp. “A dynamic programming approach to sequencing problems”. In: *J. Soc. Ind. Appl. Math.* 10(1), 1962, pages 196–210. DOI: 10.1137/0110015.

[133] Fritz Henglein and Harry G. Mairson. “The complexity of type inference for higher-order typed lambda calculi”. In: *J. Funct. Program.* 4(4), 1994, pages 435–477. DOI: 10.1017/S0956796800001143.

[134] Danny Hermelin, Stefan Kratsch, Karolina Sołtys, Magnus Wahlström, and Xi Wu. *Hierarchies of Inefficient Kernelizability.* 2011. arXiv:1110.0976.

[135] Danny Hermelin and Xi Wu. “Weak compositions and their applications to polynomial lower bounds for kernelization”. In: *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012.* 2012, pages 104–113. ACM ID: 2095125.

[136] Chính T. Hoàng, Marcin Kaminski, Vadim V. Lozin, Joe Sawada, and Xiao Shu. “Deciding k-colorability of P_5-free graphs in polynomial time”. In: *Algorithmica* 57(1), 2010, pages 74–81. DOI: 10.1007/s00453-008-9197-8.

[137] Dorit S. Hochbaum. “Approximating covering and packing problems: Set cover, vertex cover, independent set, and related problems”. In: *Approximation Algorithms for NP-hard Problems*, 1997, pages 94–143. ACM ID: 241941.

[138] John E. Hopcroft and Richard M. Karp. “An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs”. In: *SIAM J. Comput.* 2(4), 1973, pages 225–231. DOI: 10.1137/0202019.

[139] Russell Impagliazzo and Ramamohan Paturi. “On the complexity of k-sat”. In: *J. Comput. Syst. Sci.* 62(2), 2001, pages 367–375. DOI: 10.1006/jcss.2000.1727.

[140] Kyriaki Ioannidou, George B. Mertzios, and Stavros D. Nikolopoulos. “The longest path problem has a polynomial solution on interval graphs”. In: *Algorithmica* 61, 2 2011, pages 320–341. DOI: 10.1007/s00453-010-9411-3.

[141] Guy Jacobson. “Space-efficient static trees and graphs”. In: *Proceedings of the 30th Annual Symposium on Foundations of Computer Science, FOCS 1989.* 1989, pages 549–554. DOI: 10.1109/SFCS.1989.63533.

[142] Bart M. P. Jansen. *Kernelization for Maximum Leaf Spanning Tree with Positive Vertex Weights.* Technical report UU-CS-2009-027. Department of Information and Computing Sciences, Utrecht University, 2010. UU ID: UU-CS-2009-027.

[143] Bart M. P. Jansen. “Kernelization for maximum leaf spanning tree with positive vertex weights”. In: *J. Graph Algorithms Appl.* 16(4), 2012, pages 811–846. DOI: 10.7155/jgaa.00279.

[144] Bart M. P. Jansen. “Polynomial kernels for hard problems on disk graphs”. In: *Proceedings of the 12th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2010*. Volume 6139 of LNCS. 2010, pages 310–321. DOI: 10.1007/978-3-642-13731-0_30.

[145] Bart M. P. Jansen and Hans L. Bodlaender. “Vertex cover kernelization revisited”. In: *Theory Comput. Syst.* 2012. Online First, pages 1–37. DOI: 10.1007/s00224-012-9393-4.

[146] Bart M. P. Jansen and Hans L. Bodlaender. “Vertex cover kernelization revisited: Upper and lower bounds for a refined parameter”. In: *Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011*. Volume 9 of LIPIcs. 2011, pages 177–188. DOI: 10.4230/LIPIcs.STACS.2011.177.

[147] Bart M. P. Jansen and Stefan Kratsch. “Data reduction for graph coloring problems”. In: *Proceedings of the 18th International Symposium on Fundamentals of Computation Theory, FCT 2011*. Volume 6914 of LNCS. 2011, pages 90–101. DOI: 10.1007/978-3-642-22953-4_8.

[148] Bart M. P. Jansen and Stefan Kratsch. “On polynomial kernels for structural parameterizations of odd cycle transversal”. In: *Proceedings of the 6th International Symposium on Parameterized and Exact Computation, IPEC 2011*. Volume 7112 of LNCS. 2011, pages 132–144. DOI: 10.1007/978-3-642-28050-4_11.

[149] Klaus Jansen and Petra Scheffler. “Generalized coloring for tree-like graphs”. In: *Discrete Appl. Math.* 75(2), 1997, pages 135–155. DOI: 10.1016/S0166-218X(96)00085-6.

[150] Viggo Kann. “Polynomially bounded minimization problems that are hard to approximate”. In: *Nord. J. Comput.* 1(3), 1994, pages 317–331. ACM ID: 640172.

[151] Petteri Kaski, Mikko Koivisto, and Jesper Nederlof. “Homomorphic hashing for sparse coefficient extraction”. In: *Proceedings of the 7th International Symposium on Parameterized and Exact Computation, IPEC 2012*. Volume 7535 of LNCS. 2012, pages 147–158. DOI: 10.1007/978-3-642-33293-7_15.

[152] Ken-ichi Kawarabayashi, Bojan Mohar, and Bruce A. Reed. “A simpler linear time algorithm for embedding graphs into an arbitrary surface and the genus of graphs of bounded tree-width”. In: *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008*. 2008, pages 771–780. DOI: 10.1109/FOCS.2008.53.

[153] Daniel J. Kleitman and Douglas B. West. “Spanning trees with many leaves”. In: *SIAM J. Discret. Math.* 4(1), 1991, pages 99–106. DOI: 10.1137/0404010.

[154] Ton Kloks and Dieter Kratsch. "Treewidth of chordal bipartite graphs". In: *J. Algorithms* 19(2), 1995, pages 266–281. DOI: 10.1006/jagm.1995.1037.

[155] Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. "Divide-and-color". In: *Proceedings of the 32nd International on Graph-Theoretic Concepts in Computer Science, WG 2006.* Volume 4271 of LNCS. 2006, pages 58–67. DOI: 10.1007/11917496_6.

[156] Donald E. Knuth. *Axioms and hulls.* Springer-Verlag, 1992.

[157] Daniel Kobler and Udi Rotics. "Edge dominating set and colorings on graphs with fixed clique-width". In: *Discrete Appl. Math.* 126(2-3), 2003, pages 197–221. DOI: 10.1016/S0166-218X(02)00198-1.

[158] Petr Kolman and Ondrej Pangrác. "On the complexity of paths avoiding forbidden pairs". In: *Discrete Appl. Math.* 157(13), 2009, pages 2871–2876. DOI: 10.1016/j.dam.2009.03.018.

[159] Christian Komusiewicz and Rolf Niedermeier. "New races in parameterized algorithmics". In: *Proceedings of the 37th International Symposium on Mathematical Foundations of Computer Science, MFCS 2012.* Volume 7464 of LNCS. 2012, pages 19–30. DOI: 10.1007/978-3-642-32589-2_2.

[160] Dénes Kőnig. "Graphok és matrixok (Graphs and matrices)". Hungarian. In: *Matematikai és Fizikai Lapok* 38, 1931, pages 116–119.

[161] Jakub Kovác. *Complexity of the path avoiding forbidden pairs problem revisited.* 2013. arXiv:1111.3996.

[162] Jan Kratochvíl. "Precoloring extension with fixed color bound". In: *Acta Math. Univ. Comenian.* 62(2), 1993, pages 139–153. CiteSeer ID: 10.1.1.37.8238.

[163] Stefan Kratsch. "Co-nondeterminism in compositions: A kernelization lower bound for a Ramsey-type problem". In: *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012.* 2012, pages 114–122. arXiv:1107.3704.

[164] Stefan Kratsch. "Co-nondeterminism in compositions: A kernelization lower bound for a Ramsey-type problem". In: *ACM Trans. Algorithms*, 2012. To appear. arXiv:1107.3704.

[165] Stefan Kratsch, Marcin Pilipczuk, Ashutosh Rai, and Venkatesh Raman. "Kernel lower bounds using co-nondeterminism: Finding induced hereditary subgraphs". In: *Proceedings of the 13th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2012.* Volume 7357 of LNCS. 2012, pages 364–375. DOI: 10.1007/978-3-642-31155-0_32.

[166] Stefan Kratsch and Magnus Wahlström. "Compression via matroids: A randomized polynomial kernel for odd cycle transversal". In: *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012.* 2012, pages 94–103. ACM ID: 2095124.

[167] Stefan Kratsch and Magnus Wahlström. "Representative sets and irrelevant vertices: New tools for kernelization". In: *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012*. 2012, pages 450–459. DOI: 10.1109/FOCS.2012.46.

[168] Mukkai S. Krishnamoorthy. "An NP-hard problem in bipartite graphs". In: *SIGACT News* 7(1), 1975, pages 26–26. DOI: 10.1145/990518.990521.

[169] Kazimierz Kuratowski. "Sur le problème des courbes gauches en topologie". In: *Fund. Math.* 15, 1930, pages 271–283.

[170] Jens Lagergren and Stefan Arnborg. "Finding minimal forbidden minors using a finite congruence". In: *Proceedings of the 18th International Colloquium on Automata, Languages and Programming, ICALP 1991*. Volume 510 of LNCS. 1991, pages 532–543. DOI: 10.1007/3-540-54233-7_161.

[171] Michael Lampis. "A kernel of order $2k - c\log k$ for vertex cover". In: *Inf. Process. Lett.* 111(23-24), 2011, pages 1089–1091. DOI: 10.1016/j.ipl.2011.09.003.

[172] Michael Lampis. "Algorithmic meta-theorems for restrictions of treewidth". In: *Algorithmica* 64(1), 2012, pages 19–37. DOI: 10.1007/s00453-011-9554-x.

[173] Stefan Langerman and Pat Morin. "Covering things with things". In: *Discrete Comput. Geom.* 33(4), 2005, pages 717–729. DOI: 10.1007/s00454-004-1108-4.

[174] Orna Lichtenstein and Amir Pnueli. "Checking that finite state concurrent programs satisfy their linear specification". In: *Proceedings of the 12th Annual ACM Symposium on Principles of Programming Languages*. 1985, pages 97–107. DOI: 10.1145/318593.318622.

[175] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. "Known algorithms on graphs on bounded treewidth are probably optimal". In: *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011*. 2011, pages 777–789. arXiv:1007.5450.

[176] James F. Lynch. "The equivalence of theorem proving and the interconnection problem". In: *SIGDA Newsl.* 5(3), 1975, pages 31–36. DOI: 10.1145/1061425.1061430.

[177] Dániel Marx. "Parameterized coloring problems on chordal graphs". In: *Theor. Comput. Sci.* 351(3), 2006, pages 407–424. DOI: 10.1016/j.tcs.2005.10.008.

[178] Sounaka Mishra, Venkatesh Raman, Saket Saurabh, and Somnath Sikdar. "König deletion sets and vertex covers above the matching size". In: *Proceedings of the 19th International Symposium on Algorithms and Computation, ISAAC 2008*. Volume 5369 of LNCS. 2008, pages 836–847. DOI: 10.1007/978-3-540-92182-0_73.

[179] Sounaka Mishra, Venkatesh Raman, Saket Saurabh, Somnath Sikdar, and C. R. Subramanian. “The complexity of finding subgraphs whose matching number equals the vertex cover number”. In: *Proceedings of the 18th International Symposium on Algorithms and Computation, ISAAC 2007.* Volume 4835 of LNCS. 2007, pages 268–279. DOI: 10.1007/978-3-540-77120-3_25.

[180] Sounaka Mishra, Venkatesh Raman, Saket Saurabh, Somnath Sikdar, and C. R. Subramanian. “The complexity of König subgraph problems and above-guarantee vertex cover”. In: *Algorithmica* 61(4), 2011, pages 857–881. DOI: 10.1007/s00453-010-9412-2.

[181] Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, Saket Saurabh, and Somnath Sikdar. “FPT algorithms for connected feedback vertex set”. In: *J. Comb. Optim.* 24(2), 2012, pages 131–146. DOI: 10.1007/s10878-011-9394-2.

[182] Rolf H. Möhring. “Triangulating graphs without asteroidal triples”. In: *Discrete Appl. Math.* 64(3), 1996, pages 281–287. DOI: 10.1016/0166-218X(95)00095-9.

[183] Burkhard Monien and Ivan Hal Sudborough. “Min cut is NP-complete for edge weighted treees”. In: *Theor. Comput. Sci.* 58(1-3), 1988, pages 209–229. DOI: 10.1016/0304-3975(88)90028-X.

[184] Haiko Müller. “Hamiltonian circuits in chordal bipartite graphs”. In: *Discrete Math.* 156(1-3), 1996, pages 291–298. DOI: 10.1016/0012-365X(95)00057-4.

[185] Sridhar Natarajan and Alan P. Sprague. “Disjoint paths in circular arc graphs”. In: *Nordic J. of Computing* 3(3), 1996, pages 256–270. ACM ID: 642154.

[186] Jesper Nederlof, Erik Jan van Leeuwen, and Ruben van der Zwaan. “Reducing a target interval to a few exact queries”. In: *Proceedings of the 37th International Symposium on Mathematical Foundations of Computer Science, MFCS 2012.* Volume 7464 of LNCS. 2012, pages 718–727. DOI: 10.1007/978-3-642-32589-2_62.

[187] George L. Nemhauser and Leslie E. Trotter. “Vertex packings: Structural properties and algorithms.” In: *Math. Program.* 8(1), 1975, pages 232–248. DOI: 10.1007/BF01580444.

[188] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press, 2006.

[189] Rolf Niedermeier. “Reflections on multivariate algorithmics and problem parameterization”. In: *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010.* Volume 5 of LIPIcs. 2010, pages 17–32. DOI: 10.4230/LIPIcs.STACS.2010.2495.

[190] Rolf Niedermeier and Peter Rossmanith. “On efficient fixed-parameter algorithms for weighted vertex cover”. In: *J. Algorithms* 47(2), 2003, pages 63–77. DOI: 10.1016/S0196-6774(03)00005-1.

[191] Kristian G. Olesen and Anders L. Madsen. “Maximal prime subgraph decomposition of Bayesian networks”. In: *IEEE T. Syst. Man Cy. B* 32(1), 2002, pages 21–31. DOI: 10.1109/3477.979956.

[192] *Parameterized Complexity Wiki: Table of FPT races.* URL: http://fpt.wikidot.com/fpt-races.

[193] *Parameterized Complexity Newsletter of November 2010.* URL: http://fpt.wikidot.com/fpt-news:the-parameterized-complexity-newsletter.

[194] *Parameterized Complexity Newsletter of August 2012.* URL: http://fpt.wikidot.com/fpt-news:the-parameterized-complexity-newsletter.

[195] Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. “Paths, flowers and vertex cover”. In: *Proceedings of the 19th Annual European Symposium on Algorithms, ESA 2011.* Volume 6942 of LNCS. 2011, pages 382–393. DOI: 10.1007/978-3-642-23719-5_33.

[196] Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. “Finding odd cycle transversals”. In: *Oper. Res. Lett.* 32(4), 2004, pages 299–301. DOI: 10.1016/j.orl.2003.10.009.

[197] Neil Robertson and Paul D. Seymour. “Graph minors: A survey”. In: *Surveys in Combinatorics.* 1985, pages 153–171.

[198] Neil Robertson and Paul D. Seymour. “Graph minors. V. Excluding a planar graph”. In: *J. Comb. Theory, Ser. B* 41(1), 1986, pages 92 –114. DOI: 10.1016/0095-8956(86)90030-4.

[199] Neil Robertson and Paul D. Seymour. “Graph minors. XIII. The disjoint paths problem”. In: *J. Comb. Theory, Ser. B* 63(1), 1995, pages 65–110. DOI: 10.1006/jctb.1995.1006.

[200] Neil Robertson and Paul D. Seymour. “Graph minors. XX. Wagner’s conjecture”. In: *J. Comb. Theory, Ser. B* 92(2), 2004, pages 325–357. DOI: 10.1016/j.jctb.2004.08.001.

[201] Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. “Dynamic programming on tree decompositions using generalised fast subset convolution”. In: *Proceedings of the 17th Annual European Symposium on Algorithms, ESA 2009.* Volume 5757 of LNCS. 2009, pages 566–577. DOI: 10.1007/978-3-642-04128-0_51.

[202] Róbert Sasák. “Comparing 17 graph parameters”. Master’s thesis. University of Bergen, 2010. URL: http://hdl.handle.net/1956/4329.

[203] Alexander Schrijver. *Combinatorial Optimization. Polyhedra and Efficiency.* Springer, 2003.

[204] Jacob Scott, Trey Ideker, Richard M. Karp, and Roded Sharan. "Efficient algorithms for detecting signaling pathways in protein interaction networks". In: *J. Comput. Biol.* 13(2), 2006, pages 133–144. DOI: 10.1089/cmb.2006.13.133.

[205] Arezou Soleimanfallah and Anders Yeo. "A kernel of order $2k - c$ for vertex cover". In: *Discrete Math.* 311(10-11), 2011, pages 892–895. DOI: 10.1016/j.disc.2011.02.014.

[206] Ulrike Stege. "The impact of parameterized complexity to interdisciplinary problem solving". In: *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday.* Volume 7370 of LNCS. 2012, pages 56–68. DOI: 10.1007/978-3-642-30891-8_5.

[207] Stéphan Thomassé. "A $4k^2$ kernel for feedback vertex set". In: *ACM Trans. Algorithms* 6(2), 2010. DOI: 10.1145/1721837.1721848.

[208] Magnus Wahlström. "Abusing the Tutte matrix: An algebraic instance compression for the k-set-cycle problem". In: *Proceedings of the 30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013.* Volume 20 of LIPIcs. 2013, pages 341–352. DOI: 10.4230/LIPIcs.STACS.2013.341.

[209] Karsten Weihe. "Covering trains by stations or the power of data reduction". In: *Algorithms and Experiments (ALEX98).* 1998, pages 1–8. CITESEER ID: 10.1.1.57.2173.

[210] Karsten Weihe. "On the differences between "practical" and "applied"". In: *Proceedings of the 4th International Workshop on Algorithm Engineering, WAE 2000.* Volume 1982 of LNCS. 2000, pages 1–10. DOI: 10.1007/3-540-44691-5_1.

[211] Robert Wilson. *Graphs, Colourings and the Four-colour Theorem.* Oxford Science Publications, 2002.

[212] Chee-Keng Yap. "Some consequences of non-uniform conditions on uniform classes". In: *Theor. Comput. Sci.* 26, 1983, pages 287–300. DOI: 10.1016/0304-3975(83)90020-8.

[213] Jennifer Zito. "The structure and maximum number of maximum independent sets in trees". In: *J. Graph Theory* 15(2), 1991, pages 207–221. DOI: 10.1002/jgt.3190150208.

Subject Index

Author Index

Nederlandse Samenvatting

Wanneer je een Sudoku puzzel oplost, de beste zet probeert te vinden in een schaakpartij, of een tentamenrooster aan het maken bent voor studenten met verschillende vakkenpakketten, loont het vaak de moeite om te kijken of er mogelijkheden zijn waarvan gemakkelijk aangetoond kan worden dat ze niet tot de gewenste oplossing leiden. Deze mogelijkheden kunnen buiten beschouwing gelaten worden, waarna het probleem gemakkelijker wordt.

Het “voorbewerken” van een probleemstelling is niet alleen nuttig voor menselijk redeneren. Wanneer een computer wordt gebruikt voor het vinden van een oplossing, kan vaak rekentijd worden bespaard door te beginnen met een fase waarin, op basis van simpele argumenten, delen van de puzzel worden versimpeld. Er wordt dan een eenvoudigere formulering van hetzelfde probleem gevonden. Na het berekenen van een oplossing voor het versimpelde probleem, wordt die terugvertaald naar een antwoord op het originele vraagstuk.

In de informatica worden dergelijke voorbewerkingstechnieken al decennia lang met succes toegepast. Vanuit theoretisch oogpunt is het echter nog altijd de vraag waarom deze technieken zulke grote snelheidswinsten opleveren. Ze zijn namelijk ook geregeld toepasbaar op problemen die als *notoir lastig* zijn geclassificeerd. Van dit soort problemen gelooft men op theoretische gronden dat ze in het algemeen niet efficiënt oplosbaar zijn. Het is daarom een van de grote mysteries van de informatica waarom voorbewerkingstechnieken ervoor zorgen dat de instanties van notoir lastige problemen waarmee men in de praktijk te maken krijgt, toch snel door computers kunnen worden opgelost.

Dit proefschrift gaat over de theoretische analyse van voorbewerkingstechnieken om de hoeveelheid gegevens en mogelijkheden in een probleemstelling te verminderen, zonder de oplossing op essentiële wijze aan te tasten. In de beginjaren van de informatica waren er geen wiskundige technieken voorhanden om dit soort analyses mee te bewerkstelligen. De opkomst van de geparameteriseerde algoritmiek eind jaren negentig heeft de benodigde gereedschappen verschaft om een rigoureuze studie van dit onderwerp te maken. De crux van geparameteriseerde analyse ligt in het feit dat, naast de totale grootte van de invoer, een tweede *parameter* wordt onderscheiden. Deze parameter wordt gebruikt om de invloed van verschillende aspecten van het probleem op de complexiteit te bepalen. Voor een Sudoku puzzel kan de parameter bijvoorbeeld het aantal al ingevulde vakjes zijn, of het aantal nog volledig lege rijen; voor een tentamenroosteringsprobleem kan de parameter het aantal vakken meten, of het aantal studenten dat minstens twee tentamens moet maken.

Een efficiënte methode om een beslisprobleem te versimpelen wordt geformaliseerd door het wiskundige concept van een *reductie tot een probleemkern*: een algoritme dat een instantie van een geparameteriseerd beslisprobleem als invoer krijgt en een instantie van hetzelfde probleem als uitvoer geeft, zodat aan de volgende eisen wordt voldaan: (1) de grootte en parameter van de uitvoer worden begrensd door een berekenbare functie van de parameter van de invoer, (2) het aantal stappen dat het algoritme nodig heeft is begrensd door een polynoom in de lengte van de invoer, en (3) de uitvoer heeft hetzelfde antwoord als de invoer (dwz. als de invoer als antwoord JA heeft dan heeft de uitvoer dat ook, en vice versa). De kern heeft grootte $f(k)$ als een invoer met parameterwaarde k wordt gereduceerd naar een uitvoer waarvan de grootte en parameterwaarde hoogstens $f(k)$ zijn. Let op dat de grootte van de uitvoer alleen afhangt van de waarde van de parameter van de invoer, en niet van de totale grootte van de invoer instantie.

Het concept van reductie tot een probleemkern heeft de potentie om het succes van voorbewerkingstechnieken te kunnen verklaren. Als in de praktijk voorkomende instanties van notoir lastige problemen parameters hebben met kleine waarden, zodanig dat de resulterende geparameteriseerde problemen kunnen worden gereduceerd tot een probleemkern, dan verklaart dat waarom de technieken voor gegevensreductie zo effectief zijn. Zulk gedrag zou niet in tegenspraak zijn met het geloof dat notoir lastige problemen in het algemeen niet efficiënt kunnen worden opgelost. Immers, niet *alle* instanties kunnen effectief worden versimpeld, maar alleen de instanties met een *kleine parameterwaarde*.

In het algemeen verwachten we dat de invoer voor computationele problemen uit de dagelijkse praktijk structurele regulariteiten bevat, omdat deze invoer voortkomt uit processen die zelf ook gebonden worden door computationele beperkingen. Deze samenhang noemen we de *ecologie van parameters*. Aangezien we op voorhand niet weten op welke manier deze regulariteiten tot uiting komen, zijn we geïnteresseerd in het bepalen van parameterisaties van een probleem die tot een probleemkern gereduceerd kunnen worden. Daarnaast is het van belang om parameters te identificeren waarmee een probleem bewijsbaar niet tot een kern kan worden gereduceerd. We formuleren daarom een onderzoeksprogramma dat we het *parameter ecologie programma* noemen. Het heeft als doelstelling om voor fundamentele problemen, met een zo breed mogelijk scala aan parameters, te onderzoeken welke parameters een reductie tot een probleemkern toestaan. Ook zijn we geïnteresseerd in de grootte van de overblijvende kern, uitgedrukt als functie van de parameterwaarde: is de grootte polynomiaal, of exponentieel?

We richten ons bij dit onderzoek op fundamentele vraagstukken over grafen. Een graaf is een wiskundige structuur die van een verzameling objecten, *knopen* genaamd, aangeeft welke paren knopen verbonden zijn door een lijn of *kant*. Grafen worden veel gebruikt om belangrijke optimaliseringsproblemen te formuleren, bijvoorbeeld voor het modelleren van een wegennetwerk. De bijbehorende graaf heeft een knoop voor ieder kruispunt, met een kant tussen twee kruispunten als er een directe weg tussen loopt. Grafen komen ook op een natuurlijke manier voort uit sociale netwerken: hierbij is er een knoop voor iedere persoon, met een kant tussen personen die bevriend zijn. Een fundamenteel graafprobleem bestaat uit het zoeken van een zo groot mogelijke *kliek*: een verzameling knopen met de eigenschap dat ieder paar uit de verzameling verbonden is door een kant. Een kliek in een graaf voortkomend uit een sociaal netwerk is dus, zoals je verwacht, een groep mensen die onderling allemaal met elkaar bevriend zijn.

Het doel van dit proefschrift is het bestuderen van fundamentele graafproblemen binnen het parameter ecologie programma: we identificeren structurele parameters van grafen waarmee reducties tot een probleemkern bestaan. De theoretische resultaten over het wel of niet bestaan van zulke reducties worden vervolgens gebruikt als verklaringen voor het geobserveerde succes van voorbewerkingstechieken voor notoir lastige problemen. Bij het onderzoeken van probleemkernen streven we tevens naar het vinden van nieuwe technieken voor gegevensreductie, die kunnen worden toegepast om fundamentele graafproblemen efficiënter op te lossen. Onze bijdragen in dit proefschrift zijn als volgt.

Deel I van het proefschrift is gewijd aan een uitgebreide introductie waarin we de gebruikte theoretische concepten definiëren. In het eerste hoofdstuk geven we een informele inleiding op het onderwerp. In hoofdstuk 2 formuleren we het parameter ecologie programma, waarbij we veel aandacht schenken aan de onderlinge relaties tussen verschillende parameters van grafen. We beschouwen de ruimte met graafparameters als een hiërarchie, met als doel te bepalen waar de grenzen in deze hiërarchie liggen wat betreft het bestaan van reducties naar een probleemkern van polynomiale grootte.

Het bestuderen van probleemkernen volgens het parameter ecologie programma vereist wat wiskundig gereedschap. In deel II ontwikkelen we twee soorten gereedschap om dit onderzoek te faciliteren. Hoofdstuk 3 introduceert de *kruis-compositie*. Hierbij wordt een reeks van invoeren van een klassiek probleem efficiënt gecombineerd tot een enkele instantie van een geparameteriseerd probleem, zodat de combinatie de logische OF van de invoeren uitdrukt. Middels een kruis-compositie kan onder bepaalde voorwaarden worden afgeleid dat een geparameteriseerd probleem niet kan worden gereduceerd tot een kern van polynomiale grootte. De techniek geeft dus ondergrenzen op de groottes van probleemkernen. In hoofdstuk 4 ontwikkelen we stellingen waarmee juist bovengrenzen op de groottes van probleemkernen kunnen worden afgeleid. We bestuderen graafproblemen gerelateerd aan het zoeken van geïnduceerde deelgrafen met bepaalde eigenschappen, waarbij we als parameter de grootte van een gegeven knoop-bedekking gebruiken. Een knoop-bedekking in een graaf is een verzameling knopen zodat iedere kant grenst aan minstens èèn knoop in de verzameling. We presenteren drie algemene stellingen die aangeven onder welke condities een simpele reductieregel leidt tot probleemkernen van bewijsbaar klein formaat.

Het hart van het proefschrift wordt gevormd door deel III. Elk van de vier hoofdstukken in dit deel heeft een specifiek fundamenteel graafprobleem als focus, waarvan de grenzen op de mogelijkheid tot effectief voorbewerken worden opgezocht. Als eerste kijken we naar het probleem KNOOP-BEDEKKING, wat in het verleden al regelmatig is bestudeerd. We geven twee bijdragen aan het onderwerp: een bovengrens en een ondergrens. We nemen als parameter hoeveel knopen er nodig zijn om alle cykels in de graaf te doorbreken. Door nieuwe reductieregels te combineren met een nauwkeurige analyse op basis van een graaftheoretisch argument over uiterste situaties, kunnen we het probleem herleiden tot een kern met $\mathcal{O}(k^3)$ knopen, waarbij k de waarde van de parameter is. Voor de ondergrens beschouwen we de parameter die het minimum aantal knopen telt dat verwijderd moet worden om een graaf over te houden die in het vlak kan worden getekend, zodat alle knopen aan de buitenzijde van de tekening grenzen. We geven een kruis-compositie om te laten zien dat dit geparameteriseerde probleem niet te herleiden is tot een kern van polynomiale grootte, tenzij het complexiteitstheoretische vermoeden NP $\not\subseteq$ coNP/poly onjuist blijkt te zijn.

In hoofdstuk 6 bekijken we problemen gerelateerd aan de boombreedte van een graaf; dit is een maat voor de structurele gelijkenis van de graaf met een samenhangende acyclische graaf, ook wel boom genoemd. In het verleden zijn al verschillende reductieregels voor dit probleem ontwikkeld. Uit experimenten is gebleken dat deze regels zeer effectief zijn in het versimpelen van de graaf. Er was echter geen verklaring voor dit succes. Wij tonen aan dat een combinatie van enkele van deze regels bewijsbaar leidt tot een reductie tot een probleemkern voor Boombreedte, waarvan het aantal knopen kubisch is in de grootte van de kleinste knoop-bedekking van de graaf. Daarna gebruiken we als parameter wederom het aantal knopen dat nodig is om alle cykels te verbreken. We ontwikkelen verschillende nieuwe reductieregels, die leiden tot een probleemkern met $\mathcal{O}(k^4)$ knopen voor Boombreedte. We leiden voor dit probleem ook ondergrenzen op de grootte van mogelijke probleemkernen af. We nemen als parameter hoeveel knopen uit de graaf verwijderd moeten worden voordat een enkele kliek overblijft. Door een ingewikkelde kruis-compositie tonen we aan dat Boombreedte met deze parameter niet kan worden gereduceerd tot een kern van polynomiale grootte, tenzij NP $\subseteq$ coNP/poly. Verder laten we zien dat de variant Gewogen Boombreedte, waarbij knopen een gewicht dragen, minder goed voorbewerkt kan worden dan de ongewogen variant: wanneer de grootte van de kleinste knoop-bedekking als parameter wordt gebruikt, bestaat er geen reductie tot een probleemkern van polynomiale grootte tenzij NP $\subseteq$ coNP/poly.

Vervolgens richten we ons in hoofdstuk 7 op kleuringsproblemen in grafen. Hierbij is het, gegeven een graaf en een geheel getal q, de vraag of het mogelijk is om iedere knoop van de graaf èèn van de q mogelijke kleuren toe te kennen, zodat knopen die verbonden zijn met een kant verschillende kleuren dragen. Aangezien het al notoir lastig is om te beslissen of een graaf een geldige kleuring toestaat met drie kleuren, is het gebruik van het aantal kleuren als de parameter niet interessant. We gebruiken daarom structurele eigenschappen van de graaf als parameters. Deze parameters worden gevormd door te meten hoeveel knopen uit een graaf verwijderd moeten worden zodat het resultaat bevat is in een simpele graafklasse $\mathcal{F}$. We identificeren eigenschappen van de klasse $\mathcal{F}$ die het wel of niet bestaan van reducties tot een probleemkern, met de afstand tot $\mathcal{F}$ als parameter, verklaren. De belangrijkste eigenschap blijkt ruwweg te zijn of het niet-bestaan van lijstkleuringen wordt veroorzaakt door een verzameling knopen waarvan de grootte alleen afhangt van het totaal aantal beschikbare kleuren, of juist kan worden veroorzaakt door een deelgebied van de graaf met onbegrensde grootte.

Tot slot beschouwen we in hoofdstuk 8 verbindingsproblemen in grafen die draaien om paden en cykels. We beschouwen bijvoorbeeld het probleem LANG PAD, dat vraagt of er een lang pad is in de graaf dat zichzelf niet kruist. Hoewel de algemene stellingen uit hoofdstuk 4 toepasbaar zijn op veel van deze problemen, laten we zien dat het formaat van de probleemkernen verkleind kan worden door probleemspecifieke reductieregels toe te passen. Deze regels zijn gebaseerd op een nieuwe stelling over koppelingen in bipartiete grafen. De stelling laat zien hoe een bipartiete graaf verkleind kan worden zonder, voor een van de twee helften knopen van de graaf, aan te tasten welke van zijn deelverzamelingen kunnen worden bedekt door een koppeling. We gebruiken deze stelling herhaaldelijk om probleemkernen van polynomiale grootte te herleiden voor de problemen LANG PAD, LANGE CYKEL, HAMILTONIAANS PAD, HAMILTONIAANSE CYKEL, DISJUNCTE PADEN en DISJUNCTE CYKELS, waarbij de minimumgrootte van een kant-bedekking de parameter vormt. Voor de parameter die meet hoeveel knopen verwijderd moeten worden om een cluster graaf te verkrijgen, bereiken we soortgelijke resultaten. Ook passen we onze aanpak aan voor de situatie waarin de parameter uitdrukt hoeveel bladeren er maximaal in een opspannende deelboom van de graaf zitten. We identificeren daarnaast een limiet voor de mogelijkheid tot efficiënt voorbewerken. Het probleem HAMILTONIAANSE CYKEL, met als parameter het aantal knopen dat verwijderd moet worden om een buiten-planaire graaf te verkrijgen, kan niet gereduceerd worden tot een kern van polynomiale grootte tenzij NP $\subseteq$ coNP/poly. Verder doen we onderzoek naar padproblemen waarbij bepaalde paren knopen niet *allebei* bezocht mogen worden. We classificeren de complexiteit van enkele parameterisaties van zulke problemen.

In hoofdstuk 9 trekken we conclusies uit ons onderzoek. We identificeren ook enkele trends die zichtbaar worden in het gedrag van problemen onder verschillende soorten parameterisaties. De belangrijkste resultaten van ons onderzoek kunnen als volgt worden samengevat. De structuur van de invoergraaf heeft een grote invloed op de computationele complexiteit van veel fundamentele graafproblemen. Wanneer een beperkte parameterwaarde een voldoende mate van regulariteit in de graaf afdwingt, kunnen instanties van veel van deze problemen efficiënt worden herleid tot een probleemkern. Voor alle problemen is er echter een haalbaarheidsgrens in de hiërarchie van parameters: parameters onder deze grens staan teveel diversiteit in de graaf toe, waardoor er geen kern van polynomiale grootte herleid kan worden voor de corresponderende parameterisaties. Tijdens het onderzoek naar reducties tot probleemkernen zijn efficiënte reductiestrategieën gevonden die kunnen worden gebruikt om instanties uit de dagelijkse praktijk te verkleinen, waarna ze gemakkelijker kunnen worden opgelost. De theoretische resultaten over het bewijsbare effect van reductieregels zijn potentiële verklaringen voor het empirisch geobserveerde succes van zulke regels om het oplossen van notoir lastige problemen te faciliteren. Door de eigenschappen van datasets uit toepassingsgebieden te onderzoeken, moet in toekomstig werk worden bekeken of deze theoretische resultaten aansluiten bij de dagelijkse praktijk.

I would rather discover one scientific fact than become King of Persia.
—Democritus, ca. 460–370 BC

Curriculum Vitae

Bart Maarten Paul Jansen was born on the 5th of August 1986 in Nijmegen, the Netherlands. In 2004 he received his VWO-diploma Cum Laude from the Dominicus College in Nijmegen. From 2004 until 2009 he studied Computer Science at Utrecht University, where he obtained his bachelor's degree in 2007 and his master's degree Applied Computing Science in 2009; both were awarded Cum Laude. His master's thesis was titled "Fixed Parameter Complexity of the Weighted Max Leaf Problem" and was written under the supervision of dr. Hans L. Bodlaender. In 2009 he started as a Ph.D. student working on the NWO project "KERNELS: Combinatorial Analysis of Data Reduction" in the Department of Information and Computing Sciences at Utrecht University. He was supervised by dr. Hans L. Bodlaender and prof. dr. Jan van Leeuwen.

Cito rumpes arcum, semper si tensum habueris.
—Phædrus, ca. 15BC–50AD

Acknowledgments

Despite the picture that I occasionally sketch when someone asks me what I do all day (“I sit in my office, draw funny pictures on my whiteboard, and stare at them”), research is not a solitary activity. I would hereby like to acknowledge the contributions of those around me that have allowed me to do my research with such pleasure as I have experienced in the past four years.

Let me begin by thanking the people at Utrecht University. Hans Bodlaender has been an inspiration since I followed his first course on algorithms. If it was not for his follow-up course “Algorithms and Networks”, I may never have ended up in the master program *Applied Computing Science*. During my Ph.D. research his door was always open for me, be it to share in the enthusiasm about a new result, answer a question about treewidth, or simply to enjoy tea at *Chez Hans*. My promotor, Jan van Leeuwen, taught me a lot about the inner workings of academic circles. I continue to be impressed by his attention to mathematical detail. My fellow Ph.D. students Thomas van Dijk, Johan Kwisthout, and Johan van Rooij, brought a lively atmosphere to lunches and discussions. I have greatly enjoyed sharing an office with Stefan Kratsch, who served as a research collaborator and advisor for career decisions. The boardgames we played on his initiative were a good opportunity to meet as friends as well as colleagues. During the last year of my Ph.D. I was happy to have Jesper Nederlof as an office mate, who enlightened me about the intricacies of chess puzzles that I failed to solve. I extend my gratitude to all other staff members in Utrecht, for creating such a hospitable research environment.

Outside of Utrecht University I have also made many friends in the academic community. I am grateful to the kind members of the algorithms group in Bergen for hosting me on several occasions. My research visit to Daniel Lokshtanov at the University of California, San Diego, was a fantastic experience, in terms of both research and sight-seeing. I thank Mike Fellows and Fran Rosamond for hosting me in their home in Australia; I will never forget surfing together in the morning, doing research on a flip chart in the evening. I visited many conferences and workshops. The travel support I have had from several sources has been invaluable in making these trips possible.

Besides my friends in academic circles, there are many others who deserve to be mentioned here. Their company allowed me to unwind after long days of work. They also helped me to cope with the inevitable disappointments of having papers rejected or discovering bugs in proofs. My friends from the salsa dancing community, not just in Utrecht, Nijmegen, or Amsterdam, but also in Oslo and San Diego, were consistently there to let my mind rest while my body danced.

My friends in the windsurfing club Aeolus are the highlight of my social life. We traveled the globe together in search of the best combination of wind and waves. Although they are not computer scientists, they always showed interest in my work. On a windless afternoon this led to a room full of surfers trying to find an embedding of K_7 on a toilet roll! Sharing my experiences with the other surfing Ph.D. students in the club allowed me to gain a healthy perspective on the balance between life and work.

A word of thanks to my family. My parents have been a solid pillar of support, raising me with the message that I should pursue my dreams wherever they take me. They gave me the confidence to take on the world. Verbal sparring with my brother and sister over the dinner table helped me to develop who I am. I could not have wished for any better preparation for writing this thesis than the road trip through eastern Australia that I made with my sister.

Finally, I want to thank my girlfriend for her wit, charm, and unconditional support. While she might not always understand what I do, she understands me.

Bart Jansen.
January 31st 2013, Utrecht.