

THE KERNELIZATION COMPLEXITY OF SOME
DOMINATION AND COVERING PROBLEMS

GEEVARGHESE PHILIP

The Institute of Mathematical Sciences, Chennai, India.

A thesis submitted to the
Board of Studies in Mathematical Sciences
in partial fulfilment of the requirements
for the Degree of
Doctor of Philosophy
of
Homi Bhabha National Institute



September 2011

To Manju Joseph, my wife.

To A. P. Geevarghese and Saramma Philip, my parents.

Abstract

Polynomial-time preprocessing is a simple algorithmic strategy which has been widely employed in practice to tackle hard problems. The quantification and analysis of the efficiency of preprocessing algorithms are, in a certain precise sense, outside the pale of classical complexity theory. The notion of *kernelization* from parameterized complexity theory provides a framework for the mathematical analysis of polynomial-time preprocessing algorithms. Both kernelization and the closely related notion of fixed-parameter tractable (FPT) algorithms are very active areas of current research. In this thesis we describe the results of our study of the kernelization complexity of some graph domination and covering problems.

An instance of a *parameterized problem* is of the form (x, k) where x is a classical problem instance and k is a suitably-chosen *parameter*. A *fixed-parameter tractable* (FPT) algorithm for the problem is an algorithm which solves the problem in time $\mathcal{O}(f(k) \cdot |x|^c)$ for some computable function $f()$ and constant c . A *kernelization algorithm* for the problem is a polynomial-time algorithm which converts the input instance (x, k) to an *equivalent* parameterized instance (x', k') where both the size of the new instance x' and the value of the new parameter k' are bounded by some computable function $f(k)$ of the original parameter k . The new instance is called a *kernel* for the problem, and $f(k)$ is the *size* of the kernel. In the following, n denotes the number of vertices in the input graph, and k is the parameter in each case.

The study of variants of the domination problem in graphs has been a vibrant area of research for many decades, and continues to be a rich source of graph-theoretical and algorithmic problems. The prototypical problem in this field is DOMINATING SET, a classical minimization problem which asks whether the input graph has a dominating

set of size at most k . The natural parameterization of this problem — with the solution size k as the parameter (PARAMETERIZED DOMINATING SET) — is known to be $W[2]$ -complete, and hence is unlikely to have an FPT algorithm on general graphs. We begin our study of the kernelization complexity of graph domination problems by showing that for every fixed $j \geq i \geq 1$, the PARAMETERIZED DOMINATING SET problem is FPT and has a polynomial kernel on graphs that do not have the complete bipartite graph $K_{i,j}$ as a subgraph. In particular, this implies that the problem has polynomial kernels on graphs of *bounded degeneracy*. We then consider a variant of the DOMINATING SET problem, named CONNECTED DOMINATING SET, where the dominating set is required to be connected. The natural parameterized version of this problem (PARAMETERIZED CONNECTED DOMINATING SET) is also known to be $W[2]$ -complete on general graphs. We study the effect of the *girth* of the input graph on the kernelization complexity of this problem, and discover an interesting scenario : the problem is $W[2]$ -hard on graphs of girth 3 or 4, is FPT on graphs of girth at least 5, is unlikely to have polynomial-size kernels on graphs of girth 5 or 6, and has a polynomial kernel on graphs of girth at least 7.

We now move on to the study of various graph covering problems, where the objective is to find a small subset of vertices and/or edges of the input graph such that their removal deletes certain specified structures from the graph. The first problem we consider is the PARAMETERIZED PATHWIDTH-ONE VERTEX DELETION problem which asks whether one can delete at most k vertices from the input graph such that the remaining graph has pathwidth at most one; the parameter is k . A graph has pathwidth at most one if and only if it does not contain cycles or T_2 s (a specific graph on seven vertices), and thus this is a graph covering problem. We show that the problem has a quartic ($\mathcal{O}(k^4)$) vertex kernel, and has an FPT algorithm which runs in $\mathcal{O}(7^k k \cdot n^2)$ time. We then look at a connected variant of a well-studied graph covering problem, namely the FEEDBACK VERTEX SET problem. In the FEEDBACK VERTEX SET problem the question is whether one can delete at most k vertices from the input graph such that the remaining graph contains no cycles. Such a set is called a feedback vertex set of the graph. In the PARAMETERIZED CONNECTED FEEDBACK VERTEX SET problem which we investigate, the question is whether the input graph has a feedback vertex set of size at most k which induces a connected subgraph. We show that the PARAMETERIZED CONNECTED FEEDBACK VERTEX SET problem is FPT, and can be solved in time $\mathcal{O}(2^{\mathcal{O}(k)} n^{\mathcal{O}(1)})$ on general graphs and in time $\mathcal{O}(2^{\mathcal{O}(\sqrt{k} \log k)} n^{\mathcal{O}(1)})$ on graphs excluding a fixed graph H as a minor. Further, we show that the problem is unlikely to have polynomial kernels on general graphs.

We round off the thesis by investigating “partially-connected” variants of two classical graph problems. For each fixed integer t , the PARAMETERIZED t -TOTAL VERTEX COVER problem asks whether the input graph has a vertex cover S of size at most k such that

each connected component of the subgraph induced by S has at least t vertices. The **PARAMETERIZED t -TOTAL EDGE COVER** problem asks whether the input graph has an edge cover S of size at most k such that each connected component of the subgraph induced by S contains at least t edges from S . We show that for $1 \leq t \leq k$, both **PARAMETERIZED t -TOTAL VERTEX COVER** and **PARAMETERIZED t -TOTAL EDGE COVER** are FPT and can be solved in $O(c^k n^d)$ time for some constants $c, d > 0$ in each case. We further show that for every $2 \leq t \leq k$, **PARAMETERIZED t -TOTAL VERTEX COVER** is unlikely to have polynomial kernels, while **PARAMETERIZED t -TOTAL EDGE COVER** has a linear vertex kernel of size $\frac{t+1}{t}k$.

Publication List

1. Geevarghese Philip, Venkatesh Raman, and Somnath Sikdar. Polynomial Kernels for Dominating Set in Graphs of Bounded Degeneracy and Beyond. Accepted for publication in the journal *ACM Transactions on Algorithms*. Preliminary version — Solving Dominating Set in Larger Classes of Graphs: FPT Algorithms and Polynomial Kernels. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, LNCS volume 5757, pages 694–705, 2009.
2. Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, and Saket Saurabh. The effect of girth on the kernelization complexity of Connected Dominating Set. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India, LIPIcs volume 8*, pages 96–107, 2010.
3. Geevarghese Philip, Venkatesh Raman, and Yngve Villanger. A Quartic Kernel for Pathwidth-One Vertex Deletion. In Dimitrios M. Thilikos, editor, *Graph Theoretic Concepts in Computer Science - 36th International Workshop, WG 2010, Zarós, Crete, Greece, June 28-30, 2010. Revised Papers*, LNCS volume 6410, pages 196–207, 2010.
4. Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, Saket Saurabh, and Somnath Sikdar. FPT Algorithms for Connected Feedback Vertex Set. *Journal of Combinatorial Optimization*. Published online: 05 April 2011. DOI=<http://dx.doi.org/10.1007/s10878-011-9394-2>. Preliminary version — FPT Algorithms for Connected Feedback Vertex Set. In Md. Saidur Rahman and Satoshi Fujita, editors, *WALCOM: Algorithms and Computation, 4th International Workshop, WALCOM 2010, Dhaka, Bangladesh, February 10-12, 2010. Proceedings*, volume 5942, pages 269–280, 2010.

5. Henning Fernau, Fedor V. Fomin, Geevarghese Philip, and Saket Saurabh. The Curse of Connectivity: t -Total Vertex(Edge) Cover. In My T. Thai and Sartaj Sahni, editors, *Computing and Combinatorics, 16th Annual International Conference, COCOON 2010, Nha Trang, Vietnam, July 19-21, 2010. Proceedings, LNCS*, volume 6196, pages 34–43, 2010.

Acknowledgements

Many thanks go to ...

Venkatesh Raman, my PhD Adviser, for his patience and care, and for the time he spent with me and for me, trying to teach me how to think, read, write, and speak science. I(t) must have been exasperating at times, especially when your generous hints seemed to be going nowhere. I thank you for your support and forbearance; I hope you are not too disappointed with the outcome!

Saket Saurabh, my “other” Adviser, for always insisting that I can do better — regardless of *my* humble opinion on the matter — and for opening my world to so many new problems, ideas, and people. If not for you, I wouldn’t have learnt much of what I did. If I can take up a tenth of what you keep throwing at me, I should end up doing very well.

K Muralikrishnan, my teacher from pre-research days, without whose guidance I would never have taken up research. I thank you for putting the idea of a research career into my head, for helping with my preparations for admissions, for helping me choose the institute, and for giving me good counsel whenever I was in doubt over these five years.

Vikraman Arvind, Meena Mahajan, and C. R. Subramanian, who constitute my Doctoral Committee.

Vikraman Arvind, Kamal Lodaya, Meena Mahajan, Venkatesh Raman, R. Ramanujam, Saket Saurabh, and C. R. Subramanian, whose courses I attended at IMSc.

M. Praveen, my Computer Science batchmate and fellow-sufferer during the course years. We sure had a lot of fun, both “*on shore, and when Thro’ scudding drifts the rainy Hyades Vext the dim sea*”; I hope you remember those times as fondly as I do!

Krishnan Rajkumar and Srikanth Srinivasan, our go-to people during the first year for vexing (academic) problems. I thank you for listening patiently to my often muddled

questions and showing me how to think them through. Srikanth, also, for his kindness and initiative in offering a short course when he felt this might help us at one juncture.

Neeldhara Misra and Somnath Sikdar, the other “parameterized” graduate students with whom my time at IMSc overlapped the most. I thank you for the many interesting discussions we had about specific problems and otherwise.

Mubeena T C, Vinu Lukose, Narayanan N, and Sreejith A V, who have been my officemates during different periods. Also my other fellow students, who made it enjoyable to be at IMSc. I refrain from putting down names, since it is so easy to miss some name or the other. If you think your name should be here, then you know you are one of them ☺.

Abhimanyu M. Ambalath, S. Arumugam, Radheshyam Balasundaram, Robert Brederick, K. Raja Chandrasekar, Marek Cygan, Henning Fernau, Fedor V. Fomin, Chintan Rao H., Venkata Koppula, Daniel Lokshtanov, Neeldhara Misra, Matthias Mnich, André Nichterlein, Rolf Niedermeier, Marcin Pilipczuk, Michał Pilipczuk, Venkatesh Raman, M. S. Ramanujan, Saket Saurabh, Somnath Sikdar, Yngve Villanger, and Jakub Onufry Wojtaszczyk, my co-authors.

Krzysztof Diks, Henning Fernau, Fedor V. Fomin, Rolf Niedermeier, and Christophe Paul, for inviting me for research visits to their respective departments.

K Muralikrishnan and Christophe Paul, who reviewed this thesis.

The Administration at IMSc who, on the occasions when I needed help regarding official matters, have been very considerate. I especially thank Shri Vishnu Prasad, Registrar, who has found a way to help me every time I asked for help.

My parents, for having always given me the freedom to choose what I wanted to pursue.

Manju, my wife, who has borne much and forgone more for the cause of my PhD. There have been moments when you despaired of waiting for my thesis to get done. Here it is.

Contents

| | |
|--|-------------|
| Publication List | ix |
| Contents | xiii |
| I Introduction | 1 |
| 1 Overview | 3 |
| 1.1 Variants of Graph Domination | 6 |
| 1.2 Graph Covering Problems | 9 |
| 1.3 Conclusion | 14 |
| 2 Preliminaries | 15 |
| 2.1 Graph Terminology | 15 |
| 2.2 Parameterized Complexity | 17 |
| II Domination Problems | 21 |
| 3 Domination on $K_{i,j}$-free Graphs | 23 |
| 3.1 A Polynomial Kernel for $K_{i,j}$ -free Graphs | 27 |
| 3.2 A Polynomial Kernel for d -degenerate Graphs | 47 |
| 3.3 Independent Dominating Set in $K_{i,j}$ -free graphs | 48 |
| 3.4 Conclusion | 49 |

| | | |
|------------|---|------------|
| 4 | Connected Domination and Girth | 51 |
| 4.1 | On Graphs of Girth 3 and 4: $W[2]$ -hardness | 54 |
| 4.2 | On Graphs of Girth 5 or More: A Kernel of Size $2^k k^{3k}$ | 56 |
| 4.3 | On Graphs of girth 5 and 6: No Polynomial Kernels | 63 |
| 4.4 | On Graphs of girth 7 or More: A Cubic Vertex Kernel | 72 |
| 4.5 | Conclusion | 76 |
| III | Covering Problems | 79 |
| 5 | Pathwidth-One Vertex Deletion | 81 |
| 5.1 | A Single-Exponential FPT Algorithm | 83 |
| 5.2 | A Quartic Kernel | 86 |
| 5.3 | Conclusion | 96 |
| 6 | Connected Feedback Vertex Set | 99 |
| 6.1 | A Single-Exponential FPT Algorithm for General Graphs | 102 |
| 6.2 | A Faster FPT Algorithm for H -minor free Graphs | 111 |
| 6.3 | Kernelization Complexity | 119 |
| 6.4 | Conclusion | 121 |
| 7 | Total Vertex Cover and Total Edge Cover | 123 |
| 7.1 | Computing Total Vertex Covers | 126 |
| 7.2 | Computing Total Edge Covers | 135 |
| 7.3 | Conclusion | 139 |
| IV | Conclusion | 141 |
| 8 | Conclusion | 143 |
| | Bibliography | 149 |

Part I

Introduction

CHAPTER 1

Overview

LITERALLY thousands of problems derived from a multitude of fields are now known to be NP-hard [11, 27, 59], and new problems are constantly being added to this collection. Assuming the widely held $P \neq NP$ conjecture, none of these problems can be solved in polynomial time in the size of the input. Given that in most cases polynomial-time solvability coincides with *efficient* solvability, it turns out that in general, one cannot hope to solve these problems efficiently. Many of these theoretical problems are directly motivated by real-world problems which have a significant bearing on the economic efficiency, profitability, and sometimes even on the survival itself of the entities concerned. These problems can thus be brushed aside only at great cost. Many different approaches have therefore been developed to cope with such hard problems. These include heuristics (“rules of thumb”), approximation algorithms, randomized algorithms, parameterized algorithms, and probabilistic meta-heuristics such as genetic algorithms, simulated annealing, ant colony optimization, taboo search, and others.

One of the earliest and simplest methods of coping with hard problems is *preprocessing* or *data reduction*. Simply put, this involves applying some “reduction rules” to the input instance which result, most of the time, either in a solution itself, or in a simplified and small equivalent instance which can then be solved using other methods. An early example of such preprocessing is Quine’s work from the 1950’s where he applied reduction rules to solve the problem of simplifying truth functions [101]. More recent examples include work on input size reduction for various scheduling, knapsack, and social-choice problems [46, 72, 110, 111].

Polynomial-time preprocessing has thus been widely used in practice to cope with NP-hard problems, since — in practice — they turn out to be quite effective. However, there was no significant attempt at a mathematical analysis of the efficiency of such methods till comparatively recent times. A fundamental reason for this is the fact that classical, “one-dimensional” complexity theory is somewhat ill-equipped to analyze such reductions in the size of instances of NP-hard problems which are achieved in polynomial time. To see this, consider an instance I of an NP-hard problem \mathcal{P} . If there exists an algorithm \mathcal{A} which could take I as input, run in polynomial time, and return an equivalent instance I' where I' is even a single bit smaller than I , then one could recursively apply \mathcal{A} to solve the problem on I in polynomial time. Since I is an arbitrary instance of the problem, this implies that the “preprocessing” algorithm \mathcal{A} in fact *solves* the NP-hard problem \mathcal{P} in polynomial time. Thus it appears that as per classical complexity theory, polynomial-time preprocessing algorithms for NP-hard problems cannot exist unless $P=NP$.

It turns out that this gap between theory and practice — polynomial-time preprocessing algorithms exist and are quite effective, while classical complexity theory does not seem to be able to explain their existence — can be bridged by the “multidimensional” approach to problem complexity advocated by parameterized complexity theory. More specifically, the notion of *kernelization* captures the behaviour of preprocessing algorithms, and provides a framework for the rigorous mathematical analysis of such algorithms. This thesis is about kernelization algorithms for some domination and covering problems in graphs. In the following sections, we introduce the two most important notions in parameterized complexity theory — the other one being parameterized tractability — and give a brief summary of our results which are more fully described in later chapters.

Parameterized Tractability

Parameterized algorithms [41, 52, 95] constitute one approach towards solving NP-hard problems in “feasible” time. Each instance of a parameterized problem comes with an associated *parameter*, which is usually a non-negative integer, and the goal is to find algorithms that solve the problem in time polynomial in the input size, where the degree of the polynomial is *independent* of the parameter. More precisely, if k is the parameter and n the size of the input, then the goal is to obtain an algorithm that solves the problem in time $f(k) \cdot n^c$ where f is some computable function and c is a constant independent of k . Such an algorithm is called a fixed-parameter-tractable (FPT) algorithm, and the class of all parameterized problems that have FPT algorithms is called FPT; a parameterized problem that has a fixed-parameter-tractable algorithm is said to be (in) FPT.

Together with this revised notion of tractability, parameterized complexity theory

offers a corresponding notion of intractability as well, captured by the concept of *W-hardness*. In brief, the theory defines a hierarchy of complexity classes $\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \cdots \subseteq \text{XP}$, where each inclusion is believed to be strict — on the basis of evidence similar in spirit to the evidence for believing that $\text{P} \neq \text{NP}$ — and XP is the class of all parameterized problems that can be solved in $\mathcal{O}(n^{f(k)})$ time where n is the input size, k the parameter, and f is some computable function [41, 95].

Kernelization Complexity

Closely related to the notion of an FPT algorithm is the concept of a *kernel* for a parameterized problem. We say that two instances of a decision problem are *equivalent* if and only if they are either both yes-instances or both no-instances. A *kernelization algorithm* for a parameterized problem is a polynomial-time algorithm that converts an instance (x, k) of the problem to an equivalent instance (y, k') whose size $|y|$ and parameter k' are both bounded by functions of the original parameter k . The instance y output by the algorithm is said to be a *kernel* for the problem.

It is not difficult to see that if a problem has a kernelization algorithm, then the problem is FPT. Somewhat more surprisingly, the converse is also true: A folklore theorem of parameterized complexity states that a parameterized problem has a kernelization algorithm if and only if it has an FPT algorithm [41]. However, the *size* of the kernel implied by the proof of the folklore theorem is equal to the function $f(k)$ in the running time of the corresponding FPT algorithm, and hence is exponential in k . The interesting problem is, therefore, to find if the kernel size can be made smaller — in particular, whether it can be made polynomial in k .

Finding polynomial kernels for parameterized problems has been a vibrant sub-area of research in parameterized complexity for well over a decade. This has yielded a large collection of results; see, for example, the survey on kernelization results by Guo and Niedermeier [63]. A more recent and exciting development in parameterized complexity is the emergence of a corresponding lower bound theory [15, 18, 34, 39] which provides methods to prove that certain parameterized problems are unlikely to have polynomial kernels.

Organization of the rest of this Thesis

The first part of the thesis consists of two chapters of an introductory nature, including this one. In the remainder of this chapter we give an overview of the results discussed in the rest of the thesis. In [Chapter 2](#) we set down basic material and notation from graph theory and parameterized complexity theory which we use in the later chapters. This

concludes Part I of the thesis. Each of the next two parts of the thesis focuses on a specific theme; these are described in more detail below.

1.1 Variants of Graph Domination

In [Part II](#) we describe kernelization complexity results about two variants of graph domination. Both these variants are known to be $W[2]$ -hard on general graphs, and are therefore unlikely to have kernels of any size on general graphs. We investigate the effect of restricting the input graph on the kernelization complexity of these problems.

1.1.1 Dominating Set on $K_{i,j}$ -free Graphs

A *dominating set* of a graph G is a set $S \subseteq V(G)$ of vertices of G such that every vertex in $V(G) \setminus S$ is adjacent to some vertex in S . The DOMINATING SET problem is defined as follows:

DOMINATING SET

Input: A graph G and a non-negative integer k .

Question: Does G have a dominating set with at most k vertices?

The DOMINATING SET problem is NP-hard, even in very restricted graph classes such as the class of planar graphs with maximum degree 3 [59]. Hence, unless $P=NP$, there is no polynomial-time algorithm that solves the problem even in such restricted graph classes.

One natural parameter for the DOMINATING SET problem is k , the size of the solution being sought. A natural parameterized version of the DOMINATING SET problem is thus the PARAMETERIZED DOMINATING SET problem, defined as follows:

PARAMETERIZED DOMINATING SET

Input: A graph G , and a non-negative integer k .

Parameter: k

Question: Does G have a dominating set with at most k vertices?

It turns out that the DOMINATING SET problem, with this parameterization, is still hard to solve. More precisely, PARAMETERIZED DOMINATING SET is the canonical $W[2]$ -complete problem [41], and the problem remains $W[2]$ -complete even in many restricted classes of graphs — for example, it is $W[2]$ -complete in classes of graphs with bounded

average degree [60]. Thus there is no FPT algorithm that solves the problem, even when restricted to graphs of bounded average degree, unless $\text{FPT}=\text{W}[2]$, which is considered unlikely. From the equivalence of FPT and kernelization mentioned above it follows that, unless $\text{FPT}=\text{W}[2]$, there is no kernelization algorithm for PARAMETERIZED DOMINATING SET on general graphs or on graphs with a bounded average degree.

The problem does have FPT algorithms on certain restricted families of graphs, such as on planar graphs [54], graphs of bounded genus [45], nowhere-dense classes of graphs [31], K_h -topological-minor-free graphs, and graphs of bounded degeneracy [3].

The PARAMETERIZED DOMINATING SET problem has been shown to have polynomial kernels on various restricted classes of graphs, such as planar graphs [1, 24], graphs of bounded genus [53], and K_h -topological-minor-free graph classes [2, 65] (which include, for example, planar graphs). Here K_h denotes the complete graph on h vertices. The degree of the polynomial bound on the kernel size for K_h -topological-minor-free graphs depends on h .

Our Work. $K_{i,j}$ denotes the complete bipartite graph on $(i+j)$ vertices, where the two parts have i and j vertices, respectively. For fixed integers i and j , $K_{i,j}$ -free graphs are those which exclude $K_{i,j}$ as a — not necessarily induced — subgraph. In [Chapter 3](#) we consider the PARAMETERIZED DOMINATING SET problem restricted to $K_{i,j}$ -free graphs. We show that for every fixed $j \geq i \geq 1$, the PARAMETERIZED DOMINATING SET problem on $K_{i,j}$ -free graphs is FPT and has a polynomial kernel. We describe a polynomial-time algorithm that, given a $K_{i,j}$ -free graph G and a non-negative integer k , constructs a $K_{i,j}$ -free graph H and an integer k' such that (1) G has a dominating set of size at most k if and only if H has a dominating set of size at most k' , (2) H has $\mathcal{O}((j+1)^{i+1}k^{i^2})$ vertices, and (3) $k' = \mathcal{O}((j+1)^{i+1}k^{i^2})$.

Since d -degenerate graphs do not have $K_{d+1,d+1}$ as a subgraph, this immediately yields a polynomial kernel with $\mathcal{O}((d+2)^{d+2}k^{(d+1)^2})$ vertices for the PARAMETERIZED DOMINATING SET problem on d -degenerate graphs, solving an open problem posed by Alon and Gutner [2, 65].

The most general class of graphs for which a polynomial kernel was previously known for PARAMETERIZED DOMINATING SET is the class of K_h -topological-minor-free graphs [65]. Graphs of bounded degeneracy are the most general class of graphs for which an FPT algorithm was previously known for this problem. K_h -topological-minor-free graphs are $K_{i,j}$ -free for suitable values of i, j (but not vice versa), and so our results show that PARAMETERIZED DOMINATING SET has both FPT algorithms and polynomial kernels on strictly more general classes of graphs.

Using the same techniques, we also obtain an $\mathcal{O}(jk^i)$ vertex-kernel for the PARAMETERIZED INDEPENDENT DOMINATING SET problem on $K_{i,j}$ -free graphs.

This chapter is based on a paper [99] which has been accepted for publication in the journal *ACM Transactions on Algorithms*. A preliminary version appeared in the Proceedings of the European Symposium on Algorithms (ESA) 2009.

1.1.2 Connected Dominating Set and Girth

In Chapter 4 we take up the study of the kernelization complexity of a parameterized problem which is *unlikely* to have polynomial kernels on $K_{i,j}$ -free graphs. A set $S \subseteq V(G)$ of vertices of a graph G is said to be a *connected dominating set* of G if (i) S is a dominating set of G and (ii) the subgraph $G[S]$ induced on S is a connected graph. The CONNECTED DOMINATING SET problem is defined as follows:

CONNECTED DOMINATING SET

Input: A graph G and a non-negative integer k .

Question: Does G have a connected dominating set with at most k vertices?

The CONNECTED DOMINATING SET problem is NP-hard, even in very restricted graph classes such as the class of 4-regular planar graphs [59]. A natural parameter for the CONNECTED DOMINATING SET problem is k , the size of the solution being sought. A natural parameterized version of the CONNECTED DOMINATING SET problem is thus the PARAMETERIZED CONNECTED DOMINATING SET problem, defined as follows:

PARAMETERIZED CONNECTED DOMINATING SET

Input: A graph G , and a non-negative integer k .

Parameter: k

Question: Does G have a connected dominating set with at most k vertices?

The parameterized complexity of PARAMETERIZED CONNECTED DOMINATING SET has been extensively investigated, and many results are known. For instance, it is known that PARAMETERIZED CONNECTED DOMINATING SET is $W[2]$ -hard on general graphs [41], has a linear kernel on planar, or more generally, on apex-minor-free graphs [56, 62, 83], and is FPT on graphs of bounded degeneracy [60]. It has recently been shown that PARAMETERIZED CONNECTED DOMINATING SET is *unlikely* to have polynomial sized kernels on graphs of bounded degeneracy [28], and therefore, on $K_{i,j}$ -free graphs.

Our Work. In Chapter 4 we study the kernelization complexity of the Connected Dominating Set problem, when restricted to graphs that (do not) have small cycles. The *girth* of a graph is the length of a smallest cycle in the graph. We obtain the complete

kernelization complexity landscape for the PARAMETERIZED CONNECTED DOMINATING SET problem based on the girth of the problem instance. More precisely, we show that PARAMETERIZED CONNECTED DOMINATING SET

1. is $W[2]$ -hard on graphs of girth 3 or 4, and hence does not have a kernel of *any* size on these graphs unless $FPT = W[2]$;
2. has an FPT algorithm that runs in time $2^k k^{3k} n^{\mathcal{O}(1)}$ on graphs of girth 5 or 6, and hence has a kernel of size $2^k k^{3k}$ on these graphs*; but has *no* polynomial kernel (unless the polynomial hierarchy collapses to the third level) on these graphs, and,
3. has a cubic ($\mathcal{O}(k^3)$) vertex kernel on graphs of girth at least 7.

To obtain the kernel lower bound we introduce an intermediate, seemingly unrelated problem named PARAMETERIZED FAIR CONNECTED COLOURS. Using the recently developed kernel lower bound machinery due to Bodlaender et al. [15], we show that PARAMETERIZED FAIR CONNECTED COLOURS has no polynomial kernels unless the polynomial hierarchy collapses to the third level. To complete the argument, we provide a parameter-preserving reduction [18] from PARAMETERIZED FAIR CONNECTED COLOURS to PARAMETERIZED CONNECTED DOMINATING SET.

This chapter is based on a paper [89] which appeared in the Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS) 2010.

1.2 Graph Covering Problems

In Part III we describe parameterized and kernelization complexity results for some graph covering problems. In a graph covering problem, one looks for a small set of vertices or edges (or larger structures) which intersect a designated set of structures in the graph. A typical example is the FEEDBACK VERTEX SET problem, where the goal is to find a small set of vertices which intersect every cycle in the graph.

1.2.1 Pathwidth-One Vertex Deletion

The *treewidth* of a graph is a measure of how tree-like a graph is, and was introduced by Robertson and Seymour in their seminal Graph Minors series [105]. It turns out that the graphs of treewidth at most 1 are exactly the forests. The FEEDBACK VERTEX SET problem

* Throughout this thesis the symbol n denotes the number of vertices in the input graph, unless specifically mentioned otherwise.

mentioned above can equivalently be thought of as asking if there is a small set of vertices in the input graph whose deletion results in a graph of treewidth at most 1. The *pathwidth* of a graph is a notion closely related to treewidth, and was also introduced by Robertson and Seymour, in the very first paper in the Graph Minors series [104]. The pathwidth of a graph denotes how “path-like” it is. We say that a vertex in graph is a *pendant vertex* if it has degree exactly one in the graph. A graph has pathwidth at most one if and only if it is a collection of *caterpillars*, where a caterpillar is a special kind of tree: it is a tree that becomes a path (called the *spine* of the caterpillar) when all its pendant vertices are removed. More formally, a *path decomposition* of a graph G is a pair (T, χ) in which T is a path and $\chi = \{\chi_i \mid i \in V(T)\}$ is a family of subsets of $V(G)$, called *bags*, such that

- (i) $\bigcup_{i \in V(T)} \chi_i = V(G)$;
- (ii) for each edge $(u, v) \in E(G)$ there exists an $i \in V(T)$ such that both u and v belong to χ_i ; and
- (iii) for all $v \in V(G)$, the set of nodes $\{i \in V(T) \mid v \in \chi_i\}$ induces a sub-path of T .

The maximum of $|\chi_i| - 1$, over all $i \in V(T)$, is called the *width* of the path decomposition. The *pathwidth* of a graph G is the minimum width taken over all path decompositions of G .

A vertex set $S \subseteq V(G)$ of a graph G is said to be a *pathwidth-one deletion set* (PODS) if $G[V(G) \setminus S]$ has pathwidth at most one. The PATHWIDTH-ONE VERTEX DELETION problem is defined as follows:

PATHWIDTH-ONE VERTEX DELETION

Input: A graph G and a non-negative integer k .

Question: Does G have a pathwidth-one deletion set with at most k vertices?

The PATHWIDTH-ONE VERTEX DELETION problem is NP-hard; this follows directly from a classical result due to Lewis and Yannakakis [86] on the NP-hardness of hereditary vertex-deletion problems. A natural parameter for the PATHWIDTH-ONE VERTEX DELETION problem is k , the size of the solution being sought. A natural parameterized version of the PATHWIDTH-ONE VERTEX DELETION problem is thus the PARAMETERIZED PATHWIDTH-ONE VERTEX DELETION problem, defined as follows:

PARAMETERIZED PATHWIDTH-ONE VERTEX DELETION

Input: A graph G , and a non-negative integer k .

Parameter: k

Question: Does G have a pathwidth-one deletion set with at most k vertices?

Our Work. We initiated the study of the parameterized complexity of this problem, and we describe our results in [Chapter 5](#). We show that the problem has a quartic vertex-kernel: That is, given an input instance (G, k) , we can construct, in polynomial time, an instance (G', k') such that (i) (G, k) is a YES instance if and only if (G', k') is a YES instance, (ii) G' has $\mathcal{O}(k^4)$ vertices, and (iii) $k' \leq k$. We also present an FPT algorithm for the problem that runs in $\mathcal{O}(7^k k \cdot n^2)$ time.

This chapter is based on a paper [100] which appeared in the Proceedings of the International Workshop on Graph Theoretic Concepts in Computer Science (WG) 2010.

1.2.2 Connected Feedback Vertex Set

A set $S \subseteq V(G)$ of vertices of a graph G is said to be a *feedback vertex set* of G if the graph $G[V(G) \setminus S]$ obtained by removing all the vertices in S from G is a forest (i.e. has no cycles). The set S is said to be a *connected feedback vertex set* of G if (i) S is a feedback vertex set of G and (ii) the subgraph $G[S]$ induced on S is a connected graph. The CONNECTED FEEDBACK VERTEX SET problem is defined as follows:

CONNECTED FEEDBACK VERTEX SET

Input: A graph G and a non-negative integer k .

Question: Does G have a connected feedback vertex set with at most k vertices?

The CONNECTED FEEDBACK VERTEX SET problem is NP-hard, even in restricted graph classes such as the class of planar graphs [16]. A natural parameter for the CONNECTED FEEDBACK VERTEX SET problem is k , the size of the solution being sought. A natural parameterized version of the CONNECTED FEEDBACK VERTEX SET problem is thus the PARAMETERIZED CONNECTED FEEDBACK VERTEX SET problem, defined as follows:

PARAMETERIZED CONNECTED FEEDBACK VERTEX SET

Input: A graph G , and a non-negative integer k .

Parameter: k

Question: Does G have a connected feedback vertex set with at most k vertices?

The closely related and very well studied PARAMETERIZED FEEDBACK VERTEX SET problem asks if the input graph has a — not necessarily connected — feedback vertex

set of size at most k ; the parameter is k . The quest for fast FPT algorithms and small kernels for the PARAMETERIZED FEEDBACK VERTEX SET presents an illuminative case study of the evolution of the field of fixed parameter tractability, and stands out among the many success stories of this algorithmic approach towards solving hard problems. The first FPT algorithm for the PARAMETERIZED FEEDBACK VERTEX SET problem, with a running time of $\mathcal{O}(k^4! \cdot n^{\mathcal{O}(1)})$, was developed by Bodlaender [13] and by Downey and Fellows [40]. After a series of improvements [41, 74, 103], a running time of the form $\mathcal{O}(c^k \cdot n^{\mathcal{O}(1)})$ was first obtained by Guo et.al [64], whose algorithm ran in $\mathcal{O}(37.7^k \cdot n^{\mathcal{O}(1)})$ time. This was improved by Dehne et.al [33] to $\mathcal{O}(10.6^k \cdot n^{\mathcal{O}(1)})$ in 2007, and to the current best deterministic time bound of $\mathcal{O}(3.83^k \cdot n^{\mathcal{O}(1)})$ by Cao et.al [22] in 2010. The fastest known *randomized* algorithm for the problem was developed by Cygan et al. in 2011, and runs in $\mathcal{O}(3^k \cdot n^{\mathcal{O}(1)})$ time [30].

Our Work. In contrast to PARAMETERIZED FEEDBACK VERTEX SET, the PARAMETERIZED CONNECTED FEEDBACK VERTEX SET problem had, somewhat surprisingly, not been studied from the point of view of parameterized algorithms. We initiated the study of the parameterized complexity of the PARAMETERIZED CONNECTED FEEDBACK VERTEX SET problem, and we describe our results in Chapter 6. We show that CONNECTED FEEDBACK VERTEX SET can be solved in time $\mathcal{O}(2^{\mathcal{O}(k)} n^{\mathcal{O}(1)})$ on general graphs and in time $\mathcal{O}(2^{\mathcal{O}(\sqrt{k} \log k)} n^{\mathcal{O}(1)})$ on graphs excluding a fixed graph H as a minor. For obtaining our result on general undirected graphs we develop a parameterized algorithm for GROUP STEINER TREE, a well studied variant of STEINER TREE, which is of independent interest in that it could be useful for obtaining parameterized algorithms for other connectivity problems. We also show that this problem is unlikely to have polynomial kernels on general graphs.

This chapter is based on a paper [90] which appeared in the *Journal of Combinatorial Optimization*.

1.2.3 Total Vertex Cover, Total Edge Cover

A set $S \subseteq V(G)$ of vertices of a graph G is said to be a *vertex cover* of G if for every edge e in G , there is some vertex in S which is incident with e . A set $F \subseteq E(G)$ of edges of G is said to be an *edge cover* of G if every vertex v in G is incident with some edge in F .

For each fixed positive integer t , a set S of vertices of G is said to be a *t -total vertex cover* of G if (i) S is a vertex cover of G , and (ii) each connected component of the graph $G[S]$ induced by S contains at least t vertices. The t -TOTAL VERTEX COVER problem is defined as follows:

t-TOTAL VERTEX COVER

Input: A graph G and a non-negative integer k .

Question: Does G have a t -total vertex cover with at most k vertices?

Note that the t -TOTAL VERTEX COVER problem is a generalization of the well-studied NP-hard problems VERTEX COVER [75] and CONNECTED VERTEX COVER [58].

For each fixed positive integer t , a set F of edges of G is said to be a t -total edge cover of G if (i) F is an edge cover of G , and (ii) each connected component of the graph $G[F]$ induced by F contains at least t edges from F . The t -TOTAL EDGE COVER problem is defined as follows:

t-TOTAL EDGE COVER

Input: A graph G and a non-negative integer k .

Question: Does G have a t -total edge cover with at most k edges?

The t -TOTAL VERTEX COVER problem is NP-hard for all $t \geq 1$, and the t -TOTAL EDGE COVER problem is NP-hard for all $t \geq 2$ [50]. A natural parameter for each of these problems is k , the size of the set being sought. This yields the following parameterized problems:

PARAMETERIZED t -TOTAL VERTEX COVER

Input: A graph G , and a non-negative integer k .

Parameter: k

Question: Does G have a t -total vertex cover with at most k vertices?

PARAMETERIZED t -TOTAL EDGE COVER

Input: A graph G , and a non-negative integer k .

Parameter: k

Question: Does G have a t -total edge cover with at most k edges?

Our Work. The study of the parameterized complexity of these problems was initiated by Fernau and Manlove [50]. We significantly improve their results and obtain several new results, which we describe in [Chapter 7](#). In particular, we complete the picture on how even the slightest connectivity requirement dramatically changes the complexity of these problems. We show that

- both problems remain fixed-parameter tractable with these restrictions, with running times of the form $O(c^k n^d)$ for some constants $c, d > 0$ in each case;
- for every $t \geq 2$, PARAMETERIZED t -TOTAL VERTEX COVER has no polynomial kernel unless the Polynomial Hierarchy collapses to the third level;
- for every $t \geq 2$, PARAMETERIZED t -TOTAL EDGE COVER has a linear vertex kernel of size $\frac{t+1}{t}k$.

These results significantly improve the earlier work on these problems.

Our no-poly-kernel result for PARAMETERIZED t -TOTAL VERTEX COVER, and the known NP-hardness result for t -TOTAL EDGE COVER, are in stark contrast to the fact that PARAMETERIZED VERTEX COVER has a $2k$ vertex kernel, and that EDGE COVER is solvable in polynomial time. This illustrates how even the slightest connectivity requirement could result in a drastic change in the tractability of a graph covering problem.

This chapter is based on a paper [51] which appeared in the Proceedings of the Annual International Computing and Combinatorics Conference (COCOON) 2010.

1.3 Conclusion

This thesis is about certain kernelization problems on graphs. In this chapter we motivated the study of kernelization algorithms, briefly describe parameterized tractability and kernelization complexity, and give a summary introduction to the rest of this thesis. In the next chapter we collect together the notation and terminology used in the rest of the thesis, and describe various results from parameterized complexity theory used elsewhere in the thesis.

CHAPTER 2

Preliminaries

IN this chapter we lay down the notation and terminology used elsewhere in the thesis, for the sake of easy reference. We also explain concepts from parameterized complexity theory, and give a description of the recently-developed theory of kernel lower bounds.

2.1 Graph Terminology

In general we follow the graph terminology used in the textbook by Diestel [37]. We let $V(G)$ and $E(G)$ denote, respectively, the vertex and edge sets of a graph G . The *open neighbourhood* of a vertex v in a graph G , denoted $N(v)$, is the set of all vertices that are adjacent to v in G . The elements of $N(v)$ are said to be the *neighbours* of v , and $N[v] = N(v) \cup \{v\}$ is called the *closed neighbourhood* of v . For a set of vertices $X \subseteq V(G)$, the open and closed neighbourhoods of X are defined, respectively, as $N(X) = \bigcup_{u \in X} N(u) \setminus X$ and $N[X] = N(X) \cup X$. A vertex $v \in V(G)$ is said to be a *pendant* vertex of G if $|N(v)| = 1$.

A graph H is a *subgraph* of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. The subgraph H is called an *induced subgraph* (induced by the vertex set $V(H)$) of G if $E(H) = \{\{u, v\} \in E(G) \mid u, v \in V(H)\}$. For a subset $S \subseteq V(G)$ the subgraph of G induced by S is denoted by $G[S]$, and we use $G \setminus S$ to denote the subgraph induced by $V(G) \setminus S$. The *girth* of a graph is the length of a smallest cycle present in the graph.

A *planar* graph is a graph which can be drawn on the plane in such a way that no two edges cross. A graph G is said to be an *apex* graph if there is a vertex $v \in V(G)$ such that the graph G' obtained by deleting v from G is a planar graph. Given a graph G and

$A, B \subseteq V(G)$, we say that A *dominates* B if every vertex in $B \setminus A$ is adjacent in G to some vertex in A .

A *dominating set* of graph G is a vertex-subset $S \subseteq V(G)$ such that for each $u \in V(G) \setminus S$ there exists $v \in S$ such that $\{u, v\} \in E(G)$.

The operation of *contracting* an edge (u, v) consists of deleting vertex u , renaming vertex v to uv , and adding a new edge (x, uv) for each edge $(x, u); x \neq v$. Multiple edges that may possibly result from this operation are deleted. Note that the operation is symmetric with respect to u and v . A graph H is said to be a *minor* of a graph G if a graph isomorphic to H can be obtained by contracting zero or more edges of some subgraph of G . A graph class \mathcal{C} is *minor-closed* if any minor of any graph in \mathcal{C} is also an element of \mathcal{C} . A minor-closed graph class \mathcal{C} is *H -minor-free* or simply *H -free* if $H \notin \mathcal{C}$.

A *tree decomposition* of a graph G is a pair (T, χ) in which T is a tree and $\chi = \{\chi_i \mid i \in V(T)\}$ is a family of subsets of V , called *bags*, such that

- (i) $\bigcup_{i \in V(T)} \chi_i = V$;
- (ii) for each edge $(u, v) \in E$ there exists an $i \in V(T)$ such that both u and v belong to χ_i ; and
- (iii) for all $v \in V$, the set of nodes $\{i \in V(T) \mid v \in \chi_i\}$ induces a connected subgraph of T .

The maximum of $|\chi_i| - 1$, over all $i \in V(T)$, is called the *width* of the tree decomposition. The *treewidth* of a graph G is the minimum width taken over all tree decompositions of G . A *path decomposition* of a graph $G = (V, E)$ is a tree decomposition of G where the underlying tree T is a path. The *pathwidth* of G is the minimum width over all possible path decompositions of G .

A tree decomposition $(T, \mathcal{X} = \{X_t\}_{t \in V(T)})$ of a graph G is called a *nice tree decomposition* [14] if it satisfies the following conditions:

- Every node of the tree T has at most two children. A node that has no children is called a *leaf* node. The non-leaf nodes are of three kinds:
 - If a node t has two children t_1 and t_2 , then $X_t = X_{t_1} = X_{t_2}$, and t is called a *join* node.
 - if a node t has one child t_1 , then either $|X_t| = |X_{t_1}| + 1$ and $X_{t_1} \subset X_t$ (t is called an *introduce* node), or $|X_t| = |X_{t_1}| - 1$ and $X_t \subset X_{t_1}$ (t is called a *forget* node).

It is possible to transform a given tree decomposition of a graph G into a nice tree decomposition of the same width in time $\mathcal{O}(|V(G)| + |E(G)|)$ [14].

2.2 Parameterized Complexity

Parameterized complexity [41, 52, 95] is a two-dimensional generalization of classical complexity analysis where, in addition to the overall input size n , one studies how a secondary measurement (called the *parameter*), that captures additional relevant information, affects the computational complexity of the problem in question. Parameterized decision problems are defined by specifying the input, the parameter, and the question to be answered. A *parameterized problem* Π is thus a subset of $\Gamma^* \times \mathbb{N}$, where Γ is a finite alphabet. An instance of a parameterized problem is a tuple (x, k) , where k is called the parameter.

One of the two central notions in parameterized complexity is *fixed-parameter tractability (FPT)* which means, for a given instance (x, k) , decidability in time $\mathcal{O}(f(k) \cdot p(|x|))$, where f is a computable function of k and p is a polynomial. A parameterized problem that can be decided in such a time-bound is termed *fixed-parameter tractable (FPT)*, and the class of all FPT problems is also called FPT. The class FPT is the two-dimensional analogue of the classical complexity class P.

In specifying the running times of FPT algorithms (and otherwise as well), we sometimes use the following shortened notation: Given $f: \mathbb{N} \rightarrow \mathbb{N}$, we define $\mathcal{O}^*(f(n))$ to be $\mathcal{O}(f(n) \cdot p(n))$, where $p(\cdot)$ is some polynomial function. That is, the \mathcal{O}^* notation suppresses polynomial factors in the expression.

The other central notion in parameterized complexity, namely *kernelization*, is formally defined as follows:

Definition 2.1. [Kernelization, Kernel] A kernelization algorithm for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, outputs, in time polynomial in $|x| + k$, a pair $(x', k') \in \Sigma^* \times \mathbb{N}$ such that (a) $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$ and (b) $|x'|, k' \leq g(k)$, where g is some computable function. The output instance x' is called the kernel, and the function g is referred to as the size of the kernel. If $g(k) = k^{\mathcal{O}(1)}$ then we say that Π admits a polynomial kernel.

When a kernelization algorithm outputs a graph on $h(k)$ vertices, we sometimes say that the output is an $h(k)$ *vertex-kernel*.

Less formally, kernelization algorithms are polynomial-time algorithms that take an input and a positive integer k (the *parameter*) and output an equivalent instance where the size of the new instance and the new parameter are both bounded by some function $g(k)$. The new instance is called a $g(k)$ *kernel* for the problem. If $g(k)$ is a polynomial in k then we say that the problem admits *polynomial kernels*. If $p(k) = \mathcal{O}(k)$, then the problem is said to have a *linear kernel*.

The two notions are closely related, as shown by the following “first theorem of parameterized complexity” :

Theorem 2.1. *A parameterized problem is fixed-parameter tractable if and only if it has a kernel.*

The standard proof of the forward direction of this statement also implies that if the problem can be solved in $f(k) \cdot \mathcal{O}(p(|x|))$ time, then the problem has a kernel of size $f(k)$.

Kernelization is a rapidly growing sub-area of parameterized complexity. For many years, the main thrust in this line of research had been in finding “small” kernels — polynomial, or better, linear kernels — for a variety of problems. Over time, the field acquired a growing collection of problems for which it was not known whether they had polynomial kernels or not. It seemed quite hard to show that these problems had polynomial kernels, but there was no way of proving lower bounds either. A recent set of breakthrough results bridged this gap, and provided the field with a framework for proving that certain problems have no polynomial kernels, albeit under certain complexity-theoretic assumptions.

2.2.1 Kernel Lower Bound Machinery

We now describe the notions and results from the recently developed theory of kernel lower bounds [15, 18, 39] which are used to prove lower bounds on the size of kernels. We begin by associating a classical decision problem with a parameterized problem in a natural way, as follows:

Definition 2.2. [Derived Classical Problem] [18] Let $\Pi \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem, and let $1 \notin \Sigma$ be a new symbol. We define the *derived classical problem* associated with Π to be $\{x1^k \mid (x, k) \in \Pi\}$.

That is, to obtain the “unparameterized”, classical version of a parameterized problem instance, we merely write the parameter out in unary.

The notion of a composition algorithm plays a key role in the kernel lower bound machinery.

Definition 2.3. [Composition Algorithm, Compositional Problem] [15] A *composition algorithm* for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that

- takes as input a sequence $\langle (x_1, k), (x_2, k), \dots, (x_t, k) \rangle$ where each $(x_i, k) \in \Sigma^* \times \mathbb{N}$,
- runs in time polynomial in $\sum_{i=1}^t (|x_i| + k)$, and,

- outputs an instance $(y, k') \in \Sigma^* \times \mathbb{N}$ with
 1. $(y, k') \in L \iff (x_i, k) \in L$ for some $1 \leq i \leq t$, and
 2. k' is polynomial in k .

We say that a parameterized problem is *compositional* if it has a composition algorithm.

In other words, a composition algorithm for a parameterized problem acts like a polynomial-time “OR gate” for the problem, where all the input instances have the same parameter. Further, the parameter of the instance output by the composition algorithm is polynomially bounded in the input parameter.

The following theorem, due to Bodlaender et al. [15] is the cornerstone of the kernel lower bound machinery:

Theorem 2.2. [15, Lemmas 1 and 2] *Let L be a compositional parameterized problem whose derived classical problem is NP-complete. If L has a polynomial kernel, then $\text{CoNP} \subseteq \text{NP/Poly}$ and the Polynomial Hierarchy collapses to the third level.*

Another tool which we use to obtain our kernel lower bound is a notion of reductions, which is similar in spirit to those used in classical complexity to show NP-hardness results.

Definition 2.4. [18] Let P and Q be parameterized problems. We say that P is *polynomial parameter reducible* to Q , written $P \leq_{ppt} Q$, if there exists a polynomial time computable function $f : \Sigma^* \times \mathbb{N} \rightarrow \Sigma^* \times \mathbb{N}$ and a polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ such that for all $x \in \Sigma^*$ and $k \in \mathbb{N}$,

- $(x, k) \in P \iff f(x, k) \in Q$, and,
- $f(x, k) = (x', k') \implies k' \leq p(k)$

We call f a polynomial parameter transformation (or a PPT) from P to Q .

The following theorem captures the reason why this notion of a reduction is useful in showing kernel lower bounds:

Theorem 2.3. [18, Theorem 3] *Let P and Q be parameterized problems whose derived classical problems are P^c, Q^c , respectively. Let P^c be NP-complete, and $Q^c \in \text{NP}$. Suppose there exists a PPT from P to Q . Then, if Q has a polynomial kernel, then P also has a polynomial kernel.*

As a consequence, to show that the problem Q (conditionally) has no polynomial kernels, it is sufficient to show that the problem P — again, conditionally — has no polynomial kernels, and then exhibit a PPT from P to Q . Observe that this is quite similar to the way in which polynomial-time reductions are used in classical complexity to propagate NP-hardness results.

Part II

Domination Problems

Domination on $K_{i,j}$ -free Graphs

IN the PARAMETERIZED DOMINATING SET problem the input consists of a graph G and a positive integer k , and the question is whether there is a set S of at most k vertices in G — a *dominating set* of G — such that every vertex in G which is not in S is adjacent to some vertex in S ; the parameter is k . The PARAMETERIZED DOMINATING SET problem is $W[2]$ -hard, and therefore it is unlikely (See [Chapter 2](#)) that the problem has fixed-parameter tractable (FPT) algorithms or polynomial kernels.

The problem does have FPT algorithms in certain *restricted* families of graphs, such as in planar graphs [54], graphs of bounded genus [45], nowhere-dense classes of graphs [31], K_h -topological-minor-free graphs, and graphs of bounded degeneracy [3]. Before our work [99], graphs of bounded degeneracy were the most general graph class known to have an FPT algorithm for this problem. We showed that the problem has an FPT algorithm in a class of graphs that encompasses, and is strictly larger than, all the aforementioned classes — namely, the class of $K_{i,j}$ -free graphs. In this chapter, we describe these results in detail.

Recall (see [Chapter 2](#)) that for the PARAMETERIZED DOMINATING SET problem, a kernelization algorithm is an algorithm that takes (G, k) as input, runs in polynomial time, and outputs an equivalent instance (H, k') , where $k' \leq g(k)$ and H is a graph with at most $h(k)$ vertices for some computable functions g and h . Here (H, k') is equivalent to (G, k) in the sense that the graph H has a dominating set of size at most k' if and only if G has a dominating set of size at most k . H is the kernel output by this algorithm. From the equivalence of FPT and kernelization (recall the folklore [Theorem 2.1](#)) it follows that,

unless $\text{FPT}=\text{W}[2]$, there is no kernelization algorithm for PARAMETERIZED DOMINATING SET on general graphs (or on graphs with a bounded average degree, for that matter). For the same reason, the problem admits kernelization algorithms when the input is restricted to planar graphs, graphs of bounded genus, K_h -topological-minor-free graphs, or graphs of bounded degeneracy. However, the *size* of the kernel implied by the proof of [Theorem 2.1](#) is equal to the factor $f(k)$ in the running time of the corresponding FPT algorithm, and hence is exponential in k . The interesting problem is, therefore, to find if the kernel size can be made smaller — in particular, whether it can be made polynomial in k .

For the PARAMETERIZED DOMINATING SET problem, the first polynomial kernel result was obtained by Alber et al. [1] in 2004: they showed that in *planar graphs*, the problem has a *linear* kernel on at most $335k$ vertices. A linear kernel for a parameterized problem is one whose size is a linear function of the parameter k . This bound for planar graphs was later improved to $67k$ by Chen et al. [24]. Fomin and Thilikos [53] showed in 2004 that the same reduction rules as used by Alber et al. give a linear kernel (linear in $k + g$) for PARAMETERIZED DOMINATING SET restricted to graphs of genus g .

The next advances in kernelizing this problem were made by Alon and Gutner in 2008 [2, 65]. They showed that the problem has a linear kernel on $K_{3,h}$ -topological-minor-free graph classes (which include, for example, planar graphs), and a polynomial kernel in K_h -topological-minor-free graph classes. Here K_h denotes the complete graph on h vertices, and $K_{3,h}$ is the complete bipartite graph on $h + 3$ vertices where one piece of the partition has 3 vertices and the other has h . The degree of the polynomial bound on the kernel size for K_h -topological-minor-free graphs depends on h , and these are the most general class of graphs for which the problem has been previously shown to have a polynomial kernel.

In the meantime, the same authors had shown in 2007 that the problem is FPT on (the strictly larger class of) graphs of bounded degeneracy [3], but had left open the question whether the problem has a polynomial kernel on such graph classes. We answered this question in the affirmative, and showed that, in fact, even larger classes of graphs — the $K_{i,j}$ -free graph classes — admit polynomial kernels for this problem [99]. More recently, Bodlaender et al. [17] and Fomin et al. [56] have obtained general results which imply, *inter alia*, linear kernels for PARAMETERIZED DOMINATING SET in graphs of bounded genus and in apex-minor-free graphs (which are classes of graphs that exclude special graphs — called apex graphs — as a minor). In [Table 3.1](#) we summarize some FPT and kernelization results for the PARAMETERIZED DOMINATING SET problem on various classes of graphs.

| <i>Graph Class</i> | <i>FPT Algorithm Running Time</i> | <i>Kernel Size</i> |
|-------------------------------|---|---|
| Planar | $\mathcal{O}(k^4 + 2^{15.13\sqrt{k}}k + n^3)$ [54] | $\mathcal{O}(k)$ [1, 24] |
| Genus- g | $\mathcal{O}((24g^2 + 24g + 1)^k n^2)$ [45] | $\mathcal{O}(k + g)$ [53] |
| K_h -minor-free | $\mathcal{O}(n^{3.5} + 2^{\mathcal{O}(\sqrt{k})})$ [65] | $\mathcal{O}(k^c)$ [65] |
| K_h -topological-minor-free | $(\mathcal{O}(h))^{hk} \cdot n$ [3] | $\mathcal{O}(k^c)$ [65] |
| d -degenerate | $k^{\mathcal{O}(dk)} n$ [3] | $k^{\mathcal{O}(dk)}$ [3], $\mathcal{O}(k^{2(d+1)^2})^\dagger$ |
| $K_{i,j}$ -free | $\mathcal{O}(n^{i+\mathcal{O}(1)} + 2^{\mathcal{O}(k^{2i^2})})^\dagger$ | $\mathcal{O}(k^{2i^2})^\dagger$ |

Table 3.1: Some FPT and kernelization results for k -DOMINATING SET. Results described in this chapter are marked with a \dagger .

Our Results

We use $K_{i,j}$ to denote the complete bipartite graph on $i + j$ vertices where one piece of the partition has i vertices and the other part has j vertices. A graph is said to be $K_{i,j}$ -free if it does not contain $K_{i,j}$ as a (not necessarily induced) subgraph. We show that for every fixed $i, j \geq 1$, the PARAMETERIZED DOMINATING SET problem has a polynomial kernel on $K_{i,j}$ -free graphs. For input graph G and parameter k , the size of the kernel is bounded by k^c where c is a constant that depends only on i and j .

A graph G is said to be d -degenerate if every subgraph of G has a vertex of degree at most d . Since a d -degenerate graph does not have $K_{d+1,d+1}$ as a subgraph, it follows that the PARAMETERIZED DOMINATING SET problem has a polynomial kernel on graphs of bounded degeneracy. This settles a question posed by Alon and Gutner [2, 65].

A subset S of the vertex set of a graph is said to be *independent* if no two vertices in S have an edge between them in the graph. The PARAMETERIZED INDEPENDENT DOMINATING SET problem asks whether the input graph G has an independent DOMINATING SET of size at most k , with the parameter being k . We show that the PARAMETERIZED INDEPENDENT DOMINATING SET problem has a polynomial kernel on $K_{i,j}$ -free graphs.

Observe that the first three graph classes in Table 3.1 are *minor-closed* (See Chapter 2 for the definition of a minor-closed graph class.). The only other previous FPT or kernelization result for the PARAMETERIZED DOMINATING SET problem on a class of graphs which is *not* minor-closed — of which we are aware — is the $\mathcal{O}(k^3)$ kernel and the resulting

FPT algorithm for graphs that exclude triangles and 4-cycles [102]. In fact, this result can be modified to obtain similar bounds on graphs which have no 4-cycles, but may have triangles. Since a 4-cycle is a $K_{2,2}$, this latter result follows from the main result of the current chapter by setting $i = j = 2$.

Since, for a constant h , a K_h -topological-minor-free graph has bounded degeneracy [20, 65, 80], the class of $K_{i,j}$ -free graphs is more general than the class of K_h -topological-minor-free graphs. Thus we extend the class of graphs for which the PARAMETERIZED DOMINATING SET problem is known to have (1) FPT algorithms and (2) polynomial kernels, to the class of $K_{i,j}$ -free graphs. It is interesting to note that except for $K_{i,j}$ -free graphs, all the other graph classes in Table 3.1 are of bounded degeneracy, and are hence *sparse*: any d -degenerate graph on n vertices has at most dn edges. In contrast, $K_{i,j}$ -free graphs can, in general, have a super-linear number of edges; for example, Alon et al. [5] show that for sufficiently large i and for $j > (i - 1)!$, there exist $K_{i,j}$ -free graphs on n vertices with $\Omega(n^{2-1/i})$ edges.

Organization of the rest of the chapter

Throughout this chapter, n denotes the number of vertices in the input graph. In Section 3.1 we present our main kernelization algorithm that, for fixed $j \geq i \geq 2$, runs in $\mathcal{O}(n^i)$ time and constructs a kernel on $\mathcal{O}((j + 1)^{i+1}k^{i^2})$ vertices for PARAMETERIZED DOMINATING SET on $K_{i,j}$ -free graphs. As a corollary we obtain, in Section 3.2, a polynomial kernel for the problem restricted to d -degenerate graphs, where the kernelization algorithm runs in $\mathcal{O}(n^{d+1})$ time and outputs a kernel of size $\mathcal{O}((d + 2)^{d+3}k^{(d+1)^2})$. In Section 3.2.1 we describe an improvement to the above algorithm that applies to d -degenerate input graphs, yields a kernel of the same size as above, and runs in time $\mathcal{O}(2^d dn^2)$. In Section 3.3 we describe a modification of the algorithm in Section 3.1 that constructs a polynomial kernel for the PARAMETERIZED INDEPENDENT DOMINATING SET problem on $K_{i,j}$ -free graphs. This kernel has $\mathcal{O}(jk^i)$ vertices, and so implies a kernel of size $\mathcal{O}((d + 1)^2 k^{d+1})$ for this problem on d -degenerate graphs. In Section 3.4 we state our conclusions and list some open problems.

Notation

All the graphs in this chapter are finite, undirected and simple. In general we follow the graph terminology of Section 2.1. Let H be a graph obtained from a copy of a graph G by applying some changes, and let S be a vertex subset of G whose copy survives in H . For ease of presentation, we sometimes abuse notation and use S to denote the copy of S in H as well.

Note that we use the adjective “ $K_{i,j}$ -free” to denote graphs which do not contain $K_{i,j}$ as a *subgraph*. We would like to emphasize that this is different from the notion of excluding $K_{i,j}$ as an *induced* subgraph. A graph which excludes $K_{i,j}$ as an induced subgraph may indeed contain $K_{i,j}$ as a subgraph. We consider a simple example to buttress this distinction. As noted below, the PARAMETERIZED DOMINATING SET problem has a linear kernel on graphs which exclude $K_{1,4}$ as a subgraph — that is, on $K_{1,4}$ -free graphs. In stark contrast, the same problem is W[2]-hard on graphs which exclude $K_{1,4}$ as an *induced* subgraph [71].

3.1 A Polynomial Kernel for $K_{i,j}$ -free Graphs

In this section we consider the PARAMETERIZED DOMINATING SET problem on graphs that do not have $K_{i,j}$ as a subgraph, for fixed $j \geq i \geq 1$. If $k = 1$, then the problem can be solved in linear time by checking if there is a vertex which is adjacent to all other vertices in the graph. For $i = 1, j \geq i$, a graph that does not have $K_{i,j}$ as a subgraph has degree at most $j - 1$. Any set of k vertices in such a graph G can dominate at most $(j - 1)k$ other vertices, and so G is a YES instance of PARAMETERIZED DOMINATING SET only if G contains at most jk vertices. Thus the problem is (1) polynomial-time solvable when $k = 1$, and (2) has a linear vertex kernel when $i = 1, j \geq i$, and so in the rest of the chapter we restrict our attention to the cases $k > 1, j \geq i \geq 2$.

We derive a polynomial kernel for a slightly more general, coloured version of the PARAMETERIZED DOMINATING SET problem. We define an *rwb-graph* (a red-white-blue graph) to be a graph whose vertices are (arbitrarily) coloured with the three colours red, white, and blue. More precisely, an rwb-graph is a graph $G = (V, E)$ where V is partitioned into R_G, W_G , and B_G (coloured red, white, and blue, respectively). An *rwb-dominating set* of an rwb-graph G is a vertex subset $S \subseteq V$ of G such that $R_G \subseteq S$ and S dominates B_G ; that is, it contains all red vertices and dominates all blue vertices. We define the PARAMETERIZED RWB-DOMINATING SET problem as follows:

PARAMETERIZED RWB-DOMINATING SET

Input: An rwb-graph $G = (V, E)$ and a non-negative integer k .

Parameter: k

Question: Does G have an rwb-dominating set with at most k vertices?

The following simple claim shows that the coloured version of the problem is more general.

Claim 1. *Let G be a graph and H the rwb-graph obtained from G by colouring all the vertices blue. Then G has a dominating set of size at most k if and only if H has an rwb-dominating set of size at most k .*

Proof. Note that H is a copy of G with coloured vertices. Let S be a dominating set of G of size at most k . Since the set R_H of red vertices of H is empty, $R_H \subseteq S$. Since H is isomorphic to G as a graph, S dominates all vertices in H . Hence S is an rwb-dominating set of H of size at most k .

Conversely, if S is an rwb-dominating set of H of size at most k , then since all vertices in H are blue, S dominates all vertices in H . Thus S is a dominating set of G of size at most k . \square

In our kernelization algorithm for PARAMETERIZED DOMINATING SET, we first colour all the vertices of the input graph G blue to obtain an rwb-graph H . Then we apply certain *reduction rules* to H . Roughly speaking, the reduction rules try to identify (1) vertices that must necessarily be in every rwb-dominating set of H of size at most k , and (2) vertices whose deletion from H does not affect the size of a minimal rwb-dominating set of H of size at most k .

The reduction rules also colour various vertices red or white. Intuitively, the vertices coloured red are those that will be picked up by the reduction rules in the rwb-dominating set D of size at most k that we are trying to construct. In particular, if there is a k -dominating set in the graph, the rules ensure that there will be one that contains all the red vertices. Vertices which are known to have been already dominated by D are coloured white. Clearly all neighbours of red vertices are white, but our reduction rules colour some vertices white even if they have no red neighbours (at that point). These are vertices that will be dominated by one out of a small number of vertices identified by the reduction rules: See reduction rule 2 for the details. The vertices that remain blue are those that are yet to be dominated.

We first describe an algorithm that takes as input an rwb-graph G on n vertices and a positive number k , and runs in $\mathcal{O}(n^i)$ time. The algorithm either finds that G does not have any rwb-dominating set of size at most k , or it constructs an instance (H, k) on $\mathcal{O}((j+1)^{i+1}k^{i^2})$ vertices such that G has an rwb-dominating set of size at most k if and only if H has an rwb-dominating set of size at most k . To complete the kernelization procedure, we show that this instance (H, k) of PARAMETERIZED RWB-DOMINATING SET can be converted into an equivalent instance of PARAMETERIZED DOMINATING SET — that is, the colours can be removed — with a polynomially bounded increase in both the number of vertices and the parameter value.

The algorithm applies a sequence of reduction rules in a specified order. The input and output of each reduction rule are rwb-graphs.

Definition 3.1. An rwb-graph G is said to be *reduced* with respect to a reduction rule if an application of the rule to G does not change G .

The correctness of the kernelization algorithm depends on the fact that each reduction rule satisfies the following correctness condition and preserves the invariants stated below:

Definition 3.2. (*Correctness*) A reduction rule R is said to be *correct* if the following condition holds: Let (G, k) be an instance of PARAMETERIZED RWB-DOMINATING SET, and let (H, k') be the instance of PARAMETERIZED RWB-DOMINATING SET obtained from (G, k) by one application of rule R . Then H has an rwb-dominating set D' of size at most k' if and only if G has an rwb-dominating set D of size at most k .*

We ensure that the following invariants are maintained after every application of each reduction rule.

Invariants:

1. None of the reduction rules introduces a $K_{i,j}$ into a graph.
2. In the rwb-graphs constructed by the algorithm, no red vertex has a blue neighbour.
3. Let R_1 and R_2 be two reduction rules such that R_1 precedes R_2 in the order in which the rules are presented below. Suppose (G_1, k_1) is reduced with respect to R_1 and (G_2, k_2) is obtained by an application of rule R_2 to (G_1, k_1) . Then (G_2, k_2) is reduced with respect to R_1 .

3.1.1 The reduction rules and the kernelization algorithm

The kernelization algorithm assumes that the input graph is an rwb-graph. It applies the following rules exhaustively *in the given order*. Each rule is repeatedly applied till it causes no changes to the graph and then the next rule is applied.

We use some notational conventions in this section. For each rule below, (G, k) denotes the instance on which the rule is applied, and (H, k') the instance that is obtained when the rule is applied to (G, k) . Further, D, D', k and k' are as in [Definition 3.2](#): D is an rwb-dominating set of size k of G , and D' an rwb-dominating set of H of size k' .

Our first reduction rule is simple to state, and its correctness is almost self-evident:

* Note, however, that none of our reduction rules changes the value of k , and so $k' = k$ for every one of these rules.

Rule 1. Let B be the set of all isolated blue vertices in G .

1. Colour all vertices in B red.
2. Set $k' := k$.

Lemma 3.1. *Rule 1 is correct and preserves the invariants.*

Proof. Let (G, k) be the instance on which the rule is applied, and (H, k) the resulting instance. Let I be the set of isolated blue vertices in G .

Let D be an rwb-dominating set of G of size at most k . From the definition of an rwb-dominating set, $R_G \subseteq D$. Since an isolated vertex can only be dominated by itself, $I \subseteq D$. Since the only thing that the rule does is to colour isolated blue vertices of G red, $R_H = R_G \cup I$, and so $R_H \subseteq D$. Set $D' = D$ in H . Then D' dominates every vertex in H , $R_H \subseteq D'$, and $|D'| \leq k$. Thus D' is an rwb-dominating set of H of size at most k .

Conversely, let D' be an rwb-dominating set of H of size at most k . Set $D = D'$ in G . Since the only thing that the rule does is to colour isolated blue vertices of G red, $R_G \subseteq R_H \subseteq D' = D$, and so D is an rwb-dominating set of G of size at most k . Thus **Rule 1** is correct.

The rule trivially preserves the first two invariants, and vacuously preserves the third. \square

The next reduction rule is somewhat more involved, and may look mysterious at first. To motivate this rule, observe that if the *maximum degree* of a vertex in the input graph is Δ , then any set of k vertices in the graph can dominate at most $k\Delta$ vertices, and so the total number of vertices in a YES instance is at most $k(\Delta + 1)$. This is precisely the argument that we used at the beginning of this section to obtain a kernel on $\mathcal{O}(jk)$ vertices for $K_{1,j}$ -free graphs. It is not clear, however, that this observation helps in any way for bounding the size of YES instances in $K_{i,j}$ -free graphs when $i \geq 2$. A $K_{3,10}$ -free graph, for instance, can have a vertex of arbitrarily large degree, and the observation which relies on a bounded maximum degree does not seem to be relevant in this case.

It turns out, in fact, that the bounded-degree argument *does* apply, but in a slightly more involved manner, and after a bit of preprocessing. To see how, consider again the case of a $K_{3,10}$ -free graph G , which may contain vertices of arbitrarily large degree. Since a bounded-degree argument does not directly apply to G , we look instead at *pairs* of vertices which have a large *common* neighbourhood. So let u, v be two vertices which have more than $10k$ common neighbours in G , and let B be this set of common neighbours. We claim that if G has a dominating set of size at most k , then *at least one* of $\{u, v\}$ must be present in *every* such dominating set.

To see why, observe that no vertex $w \notin \{u, v\}$ has 10 or more neighbours in B , or else the subgraph of G induced by the vertices $\{u, v, w\}$ and their common neighbours contain a $K_{3,10}$, a contradiction. Thus *any* vertex other than u, v can dominate at most 10 vertices which are in B ; this maximum is attained when a vertex in B dominates 9 other vertices in B . Any set S of at most k vertices not intersecting $\{u, v\}$ can therefore dominate at most $10k$ vertices in B . Since B contains at least $10k + 1$ vertices, S cannot dominate all the vertices in B , and therefore cannot be a dominating set of G ; the claim follows.

For $K_{3,10}$ -free graphs we may thus have a reduction rule which says that if two vertices u and v have a sufficiently large *common* neighbourhood, then we can colour all the common neighbours of u and v white; this is because at least one of u, v is guaranteed to be in any solution. This rule is, admittedly, somewhat weak compared to the bounded-degree argument which we had for $K_{1,j}$ -free graphs. Further, it does not seem to cause any progress: neither does it increase the number of vertices forced into the eventual solution set, nor does it reduce the size of the instance. It turns out, however, that by repeatedly applying this rule and its variants, we can in fact reduce the input instance to a state where a bounded-degree argument applies. Our next reduction rule, which is in fact a sequence of reduction rules, is motivated by these considerations:

Rule 2. This is a sequence of $i - 2$ rules, named Rule 2.1, Rule 2.2, \dots , Rule 2. $(i - 2)$. The kernelization algorithm first applies Rule 2.1 exhaustively, till it causes no more changes in the graph. Then it applies the next rule in the sequence exhaustively, and so on. The first rule in this sequence is as follows:

Rule 2.1. Let $U = \{u_1, u_2, \dots, u_{i-1}\}$ be a set of $(i - 1)$ vertices in G , none of which is red. Let B be the set of *common* blue neighbours of the vertices in U . If $|B| > b = jk$, then:

1. Add $(i - 1)$ new (*gadget*) vertices $X = \{x_1, x_2, \dots, x_{i-1}\}$ and all the edges $\{u, x\}; u \in U, x \in X$ to G , as in [Figure 3.1](#).
2. Colour all the vertices in B white.
3. Colour all the vertices in X blue.
4. Set $k' := k$.

For $p \in \{2, 3, \dots, i - 2\}$, Rule 2. p is defined as follows:

Rule 2. p . Let $b = jk^p + k^{p-1} + k^{p-2} + \dots + k$. Let $U = \{u_1, u_2, \dots, u_{i-p}\}$ be a set of $(i - p)$ vertices in G , none of which is red. Let B be the set of *common* blue neighbours of the vertices in U . If $|B| > b$, then:

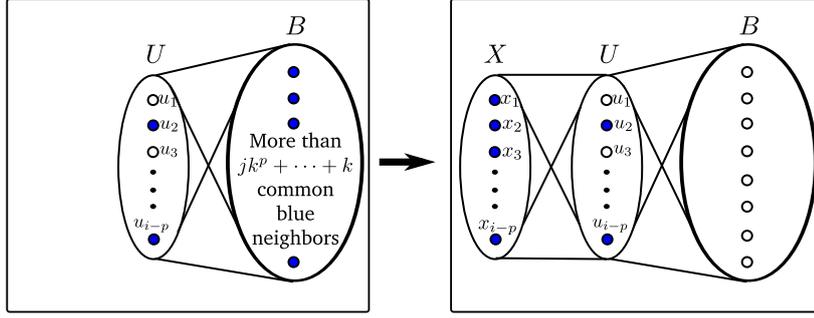


Figure 3.1: **Rule 2.p.** U is a set of $i - p$ white or blue vertices which have a sufficiently large set B of *common* blue neighbours. The rule adds a set X of $i - p$ blue neighbours adjacent to all the vertices in U and colours all the vertices in B white.

1. Add $(i - p)$ new (*gadget*) vertices $X = \{x_1, x_2, \dots, x_{i-p}\}$ and all the edges $\{u, x\}; u \in U, x \in X$ to G , as in [Figure 3.1](#).
2. Colour all the vertices in B white.
3. Colour all the vertices in X blue.
4. Set $k' := k$.

Proposition 3.1. For $1 \leq p \leq (i - 2)$, Rule 2.p preserves all the three invariants.

Proof. Since we have shown that [Rule 1](#) preserves all the invariants, we can assume inductively that all the rules that are applied before Rule 2.p preserve all the three invariants. We now consider the behaviour of Rule 2.p for each of the invariants:

Invariant 1. From the inductive assumption, and from the fact that the input graph is $K_{i,j}$ -free, it follows that the graph G on which Rule 2.p is applied is $K_{i,j}$ -free. Suppose the graph H resulting from the application of the rule contains a $K_{i,j}$, say K , that is introduced by the rule. Then K must necessarily contain at least one of the newly added vertices in X , or else $G = H \setminus X$ would also contain K . Since each vertex in X has degree exactly $(i - p) < i$ in H , no vertex in X can be part of a $K_{i,j}$ in H , and it follows that there is no $K_{i,j}$ in H .

Invariant 2. From the inductive assumption, the invariant holds for the graph G on which Rule 2.p is applied. The rule does not introduce new red vertices or colour existing non-red vertices red. Further, it does not add new vertices as neighbours to any existing red vertex — observe that all vertices in U are non-red. Hence it follows that the rule preserves this invariant.

Invariant 3. Rule 2.1 preserves the invariant since it does not introduce isolated blue vertices into the graph. Assume inductively that for $2 \leq p \leq i-2$, Rules 2.1, \dots , 2. $(p-1)$ preserve the invariant. So the graph G on which Rule 2. p is applied is reduced with respect to Rules 1, 2.1, \dots , 2. $(p-1)$. Let H be the graph that results when Rule 2. p is applied to G . Then H is reduced with respect to Rule 1, since Rule 2. p does not introduce isolated blue vertices in H .

Suppose H is *not* reduced with respect to Rule 2. q for some $1 \leq q \leq (p-1)$. Then H contains a set $U = \{u_1, u_2, \dots, u_{i-q}\}$ of $(i-q)$ non-red vertices such that U has more than b common blue neighbours B , where $b = jk$ if $q = 1$ and $b = jk^q + k^{q-1} + k^{q-2} \dots + k$ otherwise. Either U or B (or both) must necessarily contain at least one of the newly added vertices in X , or else $G = H \setminus X$ would also be not reduced with respect to Rule 2. q . Note that each vertex in X has degree exactly $(i-p)$ in H . Each vertex in U has degree at least b , and every vertex in B has degree at least $(i-q)$. Since $p > q$, we have $(i-p) < (i-q)$. Since $i \leq j$ and $k \geq 1$, it follows that $(i-p) < b$. Thus no vertex in X can be part of either U or B in H . It follows that H is reduced with respect to Rule 2. q , and hence Rule 2. p preserves this invariant.

Thus the rule preserves all the three invariants. \square

Claim 2. Consider an application of Rule 2. p , $1 \leq p \leq i-2$. If U is a set of vertices of G that satisfies the condition in Rule 2. p , then in every subset of $V(G)$ of size at most k that dominates B , there must be at least one vertex which is in U .

Proof. Let $p = 1$. Suppose there is a vertex set $S \subseteq V(G)$ of size at most k such that (1) S dominates B , and (2) S does not contain any vertex of U . Since $|B| > jk$, there is a vertex v in S that dominates at least $j+1$ vertices in B . Let $T = N(v) \cap B$. Then $|T| \geq j$, and the vertex sets $\{U \cup \{v\}, T\}$ form the two parts of a $K_{i,j}$ in G . This contradicts the $K_{i,j}$ -free property of the input graph or the first invariant.

Now let $2 \leq p \leq (i-2)$. Let $S \subseteq V(G)$ be a set of size at most k that dominates B and does not contain any vertex of U . Since $|B| > b$, there is a vertex $v \in S$ that dominates at least $(b/k)+1$ vertices in B . Because of the second invariant, v is not red. Let $T = N(v) \cap B$. Then $|T| \geq (b/k)$, and T is in the common neighbourhood of $(U \cup \{v\})$. Thus $(U \cup \{v\})$ is a set of $(i-(p-1))$ vertices in G , none of which is red, and which have at least $b/k > jk^{p-1} + k^{p-2} + \dots + k$ common blue neighbours. This contradicts the fact that G is reduced with respect to Rule 2. $(p-1)$. \square

We now argue that the gadget and the colouring correctly capture the structural property guaranteed by Claim 2.

Proposition 3.2. *Rule 2.p is correct for $1 \leq p \leq (i - 2)$.*

Proof. Let D be an rwb-dominating set of G of size at most k , and let U be as in the statement of Rule 2.p. Set $D' := D$. Since D is an rwb-dominating set of G , $R_G \subseteq D$. Since the rule does not add any new red vertices in H , it follows that $R_H \subseteq D'$. Since (1) D dominates all blue vertices of G , and (2) the rule removes no edges from G , it follows that $D' = D$ dominates all blue vertices in H that are copies of blue vertices in G . By Claim 2, $D \cap U \neq \emptyset$, and so $D' \cap U \neq \emptyset$. Since all the new blue vertices added to H — namely, those which constitute the set X — are adjacent to every vertex in U by construction, D' dominates all blue vertices in H . Thus D' is an rwb-dominating set of H of size at most k .

Conversely, let D' be an rwb-dominating set of H of size at most k . We consider three cases:

$D' \cap U = \emptyset$. In this case, since D' dominates X and X is an independent set, $X \subseteq D'$. Set $D := (D' \setminus X) \cup U$. Since X and U are disjoint sets of equal cardinality, $|D| \leq |D'| \leq k$. Since D' is an rwb-dominating set of H , $R_H \subseteq D'$. Since all vertices in X are blue and since the reduction rule does not delete any red vertex from G to obtain H , it follows that $R_G \subseteq D$. Now, all the blue vertices dominated by X in H are contained in U and $U \subseteq D$. Further, the set of vertices which are blue in G and white in H is exactly the set B , and each vertex in U dominates all vertices in B . Therefore D dominates all blue vertices in G , and thus D is an rwb-dominating set of G of size at most k .

$D' \cap X = \emptyset$. In this case, since D' dominates X , it follows that $D' \cap U \neq \emptyset$. Set $D := D'$. Then $|D| = |D'| \leq k$. For the same reasons as above, $R_G \subseteq D$, and D dominates all vertices (namely, the set B) which are blue in G and white in H . Since D' dominates all blue vertices in H , and since G can be obtained from H by deleting a set (namely, X) of blue vertices and making the set B of white vertices blue, $D = D'$ dominates all blue vertices in G . Thus D is an rwb-dominating set of G of size at most k .

$D' \cap U \neq \emptyset, D' \cap X \neq \emptyset$. In this case, pick an arbitrary vertex $v \in B$ and set $D := (D' \setminus X) \cup \{v\}$. Since $D' \cap X \neq \emptyset$, it follows that $|D| \leq |D'| \leq k$. For the same reasons as above, $R_G \subseteq D$, and D dominates all vertices (namely, the set B) which are blue in G and white in H . Since D' dominates all blue vertices in H , $D' \setminus X$ dominates all blue vertices in $H \setminus X$, except possibly for some blue vertices in U whose only neighbours in D' belong to X . But the vertex v dominates all vertices in U , and so D is an rwb-dominating set of G of size at most k .

These three cases are exhaustive, and so it follows that for $1 \leq p \leq (i - 2)$, Rule 2.p is correct. \square

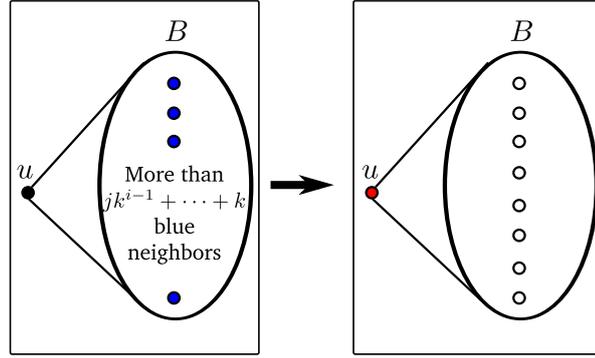


Figure 3.2: **Rule 3.** u is a white or blue vertex which has a sufficiently large set B of blue neighbours. The rule colours u red, and colours all vertices in B white.

Putting together [Proposition 3.2](#) and [Proposition 3.1](#) we obtain

Lemma 3.2. For $1 \leq p \leq (i - 2)$, Rule 2. p is correct and preserves all the three invariants.

At this point we are in a position to invoke a bounded-degree argument: every vertex with sufficiently many *blue* neighbours is — as we justify below — forced in any solution. Hence we have:

Rule 3. Let u be a blue or white vertex in G , and let B be the set of blue neighbours of u . If $|B| > h = jk^{i-1} + k^{i-2} + \dots + k^2 + k$, then (See [Figure 3.2](#)):

1. Colour u red.
2. Colour all vertices in B white.
3. Set $k' := k$.

Claim 3. Consider an instance of applying [Rule 3](#). If u is a vertex of G that satisfies the condition in the rule, then u must be in every subset of $V(G)$ of size at most k that dominates B .

Proof. Let $S \subseteq V(G)$ be a set of size at most k that dominates B . If S does not contain u , then there is a $v \in S$ that dominates at least $(h/k) + 1$ of the vertices in B . The vertex v is not red (because of the second invariant), and u, v have $h/k > jk^{i-2} + k^{i-3} + \dots + 1$ common blue neighbours, a contradiction to the fact that G is reduced with respect to Rule 2. $(i - 2)$. \square

Lemma 3.3. [Rule 3](#) is correct and preserves all the three invariants.

Proof. Let D be an rwb-dominating set of G of size at most k , and let u be as in the statement of [Rule 3](#). Set $D' := D$. Since D is an rwb-dominating set of G , $R_G \subseteq D$. From [Claim 3](#), $u \in D$. The rule does not add any new vertex to G to obtain H . Since u is the only vertex that is red in H and not red in G , $R_H = (R_G \cup \{u\}) \subseteq D'$. Since (1) D dominates all blue vertices of G , (2) the rule does not add new blue vertices or make non-blue vertices blue, and (3) the rule removes no edges from the graph, it follows that $D' = D$ dominates all blue vertices in H . Thus D' is an rwb-dominating set of H of size at most k .

Conversely, let D' be an rwb-dominating set of H of size at most k . Then from [Claim 3](#), $u \in D'$. Set $D := D'$. Since D' is an rwb-dominating set of H , $R_H \subseteq D'$. Since $R_G = R_H \setminus \{u\}$, $R_G \subseteq D$. Since (1) D' dominates all blue vertices in H , (2) the only blue vertices in G that are not blue in H are in $B \cup \{u\}$, and (3) $u \in D$ dominates $B \cup \{u\}$, it follows that $D = D'$ dominates all blue vertices in G . Thus D is an rwb-dominating set of G of size at most k .

As for the invariants, [Rule 3](#) does not change the structure of the graph. Further, it gives the colour white to all blue neighbours of the only vertex — namely, u — whose colour it changes to red. It follows that [Rule 3](#) preserves all the three invariants. \square

Recall that we started the reduction from an rwb-graph which had *no* white vertices, and we ensured that every reduction rule is correct. It follows from our notion of correctness (See [Definition 3.2](#)) that if the current rwb-graph G has an rwb-dominating set of size at most k , then all vertices which are coloured white in G are guaranteed to be dominated by *some* rwb-dominating set D of G of size at most k . This opens up the possibility that we may be able to deduce that some white vertices are not required in some solution, essentially since every blue vertex that they dominate is also dominated by some other vertices in the solution. The next two reduction rules remove white vertices which are dispensable, in the sense that there exists a solution which does not contain these vertices.

It is intuitively clear that a white vertex that has no blue neighbour is dispensable. It is also true that a white vertex which has just one blue neighbour is dispensable as well; the intuition is that we might instead pick into any solution the one blue neighbour, with no loss of generality. This motivates our next reduction rule:

Rule 4. If a white vertex u is adjacent to at most one blue vertex in G , then

1. Delete u from G ,
2. Set $k' := k$, and
3. Apply [Rule 1](#).

Lemma 3.4. *Rule 4 is correct and preserves all the three invariants.*

Proof. Since we have already proved that Rule 1 is correct and preserves the three invariants, it suffices to show that the first two steps of Rule 4 have the stated properties*. We now proceed to do this; in the following, whenever we refer to Rule 4, we mean the first two steps of this rule.

Let D be an rwb-dominating set of G of size at most k , and let u be as in the statement of Rule 4. We consider two cases:

$u \notin D$. In this case, set $D' := D$. Since D is an rwb-dominating set of G , and since the rule only deletes a white vertex $u \notin D$ to obtain H , it follows that D' is an rwb-dominating set of H of size at most k .

$u \in D$. In this case, let $A = (N(u) \cap B_G)$ be the set of blue neighbours of u in G . Note that $|A| \leq 1$. Set $D' := (D \setminus \{u\}) \cup A$. Since $|A| \leq 1$, $|D'| \leq |D| \leq k$. Since D is an rwb-dominating set of G , $R_G \subseteq D$. Since the rule only deletes a white vertex u to obtain H , it follows that $R_H = R_G \subseteq D'$. Since (1) D dominates all blue vertices of G , (2) D' contains A , the set of all blue vertices of G dominated by the vertex u that the rule deletes, and (3) the rule removes no edges from the graph other than those adjacent to the removed white vertex u , it follows that $D' = D$ dominates all blue vertices in H . Thus D' is an rwb-dominating set of H of size at most k .

Conversely, let D' be an rwb-dominating set of H of size at most k . Set $D := D'$. Since D' is an rwb-dominating set of H , and since G can be obtained from H by adding a white vertex u and some edges incident on u to H , $D = D'$ is an rwb-dominating set of G of size at most k .

As for the invariants, since Rule 4 only deletes a vertex, its application cannot introduce any of the subgraphs that make it possible to apply Rule 2 or Rule 3 to H . It is possible that new isolated blue vertices may be introduced in H , but then the application of Rule 1 ensures that such vertices do not survive once Rule 4 is completely applied. It follows that Rule 4 preserves all the three invariants. \square

At this stage each white vertex has at least two blue neighbours. If two white vertices u, v have the *same* set of blue vertices as their respective neighbourhoods, then it is intuitively clear that one of these white vertices — say u — is dispensable in the sense described earlier: we can, without loss of generality, replace u with v in any rwb-dominating set. This intuition is formalized below, and justifies our next reduction rule:

* Except for arguing that the rule preserves the invariants: see below.

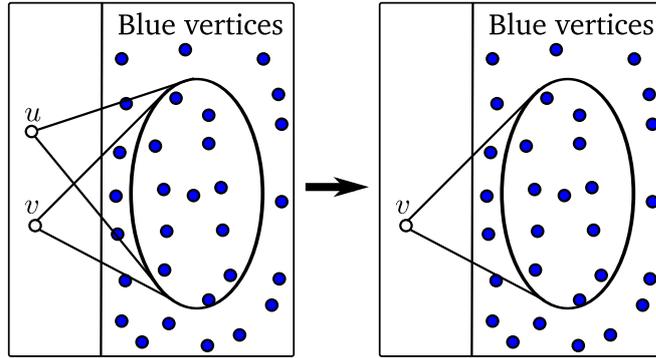


Figure 3.3: **Rule 5.** u is a white vertex whose set of blue neighbours is identical to the blue neighbourhood of a white or blue vertex v . The rule deletes u .

Rule 5. Let u be a white vertex in G , and let v be a white or blue vertex. If these two vertices have identical sets of blue neighbours — that is, if $N[u] \cap B_G = N[v] \cap B_G$ — then (See Figure 3.3):

1. Delete u from G
2. Set $k' := k$

Lemma 3.5. *Rule 5 is correct and preserves all the three invariants.*

Proof. Let D be an rwb-dominating set of G of size at most k , and let u, v be as in the statement of Rule 5. We consider two cases:

$u \notin D$. In this case, set $D' := D$. Since D is an rwb-dominating set of G , and since the rule only deletes a white vertex $u \notin D$ to obtain H , it follows that D' is an rwb-dominating set of H of size at most k .

$u \in D$. In this case, set $D' := (D \setminus \{u\}) \cup \{v\}$. Then $|D'| = |D| \leq k$. Since D is an rwb-dominating set of G , $R_G \subseteq D$. Since the rule only deletes a white vertex u to obtain H , it follows that $R_H = R_G \subseteq D'$. Since (1) D dominates all blue vertices of G , (2) D' contains a vertex — namely, v — that dominates all blue vertices dominated by the vertex u that the rule deletes, and (3) the rule removes no edges from the graph other than those adjacent to the removed white vertex u , it follows that $D' = D$ dominates all blue vertices in H . Thus D' is an rwb-dominating set of H of size at most k .

Conversely, let D' be an rwb-dominating set of H of size at most k . Set $D := D'$. Since D' is an rwb-dominating set of H , and since G can be obtained from H by adding a white vertex u and some edges incident on u to H , $D = D'$ is an rwb-dominating set of G of size at most k .

Since [Rule 5](#) only deletes a vertex, its application cannot introduce any of the subgraphs that make it possible to apply [Rule 2](#), [Rule 3](#), or [Rule 4](#) to H . No new isolated blue vertex is introduced in H , since u is the only deleted vertex, and all blue neighbours of u have at least one other neighbour that survives in H , namely the vertex v . It follows that [Rule 5](#) preserves all the three invariants. \square

If there are too many red or blue vertices, then we can conclude that the input is a No instance:

Rule 6. If the graph G contains more than k red vertices or more than $jk^i + k^{i-1} + k^{i-2} + \dots + k^2$ blue vertices, then set (H, k') to be a trivial No-instance of the problem; for instance, make H the independent set on two blue vertices and set $k' = 1$. If neither of these conditions hold, set $H := G, k' := k$.

Lemma 3.6. [Rule 6](#) is correct and preserves all the three invariants.

Proof. Note that if the instance (G, k) satisfies neither of the two conditions, then the rule returns the instance unchanged. Therefore, to show that the rule is correct, it is sufficient to show that an instance (G, k) that satisfies either of the two conditions is a No-instance.

If $|R_G| > k$, then since every rwb-dominating set of G must contain all of R_G , G has no rwb-dominating set of size at most k .

From the second invariant, no blue vertex of G has a red neighbour. Since G is reduced with respect to [Rule 1](#) to [Rule 5](#), no white or blue vertex in G has more than $jk^{i-1} + k^{i-2} + \dots + k$ blue neighbours, or else [Rule 3](#) would have applied, contradicting the third invariant. So k white or blue vertices in G can dominate at most $jk^i + k^{i-1} + k^{i-2} + \dots + k^2$ blue vertices. Hence if $|B_G| > jk^i + k^{i-1} + k^{i-2} + \dots + k^2$, then no set of k vertices in G can dominate all of B_G , and so in this case G has no rwb-dominating set of size at most k .

The reduction rule either returns the instance unchanged or returns a simple No-instance. In both cases, it trivially satisfies all the three invariants. \square

3.1.2 Algorithm correctness, running time, and kernel size

Recall that the input to the kernelization algorithm is a pair (G, k) where G is an rwb-graph and k is a non-negative integer. The algorithm applies [Rule 1](#) to [Rule 6](#), in this order, to (G, k) , exhaustively applying each rule before applying the next. From the correctness of [Rule 1](#) to [Rule 6](#) — see [Lemma 3.1](#) to [Lemma 3.6](#) — we obtain

Lemma 3.7. *The kernelization algorithm is correct: Let (G, k) be the input to the algorithm. If the algorithm says No, then G does not have an rwb-dominating set of size at most k . Otherwise, let H be the rwb-graph output by the algorithm. Then G has an*

*rwb-dominating set of size at most k if and only if H has an *rwb-dominating set of size at most k .**

We now show that the kernelization algorithm runs in polynomial time. To do so, we first show that the algorithm does not add too many gadget vertices to the input graph.

Claim 4. *Let (G, k) be the input to the kernelization algorithm, where G is a $K_{i,j}$ -free *rwb-graph on n vertices. The total number of gadget vertices that the algorithm adds to the graph, over ALL applications of rules 2.1 to 2. $(i-2)$, is less than n .**

Proof. We reuse the notation used to describe rules 2.1 to 2. $(i-2)$. Rule 2.1 colours all vertices in B white, and adds the new all-blue gadget vertex set X to the graph. The set B contains at least $(jk+1)$ blue vertices, and the set X has exactly $(i-1)$ blue vertices. Thus one application of Rule 2.1 reduces the count of blue vertices in the graph by at least $(jk-i+2)$. By a similar argument, we can see that for $2 \leq p \leq i-2$, each application of Rule 2. p reduces the count of blue vertices by at least $(jk^p + k^{p-1} + \dots + k - i + p + 1)$. Since, by assumption, $j \geq i \geq 2$ and $k \geq 2$, it follows that $(jk-i+2) \leq (jk^p + k^{p-1} + \dots + k - i + p + 1)$ for $2 \leq p \leq i-2$. Thus each application of one of the rules 2.1 to 2. $(i-2)$ reduces the total number of blue vertices in the graph by at least $(jk-i+2)$. Also, observe that the number of gadget vertices added to the graph in each application of one of the rules 2.1 to 2. $(i-2)$ is at most $(i-1)$, where this maximum is attained for Rule 2.1.

Consider an application of Rule 2. p for some $1 \leq p \leq i-2$. A blue or white vertex can be part of a set U as mentioned in the rule only if it has at least $jk^p + k^{p-1} + k^{p-2} + \dots + k$ blue neighbours. Since the maximum number of blue neighbours that a gadget vertex can have is $i-1$, it follows that no gadget vertex will ever be part of the set U in any application of Rule 2. p . Since the *rwb-graph* given as input to the kernelization algorithm has exactly n blue vertices, rules 2.1 to 2. $(i-2)$ can thus be applied at most $n/(jk-i+2)$ times in total, over the full course of the algorithm. So the total number of gadget vertices added to the graph over all applications of rules 2.1 to 2. $(i-2)$ is $n(i-1)/(jk-i+2)$, and this number is less than n since we assume that k is at least 2. \square

This bound on the number of gadget vertices helps in showing that the algorithm runs in polynomial time.

Lemma 3.8. *The kernelization algorithm can be implemented in such a way as to run in $O(\max(n^2, in^i))$ time when the input instance is a $K_{i,j}$ -free *rwb-graph G on n vertices and m edges.**

Proof. We assume that the input *rwb-graph* is given in the form of a modified adjacency list. This representation differs from the standard adjacency list representation in two ways:

1. There is a provision for colouring each vertex red, white, or blue.
2. Let u, v be two vertices such that $\{u, v\}$ is an edge in the graph. Let v_u be the node for v in the adjacency list of u , and u_v the node for u in the adjacency list of v . Then v_u contains a pointer to u_v , and u_v contains a pointer to v_u .

This is not a costly assumption: Observe that we can add a new vertex x to such a modified adjacency list L in time linear in the number of edges from x to the vertices which are already present in L . It follows that one can convert an adjacency matrix or adjacency list representation of the input rwb -graph to the modified form in time linear in the size of the original representation.

Observe that [Claim 4](#) implies that the total number of vertices in the graph at any point during the execution of the algorithm does not exceed $2n$. We now analyze the time taken to exhaustively apply each rule.

Rule 1, Rule 3 Each application of one of these two rules colours at least one blue vertex red. No reduction rule changes the colour of a red vertex. From the bound on the total number of vertices in the graph, it follows that [Rule 1](#) and [Rule 3](#) can be applied at most $2n$ times each. One application of each of these two rules can be done in $\mathcal{O}(n)$ time by a constant number of scans of the vertex list in the input graph, and so both these rules can be applied exhaustively in $\mathcal{O}(n^2)$ time.

Rule 2 As argued in the proof of [Claim 4](#), no gadget vertex need ever be considered for inclusion in the set U in any application of [Rule 2.p](#). For applying [Rule 2.p](#) for a fixed $1 \leq p \leq i - 2$, therefore, the algorithm iterates over all $(i - p)$ -subsets of the set of all *original* (not gadget) vertices which are blue or white (at this point). This can be done in $\mathcal{O}\binom{n}{i-p}$ time, as was first shown by Ehrlich [[44](#)]. For each such subset S , the algorithm finds the set of common blue neighbours of S as in [Algorithm 1](#). Since the total number of possible blue vertices — including gadget vertices — is at most $2n$ (see [Claim 4](#)), this can be done in time $\mathcal{O}((i - p)n)$. A straightforward implementation of the remaining part of [Rule 2.p](#) runs in $\mathcal{O}(n + (i - p)^2) = \mathcal{O}(n)$ time (since $i - p = \mathcal{O}(1)$), and so [Rule 2.p](#) can be exhaustively applied in $\binom{n}{i-p} \cdot (\mathcal{O}((i - p)n) + \mathcal{O}(n)) = \frac{n}{i-p} \binom{n-1}{i-p-1} \cdot \mathcal{O}((i - p)n) = \mathcal{O}(n^{i-p+1})$ time. All the rules [2.p](#); $1 \leq p \leq i - 2$ can therefore be exhaustively applied in $\mathcal{O}(in^i)$ time.

Rule 4 Each application of this rule deletes at least one white vertex. From the bound on the total number of vertices in the graph, it follows that the rule can be applied at most $2n$ times. Each application of the rule essentially consists of the deletion

Algorithm 1 Rule 2. p : Finding the set of common blue neighbours of a vertex subset S .

```

1:  $A \leftarrow$  A binary array with indices ranging from 1 to  $|V(G)|$ , initialized to all 0s.
2:  $x \leftarrow$  A vertex in  $S$ .
3: for Each vertex  $y$  in the adjacency list of  $x$  do
4:   if  $y$  is blue then
5:      $A[y] \leftarrow 1$ 
6:   end if
7: end for
8: for Each vertex  $z \in S \setminus \{x\}$  do
9:   for Each vertex  $y$  in the adjacency list of  $z$  do
10:    if  $y$  is not blue then
11:       $A[y] \leftarrow 0$ 
12:    end if
13:   end for
14: end for
15: return  $\{v \in V(G); A[v] = 1\}$ 

```

of one vertex from the graph. This can be done in time linear in the degree of this vertex, by making use of the pointers present in the data structure.

From [Claim 4](#), less than n new (gadget) vertices are added to the graph by the kernelization algorithm. As argued in the proof of [Claim 4](#), no gadget vertex need ever be considered for inclusion in the set U in any application of Rule 2. p . Thus the only edges at a gadget vertex a , at any point during the algorithm, are the ones added by the particular application of Rule 2. p which introduced the vertex a in the graph. Therefore each new vertex has degree at most $i - 1$ — this bound is attained for vertices added by Rule 2.1. Thus the total number of edges added to the graph is at most $(n - 1)(i - 1)$, and so the total number of edges in the graph is at most $m + (i - 1)(n - 1) = \mathcal{O}(m + in)$. It follows that the rule can be applied exhaustively in $\mathcal{O}(m + in) = \mathcal{O}(n^2)$ time.

Rule 5 This rule can be exhaustively applied in $\mathcal{O}(|G|) = \mathcal{O}(n^2)$ time, as follows. Two vertices in a graph are said to be *twins* if they have identical neighbourhoods in the graph. Observe that this defines an equivalence relation on the set of vertices. Habib et al. [66] show how to find the equivalence classes of this relation in a graph G — that is, how to partition the vertex set of G into classes of twins — in $\mathcal{O}(|G|)$ time. A small modification of their algorithm yields a partition of the vertex set of the rwb-graph G , where each class consists of all white or blue vertices which have identical blue neighbourhoods. More specifically, we set the pivots — as defined in their algorithm — to be the set of blue vertices, and the pivot set of each blue

vertex to be its closed neighbourhood. It is straightforward to verify that with this modification, their algorithm yields a partition of $V(G)$ of the desired kind in $\mathcal{O}(|G|)$ time. We now go through each equivalence class, and if a class contains a white vertex u and at least one other vertex, then we delete u . By making use of the pointers present in the data structure that represents the graph G , all these deletions can be effected in $\mathcal{O}(|G|)$ time.

Putting all these together, the kernelization algorithm can be implemented to run in $\mathcal{O}(\max(n^2, in^i))$ time. \square

Now we prove a polynomial bound on the size of the reduced instance.

Lemma 3.9. *Let (G, k) be the input to the kernelization algorithm. If the algorithm outputs the instance (H, k) , then $|V(H)| = \mathcal{O}((j+1)^{i+1}k^i)$.*

Proof. From [Rule 6](#), we get $|R_H| \leq k$ and $b = |B_H| \leq jk^i + k^{i-1} + \dots + k \leq (j+1)k^i$. Now we bound $|W_H|$. Note that no two white vertices in H can have identical blue neighbourhoods, or else [Rule 5](#) would have applied. Also, each white vertex has at least two blue neighbours, or else [Rule 4](#) would have applied. Hence the number of white vertices in H that have less than i blue neighbours is at most $\binom{b}{2} + \binom{b}{3} + \dots + \binom{b}{i-1} \leq 2b^{i-1}$. No set of i blue vertices in H has more than $(j-1)$ common white neighbours, or else these form a $K_{i,j}$. Hence the number of white vertices that have i or more blue neighbours in H is at most $\binom{b}{i}(j-1) \leq (j-1)b^i$. So the total number of white vertices in H ,

$$\begin{aligned} |W_H| &\leq 2b^{i-1} + (j-1)b^i \\ &= (2 + (j-1)b)b^{i-1} \\ &\leq (j+1)b^i \\ &\leq (j+1)((j+1)k^i)^i \\ &= (j+1)^{i+1}k^i \end{aligned}$$

The bound in the lemma follows. \square

From [Lemma 3.8](#) [Lemma 3.9](#) we obtain

Corollary 3.1. *For every fixed $j \geq i \geq 1$, the PARAMETERIZED RWB-DOMINATING SET problem on $K_{i,j}$ -free graphs has a polynomial kernel with $\mathcal{O}((j+1)^{i+1}k^i)$ vertices.*

3.1.3 Removing the colours

By [Claim 1](#), the PARAMETERIZED RWB-DOMINATING SET problem is a more general version of the PARAMETERIZED DOMINATING SET problem. By [Corollary 3.1](#), PARAMETERIZED RWB-DOMINATING SET has a polynomial kernel on $K_{i,j}$ -free graphs, and therefore, intuitively, so should PARAMETERIZED DOMINATING SET. This intuition is in fact justified, because the notion of PARAMETERIZED RWB-DOMINATING SET being “more general” than PARAMETERIZED DOMINATING SET captures the fact that there is a “nice” polynomial-time many-to-one reduction from PARAMETERIZED DOMINATING SET to PARAMETERIZED RWB-DOMINATING SET. To be more precise:

- By [Claim 1](#), PARAMETERIZED DOMINATING SET polynomial-time reduces to PARAMETERIZED RWB-DOMINATING SET, and the reduction *preserves the parameter* — k goes to k .
- PARAMETERIZED RWB-DOMINATING SET is in NP — a solution by itself is a certificate which is verifiable in polynomial time.
- PARAMETERIZED DOMINATING SET is NP-hard in $K_{i,j}$ -free graphs, since it is NP-hard in $K_{2,2}$ -free graphs; see, for example, the reduction from the 3-SAT problem attributed to David Johnson [[69](#), Theorem 1.7].

Therefore, to obtain a polynomial kernel for PARAMETERIZED DOMINATING SET in $K_{i,j}$ -free graphs, one can do the following:

- Use [Claim 1](#) to reduce PARAMETERIZED DOMINATING SET to PARAMETERIZED RWB-DOMINATING SET in polynomial time, preserving the parameter.
- Use [Corollary 3.1](#) to obtain a polynomial kernel for the PARAMETERIZED RWB-DOMINATING SET instance obtained in the previous step.
- Apply the polynomial-time many-to-one reduction from PARAMETERIZED RWB-DOMINATING SET to PARAMETERIZED DOMINATING SET in $K_{i,j}$ -free graphs, to the PARAMETERIZED RWB-DOMINATING SET kernel obtained in the above step.

Since the PARAMETERIZED RWB-DOMINATING SET kernel is of polynomial size in the original parameter k , and the last reduction runs in polynomial time, the resulting PARAMETERIZED DOMINATING SET instance has size, and hence parameter, polynomial in k . This argument shows that PARAMETERIZED DOMINATING SET has a polynomial kernel when restricted to $K_{i,j}$ -free graphs, but does not give an explicit bound on the kernel size. We now describe a specific polynomial-time many-to-one reduction from

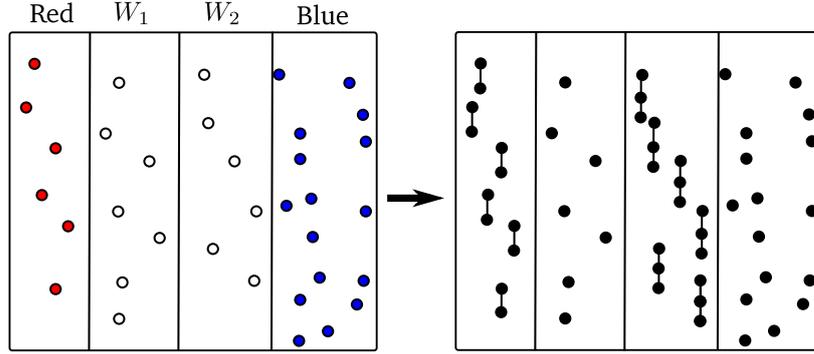


Figure 3.4: **Removing the colours.** W_1 is the set of white vertices which are adjacent to red vertices, and W_2 are the other white vertices. The rule attaches a pendant to each red vertex, a pendant path of length two to each vertex in W_2 , and removes all the colours.

PARAMETERIZED RWB-DOMINATING SET TO PARAMETERIZED DOMINATING SET in $K_{i,j}$ -free graphs and derive a concrete upper bound on the size of the kernel.

Let (G, k) be an instance of the PARAMETERIZED DOMINATING SET problem on $K_{i,j}$ -free graphs. To obtain a polynomial kernel for this instance, we first colour all the vertices of G blue to obtain an equivalent instance of the PARAMETERIZED RWB-DOMINATING SET problem. Then we apply [Corollary 3.1](#) on this PARAMETERIZED RWB-DOMINATING SET instance to obtain a reduced instance (G', k) of the problem, where $|V(G')| = \mathcal{O}((j+1)^{i+1}k^i)$.

We then apply the following steps (See [Figure 3.4](#)) to transform the reduced coloured instance (G', k) to an instance $(H, k+w)$ of (uncoloured) PARAMETERIZED DOMINATING SET, where $w = \mathcal{O}((j+1)^{i+1}k^i)$ is the number of white vertices in G' which have no red neighbours. This involves a significant increase in the value of the parameter, which is somewhat unusual: for most known kernels the new parameter is upper bounded by the original value of the parameter, or by a constant additive or multiplicative factor of the original value.

1. For each white vertex u that has no red neighbour, add two new vertices u_1, u_2 and the edges $\{u, u_1\}, \{u_1, u_2\}$. That is, create a new path with two edges starting at u . Let M be the set of all “middle” vertices u_1 added in this manner.
2. For each red vertex v , add a new vertex v_1 and the edge $\{v, v_1\}$. That is, add a new pendant vertex attached to v .
3. Remove all colours from the vertices.

Note that this construction does not introduce a $K_{i,j}$ into the graph, and that it increases the number of vertices in the graph by at most a factor of 3. We use the extra vertices

to encode the information which is captured by colours in the coloured instance. More precisely, suppose G' has an rwb-dominating set S of size at most k . By definition, the vertex set S contains all the red vertices and dominates all the blue vertices in G' . Therefore, in the graph H the same set S dominates the following sets of vertices: (1) all vertices which were red in G' , and all their neighbours, including all the pendant vertices added in Step 2 above, and, (2) all the vertices which were blue in G' . The only vertices in H which are *not* dominated by S are: (1) those white vertices in G' which had no red neighbours, and (2) the new vertices added to G' by Step 1 of the above construction. The set M of “middle” vertices added in Step 1 above dominates all these vertices in H , and so $S \cup M$ is a dominating set of H of size at most $k + w$.

Conversely, suppose H has a dominating set of size at most $k + w$, and let X be an inclusion-minimal dominating set of H of size at most $k + w$. Then we may assume the following about X without loss of generality:

1. X contains all the vertices which were red in G' : $R_{G'} \subseteq X$,
2. X contains all the “middle” vertices added in Step 1: $M \subseteq X$, and,
3. X does not contain any pendant vertex added in Step 1 or Step 2.

To see these, observe first that if X does not contain a vertex v which was red in G' , then it must contain the pendant vertex v_1 added in Step 2 of the construction, because X must dominate v_1 . The set $(X \setminus \{v_1\}) \cup \{v\}$ is then a dominating set of H of the same size as X which contains one more vertex which was red in G' , which justifies the first assumption. A similar argument using the pendant vertices added in Step 1 of the construction shows that we may assume that X contains all the “middle” vertices added in this step. Finally, if $v_1 \in X$ is a pendant vertex added in Step 1 or Step 2, then since X contains — by the first two assumptions — a vertex v adjacent to v' , the set $X \setminus \{v'\}$ is a smaller dominating set of H , and this contradicts the minimality of X .

Now observe that the vertices in M do not dominate any vertex which was blue in G' . Since (1) $(R_{G'} \cup M) \subseteq X$, (2) the set X does not contain any pendant vertex added by the construction, and (3) $|M| = w$, it follows that $X \setminus M$ is a set of at most k vertices in G' which contains all the red vertices and dominates all the blue vertices. Thus the above reduction from PARAMETERIZED RWB-DOMINATING SET to PARAMETERIZED DOMINATING SET is sound, and so from [Corollary 3.1](#) we have

Theorem 3.1. *For every fixed $j \geq i \geq 1$, the PARAMETERIZED DOMINATING SET problem on $K_{i,j}$ -free graphs has a polynomial kernel with $\mathcal{O}((j+1)^{i+1}k^2)$ vertices.*

3.2 A Polynomial Kernel for d -degenerate Graphs

A d -degenerate graph does not contain $K_{d+1,d+1}$ as a subgraph, and so the kernelization algorithm of the previous section can be applied to a d -degenerate graph, setting $i = j = d + 1$. The algorithm runs in $\mathcal{O}(\max(n^2, (d + 1)n^{d+1}))$ time and constructs a kernel with $\mathcal{O}((d + 2)^{d+2} \cdot k^{(d+1)^2})$ vertices. Since a d -degenerate graph on v vertices has at most dv edges, we have:

Corollary 3.2. *The PARAMETERIZED DOMINATING SET problem on d -degenerate graphs has a kernel on $\mathcal{O}((d + 2)^{d+3} \cdot k^{(d+1)^2})$ vertices and edges.*

Corollary 3.2 settles an open problem posed by Alon and Gutner [2, 65].

3.2.1 Improving the running time

We describe a modification to our algorithm that reduces the running time to $\mathcal{O}(2^d \cdot d \cdot n^2)$ when the input is restricted to d -degenerate graphs; the bound on the kernel size remains the same. The modified algorithm makes use of the following well-known property of d -degenerate graphs:

Fact 1. [57, Theorem 2.10] *Let G be a d -degenerate graph on n vertices. Then one can compute, in $\mathcal{O}(dn)$ time, an ordering v_1, v_2, \dots, v_n of the vertices of G such that for $1 \leq i \leq n$, v_i has at most d neighbours in the subgraph of G induced on $\{v_i, \dots, v_n\}$.*

The modification to the algorithm pertains to the way in which rules 2.1 to 2. $(d - 1)$ are implemented: the rest of the algorithm remains the same.

The previous implementation of Rule 2. p , $1 \leq p \leq (d - 1)$, checks each $(d - p + 1)$ -subset of vertices in the graph to see if it satisfies the condition in the rule. When the graph is degenerate, we instead make use of Fact 1 to quickly find such a set of vertices, if it exists. Let G be the graph instance on n vertices on which Rule 2. p is to be applied. First we delete, temporarily, all the red vertices in G . We then find an ordering v_1, v_2, \dots, v_n of the kind described in Fact 1, of all the remaining vertices in G . Let U and B be as defined in the rule. Since each vertex in U has degree greater than d , the first vertex v_l in $U \cup B$ that appears in the ordering has to be from B . The vertex v_l will then have a neighbourhood of size $d - p + 1$ that in turn has B as its common neighbourhood. We use this fact to look for such a pair (U, B) and exhaustively apply Rule 2. p to G ; see Algorithm 2. We then add back the red vertices that we deleted prior to this step, along with all their edges to the rest of the graph.

As $|N| \leq d$, the inner **for** loop is executed at most $\binom{d}{p-1}$ times for each iteration of the outer loop. Each of the individual steps in the algorithm can be done in $\mathcal{O}(dn)$ time, and

Algorithm 2 A faster implementation of Rule 2. p in d -degenerate graphs.

```

1: for  $l \leftarrow 1$  to  $n$  do
2:   if  $v_l$  is blue and its degree in  $G[v_{l+1}, \dots, v_n]$  is at least  $d - p + 1$  then
3:     Find the neighbourhood  $N$  of  $v_l$  in  $G[v_{l+1}, \dots, v_n]$ 
4:     for each  $(d - p + 1)$ -subset  $S$  of  $N$  do
5:       if  $S$  has more than  $(d + 1)k^p + k^{p-1} + \dots + k$  common blue neighbours in  $G$  then
6:         Apply the three steps of Rule 2. $p$ , taking  $S$  as  $U$ 
7:       end if
8:     end for
9:   end if
10: end for

```

so Rule 2. p can be applied in $\mathcal{O}(dn \sum_{l=1}^n \binom{d}{p-1})$ time. All the rules 2. p can therefore be applied in $\mathcal{O}(dn \sum_{l=1}^n \sum_{p=1}^{d-1} \binom{d}{p-1}) = \mathcal{O}(2^d \cdot dn^2)$ time. Since the time taken to apply each of the other rules exhaustively is $\mathcal{O}(n^2)$ (see Lemma 3.8), we have:

Theorem 3.2. *For every fixed $d \geq 1$, the PARAMETERIZED DOMINATING SET problem on d -degenerate graphs has a kernel on $\mathcal{O}((d + 2)^{d+3} \cdot k^{(d+1)^2})$ vertices and edges, and this kernel can be found in $\mathcal{O}(2^d \cdot d \cdot n^2)$ time for an input graph on n vertices.*

3.3 Independent Dominating Set in $K_{i,j}$ -free graphs

The PARAMETERIZED INDEPENDENT DOMINATING SET problem asks, for a graph G and a positive integer k given as inputs, whether G has a dominating set S of size at most k such that S is an independent set in G (that is, no two vertices in S are adjacent in G). This problem is known to be NP-hard for general graphs [59], and the problem parameterized by k is $W[2]$ -complete [41]. Using a modified version of the set of reduction rules in Section 3.1 we show that PARAMETERIZED INDEPENDENT DOMINATING SET has a polynomial kernel on $K_{i,j}$ -free graphs for $j \geq i \geq 1$. For $i = 1, j \geq 1$ we can easily obtain trivial kernels as before, and for $i = 2, j \geq 2$ a simplified version of the following algorithm gives a kernel of size $\mathcal{O}(j^3 k^4)$.

3.3.1 The reduction rules

Rule 1 is the same as for the PARAMETERIZED DOMINATING SET kernel for $K_{i,j}$ -free graphs (Section 3.1.1). Rules 2.1 to 2. $(i - 2)$ and Rule 3 are modified to make use of the fact that we are looking for a dominating set that is independent. A vertex u that is made white will never be part of the independent dominating set D that is sought to be constructed by the algorithm, since u is adjacent to some vertex $v \in D$. So a vertex can be deleted as soon as it is made white. Also, rules 1, 2.1 \dots 2. $(i - 2)$ and 3 are the only rules. Rules 4 and 5 from that section do not apply, because of the same reason as above. The modified rules ensure

that no vertex is coloured white, and so they work on *rb-graphs*: graphs whose vertex set is partitioned into red and blue vertices. Using these modified rules, the bounds of $|R_H|$ and $|B_H|$ in the proof of [Lemma 3.9](#), and the fact that there are no white vertices, we have

Theorem 3.3. *For every fixed $j \geq i \geq 1$, the PARAMETERIZED INDEPENDENT DOMINATING SET problem on $K_{i,j}$ -free graphs has a polynomial kernel with $\mathcal{O}(jk^i)$ vertices.*

For d -degenerate graphs, we have $i = j = d + 1$. Since a d -degenerate graph on v vertices has at most dv edges, we have:

Corollary 3.3. *For every fixed $d \geq 1$, the PARAMETERIZED INDEPENDENT DOMINATING SET problem on d -degenerate graphs has a polynomial kernel with $\mathcal{O}(d(d+1)k^{d+1})$ vertices and edges.*

3.4 Conclusion

In this chapter we derived a polynomial kernel for the PARAMETERIZED DOMINATING SET problem on graphs that do not have $K_{i,j}$ as a subgraph, for every fixed $j \geq i \geq 1$. We used this result to show that the PARAMETERIZED DOMINATING SET problem has a polynomial kernel of size $\mathcal{O}((d+2)^{d+3} \cdot k^{(d+1)^2})$ on graphs of degeneracy at most d , thereby settling an open problem posed by Alon and Gutner [[2](#), [65](#)]. A modified version of our kernelization algorithm for PARAMETERIZED DOMINATING SET yields a (smaller) kernel for the PARAMETERIZED INDEPENDENT DOMINATING SET problem on $K_{i,j}$ -free and d -degenerate graphs, as well. All our kernelization algorithms are based on simple reduction rules that look at the common neighbourhoods of sets of vertices.

These results are primarily of theoretical interest in that the kernel sizes are too large to be of practical use. For example, using the fact that planar graphs are $K_{3,3}$ -free, our kernelization algorithm can be used to obtain a polynomial kernel for the PARAMETERIZED DOMINATING SET on planar graphs. The upper bound that we derive on the size of this kernel is $\mathcal{O}(k^9)$, while the problem is known [[24](#)] to have a kernel on at most $67k$ vertices in planar graphs.

Using the kernel lower-bound techniques of Bodlaender et al. [[15](#)], Dom et al. [[39](#)] have shown that the PARAMETERIZED DOMINATING SET problem on d -degenerate graphs does not have a kernel of size polynomial in *both* d and k unless the Polynomial Hierarchy collapses to the third level. This shows that it is unlikely that the kernel size that we have obtained for this class of graphs can be significantly improved.

Many interesting classes of graphs are of bounded degeneracy. These include all nontrivial minor-closed families of graphs such as planar graphs, graphs of bounded

genus, graphs of bounded treewidth, and graphs excluding a fixed minor, and some non-minor-closed families such as graphs of bounded degree. Graphs of degeneracy d are $K_{d+1,d+1}$ -free. Since every $K_{i,j}$; $j \geq i \geq 2$ contains a 4-cycle, every graph of girth 5 is $K_{i,j}$ -free. Sachs [106, Theorem 1] showed that there exist graphs of girth 5 and arbitrarily large degeneracy (See Chapter III, Theorem 1.1 of Bollobás' Extremal Graph Theory [19] for a short proof). Therefore, $K_{i,j}$ -free graphs are strictly more general than graphs of bounded degeneracy. To the best of our knowledge, $K_{i,j}$ -free graphs form the largest class of graphs for which FPT algorithms and polynomial kernels are known for the dominating set problem variants discussed in this chapter.

One interesting direction of future work is to try to demonstrate (no) kernels of size $f(d) \cdot k^c$ for the PARAMETERIZED DOMINATING SET problem on d -degenerate graphs, where c is independent of d . Note that the result of Dom et al. mentioned above does *not* suggest that such kernels are unlikely.

A graph property Π is a set of graphs. The *vertex deletion problem for Π* asks, given a graph G and a non-negative integer k as inputs, whether there exist at most k vertices in G whose deletion from G results in a graph that belongs to Π . A graph property Π is said to be (1) *nontrivial* if neither Π nor its complement is finite, and (2) *hereditary* if $(G \in \Pi, H \text{ is a subgraph of } G) \implies H \in \Pi$. Dell and van Melkebeek [34] have recently developed a lower-bound technique which allows them to show, *inter alia*, that the vertex deletion problem for any nontrivial hereditary graph class has no kernel of size $\mathcal{O}(k^{2-\epsilon})$ for any $\epsilon > 0$. It will be interesting to see if this new machinery can be extended to show that the PARAMETERIZED DOMINATING SET problem does not have kernels of size $f(d) \cdot k^c$ on d -degenerate graphs.

Another challenge is to improve the running times of the kernelization algorithms: to remove the exponential dependence on d of the running time for d -degenerate graphs, and to obtain a running time of the form $\mathcal{O}(n^c)$ for $K_{i,j}$ -free graphs where c is independent of i and j . It would also be interesting to see if the natural parameterized versions of other NP-hard variants of DOMINATING SET — of which there are many [68, 69] — have FPT algorithms and polynomial kernels on $K_{i,j}$ -free graphs and graphs of bounded degeneracy. Very recently, Cygan et al. [28] showed that PARAMETERIZED CONNECTED DOMINATING SET, where one asks for a dominating set of size at most k that induces a connected subgraph of the input graph, has no polynomial kernels on graphs of degeneracy $d \geq 2$ unless the Polynomial Hierarchy collapses to the third level. Note that our kernelization procedure breaks down (as it should) when we insist that the solution be connected, since Rule 4 and Rule 5 can no longer be applied: white vertices which are useless in dominating blue vertices may still be useful in providing connectivity for the dominating set that is being constructed.

Connected Domination and Girth

IN the PARAMETERIZED CONNECTED DOMINATING SET problem the input consists of a graph G and a positive integer k , and the question is whether there is a set S of at most k vertices in G — a *connected dominating set* of G — such that (i) S is a dominating set of G , and (ii) the subgraph $G[S]$ induced by S is connected; the parameter is k . The underlying decision problem is a basic connectivity problem which is long known to be NP-complete, and it has been extensively studied using several algorithmic approaches.

PARAMETERIZED CONNECTED DOMINATING SET is $W[2]$ -complete, and therefore it is unlikely (See [Chapter 2](#)) that the problem has fixed-parameter tractable (FPT) algorithms or polynomial kernels on graphs in general.

The problem does have FPT algorithms in certain restricted families of graphs, such as in planar graphs [49, 62, 83, 85], graphs of bounded genus [17], apex-minor-free graphs [56], nowhere-dense classes of graphs [31], and graphs of bounded degeneracy [60].

Recall (see [Chapter 2](#)) that for the PARAMETERIZED CONNECTED DOMINATING SET problem, a kernelization algorithm is an algorithm that takes (G, k) as input, runs in polynomial time, and outputs an equivalent instance (H, k') , where $k' \leq g(k)$ and H is a graph with at most $h(k)$ vertices for some computable functions g and h . Here (H, k') is equivalent to (G, k) in the sense that the graph H has a connected dominating set of size at most k' if and only if G has a connected dominating set of size at most k . H is the kernel output by this algorithm. From the equivalence of FPT and kernelization (recall the folklore [Theorem 2.1](#)) it follows that, unless $FPT=W[2]$, there is no kernelization

algorithm for PARAMETERIZED CONNECTED DOMINATING SET on general graphs (or on graphs with a bounded average degree, for that matter). For the same reason, the problem admits kernelization algorithms when the input is restricted to planar graphs, nowhere-dense classes of graphs, or graphs of bounded degeneracy. However, the *size* of the kernel implied by the proof of [Theorem 2.1](#) is equal to the factor $f(k)$ in the running time of the corresponding FPT algorithm, and hence is exponential in k . The interesting problem is, therefore, to find if the kernel size can be made smaller — in particular, whether it can be made polynomial in k .

Recall from [Chapter 3](#) that for the PARAMETERIZED DOMINATING SET problem, many polynomial kernel results have been obtained on different graph classes starting with the linear kernel of Alber et al. [1] from 2004 for planar graphs. In contrast, no such results were known for PARAMETERIZED CONNECTED DOMINATING SET until more recently. The first polynomial kernels for the problem were obtained by Lokshtanov et al. [83], and independently by Gu and Imani [62]. Both groups of authors showed that the PARAMETERIZED CONNECTED DOMINATING SET problem restricted to planar graphs has *linear* kernels. Very recently, Luo et al. [85] improved the multiplicative factor in the size of this kernel, to obtain a kernel on $130k$ vertices for the planar PARAMETERIZED CONNECTED DOMINATING SET problem. The general results obtained by Bodlaender et al. [17] and Fomin et al. [56] imply, *inter alia*, that PARAMETERIZED CONNECTED DOMINATING SET has linear kernels on more general classes of graphs, namely in graphs of bounded genus and in apex-minor-free graphs (which are classes of graphs that exclude special graphs — called apex graphs — as a minor).

Note that all the kernelization results mentioned above are on graph classes which are characterized by the exclusion of some fixed graph(s) as a *minor* (See [Chapter 2](#) for definitions.). This is in fact indicative of the general state of the art in kernelization: while there are many polynomial kernel results known for hard parameterized problems restricted to graphs excluding some fixed graph as a minor, there have only been a handful of such results on graph classes that are defined by excluding some fixed graph(s) as *subgraph(s)*. The first result of this kind was obtained by Raman and Saurabh [102] who showed that PARAMETERIZED DOMINATING SET has a kernel on $\mathcal{O}(k^3)$ vertices on graphs which do not contain cycles of length 3 or 4 as subgraphs. Their argument can in fact be modified to work for graphs which exclude just the cycle of length 4 as a subgraph. This latter condition can equivalently be thought of as excluding a $K_{2,2}$, the complete bipartite graph where each part has size exactly 2. As described in [Chapter 3](#), this was later generalized [99] to show that PARAMETERIZED DOMINATING SET has a polynomial kernel of size k^h on $K_{i,j}$ -free graphs for any fixed i and j , where h is a constant that depends only on i and j . Recall from [Chapter 3](#) that this implies that the PARAMETERIZED DOMINATING

SET problem has polynomial kernels on graphs of bounded degeneracy. In contrast, it was recently shown by Cygan et al. [28], using the lower bound machinery mentioned above, that the PARAMETERIZED CONNECTED DOMINATING SET problem does not have polynomial kernels on graphs of bounded degeneracy unless the Polynomial Hierarchy collapses to the third level.

Our Results

The *girth* of a graph G is the length of a shortest cycle in G . We study the effect of girth on the kernelization complexity of PARAMETERIZED CONNECTED DOMINATING SET. We show that the PARAMETERIZED CONNECTED DOMINATING SET problem is hard on graphs with small cycles, and becomes progressively easier as the girth increases. More precisely, we obtain the following kernelization landscape:

- PARAMETERIZED CONNECTED DOMINATING SET is $W[2]$ -hard in graphs which contain cycles of length 3 or 4, and so it does not have a kernel of *any* size in such graphs unless $FPT=W[2]$.
- On any class of graphs which have girth at least 5, PARAMETERIZED CONNECTED DOMINATING SET has an FPT algorithm which runs in $\mathcal{O}(2^k k^{3k} \cdot n^c)$ time where n is the number of vertices in the input graph and c is a constant independent of n and k . As a consequence, the problem has a kernel of size $2^k k^{3k}$ on graphs of girth at least 5.
- Unless the Polynomial Hierarchy (PH) collapses to the third level, PARAMETERIZED CONNECTED DOMINATING SET has *no* polynomial kernel on graphs which contain cycles of length at most 6.
- On any class of graphs which have girth at least 7, PARAMETERIZED CONNECTED DOMINATING SET has a kernel on $\mathcal{O}(k^3)$ vertices.

While there is a large and growing collection of parameterized complexity results available for problems on graph classes characterized by excluded minors, our results add to the very few known in the field for graph classes characterized by excluded *subgraphs*.

Organization of the rest of the chapter

In [Section 4.1](#) we reuse a construction due to Raman and Saurabh [102] to show that PARAMETERIZED CONNECTED DOMINATING SET is $W[2]$ -hard on graphs of girth at most 4. In [Section 4.2](#) we derive an FPT algorithm for PARAMETERIZED CONNECTED DOMINATING

SET on graphs of girth at least 5 which runs in $2^k k^{3k} \cdot n^{O(1)}$ time, thereby showing that the problem has a kernel of size $2^k k^{3k}$ in these graphs. In [Section 4.3](#) we describe a proof that the PARAMETERIZED CONNECTED DOMINATING SET problem has no polynomial kernels on graphs of girth at most 6 unless PH collapses to the third level. To obtain this result we introduce an intermediate, seemingly unrelated problem named PARAMETERIZED FAIR CONNECTED COLOURS, show that PARAMETERIZED FAIR CONNECTED COLOURS has no polynomial kernels (unless PH collapses to the third level) using the recent kernel lower bound machinery developed by Bodlaender et al. [[15](#)], and then provide a parameter-preserving reduction [[18](#)] from PARAMETERIZED FAIR CONNECTED COLOURS to PARAMETERIZED CONNECTED DOMINATING SET. In [Section 4.4](#) we derive a cubic ($O(k^3)$) vertex kernel for PARAMETERIZED CONNECTED DOMINATING SET in graphs of girth at least 7. We conclude in [Section 4.5](#).

Notation

All the graphs in this chapter are finite, undirected and simple. In general we follow the graph terminology of [Section 2.1](#). We use \mathbb{G}_r to denote the class of all graphs with girth at least $r \in \mathbb{N}$.

4.1 On Graphs of Girth 3 and 4 : $W[2]$ -hardness

Raman and Saurabh show [[102](#), Theorem 1] that the PARAMETERIZED DOMINATING SET problem is $W[2]$ -hard on graphs of girth 4 by reduction from PARAMETERIZED DOMINATING SET on general graphs, which is a canonical $W[2]$ -hard problem [[41](#)]. It turns out that their reduction, reproduced below, suffices to show that PARAMETERIZED CONNECTED DOMINATING SET is also $W[2]$ -hard on graphs of girth at most 4.

Theorem 4.1. *PARAMETERIZED CONNECTED DOMINATING SET is $W[2]$ -hard on graphs of girth 3 and on graphs of girth 4.*

Proof. We first show that the PARAMETERIZED CONNECTED DOMINATING SET problem is $W[2]$ -hard on graphs of girth 4, by reducing from the PARAMETERIZED DOMINATING SET problem which is known to be $W[2]$ -hard. Recall from [Chapter 3](#) that the input to this problem consists of a graph G and a positive integer parameter k , and the question is whether G has a dominating set of size at most k . Given an instance (G, k) of PARAMETERIZED DOMINATING SET, we construct a bipartite graph H as depicted in [Figure 4.1](#). For each vertex v in G , we add two vertices v_1, v_2 and the edge $\{v_1, v_2\}$ to H . Let $V_1 = \{v_1 \mid v \in V(G)\}$, and $V_2 = \{v_2 \mid v \in V(G)\}$. For each edge $\{u, v\}$ in G , we add the

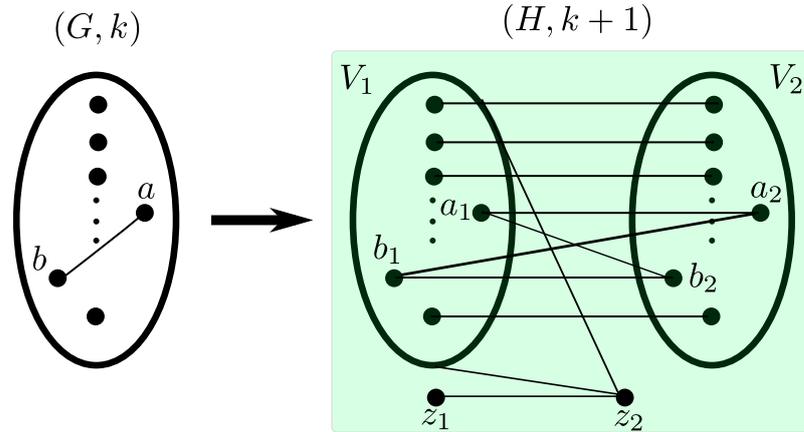


Figure 4.1: **FPT reduction from PARAMETERIZED DOMINATING SET to PARAMETERIZED CONNECTED DOMINATING SET.** This shows that PARAMETERIZED CONNECTED DOMINATING SET is $W[2]$ -hard on graphs of girth 4. The reduction constructs a bipartite graph H from the input graph G . For each vertex in graph G , two vertices are added to H , and an edge is put between them. Each edge in G gives rise to two edges in H , in the natural fashion. The construction is completed by adding two new vertices z_1 and z_2 to the two sets of vertices in H , and making z_2 adjacent to every vertex in the other part. The parameter goes from k to $k + 1$.

two edges $\{v_1, u_2\}, \{u_1, v_2\}$ to H . Finally, we add two new vertices $z_1 \in V_1, z_2 \in V_2$, and add an edge from z_2 to each vertex in V_1 . This completes the construction of H .

Observe that the construction can be done in time polynomial in the input size. Without loss of generality, we assume that the graph G has at least one edge. The girth of the reduced instance H is at least 4 because H is bipartite, and H has girth exactly 4 because the reduction takes an edge in the original instance G to a cycle of length 4 in H .

If G has a dominating set S of size at most k , then let $S_1 = \{s_1 \in V_1 \mid s \in S\}$, and $S' = S_1 \cup \{z_2\}$. The vertex z_2 in S' dominates all vertices in V_1 , and the set S_1 dominates all vertices in V_2 . Since z_2 is adjacent to all vertices in S_1 , $H[S']$ is a connected subgraph. Thus S' is a connected dominating set of H of size at most $k + 1$.

Conversely, if H has a connected dominating set of size at most $k + 1$, then let S' be a *minimal* such set, in the sense that no proper subset of S' is a connected dominating set of H . Suppose $z_1 \in S'$. Observe that z_1 has only one neighbour in H , namely z_2 , and so z_1 and z_2 are the only vertices which are dominated by z_1 . Since H contains vertices other than z_1 and z_2 , it follows that z_1 is not the only vertex in S' . Let T' be a spanning tree of the connected subgraph $H[S']$. Then z_1 is a leaf in T' , and its neighbour in T' is the vertex z_2 . Consider the graph T'' obtained by deleting z_1 from T' . Then (i) T'' is a tree since z_1 is a leaf in T' , and (ii) the set $V(T'')$ dominates all vertices in H since $z_2 \in V(T'')$. Thus $V(T'')$ is a connected dominating set of H which is properly contained

in S' , which contradicts the assumption that S' is a minimal connected dominating set of H . Therefore $z_1 \notin S'$, and since S' dominates z_1 , it follows that $z_2 \in S'$.

Thus S' consists of z_2 , some vertices in V_1 , and possibly some vertices in V_2 . Now let $S = \{u \mid u \in V(G); u_1 \in S' \text{ or } u_2 \in S'\}$. Observe that the natural projection from $S' \setminus \{z_2\}$ to S is many-to-one and onto. Hence there are at most k vertices in S . Now consider any vertex $v \in V(G) \setminus S$. Since S' is a dominating set of the graph H , the vertex v_2 in H is dominated by some vertex $u_1 \in (V_1 \cap S')$. From the construction of H , this implies that the edge $\{u, v\}$ is present in the graph G , and from the definition of S , it follows that $u \in S$. Thus the vertex v is adjacent to some vertex $u \in S$, in the graph G . Since this holds for any vertex $v \notin S$, it follows that S is a dominating set of G of size at most k . Thus the reduction is sound, and so the PARAMETERIZED CONNECTED DOMINATING SET problem is $W[2]$ -hard when restricted to graphs of girth 4.

To see that PARAMETERIZED CONNECTED DOMINATING SET is $W[2]$ -hard on graphs of girth 3 as well, modify the construction of H as follows: Add a new vertex z_3 and the two edges $\{z_2, z_3\}, \{z_1, z_3\}$ to H to form a triangle. Now H has girth 3, and the reduced instance is $(H, k + 1)$. Essentially the same argument as above shows that this reduction is sound. □

4.2 On Graphs of Girth 5 or More : A Kernel of Size $2^k k^{3k}$

Recall that \mathbb{G}_r denotes the set of all graphs with girth at least $r \in \mathbb{N}$. As we show later in this chapter (see [Lemma 4.7](#)), the PARAMETERIZED CONNECTED DOMINATING SET problem remains NP-complete in graphs which do not have cycles of length 3 or 4 — that is, on \mathbb{G}_5 . We now show that the PARAMETERIZED CONNECTED DOMINATING SET problem is fixed-parameter tractable and can be solved in $2^k k^{3k} n^{O(1)}$ time on \mathbb{G}_5 . It follows a folklore theorem of parameterized complexity (See [Chapter 2, Theorem 2.1.](#)) that the PARAMETERIZED CONNECTED DOMINATING SET problem restricted to \mathbb{G}_5 has a kernel of size $2^k k^{3k}$.

To show that PARAMETERIZED CONNECTED DOMINATING SET is FPT on \mathbb{G}_5 , we first derive an FPT algorithm for a slightly more general, vertex-coloured version of the PARAMETERIZED CONNECTED DOMINATING SET problem on \mathbb{G}_5 . It is straightforward to reduce PARAMETERIZED CONNECTED DOMINATING SET to this more general version in FPT — in fact, polynomial — time, and thus we get an FPT algorithm for PARAMETERIZED CONNECTED DOMINATING SET on \mathbb{G}_5 . In the more general problem, called PARAMETERIZED CONNECTED RWB-DOMINATING SET, the vertices of the input graph are partitioned into

three colour classes, called red, white, and blue. Recall that we used such a partitioning in [Section 3.1](#) to obtain a polynomial kernel for the PARAMETERIZED DOMINATING SET problem on $K_{i,j}$ -free graphs. The general idea is similar in this case as well, but now the colouring is subject to some constraints. This approach is motivated by a similar colouring used by Raman and Saurabh to obtain a polynomial kernel for the PARAMETERIZED DOMINATING SET problem on \mathbb{G}_5 graphs [[102](#)].

We define an *rw-graph* (a red-white-blue graph) to be a graph whose vertices are coloured with the three colours red, white, and blue, such that (i) every white vertex is the neighbour of some red vertex, and (ii) blue vertices have no red neighbours. More formally, an *rw-graph* is a graph G whose vertex set $V(G)$ is partitioned into R_G , W_G , and B_G — coloured red, white, and blue, respectively — such that (i) for every $w \in W_G$, there is at least one $v \in R_G$ such that $\{v, w\} \in E(G)$, and (ii) there is no pair $v \in B_G, w \in R_G$ such that $\{v, w\} \in E(G)$. A *connected rw-dominating set* of an *rw-graph* G is a dominating set $S \subseteq V(G)$ of G such that (i) $G[S]$ is connected and (ii) $R_G \subseteq S$. Observe that, in contrast to the definition in [Section 3.1](#), here we stipulate that S dominate *all* vertices in G . We define the PARAMETERIZED CONNECTED RWB-DOMINATING SET problem as follows:

PARAMETERIZED CONNECTED RWB-DOMINATING SET

Input: An *rw-graph* G , and a positive integer k .

Parameter: k

Question: Does G have a connected *rw-dominating set* of size at most k ?

The semantics of the colours are similar to those defined by Raman and Saurabh [[102](#)]. A vertex is coloured red if it is definitely present in the connected dominating set S that our algorithm is trying to construct. A white vertex is one that is not yet in S but is known to be dominated by some vertex in S . All the remaining vertices are yet to be dominated, and are coloured blue.

We note that Raman and Saurabh claimed [[102](#), Corollary 3] that PARAMETERIZED CONNECTED DOMINATING SET restricted to \mathbb{G}_5 has a kernel on $\mathcal{O}(k^3)$ vertices, and hence is fixed-parameter tractable. But the argument that they present is incorrect; in fact, as we show later ([Theorem 4.3](#)), PARAMETERIZED CONNECTED DOMINATING SET restricted to \mathbb{G}_5 *cannot* have any polynomial-sized kernel unless the Polynomial Hierarchy collapses to the third level. The error in their argument arises from the assumption that the reduction rules which they use for PARAMETERIZED DOMINATING SET also work for PARAMETERIZED CONNECTED DOMINATING SET — but rules like deleting a white vertex or deleting edges between white vertices *do not* apply to PARAMETERIZED CONNECTED DOMINATING SET.

This is because such vertices and edges may be needed to provide connectivity to a dominating set. However, the fixed-parameter tractability result still holds, because we show by a different argument that

Lemma 4.1. *On graphs of girth at least 5, PARAMETERIZED CONNECTED RWB-DOMINATING SET is fixed-parameter tractable and can be solved in $\mathcal{O}^*(2^k k^{3k})$ time.*

We can reduce an instance of PARAMETERIZED CONNECTED DOMINATING SET to an equivalent instance of PARAMETERIZED CONNECTED RWB-DOMINATING SET in polynomial time, with no change in the parameter:

Lemma 4.2. *An instance (G, k) of PARAMETERIZED CONNECTED DOMINATING SET can be converted to an equivalent instance (H, k) of PARAMETERIZED CONNECTED RWB-DOMINATING SET in polynomial time.*

Proof. Colour all vertices of G blue to obtain the rwb-graph H .

If G has a connected dominating set S of size at most k , then the same set S is a connected rwb-dominating set of H of size at most k : since we did not change the structure of the graph, $H[S]$ is connected, and S is a dominating set of H . Since the set R_H of red vertices in H is the empty set, $R_H \subseteq S$ holds vacuously.

Conversely, if S' is a connected rwb-dominating set of H of size at most k , then since G and H are isomorphic as graphs (ignoring the colours), S' itself is a connected dominating set of G of size at most k . \square

Observe that the construction in Lemma 4.2 does not decrease the girth of the input graph, since it does not change the structure of the graph at all. To solve the PARAMETERIZED CONNECTED DOMINATING SET problem on \mathbb{G}_5 in FPT time, we first apply Lemma 4.2 to the input instance to obtain an equivalent PARAMETERIZED CONNECTED RWB-DOMINATING SET instance where the underlying graph has girth at least 5. We then solve this instance of PARAMETERIZED CONNECTED RWB-DOMINATING SET in FPT time using Lemma 4.1. Thus we have

Theorem 4.2. *PARAMETERIZED CONNECTED DOMINATING SET can be solved in $\mathcal{O}^*(2^k k^{3k})$ time on graphs of girth at least 5.*

For the reasons stated above, this directly yields

Corollary 4.1. *The PARAMETERIZED CONNECTED DOMINATING SET problem has a kernel of size $2^k k^{3k}$ on graphs of girth at least 5.*

In the remaining part of this subsection we prove [Lemma 4.1](#).

A key ingredient in the proof of [Lemma 4.1](#) is the fact that if G is a graph of girth at least 5, then every vertex in G with degree more than k is present in every dominating set of G of size at most k . To see this, observe first that the neighbours of any vertex in G form an independent set, or else there would be a triangle in G . Further, no two vertices $u, v \in V(G)$ can have more than one *common* neighbour. For, if x, y are two common neighbours of u and v , then u, v, x, y form a cycle of length 4 in G . Thus we have:

Observation 1. *If G is a graph of girth at least 5, then the open neighbourhood $N(v)$ of any vertex $v \in V(G)$ is an independent set, and no two vertices of G have more than one common neighbour.*

This observation implies that if a vertex $v \in V(G)$ with degree more than k is not allowed to be picked as part of a dominating set of G , then we need more than k vertices to dominate all the vertices in G :

Lemma 4.3. *Let G be a graph of girth at least 5. If a vertex v in G has more than k neighbours, then v is present in every dominating set of G of size at most k .*

Proof. Assume for the sake of contradiction that S is a dominating set of G of size at most k such that $v \notin S$. Then $N(v) \cap S \neq \emptyset$, or else S does not dominate v . Let $|N(v) \cap S| = \ell$. Then $1 \leq \ell \leq k$, and $|N(v) \setminus S| \geq k + 1 - \ell$. Since $G[N(v)]$ is an independent set as per [Observation 1](#), none of the vertices in $N(v) \setminus S$ is dominated by any of the ℓ vertices in $N(v) \cap S$. Hence all the (at least $k + 1 - \ell$) vertices in $N(v)$ are dominated by at most $k - \ell$ vertices from $S \setminus N[v]$. This is clearly impossible if $\ell = k$, and so $\ell < k$. Thus — by the pigeonhole principle — there is at least one vertex $u \in (S \setminus N[v])$ that dominates at least two vertices x, y in $N(v)$. The vertices u, v thus have two common neighbours x, y in G , which contradicts [Observation 1](#). \square

A second key ingredient in the proof of [Lemma 4.1](#) is the existence of an FPT algorithm for the PARAMETERIZED STEINER TREE problem, which is a parameterized version of the classical STEINER TREE problem. Let H be a graph and $T \subseteq V(H)$ a set of designated “terminal” vertices of H . A Steiner tree of H for the terminal set T is a connected subgraph of H with the minimum number of edges which includes all the terminal vertices; it is easy to see that such a subgraph, if it exists, is a tree. The STEINER TREE problem is defined as follows:

STEINER TREE

Input: A graph H , a set $T \subseteq V(H)$ of designated “terminal” vertices, and a positive integer c .

Question: Does H have a Steiner tree for the terminal set T , with at most c edges?

The STEINER TREE problem is NP-hard; in fact, it is one among Karp's original list [75] of 21 NP-complete problems. Of interest to us is the parameterized version of the problem where the parameter is $|T|$, the number of terminals:

PARAMETERIZED STEINER TREE

Input: A graph H , a set $T \subseteq V(H)$ of designated “terminal” vertices, and a positive integer c .

Parameter: $|T|$

Question: Does H have a Steiner tree for the terminal set T , with at most c edges?

This problem is FPT. In fact, an FPT algorithm for this problem predates the invention of the notion of parameterized complexity. The classical algorithm for STEINER TREE due to Dreyfus and Wagner [43] from the year 1972 solves the problem in $\mathcal{O}^*(3^{|T|})$ time. After many improvements, the current fastest FPT algorithm for the problem, due to Nederlof [94], runs in $\mathcal{O}^*(2^{|T|})$ time and polynomial space.

Fact 1. [94] *The PARAMETERIZED STEINER TREE problem can be solved in $\mathcal{O}^*(2^{|T|})$ time and polynomial space. Further, if the input graph H contains a Steiner tree for the terminal set T with at most c edges, then such a Steiner tree can be found within the same time bound.*

We are now ready to prove [Lemma 4.1](#).

Proof of Lemma 4.1. Let (G, k) be an instance of PARAMETERIZED CONNECTED RWB-DOMINATING SET where G has girth at least 5. Let A be the set of white and blue vertices in G , each of which has at least $k + 1$ neighbours. By [Lemma 4.3](#), every vertex in A is part of every dominating set — connected or otherwise — of G of size at most k . Therefore, if $|R_G \cup A| > k$ then G does not have any dominating set of size at most k which contains all the vertices of R_G . The algorithm finds this set A and returns No if $|R_G \cup A| > k$. This can be done in polynomial time, and henceforth we assume that $|R_G \cup A| \leq k$.

The algorithm now colours all the vertices of the set A red, and all the blue neighbours of these vertices white. This can be done in polynomial time, and from [Lemma 4.3](#), the resulting instance is equivalent to the original instance. Observe that the resulting graph is also an rwb-graph. To keep the notation simple, we use G to refer to this resulting graph as well.

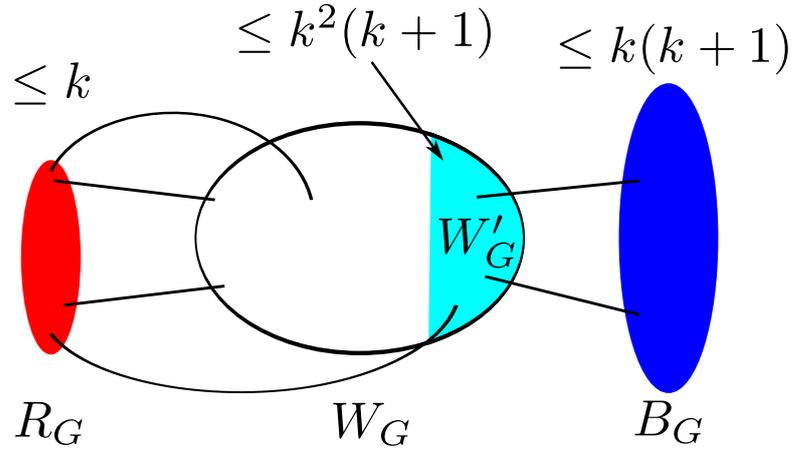


Figure 4.2: **Bounds on the sizes of various subsets of vertices of an input graph G of girth at least 5 in a PARAMETERIZED CONNECTED RWB-DOMINATING SET instance.** R_G , W_G , and B_G are, respectively, the sets of red, white, and blue vertices in G . W'_G is the set of white vertices which have at least one blue neighbour.

We now bound the number of blue vertices in the resulting graph G . Observe that in this graph, every blue or white vertex has at most k neighbours — or else the vertex would have been coloured red — and no red vertex has any blue neighbour. If G has a connected rwb-dominating set S of size at most k , then the $k' = k - |R_G|$ white and blue vertices in S can together dominate at most $k'(k + 1) \leq k^2 + k$ blue vertices in G . It follows that if (G, k) is a YES instance of the problem, then the number of blue vertices in G , $|B_G| \leq k^2 + k$. The algorithm computes $|B_G|$ — which can be done in linear time — and returns No if $|B_G| > k^2 + k$.

At this point, the graph G has at most k red vertices, and at most $k^2 + k$ blue vertices. As we show in [Section 4.3](#), it is unlikely that we will be able to give a polynomial bound on the number of white vertices in G . To get an FPT algorithm, we bound instead the number of white vertices *which are adjacent to blue vertices* in G by $k^2(k + 1)$.

Let W'_G be the set of white vertices which have at least one blue neighbour. As observed above, every blue vertex has at most k white neighbours. Since the number of blue vertices is at most $k^2 + k$, it follows that the number of white vertices which are adjacent to blue vertices, $|W'_G| \leq k|B_G| \leq k^2(k + 1)$; see [Figure 4.2](#).

Consider a connected rwb-dominating set S of G , of size at most k . By definition, $R_G \subseteq S$, and this subset of S dominates itself and all the white vertices. One can think of the remaining part of S , namely $S \setminus R_G$, as consisting of two parts: a set S_1 of vertices which serve to dominate all the blue vertices, and a set S_2 which serves to connect the vertices in R_G and S_1 . To be more precise, let S_1 be any minimal subset of $S \setminus R_G$ which

dominates B_G , and let $S_2 = S \setminus (R_G \cup S_1)$. The algorithm guesses the subset S_1 , and verifies the guess by computing the set S_2 and checking if $|S_2| \leq k - |R_G \cup S_2|$.

To see how the inclusion-minimal set S_1 which dominates B_G can be computed in FPT time, observe that no vertex in $R_G \cup (W_G \setminus W'_G)$ can dominate any blue vertex. It follows that the only vertices which can be part of the set S_1 are those present in $B_G \cup W'_G$. In other words, $S_1 \subseteq (B_G \cup W'_G)$. The algorithm guesses the set S_1 by trying out each subset of $B_G \cup W'_G$ of size at most $k - |R_G|$. Since $|(B_G \cup W'_G)| \leq k(k+1)^2$, this can be done in $\mathcal{O}\left(\binom{k(k+1)^2}{k-|R_G|}\right) = \mathcal{O}(k^{3k})$ time.

It is clear from the above discussion that to verify a given guess for the set S_1 , it is necessary and sufficient to ensure two things, namely: (i) the set $R_G \cup S_1$ is a dominating set of G , and (ii) there exists a set $S_2 \subseteq V(G) \setminus (R_G \cup S_1)$ such that $G[R \cup S_1 \cup S_2]$ is a connected subgraph and $|R \cup S_1 \cup S_2| \leq k$. It is straightforward to see that the first condition can be checked in polynomial time.

To check the second condition, we make use of the FPT algorithm for PARAMETERIZED STEINER TREE from [Fact 1](#). We create an instance of PARAMETERIZED STEINER TREE as follows: the input graph is G itself, the terminal set $T = R_G \cup S_1$, and the budget $c = k - 1$. We now invoke the algorithm of [Fact 1](#) to solve this instance in $\mathcal{O}^*(2^{|T|}) = \mathcal{O}^*(2^k)$ time.

If the algorithm finds a Steiner tree T_S of G on the terminal set T and with at most $k - 1$ edges, then the vertex set $V(T_S)$ of this Steiner tree is a connected rwb-dominating set of G of size at most k : Since $T = R_G \cup S_1 \subseteq V(T_S)$, $V(T_S)$ dominates all vertices in G . The subgraph $G[V(T_S)]$ is connected, as witnessed by the spanning tree T_S of this subgraph. $R_G \subseteq V(T_S)$, and since the tree T_S has at most $k - 1$ edges, $|V(T_S)| \leq k$.

Conversely, if there exists a set $S_2 \subseteq V(G) \setminus (R_G \cup S_1)$ such that $G[R \cup S_1 \cup S_2]$ is a connected subgraph and $|R \cup S_1 \cup S_2| \leq k$, then any spanning tree of $G[R \cup S_1 \cup S_2]$ is a Steiner tree of G on the terminal set $T = R_G \cup S_1$ and with at most $k - 1$ edges, and so the algorithm of [Fact 1](#) will find such a Steiner tree T_S .

Therefore, to verify a given guess for the set S_1 , our algorithm for PARAMETERIZED CONNECTED RWB-DOMINATING SET runs the FPT algorithm for PARAMETERIZED STEINER TREE from [Fact 1](#). If this latter algorithm returns NO, then our algorithm rejects this guess for S_1 . Otherwise it returns the vertex set of the Steiner tree found as a connected rwb-dominating set of G of size at most k . If all guesses for the set S_1 are rejected, then our algorithm returns NO.

The correctness of our algorithm is clear from the above discussion. To bound its running time, observe that apart from guessing the set S_1 and verifying the guess, all the other steps in the algorithm can be done in polynomial time. As described above, the number of different possibilities for the set S_1 is $\mathcal{O}(k^{3k})$, and all these possibilities can be enumerated in $\mathcal{O}(k^{3k})$ time. For each subset S_1 , the size of the terminal set $T = R_G \cup S_1$

which occurs in the PARAMETERIZED STEINER TREE problem is at most k , and so this problem can be solved in $\mathcal{O}^*(2^k)$ time. Thus the total time taken by the algorithm is bounded by $\mathcal{O}^*(2^k k^{3k})$. This concludes the proof of theorem. \square

4.3 On Graphs of girth 5 and 6: No Polynomial Kernels

In the previous section, we saw how the PARAMETERIZED CONNECTED DOMINATING SET problem is FPT and has a kernel of size $\mathcal{O}^*(2^k k^{3k})$ on \mathbb{G}_5 . The next natural question to ask is whether PARAMETERIZED CONNECTED DOMINATING SET has a polynomial kernel on \mathbb{G}_5 . In this section, we answer this question in the negative: we show that PARAMETERIZED CONNECTED DOMINATING SET restricted to graphs of girth 5 or 6 does not have a polynomial kernel unless the Polynomial Hierarchy collapses to the third level. To this end, we use various notions and results from the recently developed theory of kernel lower bounds [15, 18, 39]; these are described in [Section 2.2.1](#).

4.3.1 Kernel Lower Bounds for PARAMETERIZED CONNECTED DOMINATING SET

As we show later in this section, the derived classical problem — the “unparameterized” version — of PARAMETERIZED CONNECTED DOMINATING SET is NP-complete when restricted to \mathbb{G}_5 . By [Theorem 2.2](#), to show that PARAMETERIZED CONNECTED DOMINATING SET has no polynomial kernels* on graphs of girth at least 5, it is sufficient to exhibit a composition algorithm for the problem on this class of graphs. Unfortunately, this task turns out to be quite hard, and we have not been able to devise a composition algorithm for this problem. To get around this difficulty, we make use of the second tool for obtaining kernel lower bounds, namely polynomial parameter transformations (See [Section 2.2.1](#)) and [Theorem 2.3](#). We introduce an intermediate problem, named PARAMETERIZED FAIR CONNECTED COLOURS, and show that its unparameterized version is NP-complete. The PARAMETERIZED FAIR CONNECTED COLOURS problem is easy to compose, and hence by [Theorem 2.2](#), it has no polynomial kernel. We then give a polynomial parameter transformation from PARAMETERIZED FAIR CONNECTED COLOURS to PARAMETERIZED CONNECTED DOMINATING SET in \mathbb{G}_6 , which implies, by [Theorem 2.3](#), that the latter problem has no polynomial kernel. A small modification in the reduction yields the same negative result for PARAMETERIZED CONNECTED DOMINATING SET on \mathbb{G}_5 as well.

* Unless $\text{CoNP} \subseteq \text{NP/Poly}$ and the Polynomial Hierarchy collapses to the third level; see [Chapter 2](#). For ease in reading, we drop the mention of this condition from now onwards; it is to be taken as being implicitly present whenever we mention the absence of polynomial kernels.

Recently, Cygan et al showed that the PARAMETERIZED CONNECTED DOMINATING SET problem does not have polynomial kernels on graphs of bounded degeneracy [28]. In order to do this, they introduced a new problem named PARAMETERIZED CONNECTED COLOURS which “nicely encapsulates the hardness of the connectivity requirement”.

PARAMETERIZED CONNECTED COLOURS

Input: A graph $G = (V, E)$, where the vertices V are (arbitrarily) coloured with k colours.
Parameter: k
Question: Does G contain a tree T on k vertices as a subgraph, where each vertex of T has a distinct colour?

For our purpose we define a variant of this problem, which we call PARAMETERIZED FAIR CONNECTED COLOURS. An assignment of colours to the vertices of a graph is said to be *proper* if the two end points of each edge in the graph have distinct colours. The colouring is said to be *fair* if all neighbours of each vertex in the graph receive distinct colours. In PARAMETERIZED FAIR CONNECTED COLOURS, the input graph is properly and fairly coloured:

PARAMETERIZED FAIR CONNECTED COLOURS

Input: A graph G , where the vertices $V(G)$ are *properly and fairly* coloured with k colours.
Parameter: k
Question: Does G contain a tree T on k vertices as a subgraph, where each vertex of T has a distinct colour?

The unparameterized version of this problem is NP-complete, by reduction from CNF SAT.

Lemma 4.4. *The derived classical problem associated with PARAMETERIZED FAIR CONNECTED COLOURS problem is NP-complete.*

Proof. Note that the input to the derived classical problem associated with PARAMETERIZED FAIR CONNECTED COLOURS consists of the input of PARAMETERIZED FAIR CONNECTED COLOURS, together with a unary encoding of the number k . The question is the same as that for PARAMETERIZED FAIR CONNECTED COLOURS. Let T be a subgraph of the input graph G such that T is a tree on k vertices which has all its vertices coloured with distinct colours. Such a subgraph T constitutes a polynomial-time verifiable witness to a

YES-instance of the problem, and so the PARAMETERIZED FAIR CONNECTED COLOURS problem is in NP.

To show NP-hardness, we reduce from the NP-complete CNF SAT problem [75]. In the CNF SAT problem, the input is a Boolean formula in conjunctive normal form — it is an “AND” of clauses, where each clause is an “OR” of (positive or negative) literals — and the question is whether there is an assignment of 0, 1 values to the variables of the formula such that the formula evaluates to 1. Let ϕ be a Boolean formula in conjunctive normal form, on the variables x_1, \dots, x_n and with clauses C_1, \dots, C_m . If a clause contains both a variable and its negation, then we can safely remove the clause and solve the problem on the remaining formula, since this clause evaluates to 1 for any assignment of values to the variables. So we assume, without loss of generality, that there is no clause that contains both a variable and its negation.

Given ϕ , we construct a graph* G on $m + 2n + 3$ vertices as follows, where the vertices are coloured using $m + n + 3$ colours. We add one vertex for each variable in ϕ , one for the negation of each variable, one for each clause, and three special vertices named r, a, b . That is, we define the vertex set to be $V(G) := \{r, a, b, x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n, C_1, \dots, C_m\}$. We add edges joining r to a and b , edges from a to each “variable” vertex, from b to each “negated variable” vertex, and from each “clause” vertex to the vertices corresponding to all the literals which appear in the clause. That is, we add the edges $\{r, a\}, \{r, b\}$ and $\{a, x_1\}, \{a, x_2\}, \dots, \{a, x_n\}, \{b, \bar{x}_1\}, \{b, \bar{x}_2\}, \dots, \{b, \bar{x}_n\}$, and for each vertex C_i , we add an edge from C_i to vertex $y \in \{x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n\}$ if and only if the literal y appears in clause C_i in the formula ϕ . This completes the construction of the graph G . We assign the colours 0, +, − to vertices r, a, b , respectively. For $1 \leq i \leq n$, we assign colour i to vertices x_i and \bar{x}_i , and for $1 \leq j \leq m$, we assign colour $n + j$ to vertex C_j . Finally†, we append a unary encoding of the number $m + n + 3$ to the encoding of this graph. This completes the construction; see Figure 4.3 for an illustration of the coloured graph in the construction.

Note that both the neighbours of the vertex r have distinct colours which are different from the colour of the vertex r . Similarly, all the neighbours of the vertex a have distinct colours different from the colour of a , and so also for b and each of the vertices x_i and \bar{x}_i . Since no clause contains both a variable and its negation, the same holds for all the “clause” vertices C_i as well. Thus the vertices of G are properly and fairly coloured with $n + m + 3$ colours. The reduced instance of the unparameterized version of PARAMETERIZED FAIR CONNECTED COLOURS is $(G\#m + n + 3)$, where $\#$ is a special symbol and the number

* This graph is somewhat similar to the clause-variable incidence graph of the formula ϕ .

† To meet the requirement stated in the definition of a derived classical problem — see Section 2.2.1.

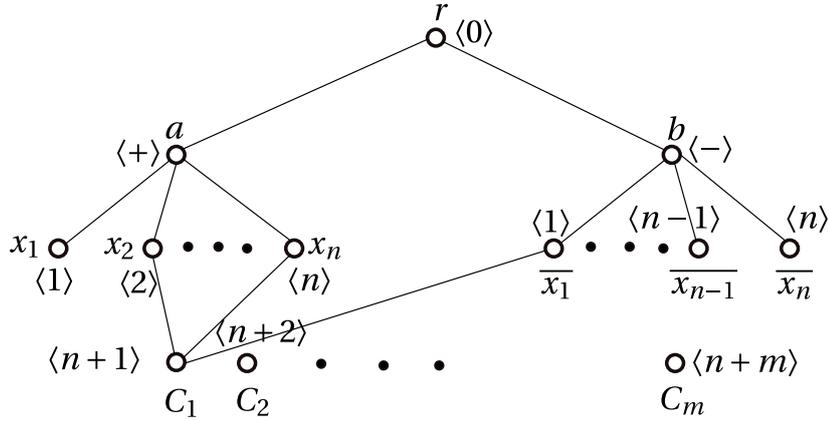


Figure 4.3: **Reduction from CNF SAT to unparameterized PARAMETERIZED FAIR CONNECTED COLOURS.** The colour of each vertex is indicated within angled brackets near the vertex.

$m + n + 3$ is encoded in unary. It remains to show that ϕ is satisfiable if and only if G contains an $m + n + 3$ -vertex tree as a subgraph whose vertices are all coloured distinctly.

Suppose ϕ is satisfiable, and let S be the set of vertices labelled by the literals (negative as well as positive) which are set to true by a satisfying assignment A of ϕ . Notice that A sets at least one literal in each clause of ϕ to true. Also, for each variable x_i , A sets exactly one of x_i, \bar{x}_i to true. Thus each vertex $C_i; 1 \leq i \leq m$ is adjacent to at least one of the vertices in S , and S contains exactly one vertex with each of the colours $\{1, 2, \dots, n\}$. It follows that the subgraph H of G induced on the vertex set $\{r, a, b, C_1, C_2, \dots, C_m\} \cup S$ is connected and has one vertex from each of the $n + m + 3$ colours $\{0, +, -, 1, 2, \dots, n + m\}$. Therefore G contains an $m + n + 3$ -vertex tree as a subgraph whose vertices are all coloured distinctly: indeed, any spanning tree of H is such a tree.

Now suppose G contains an $m + n + 3$ -vertex tree T as a subgraph whose vertices are all coloured distinctly. Because of the constraint on colours, the vertex set $V(T)$ of T must consist of $\{r, a, b, C_1, \dots, C_m\}$, and exactly n vertices from the set $X = \cup_{i=1}^n \{x_i, \bar{x}_i\}$ where exactly one vertex is chosen from $\{x_i, \bar{x}_i\}; 1 \leq i \leq n$. The unique path in T from any vertex $C_i; 1 \leq i \leq n$ to r must use a vertex in $S = X \cap V(T)$. Consider the assignment A of the formula ϕ which sets to true exactly those literals that appear in S . Since $|S \cap \{x_i, \bar{x}_i\}| = 1$ for $1 \leq i \leq n$, A is a valid assignment. Since each vertex C_i is adjacent to at least one vertex in S , the assignment satisfies every clause in ϕ , and so ϕ is satisfiable. Thus the reduction is sound, and the proof is complete. \square

The PARAMETERIZED FAIR CONNECTED COLOURS problem is easy to compose:

Lemma 4.5. *The PARAMETERIZED FAIR CONNECTED COLOURS problem is compositional.*

Proof. The composition consists of merely computing the disjoint union of the input graphs. That is, given t instances $(G_1, k), (G_2, k), \dots, (G_t, k)$ of PARAMETERIZED FAIR CONNECTED COLOURS, the composition algorithm constructs the disjoint union $G = \cup_{i=1}^t G_i$, and returns the instance (G, k) . If at least one of the input instances is a YES instance, then the k -coloured tree from that instance survives intact in G as well, and so (G, k) is a YES instance. Conversely, if (G, k) is a YES instance, then any k -coloured tree on k vertices that is a subgraph of G cannot span two different connected components of G , and so must appear intact in some graph G_i in the input; (G_i, k) is then a YES instance. \square

From [Theorem 2.2](#), [Lemma 4.4](#), and [Lemma 4.5](#) we get:

Lemma 4.6. *The PARAMETERIZED FAIR CONNECTED COLOURS problem does not have a polynomial kernel unless the Polynomial Hierarchy collapses to the third level.*

We now prove our main result by giving a polynomial parameter transformation (PPT) from PARAMETERIZED FAIR CONNECTED COLOURS to PARAMETERIZED CONNECTED DOMINATING SET on graphs with girth 5 or 6. Recall — [Definition 2.4](#) — that such a transformation runs in polynomial time and constructs an equivalent instance with a polynomially bounded parameter.

Theorem 4.3. *The PARAMETERIZED CONNECTED DOMINATING SET problem restricted to graphs of girth 5 or 6 does not admit a polynomial kernel unless the Polynomial Hierarchy collapses to the third level.*

Proof. Note that by [Theorem 2.3](#) and [Lemma 4.6](#) it is sufficient to show that there is a polynomial parameter transformation (PPT) from PARAMETERIZED FAIR CONNECTED COLOURS to each of these problems. We first describe a PPT from PARAMETERIZED FAIR CONNECTED COLOURS to PARAMETERIZED CONNECTED DOMINATING SET on graphs of girth six. Given an instance (G, k) of PARAMETERIZED FAIR CONNECTED COLOURS, we construct an instance (H, k') of PARAMETERIZED CONNECTED DOMINATING SET where H has girth six and k' is bounded by a polynomial in k .

We start with a copy of G . For each colour class (set of vertices of the same colour) \mathcal{C}_i of G , we add a new vertex v_i adjacent to all vertices of \mathcal{C}_i , and a new vertex g_i adjacent to v_i . The vertex g_i is essentially a guard vertex that will force v_i to be selected in any solution (to the PARAMETERIZED CONNECTED DOMINATING SET instance). We add a new vertex uv for each edge $\{u, v\}$ of G , and replace the edge $\{u, v\}$ by two new edges $\{u, uv\}, \{uv, v\}$. That is, we split each edge of G once. This helps to ensure that the girth of the resulting graph is more than four. For every pair of colour classes $\mathcal{C}_i, \mathcal{C}_j; i < j$ of G ,

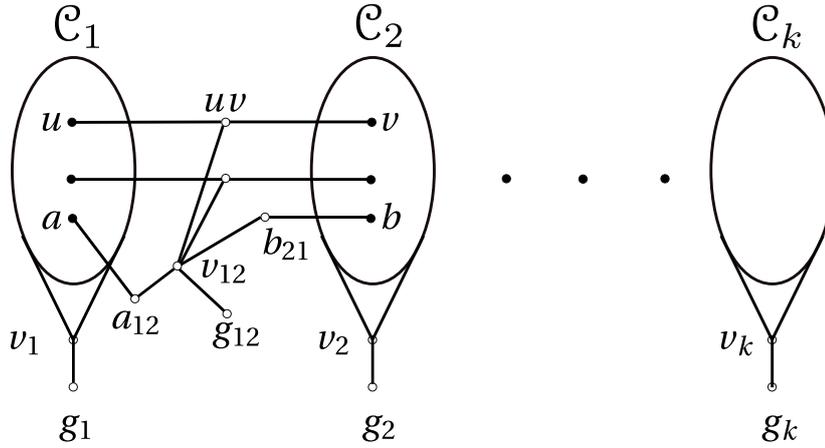


Figure 4.4: **Polynomial parameter transformation from PARAMETERIZED FAIR CONNECTED COLOURS to PARAMETERIZED CONNECTED DOMINATING SET.** Each \mathcal{C}_i consists of the vertices from one colour class of the input graph. See the text for a detailed description of the construction.

1. We add two new vertices v_{ij} and g_{ij} and the edge $\{v_{ij}, g_{ij}\}$. The vertex g_{ij} is a guard vertex which forces v_{ij} to be in any solution.
2. For each edge $\{u, v\}$ in G where $u \in \mathcal{C}_i, v \in \mathcal{C}_j$, we add the edge $\{uv, v_{ij}\}$ where uv is the new vertex that splits $\{u, v\}$. Roughly put, this construction ensures that if a solution to the PARAMETERIZED FAIR CONNECTED COLOURS instance — which is a tree — contains an edge from a vertex in \mathcal{C}_i to a vertex in \mathcal{C}_j , then this edge can be “used” to connect v_{ij} to the rest of the connected dominating set.
3. For each vertex $u \in \mathcal{C}_i$ that has no neighbour in \mathcal{C}_j , we add a new vertex u_{ij} and the edges $\{u, u_{ij}\}, \{u_{ij}, v_{ij}\}$ where v_{ij} is the vertex added in step 1. Roughly put, this construction ensures that if a solution to the PARAMETERIZED FAIR CONNECTED COLOURS instance does not contain an edge between \mathcal{C}_i and \mathcal{C}_j , then the vertex u_{ij} can be used to connect v_{ij} to the rest of the connected dominating set.
4. Symmetrically, for each vertex $u \in \mathcal{C}_j$ that has no neighbour in \mathcal{C}_i , we add a new vertex u_{ji} and the edges $\{u, u_{ji}\}, \{u_{ji}, v_{ij}\}$.

This completes the construction of H ; see [Figure 4.4](#). For later reference, let

- S be the set of vertices of the form uv introduced in H to split the edges of G ,
- $C = \mathcal{C}_1 \cup \dots \cup \mathcal{C}_k$,
- $X = \{g_i; 1 \leq i \leq k\}$,

- $Y = \{v_{ij}; 1 \leq i < j \leq k\}$,
- $Z = \{v_1, v_2, \dots, v_k\}$,
- $W = \{g_{ij}; 1 \leq i < j \leq k\}$,
- U be the set of all new vertices added in steps (3) and (4) above.

Let $A = C \cup X \cup Y$, $B = S \cup U \cup W \cup Z$. Then

1. $A \cup B$ is the vertex set of the constructed graph H — all the original vertices in the graph G are present in C , and all the vertices added by the construction are present in one of the other sets.
2. $H[A]$ is an independent set:
 - (a) Every edge which was present in G between two vertices in the set C is split by the construction, and so C is an independent set in H . From the construction, the sets X and Y are independent sets as well.
 - (b) There is no edge from any vertex in C to any vertex in $X \cup Y$: the latter set consists entirely of newly added vertices, and the construction does not add edges from C to any vertex in $X \cup Y$.
 - (c) The only neighbour of each $g_i \in X$ is the corresponding vertex $v_i \notin Y$. Hence there is no edge between X and Y .
3. $H[B]$ is an independent set.
 - (a) Each of the vertex sets S, U, W, Z is an independent set, and
 - (b) The construction adds no edges between any of these vertex sets.

Thus H is bipartite with the bipartition $A \uplus B$. Hence every cycle in H is of even length, and the smallest cycle has length at least 4. Also, H contains a 4-cycle if and only if there are two vertices in A which have two common neighbours in $V(H) \setminus A$. But no two vertices in A can have two common neighbours, as we argue below:

- The vertices in X are all of degree exactly one, and so they are not part of any cycle.
- In each of the remaining ways of forming a pair a, b of vertices from A , a and b have at most one common neighbour:
 - Any two vertices $a, b \in Y$ are at a distance of 4 from each other, so they have no common neighbour.

- For any colour class \mathcal{C}_i , two vertices $a, b \in C_i$ have exactly one common neighbour, namely v_i .
- For two distinct colour classes $\mathcal{C}_i, \mathcal{C}_j$, let $a \in C_i, b \in C_j$. If a, b are not adjacent in G , then they have no common neighbour in H . Otherwise, the new vertex that splits the edge $\{a, b\}$ is their only common neighbour in H .
- The only remaining possibility is $a \in C, b \in Y$. Without loss of generality, let $a \in \mathcal{C}_i, b = v_{ij}$. Since G is fairly coloured, the vertex a has either no neighbour or has exactly one neighbour (say a') in \mathcal{C}_j . In either case, a and b share exactly one neighbour, namely the new vertex (named a_{ij} or aa' , respectively) added to H to denote the presence or absence of such a neighbour in C_j .

It follows that H does not contain a 4-cycle, and so the smallest cycle in H has length at least 6. To see that the girth of H is indeed 6, note that we can assume without loss of generality that \mathcal{C}_1 contains at least two vertices, say a, b . Observe that there is a path of length two from a to v_{12} , and a path of length two from v_{12} to b . These paths meet only at v_{12} , and together with the two edges $\{b, v_1\}, \{v_1, a\}$ they form a cycle of length 6. Let $(H, k^2 + k)$ be the reduced instance. Now we argue that the reduction is sound.

Forward direction. Suppose G contains a tree T on k vertices, where each vertex of T has a distinct colour. Let $V(T) = \{t_1, t_2, \dots, t_k\}$, where $t_i \in \mathcal{C}_i$ for all i . Let T' be the “corresponding” tree in H : the vertex set of T' consists of $V(T)$ and all the new vertices in H that split the edges of T , and the edge set consists of all the new edges formed by splitting the edges of T . Thus T' is a tree on $2k - 1$ vertices. We now add more vertices and edges to T' to obtain a tree on $k^2 + k$ vertices that dominates all of H .

- For $1 \leq i \leq k$, we add the vertex v_i and the edge $\{v_i, t_i\}$ to T' . This adds k vertices. We also add a vertex v_{ij} for $1 \leq i < j \leq k$. This adds $\binom{k}{2}$ new vertices.
- For $1 \leq i < j \leq k$, if the vertex $t_i t_j$ is present in T' , then we add the edge $\{t_i t_j, v_{ij}\}$ to T' . Otherwise, let $a = t_i$. We add the vertex a_{ij} and the edges $\{a, a_{ij}\}, \{a_{ij}, v_{ij}\}$ to T' . This adds one vertex for each “non-edge” in T , and thus a total of $\binom{k}{2} - (k - 1)$ new vertices to T' .

This completes the construction of T' . Note that T' is a tree on $(2k - 1) + k + \binom{k}{2} + (\binom{k}{2} - (k - 1)) = k^2 + k$ vertices. In H ,

- the set $\{v_i \mid 1 \leq i \leq k\} \subseteq V(T')$ dominates all the vertices copied over from G and the new vertices $\{g_1, \dots, g_k\}$, and

- the set $\{v_{ij} \mid 1 \leq i < j \leq k\} \subseteq V(T')$ dominates all the other newly added vertices.

Thus T' is a connected dominating set of H on $k^2 + k$ vertices.

Reverse direction. Let D be a minimal connected dominating set of H with $1 < |D| \leq k^2 + k$. Observe first that vertices in $X \cup W$ are all pendant vertices, and all of their neighbours have degree at least 2. So, using an argument similar to the one used in the proof of [Theorem 4.1](#), $D \cap (X \cup W) = \emptyset$ and $N(X \cup W) = (Y \cup Z) \subseteq D$. Now since $G[D]$ is connected and $|D| \geq 2$, at least one neighbour of each vertex in D must also be in D . Observe that for any two vertices $u, v \in Y \cup Z$, $N[u] \cap N[v] = \emptyset$, and so each vertex in D can be the neighbour of at most one vertex in $Y \cup Z \subseteq D$. Thus for each vertex $v \in Y \cup Z$, D contains at least one distinct vertex $u \in (N(v) \setminus (Y \cup Z))$, and so $|D| \geq 2|Y \cup Z| = 2\binom{k}{2} + k = k^2 + k$. But $|D| \leq k^2 + k$ by assumption, and so $|D| = k^2 + k$. Thus exactly one neighbour of each vertex in $Y \cup Z$ is in D , and these neighbours are all distinct. In particular, D contains exactly one vertex from each set $\mathcal{C}_i; 1 \leq i \leq k$. Further, $D = (Y \cup Z) \cup N(Y \cup Z)$.

Let T_1 be a spanning tree of $H[D]$. Then $|V(T_1)| = |D| = k^2 + k$. From the above arguments we see that all vertices in $Y \cup Z$ are leaves in T_1 , and so $T_2 = T_1 \setminus (Y \cup Z)$ is also a tree, and $|T_2| = k^2 + k - \binom{k}{2} - k = k(k+1)/2$. Observe that for each vertex v in Y , the neighbour w of v in T_1 is either in U or in S . For each such v and the corresponding vertex w ,

- If $w \in U$, then w is a leaf in the tree T_2 ; remove w from T_2 .
- If $w \in S$, then w is either a leaf or has degree exactly two in T_2 . In either case, remove w from T_2 . If w was not a leaf in T_2 , then add an edge between its two neighbours in T_2 .

Observe that these steps yield a tree, say T . The tree T_2 had $k(k+1)/2$ vertices, and the steps above remove exactly $k(k-1)/2$ vertices from T_2 , one for each vertex in the set Y . Thus T has exactly k vertices. Since T_1 contained exactly one vertex from each set $\mathcal{C}_i; 1 \leq i \leq k$, and since the above operations did not remove any of these vertices, it follows that the vertex set of T consists of exactly one vertex of each colour. Observe that each edge added in the construction of H has at least one end point in a vertex which is not present in G . Therefore, if such a “new” edge is present in T , then T would have contained at least one of these “new” vertices. Since this is not the case, it follows that T contains none of the new edges added to H by the construction. Thus T is (isomorphic to) a subgraph of G . But T is a tree on k vertices where each vertex has a distinct colour, and so (G, k) is a YES instance of PARAMETERIZED FAIR CONNECTED COLOURS.

Graphs of Girth 5. A small modification to the above reduction suffices to show that the PARAMETERIZED CONNECTED DOMINATING SET problem has no polynomial kernel on graphs of girth 5 as well: Add three new vertices a, b, c and the four new edges required to complete the 5-cycle v_1, a, b, c, g_1 so that H has girth 5. The reduced instance is $(H, k^2 + k + 2)$. In the argument to show that this reduction is sound, both the directions go through exactly as above, once we observe that exactly one of the sets $\{v_1, a, g_1\}, \{v_1, a, b\}, \{v_1, g_1, c\}$ is contained in any minimal connected dominating set of H . \square

Observe that the polynomial parameter transformations described in the proof of [Theorem 4.3](#) are in fact polynomial-time many-to-one reductions. It follows that the PARAMETERIZED CONNECTED DOMINATING SET problem is NP-hard when restricted to graphs of girth 5 or 6.

Lemma 4.7. *The PARAMETERIZED CONNECTED DOMINATING SET problem is NP-complete on graphs of girth 5 or 6.*

4.4 On Graphs of girth 7 or More: A Cubic Vertex Kernel

We saw in the previous section that if cycles of length 5 or 6 are allowed in the input, then the PARAMETERIZED CONNECTED DOMINATING SET problem has no polynomial kernel. We now show that if we disallow cycles of length at most 6 in the input, then the problem has a kernel with $\mathcal{O}(k^3)$ vertices. Recall the FPT algorithm for PARAMETERIZED CONNECTED DOMINATING SET of graphs of girth at least 5 — [Section 4.2](#) — where we partitioned the vertex set of the input graph into three “colours” red, white and blue. We were able to bound the sizes of the red and blue sets by $\mathcal{O}(k^2)$, but could (implicitly) give only an exponential bound on the number of white vertices. The absence of cycles of length less than 7 helps us get a polynomial bound on the size of the white set of vertices as well, and this gives us a polynomial kernel for the problem on these graphs.

4.4.1 A Cubic Kernel for the Coloured Version

To show that PARAMETERIZED CONNECTED DOMINATING SET has an $\mathcal{O}(k^3)$ vertex kernel on \mathbb{G}_7 , we first derive such a kernel for the PARAMETERIZED CONNECTED RWB-DOMINATING SET problem which we defined in [Section 4.2](#). As we saw in that section, it is straightforward to reduce PARAMETERIZED CONNECTED DOMINATING SET to PARAMETERIZED CONNECTED RWB-DOMINATING SET in polynomial time, with no change in the parameter. A reduction in the reverse direction is only slightly more involved, and

can be done in polynomial time with an $\mathcal{O}(k)$ additive increase in the number of vertices. This yields the desired kernel for PARAMETERIZED CONNECTED DOMINATING SET. As before, our reduction rules colour the vertices of G red, white, and blue. Red vertices are those that must necessarily be in any connected dominating set of G of size at most k . White vertices are those non-red vertices that are dominated by the red vertices, and blue vertices are the rest. Recall that we use R_G, W_G, B_G , respectively, to denote the red, white, and blue vertices in a graph G .

Let (G, k) be the input to PARAMETERIZED CONNECTED DOMINATING SET, where $G \in \mathbb{G}_7$. Our kernelization algorithm starts by colouring all the vertices of G blue to obtain an rwb-graph H . By the proof of [Lemma 4.2](#), this step is sound. The algorithm now exhaustively applies the following reduction rules in the given order:

(Rule 1) Let S be the set of blue vertices in H , each of which has at least $k + 1$ blue neighbours. Colour all the vertices of S red and all the blue neighbours in $N(S)$ white.

(Rule 2) If $|R_H| > k$ or $|B_H| > k^2 + k$, then say No and stop.

(Rule 3) If H contains an isolated blue vertex, then say No and stop.

(Rule 4) If H contains a pendant blue or white vertex u adjacent to a vertex v , then remove u from H . If v is not red, then colour v red and colour all the remaining blue neighbours of v white.

The correctness of Rule 1 follows from [Lemma 4.3](#); see also the first paragraph in the proof of [Lemma 4.1](#). The bound obtained for $|B_G|$ in the proof of [Lemma 4.1](#) — see the third paragraph of that proof — justifies Rule 2. Observe that we need to include any isolated blue vertex in the dominating set (to dominate that vertex), but as this vertex is isolated, the resulting dominating set will not induce a connected graph. This implies that if an isolated blue vertex is present in H , then H has no connected rwb-dominating set, and this in turn justifies Rule 3.

Let u be a pendant blue or white vertex in H , adjacent to a vertex v . Observe that at least one of u, v must be in any dominating set of H , in order to dominate the vertex u . Since u is a pendant vertex which is not red, it is not present in any *minimal* connected rwb-dominating set of H . This follows by an argument similar to the one used in the proof of [Theorem 4.1](#). Thus the vertex v is present in every minimal connected rwb-dominating set of H , and this justifies Rule 4.

We now show how to bound $|W_H|$ for the resulting graph H using the fact that H has no cycles of length 6 or less.

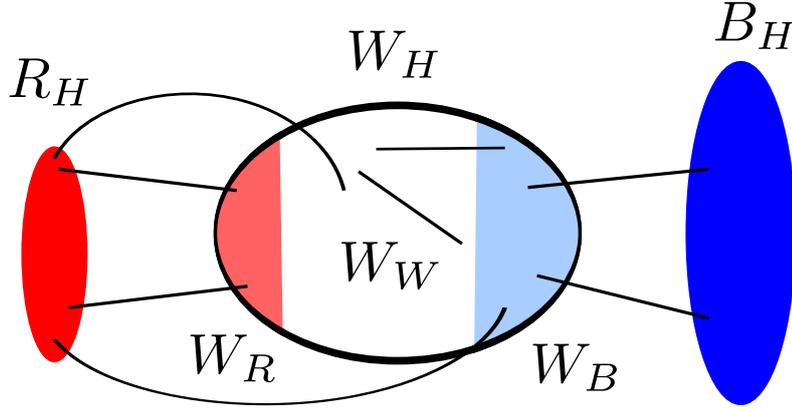


Figure 4.5: **Partitioning the set of white vertices.** R_H , W_H , B_H are, respectively, the red, white, and blue vertex sets in the graph H . W_R is the set of white vertices which have *only* red neighbours. W_W is the set of white vertices which have at least one white neighbour. W_B is the remaining set of white vertices, which have at least one blue neighbour.

Lemma 4.8. *Let (G, k) be a YES instance of PARAMETERIZED CONNECTED DOMINATING SET where $G \in \mathbb{G}_7$, and let H be the graph obtained from G by exhaustively applying Rule 1 to Rule 4. Then $|W_H| \leq k^3 + \frac{5}{2}k^2 - \frac{3}{2}k$.*

Proof. As in Figure 4.5 we divide W_H into three parts, $W_H = W_B \cup W_R \cup W_W$, where

- W_B is the set of all white vertices that have *at least one* blue neighbour,
- W_R is the set of all white vertices that have *only* red neighbours, and
- W_W is the set of all white vertices in $W_H \setminus W_B$ that have *at least one* white neighbour.

We now bound the size of each of these sets.

Since Rule 1 has been applied to the graph H , any blue vertex in H has degree at most k , and so can have at most k white neighbours. Thus $|W_B| \leq k|B_H| \leq k(k^2 + k)$.

Since Rule 4 has been applied to H , each vertex in W_R has at least two red neighbours. Further, recall — [Observation 1](#) — that no two vertices in H have more than one common neighbour. It follows that $|W_R| \leq \binom{|R_H|}{2} \leq \binom{k}{2}$. We note in passing that although the vertices in W_R do not contribute to dominating the graph, we cannot just remove these vertices from H since they could be useful in providing connectivity in some smallest connected dominating set.

We now bound the size of the last piece of W_H , namely the set W_W of all white vertices in $W_H \setminus W_B$ which have at least one white neighbour. Let E_W be the set of all edges $e \in E(H)$ where both end vertices of e are white. Consider a map which associates each vertex $w \in W_W$ with an arbitrary edge $e \in E_W$ with which w is incident. Observe that this

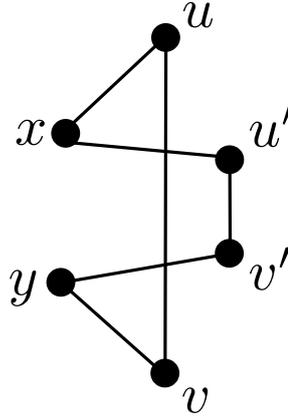


Figure 4.6: **A short cycle in H .** Let x, y be a pair of vertices in the graph H . If there are two distinct edges $\{u, v\}, \{u', v'\}$ in H where x is incident with u, u' , and y is incident with v, v' , then there is a cycle of length at most 6 in H — the length would be 5 if the two edges $\{u, v\}, \{u', v'\}$ share an end vertex.

associates every vertex in W_w with some edge in E_w , and at most two vertices in W_w to any edge in E_w . It follows that $|W_w| \leq 2|E_w|$. We now bound $|E_w|$ by a function of k .

Observe that each white vertex in H is adjacent to some red vertex. Also, the two end points of any edge in E_w are *not* adjacent to the same red vertex, or else there would be a 3-cycle in H . Therefore, for any edge $(u, v) \in E_w$, there is a pair x, y of red vertices in H such that u is adjacent to x and v is adjacent to y .

Consider any pair x, y of red vertices. If possible, let $\{u, v\}, \{u', v'\} \in E_w$ be two distinct edges where x is incident with u, u' , and y is incident with v, v' . Then the subgraph of H induced by the vertex set $\{x, y, u, v, u', v'\}$ forms a cycle of length at most 6 (Figure 4.6), which contradicts the fact that H has girth at least 7. Therefore, for any pair x, y of red vertices in H , there is *at most* one edge $(u, v) \in E_w$ such that u is adjacent to x and v is adjacent to y .

Putting these two together, we get $|E_w| \leq \binom{|R|}{2} \leq \binom{k}{2}$, and so $|W_w| \leq 2|E_w| \leq k^2 - k$.

Putting all these bounds together, if the graph G has a connected dominating set of size at most k , then the number of white vertices in the graph H is at most $k^3 + \frac{5}{2}k^2 - \frac{3}{2}k$. \square

Note that all these rules can be applied in polynomial time. Once this is done we have — because of Rule 2 — that $|R_H| \leq k$ and $|B_H| \leq k^2 + k$. By Lemma 4.8, $|W_H| \leq k^3 + \frac{5}{2}k^2 - \frac{3}{2}k$, and so we have

Lemma 4.9. *The PARAMETERIZED CONNECTED RWB-DOMINATING SET problem has a kernel on at most $k^3 + \frac{7}{2}k^2 + \frac{k}{2}$ vertices on the class of graphs of girth at least 7.*

4.4.2 Removing the Colours

Starting with an instance (G, k) of PARAMETERIZED CONNECTED DOMINATING SET, let (G', k) be the equivalent instance of PARAMETERIZED CONNECTED RWB-DOMINATING SET obtained by applying [Lemma 4.9](#). To obtain an instance of (uncoloured) PARAMETERIZED CONNECTED DOMINATING SET from this cubic vertex kernel for PARAMETERIZED CONNECTED RWB-DOMINATING SET, we do the following:

- Attach a new pendant vertex to each red vertex in G' , and,
- remove all colours from the vertices of the resulting graph to obtain a graph H .

The new instance is (H, k) . Let S be a connected rwb-dominating set of the rwb-graph G' , of size at most k . Observe that every red vertex in G' is contained in the set S , by definition; $R_{G'} \subseteq S$. Also, S dominates all the vertices in G' . Since all the new vertices added to G' are dominated by $R_{G'}$, it follows that S dominates all vertices in H as well. Since $G'[S]$ is connected, so is $H[S]$, and so S is a connected dominating set of H of size at most k .

Conversely, if S' is a minimal connected dominating set of H of size at most k , then it follows by the same reasoning as in the proof of [Theorem 4.1](#) that

- no pendant vertex added during the construction is in S' , and,
- all the neighbours of these pendant vertices are in S' .

Thus $S' \subseteq V(G')$ and $R_{G'} \subseteq S'$, and so S' itself is a connected rwb-dominating set of G' of size at most k . The procedure for removing colours is thus sound, and it adds at most k new vertices. Thus we get

Theorem 4.4. *The PARAMETERIZED CONNECTED DOMINATING SET problem has a kernel with at most $k^3 + \frac{7}{2}k^2 + \frac{3k}{2} = \mathcal{O}(k^3)$ vertices on the class of graphs of girth at least 7.*

4.5 Conclusion

In this chapter we studied the effect of excluding short cycles from the input graph, on the kernelization complexity of the PARAMETERIZED CONNECTED DOMINATING SET problem. We obtained a diverse kernelization landscape, showing that the problem becomes progressively easier as the girth of the input graph increases. Specifically, we show that

- PARAMETERIZED CONNECTED DOMINATING SET is $W[2]$ -hard on graphs which contain cycles of length 3 or 4, and so it does not have a kernel of *any* size on such graphs unless $FPT=W[2]$.
- On any class of graphs which have girth at least 5, PARAMETERIZED CONNECTED DOMINATING SET has an FPT algorithm which runs in $\mathcal{O}(2^k k^{3k} \cdot n^c)$ time where n is the number of vertices in the input graph and c is a constant independent of n and k . As a consequence, the problem has a kernel of size $2^k k^{3k}$ on graphs of girth at least 5.
- Unless the Polynomial Hierarchy (PH) collapses to the third level, PARAMETERIZED CONNECTED DOMINATING SET has *no* polynomial kernel on graphs which contain cycles of length at most 6.
- On any class of graphs which have girth at least 7, PARAMETERIZED CONNECTED DOMINATING SET has a kernel on $\mathcal{O}(k^3)$ vertices.

On the way to proving the kernel lower bound on graphs with girth at most 6 we introduced a new problem, namely PARAMETERIZED FAIR CONNECTED COLOURS, and showed that this problem has no polynomial kernels. This intermediate result is interesting in its own right; we feel that it could be used to show kernelization lower bounds for other connectivity problems on graphs that exclude small cycles.

As noted before, most kernelization and FPT results for W -hard problems are for graph classes characterized by excluded *minors*. Our results add to the small but growing collection of such results for graph classes characterized by excluded *subgraphs*. One interesting direction of future work is to study how the exclusion of small cycles — or other small graphs — as subgraphs affects the kernelization complexity of other W -hard parameterized problems.

Part III

Covering Problems

Pathwidth-One Vertex Deletion

PATHWIDTH is a notion introduced by Robertson and Seymour in the Graph Minors series [104] as a measure of how “path-like” a graph is. A graph has pathwidth at most one if and only if it is a collection of *caterpillars*, where a caterpillar is a special kind of tree: it is a tree that becomes a path (called the *spine* of the caterpillar) when all its pendant vertices are removed. Graphs of pathwidth at most one are thus a very special kind of forests, and have even less structure than forests (which are themselves very “simple” graphs). As a consequence, some problems that are NP-hard even on forests can be solved in polynomial time on graphs of pathwidth at most one. Examples include (Weighted) Bandwidth [10, 82, 98], the Proper Interval Coloured Graph problem, and the Proper Coloured Layout problem [6].

In this chapter we focus on the following problem : Given a graph G and an integer k as input, find whether G contains a set of at most k vertices whose removal from G results in a graph of pathwidth at most one. We call such a set of vertices a pathwidth-one deletion set (PODS), and the problem the PATHWIDTH-ONE VERTEX DELETION problem. It follows from an NP-hardness “meta-result” of Lewis and Yannakakis that this problem is NP-complete. We initiate the study of the parameterized complexity of the PATHWIDTH-ONE VERTEX DELETION problem, parameterized by the solution size k .

The FEEDBACK VERTEX SET problem is closely related to PATHWIDTH-ONE VERTEX DELETION, and has been extensively studied. A *feedback vertex set* (FVS) of a graph is a set of vertices whose deletion from the graph results in a forest. Given a graph G and an integer k as input, the FEEDBACK VERTEX SET problem asks whether G has an FVS

of size at most k . This problem is NP-complete [75]. Its natural parameterized version — with the solution size k as the parameter, which we call PARAMETERIZED FEEDBACK VERTEX SET — is fixed parameter tractable (FPT) and has a polynomial kernel. The best known deterministic FPT algorithm for the problem [22] runs in $\mathcal{O}^*(3.83^k)$ time, and the smallest known kernel [107] has size $\mathcal{O}(k^2)$.

Our Results

We show that the PATHWIDTH-ONE VERTEX DELETION problem parameterized by the solution size k — which we call PARAMETERIZED PATHWIDTH-ONE VERTEX DELETION — (i) has a kernel with $\mathcal{O}(k^4)$ vertices, and (ii) can be solved in $\mathcal{O}^*(7^k)$ time.

Note that, in general, a pathwidth-one deletion set “does more” than a feedback vertex set: It “kills” all cycles in the graph, like a feedback vertex set, and, in addition, it kills all non-caterpillar trees in the graph. In fact, the difference in the sizes of a smallest feedback vertex set and a smallest pathwidth-one deletion set of a graph can be arbitrarily large. For example, the treewidth of any binary tree is 1, while for any integer c there exists [104] a binary tree T_c of pathwidth at least $c + 1$. Removing a single vertex from a graph will reduce the pathwidth by at most one, and so for T_c , the difference between the two numbers is at least c . Partly as a consequence of such differences, many of the techniques and reduction rules that have been developed for obtaining FPT algorithms and kernels for the PARAMETERIZED FEEDBACK VERTEX SET problem do *not* carry over to the PARAMETERIZED PATHWIDTH-ONE VERTEX DELETION problem. Instead, we use a characterization of graphs of pathwidth at most 1 to obtain the FPT algorithm and the polynomial kernel.

Organization of the rest of the chapter

In [Section 5.1](#) we show that the PARAMETERIZED PATHWIDTH-ONE VERTEX DELETION problem is NP-complete, and describe an FPT algorithm for the problem which runs in $\mathcal{O}^*(7^k)$ time. We show in [Section 5.2](#) that the problem has a kernel with $\mathcal{O}(k^4)$ vertices. We conclude in [Section 5.3](#).

Notation

All the graphs in this chapter are finite and undirected. In general we follow the graph terminology of [Section 2.1](#). A *caterpillar* is a tree that becomes a path (called the *spine* of the caterpillar) when all its pendant vertices are removed. A *nontrivial* caterpillar is one that contains at least two vertices. A T_2 is the graph on seven vertices shown in [Figure 5.1](#).

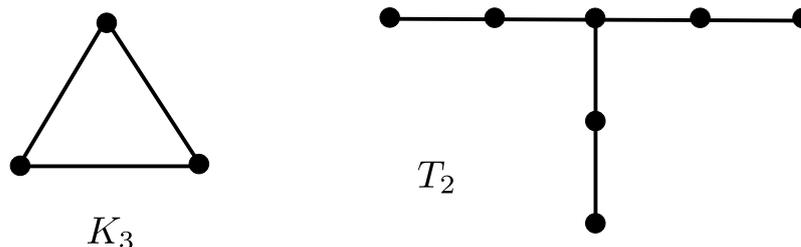


Figure 5.1: The set of excluded minors for graphs of pathwidth at most one.

The *centre* of a T_2 is the one vertex of degree 3, and its *leaves* are the three vertices of degree 1. A K_3 is the complete graph on three vertices — equivalently, the cycle on three vertices — depicted in Figure 5.1. A C_4 is the cycle on four vertices.

A *graph property* is a subset of the set of all graphs. Graph property Π is said to hold for graph G if $G \in \Pi$. Π is said to be *nontrivial* if Π and its complement set are both infinite. Π is said to be *hereditary* if Π holds for every induced subgraph of graph G whenever it holds for G . The *membership testing* problem for Π is to test whether Π holds for a given input graph.

5.1 A Single-Exponential FPT Algorithm

In this section we formally define the PARAMETERIZED PATHWIDTH-ONE VERTEX DELETION problem, show that it is NP-complete, and describe a simple $\mathcal{O}^*(7^k)$ FPT algorithm for the problem. We begin with the observation that caterpillars are the quintessential graphs of pathwidth at most one:

Fact 1. [9] *A graph G has pathwidth at most one if and only if it is a collection of vertex-disjoint caterpillars.*

To show that PATHWIDTH-ONE VERTEX DELETION is NP-hard, we make use of the following general NP-completeness result is due to Lewis and Yannakakis [86]:

Fact 2. *The following problem is NP-complete for any nontrivial hereditary graph property Π for which the membership testing problem can be solved in polynomial time:*

INPUT: Graph $G = (V, E)$, positive integer k .

QUESTION: Is there a subset $S \subseteq V$, $|S| \leq k$ such that $G[V \setminus S] \in \Pi$?

The NP-completeness of the PATHWIDTH-ONE VERTEX DELETION problem is a simple consequence of this result:

Corollary 5.1. *The PATHWIDTH-ONE VERTEX DELETION problem is NP-complete.*

Proof. Let Π be the set of all graphs of pathwidth at most one. Since Π contains all caterpillars, it is an infinite set. Since every cycle is in the complement $\bar{\Pi}$, the set $\bar{\Pi}$ is infinite as well. Thus Π is nontrivial. Deleting a vertex from a graph that does not have a cycle or T_2 as a minor will not introduce such a minor, and so the set Π is hereditary.

By a direct application of the definition of a caterpillar, we can check in polynomial time whether each component of a graph G is a caterpillar. Together with [Fact 1](#), this implies that membership testing for Π can be done in polynomial time. The NP-completeness now follows from [Fact 2](#). \square

In the rest of the chapter we focus on the parameterized complexity of the PARAMETERIZED PATHWIDTH-ONE VERTEX DELETION problem. We now describe an $\mathcal{O}^*(7^k)$ time FPT algorithm, and in the next section we describe a kernel for the problem with $\mathcal{O}(k^4)$ vertices. Let $(G = (V, E), k)$ be the input instance, where $|V| = n$. Let $S \subseteq V$ be a pathwidth-one deletion set of G of size at most k . Observe that if (G, k) is a YES instance, then the number of edges in G is at most $k(n-1) + (n-1) = (k+1)(n-1)$. The first term on the left is a trivial upper bound on the number of edges that are incident on the vertices in S ; the second term is an upper bound on the number of edges in $G \setminus S$ — recall that $G \setminus S$ is a forest. So, if G has more than $(k+1)(n-1)$ edges, then we can immediately reject the input. Since each transformation (reduction rule) which we describe below is sound, and no rule increases the number of vertices or edges, from now on we assume, without loss of generality, that the graph has at most $(k+1)(n-1)$ edges.

The kernel arguments are based on [Fact 1](#), while our starting point for the FPT algorithm is the following characterization, in terms of excluded minors, of graphs of pathwidth at most one [[21](#), [48](#)]:

Fact 3. *A graph G has pathwidth at most one if and only if it does not contain K_3 or T_2 as a minor.*

[Fact 3](#) is not very helpful in the given form in checking for a small pathwidth-one deletion set. Instead, we derive and use the following alternate characterization and the two succeeding lemmas:

Lemma 5.1. *A graph G has pathwidth at most one if and only if it does not contain a cycle or a T_2 as a subgraph.*

Proof. If G has pathwidth at most one, then by [Fact 3](#) it does not contain K_3 or T_2 as a minor; it follows that G does not contain any cycle or T_2 as a subgraph.

Conversely, assume that G does not contain a cycle or a T_2 as a subgraph. Suppose G contains a K_3 , say C , as a minor, obtained by contracting a set of edges E' and deleting a set of edges E'' of a subgraph H of G . If we replace the contracted vertices with the original edges E' and add the deleted edges E'' , we obtain a cycle which is a subgraph of the original graph (from which we obtained C as a minor), a contradiction.

Similarly, if G has a T_2 as a minor, then replacing the contracted vertices with the original edges and adding the deleted edges gives rise to a supergraph of T_2 , a contradiction. Thus G does not contain either K_3 or T_2 as a minor, and so by [Fact 3](#) the graph G has pathwidth at most one. \square

The next two lemmas imply that we can decide in polynomial time whether a graph has pathwidth at most 1. Moreover, if the pathwidth is more than 1, then we can, in effect, find “all” the obstructions in polynomial time as well.

Lemma 5.2. *Let $\mathcal{S} = \{T_2, K_3, C_4\}$. Given a graph $G = (V, E); |V| = n$, we can find whether G contains a subgraph H that is isomorphic to one of the graphs in \mathcal{S} , and also locate such an H if it exists, in $\mathcal{O}(kn^2)$ time.*

Proof. It is well-known that we can find the girth (length of a shortest cycle) of a graph by doing a breadth-first search (BFS) from each vertex. The same algorithm finds a smallest cycle in the graph as well, and so we can use it to check for and locate a K_3 or C_4 in G . Suppose G contains neither of these graphs as a subgraph. To check if G contains a T_2 , we guess the centre vertex v of the T_2 and do a BFS starting from v (the level 0 vertex). Since G does not contain K_3 as a subgraph, there is a T_2 with v as the centre if and only if at least three vertices in level 1 of the BFS have at least one neighbour each in level 2. We can combine the two tests to obtain an algorithm of the required kind that runs in $\mathcal{O}((|V|)(|V| + |E|)) = \mathcal{O}(n(n + (k + 1)(n - 1))) = \mathcal{O}(kn^2)$ time. \square

A graph which does not contain any of $\{T_2, K_3, C_4\}$ as a subgraph has a particularly simple structure.

Lemma 5.3. *Let $\mathcal{S} = \{T_2, K_3, C_4\}$. If G is a graph that does not contain any element of \mathcal{S} as a subgraph, then each connected component of G is either a tree, or a cycle with zero or more pendant vertices (“hairs”) attached to it.*

Proof. Let G be a graph that does not contain any element of \mathcal{S} as a subgraph, and let X be a connected component of G that is not a tree. Then X contains a cycle. Let C be a smallest cycle in X . Then C has length at least 5. Suppose there is a path $\langle a, b, c \rangle$ in X , where a is a vertex on C and b is not. Then $c \notin V(C)$, or else a, b, c and the shorter path from c to a on C form a cycle of length at most 4, a contradiction. Let x, y be the two

neighbours of a on C , and let $x' \neq a, y' \neq a$ be neighbours of x, y on C . Then x, y, x', y' are all distinct, and a, b, c, x, y, x', y' form a T_2 in X with a at the centre, a contradiction. It follows that for any vertex $u \in V(C)$, any neighbour $v \notin V(C)$ of u is a pendant vertex in X , and the lemma follows. \square

5.1.1 An FPT algorithm for PARAMETERIZED PATHWIDTH-ONE VERTEX DELETION

We are now ready to describe the simple FPT algorithm. Let $(G = (V, E), k)$ be the input instance, where $|V| = n$. We use a branching strategy inspired by [Lemma 5.1](#) and [Lemma 5.3](#). First we locate a (not necessarily induced) subgraph T of G that is isomorphic to one of $\mathcal{S} = \{T_2, K_3, C_4\}$. By [Lemma 5.2](#), this can be done in $\mathcal{O}(kn^2)$ time. At least one of the (at most seven) vertices of T must be in any pathwidth-one deletion set of G . So we branch on the vertices of T : We pick each one, in turn, into the minimal pathwidth-one deletion set that we are constructing, delete the picked vertex and all its adjacent edges, and recurse on the remaining graph after decrementing the parameter by one.

The leaves of this recursion tree correspond to graphs which do not have a subgraph isomorphic to any graph in \mathcal{S} . By [Lemma 5.3](#), each connected component of such a graph is a tree, or a cycle with zero or more pendant vertices (“hairs”) attached to it. The trees can be ignored — they do not have a T_2 as a subgraph — and each cycle (with or without hairs) forces exactly one vertex into any minimal solution. Thus the base case of the recursion can be solved in linear time.

This is a 7-way branching, where the depth of the recursion is at most k , and where the algorithm spends $\mathcal{O}(kn^2)$ time at each node. Hence we have

Theorem 5.1. *The PATHWIDTH-ONE VERTEX DELETION problem parameterized by the solution size k has an FPT algorithm that runs in $\mathcal{O}(n^2 \cdot 7^k k)$ time.*

By a folklore result of parameterized complexity (See [Chapter 2](#)), it follows immediately from [Theorem 5.1](#) that the PATHWIDTH-ONE VERTEX DELETION problem parameterized by the solution size k has a kernel of size $\mathcal{O}(7^k k)$. We now show that the kernel size can be brought down significantly from this trivial bound.

5.2 A Quartic Kernel

We now turn to the main result of this chapter. We describe a polynomial-time algorithm (the *kernelization algorithm*) that, given an instance (G, k) of PARAMETERIZED PATHWIDTH-ONE VERTEX DELETION, returns an instance (G', k') (the *kernel*) of PARAMETERIZED PATHWIDTH-ONE VERTEX DELETION such that (i) (G, k) is a YES instance

Algorithm 3 The kernelization algorithm

```

1: procedure KERNELIZE( $G, k$ )
2:    $CurrentInstance \leftarrow (G, k)$ 
3:   repeat
4:     Apply Rule 1 to Rule 6, updating  $CurrentInstance$  with the output of each
       rule.
5:   until None of the rules cause any change to  $CurrentInstance$ .
6: end procedure

```

if and only if (G', k') is a YES instance, (ii) G' has $\mathcal{O}(k^4)$ vertices, and (iii) $k' \leq k$. The kernelization algorithm — [Algorithm 3](#) — exhaustively applies the reduction rules of [Section 5.2.1](#) to the input instance. The resulting instance, to which no rule applies, is said to be *reduced* with respect to the reduction rules. To demonstrate a kernel for the problem with $\mathcal{O}(k^4)$ vertices, it suffices to show that

1. The rules can be exhaustively applied in polynomial time;
2. Each rule is *sound*: the output of a rule is a YES instance if and only if its input is a YES instance; and
3. If the input instance (G, k) is a YES instance, then the reduced instance (G', k') has $\mathcal{O}(k^4)$ vertices.

The reduction rules are based on the following idea: Suppose $(G = (V, E), k)$ is a YES instance of the problem that is reduced with respect to the reduction rules. Then there is a set $S \subseteq V$, $|S| \leq k$ such that $G[V \setminus S]$ is a collection of caterpillars, and it suffices to show that $|V \setminus S| = \mathcal{O}(k^4)$. We express $V \setminus S$ as the union of different kinds of vertices, and devise reduction rules that help us bound the total number of vertices of each kind. To be more specific, we set $V \setminus S = V_1 \cup V_2 \cup V_3 \cup V_4 \cup V_5$ where

1. $V_1 = \{v \in (V \setminus S); N(v) \cap (V \setminus S) = \emptyset \text{ and } |N(v) \cap S| \leq 1\}$
2. $V_2 = \{v \in (V \setminus S); N(v) \cap (V \setminus S) = \emptyset \text{ and } |N(v) \cap S| \geq 2\}$
3. $V_3 = \{v \in ((V \setminus S) \setminus V_1); v \text{ lies on the spine of a nontrivial caterpillar in } G[V \setminus S]\}$
4. $V_4 = \{v \in (V \setminus S); |N(v) \cap S| = 0 \text{ and } v \text{ is a pendant vertex in } G[V \setminus S]\}$
5. $V_5 = \{v \in (V \setminus S); |N(v) \cap S| \geq 1 \text{ and } v \text{ is a pendant vertex in } G[V \setminus S]\}$

It is not difficult to verify that these sets together exhaust $V \setminus S$. We now state the reduction rules and describe their consequences .

5.2.1 The Reduction Rules

For each rule below, let $(H = (V_H, E_H), k)$ be the instance on which the rule is applied, and (H', k') the resulting instance. Let $G = (V, E)$ be a YES instance of the problem that is reduced with respect to all the reduction rules, and let S, V_1, \dots, V_5 be as described above. To bound the sizes of various subsets of $V \setminus S$, we use the fact that no reduction rule applies to G .

We show that each reduction rule is sound. That is, we show that for each rule, (H, k) is a YES instance if and only if (H', k') is a YES instance. We also show that each rule can be implemented in polynomial time. In each case, n is the number of vertices in the input to the kernelization algorithm.

Rule 1. If a connected component $H[X]; X \subseteq V_H$ of H has pathwidth at most 1, then remove X from H . The resulting instance is $(H' = H[V_H \setminus X], k' = k)$.

Claim 1. *Rule 1 is sound, and can be applied in $\mathcal{O}(kn)$ time.*

Proof. The connected component $H[X]$ does not intersect any forbidden structure and thus does not affect any solution of the problem.

By doing a single breadth-first search (BFS) of H , we can find all the acyclic connected components of H . To check if an acyclic component C contains a T_2 , we delete all the leaves (vertices of degree one) in C and check if the remaining graph is a simple path. C contains a T_2 if and only if the remaining graph is *not* a simple path. Using a queue and an array to keep track of the leaves and the degrees of the vertices in C , respectively, all this can be done in $\mathcal{O}(|V_H| + |E_H|) = \mathcal{O}(n + (k+1)(n-1)) = \mathcal{O}(kn)$ time. \square

Rule 2. If a vertex u in H has two or more pendant neighbours, then delete all but one of these pendant neighbours to obtain H' . The resulting instance is $(H', k' = k)$.

We need the following simple observation and an additional lemma to prove the soundness of [Rule 2](#).

Observation 1. *Let G be any graph of pathwidth at most one. Adding new degree zero vertices to G or adding new pendant neighbours to an isolated vertex u of G does not add a cycle or a T_2 to G .*

Lemma 5.4. *Let G be any graph that does not have any subgraph isomorphic to T_2 . If v is a vertex in G such that by adding some number $l \geq 1$ of pendant vertices as neighbours to v we can obtain a graph H that contains a T_2 as a subgraph, then v has no pendant neighbours in G .*

Proof. Assume to the contrary that v has a pendant neighbour u in G . Consider any T_2 , say t , in H . t contains at least one of the new pendant vertices; say it contains w ; $w \neq u$. Since w is pendant in H , the degree of w in t is exactly 1, and so w is one of the leaf vertices of t ; its only neighbour in t is v , which in turn is a non-central internal vertex of t . None of the internal vertices of a T_2 has two distinct pendant neighbours, and so u is not in t . It is evident that one can remove w from t and add u and the edge $\{u, v\}$ to the resulting subgraph to obtain a T_2 consisting entirely of vertices in G , a contradiction. \square

We are now ready to prove the soundness of [Rule 2](#).

Claim 2. [Rule 2](#) is sound, and can be applied in $\mathcal{O}(kn)$ time.

Proof. The argument for soundness is somewhat involved; the running time bound is not difficult to see.

Soundness. Let L be the set of pendant neighbours of u in H that are deleted to obtain the graph H' , and let r be the pendant neighbour of u remaining in H' . Let $H = (V_H, E_H)$, $H' = (V'_H, E'_H)$, so that $V'_H = V_H \setminus L$ and $H' = H[V_H \setminus L]$. We have to show that

There exists a set $S \subseteq V_H, |S| \leq k$ such that $H[V_H \setminus S]$ contains no cycles or T_2 s (i.e. has pathwidth one) if and only if there exists a set $S' \subseteq V'_H, |S'| \leq k$ such that $H'[V'_H \setminus S']$ contains no cycles or T_2 s.

(\implies): If there exists an $S \subseteq V_H, |S| \leq k$ such that $H[V_H \setminus S]$ contains no cycles or T_2 s, then let $S' = S \setminus L$. Clearly $S' = S \setminus L \subseteq V_H \setminus L = V'_H$, and $|S'| = |S| - |S \cap L| \leq |S| \leq k$. Now $H'[V'_H \setminus S'] = H'[(V_H \setminus L) \setminus (S \setminus L)] = H[(V_H \setminus L) \setminus (S \setminus L)] = H[(V_H \setminus S) \setminus L]$, and since $H[V_H \setminus S]$ contains no cycles or T_2 s, neither does its induced subgraph $H[(V_H \setminus S) \setminus L] = H'[V'_H \setminus S']$.

(\impliedby): If there exists a set $S' \subseteq V'_H, |S'| \leq k$ such that $H'[V'_H \setminus S']$ contains no cycles or T_2 s, then let $K = H[V_H \setminus S']$. Now, the vertices in L have degree at most one in K , and so do not belong to any cycle in K . Therefore, if K contains a cycle, then so does $K \setminus L = H[V_H \setminus S' \setminus L] = H[(V_H \setminus L) \setminus S'] = H'[V'_H \setminus S']$, which contradicts the assumption that $H'[V'_H \setminus S']$ contains no cycles or T_2 s. So K does not contain a cycle.

If K does not contain a T_2 either, then setting $S = S'$ completes the argument. So let K contain a T_2 . Then $H'[V'_H \setminus S'] = H[(V_H \setminus L) \setminus S'] = H[V_H \setminus S' \setminus L]$ contains no T_2 , and $K = H[V_H \setminus S']$ contains a T_2 . The vertices in L have degree at most one in K , and from the first part of [Observation 1](#) it follows that these vertices have degree exactly one in K . So $u \in K$, i.e., $u \in V_H \setminus S'$, and by the definition of L , $u \notin L$. Thus $u \in V_H \setminus S' \setminus L$, and by

Lemma 5.4. $r \notin V_H \setminus S' \setminus L$. But by definition $r \notin L$, and so r must be in S' . Now, if there is a T_2 t in K that does not contain u , then t does not contain any vertex from L either, and so t is present in $K \setminus L = H[V_H \setminus S' \setminus L]$, a contradiction. So every T_2 in K contains u .

Set $S := (S' \setminus \{r\}) \cup \{u\}$. Clearly $S \subseteq V_H$, and $|S| = |S'| \leq k$. Since $K = H[V_H \setminus S']$ does not contain a cycle, and since the only neighbour of r in H is u , adding r to K and removing u does not introduce a cycle. Since every T_2 in K contains u , removing u from K also removes all T_2 s from K . Since the only neighbour of r in H is u , adding r to K and removing u does not introduce a T_2 . Thus $H[V_H \setminus ((S' \setminus \{r\}) \cup \{u\})] = H[V_H \setminus S]$ contains no cycles or T_2 s.

Running time. It is not difficult to see that this rule can be applied in $\mathcal{O}(|V_H| + |E_H|) = \mathcal{O}(n + (k+1)(n-1)) = \mathcal{O}(kn)$ time. \square

Rule 1 and **Rule 2** together ensure that every caterpillar in $G[V \setminus S]$ has at least one neighbour in S , and that $|V_1| \leq k$.

Lemma 5.5. Let $(G = (V, E), k)$ be a YES instance of the problem that is reduced with respect to **Rule 1** and **Rule 2**, and let S be a pathwidth-one deletion set of G of size at most k . Let $V_1 = \{v \in (V \setminus S); (N(v) \cap (V \setminus S)) = \emptyset \text{ and } |N(v) \cap S| \leq 1\}$. Then every caterpillar in $G[V \setminus S]$ has at least one neighbour in S , and $|V_1| \leq k$.

Proof. If a caterpillar in $G[V \setminus S]$ has no neighbour in S , then **Rule 1** would apply to G , a contradiction. Thus every caterpillar in $G[V \setminus S]$, and therefore every vertex $v \in V_1$, has at least one neighbour in S . If two vertices in V_1 have the same neighbour in S , then **Rule 2** would apply to G , a contradiction. Thus every vertex in V_1 has a distinct neighbour in S , and so $|V_1| \leq |S| = k$. \square

Rule 3. Let u be a vertex of H with at least two neighbours. If for every two vertices $\{v, w\} \subseteq N(u)$ there exist $k+2$ vertices excluding u that are adjacent to both v and w , then delete u from H . The resulting instance is $(H' = H[V_H \setminus \{u\}], k' = k)$.

We need the following lemma to show that **Rule 3** is sound.

Lemma 5.6. Let G be any graph of pathwidth at most one. If v is a vertex of degree at least 2 in G and H is a graph obtained from G by adding some number $l \geq 1$ of pendant vertices as neighbours to v , then H also has pathwidth at most one.

Proof. Let u_1, u_2 be two neighbours of v in G . Assume to the contrary that H has pathwidth more than one. It is clear that H does not contain a cycle, and so by **Lemma 5.1** H contains a subgraph K isomorphic to T_2 . K contains at least one of the new pendant vertices; say it

contains w . Since w is pendant in H , the degree of w in K is exactly 1, and so w is one of the leaf vertices of K ; its only neighbour in K is v , which in turn is a degree two vertex of K . Further, one of the neighbours of v in G is the central vertex of K ; say u_1 is the central vertex of K . Vertex u_2 is not part of K , or else the edges $(v, u_1), (v, u_2)$ and the path in K from u_1 to u_2 would form a cycle in G . So we can remove w from K and add u_2 and the edge (v, u_2) to the resulting subgraph to obtain a T_2 consisting entirely of vertices in G , a contradiction. \square

We are now ready to show that [Rule 3](#) is sound.

Claim 3. *[Rule 3](#) is sound, and can be applied in $\mathcal{O}(n^3)$ time.*

Proof. We first prove the soundness, and then the bound on the running time.

Soundness. Let X be a set of at most k vertices of H whose removal results in a graph of pathwidth one. Then removing $X' = (X \setminus \{u\})$ from H' results in a graph of pathwidth one, and $|X'| \leq k$.

For the other direction, let X' be a set of at most k vertices of H' such that $H'[V'_H \setminus X'] = H[V'_H \setminus X']$ has pathwidth at most one. It is sufficient to show that removing X' from H results in a graph of pathwidth at most one. This, in turn, is equivalent to showing that adding u (and all the edges from u to $V_H \setminus X'$ in H) to $H[V'_H \setminus X']$ will result in a graph (which is $H[V_H \setminus X']$) with pathwidth at most one.

Now, since $H[V'_H \setminus X']$ has pathwidth at most one, X' has at least one vertex in common with every cycle in $H[V'_H]$. Also, for any two vertices $\{v, w\} \subseteq N(u)$ there are $k+2$ vertex disjoint paths from v to w in $H[V'_H]$, and so either v or w has to be in X' . It follows that $|N(u) \setminus X'| \leq 1$.

If $N(u) \subseteq X'$, then u is isolated in $H[V_H \setminus X']$, and it follows from [Observation 1](#) that $H[V_H \setminus X'] = H[(V'_H \setminus X') \cup \{u\}]$ has pathwidth at most one. So suppose $N(u) \not\subseteq X'$, and let v be the single vertex in $N(u) \setminus X'$. Now in H' , v has at least $k+2$ neighbours excluding u , and so there are at least two such neighbours of v , say y_1, y_2 , that are not in X' . Thus (i) v has degree at least 2 in the graph $H[V'_H \setminus X']$ of pathwidth at most one, and (ii) $H[V_H \setminus X'] = H[(V'_H \setminus X') \cup \{u\}]$ is obtained by adding a pendant vertex adjacent to v to $H[V'_H \setminus X']$, and so by [Lemma 5.6](#), $H[V_H \setminus X']$ has pathwidth at most one.

Running time. The rule can be applied in $\mathcal{O}(n^3)$ time as follows. Construct a new graph $K = (V_K = V_H, E_K)$, where for each pair of vertices $x, y \in V_H$, add edge (x, y) to E_K if in the graph H , $|(N(x) \cap N(y)) \setminus \{x, y\}| \geq k+3$. To see if $u \in V_H$ qualifies for deletion from H as per the rule, check if $N(u)$ induces a clique in K . \square

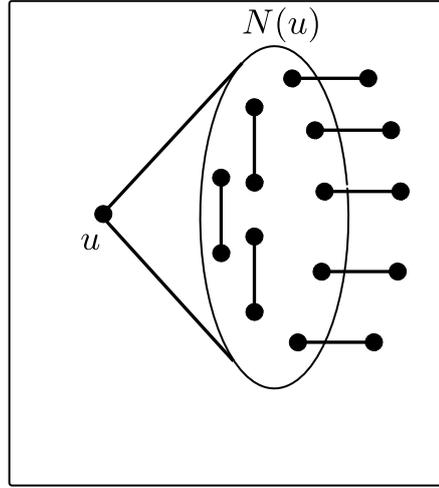


Figure 5.2: **The condition for applying Rule 4.** Each edge shown has at least one end point in $N(u)$, and is not incident on u . The rule applies if there exists a matching consisting of such edges, of size at least $k + 3$. The rule deletes u and decrements k by one.

Rule 3 ensures that $|V_2| \leq \binom{k}{2}(k + 2)$; we get this by setting $A \leftarrow V_2$ and $X \leftarrow S$ in the following Lemma.

Lemma 5.7. *Let (G, k) be a YES instance of the problem that is reduced with respect to Rule 3. For a set $X \subseteq V$, if $A \subseteq V \setminus X$ is such that every $v \in A$ has (i) at least two neighbours in X , and (ii) no neighbours outside X , then $|A| \leq \binom{|X|}{2}(k + 2)$.*

Proof. To prove this bound on $|A|$, we start by associating an integer $x_{ij} = 0$ with each pair of vertices $\{v_i, v_j\} \subseteq X$. We then go through the vertices of A , and for each vertex $u \in A$, we find a pair of vertices $\{v_i, v_j\} \subseteq N(u)$ such that $x_{ij} < (k + 2)$, and increment this x_{ij} by one. We will always be able to do this, or else **Rule 3** would apply to vertex u , a contradiction. At the end of this process, $|A| = \sum_{\{v_i, v_j\} \subseteq X} x_{ij}$. But for each pair of vertices $\{v_i, v_j\} \subseteq X$, $x_{ij} \leq (k + 2)$, and it follows that $|A| \leq \binom{|X|}{2}(k + 2)$. \square

Rule 4. For a vertex u of H , if there is a matching M of size $k + 3$ in H where (i) each edge in M has at least one end vertex in $N(u)$, and, (ii) u is not incident with any edge in M (See **Figure 5.2**), then delete u and decrement k by one. The resulting instance is $(H' = H[V \setminus \{u\}], k' = k - 1)$.

Claim 4. *Rule 4 is sound, and can be applied in $\mathcal{O}(kn^{1.5})$ time.*

Proof. We first prove the soundness, and then the bound on the running time.

Soundness. Let X be a set of at most k vertices of H whose removal results in a graph of pathwidth at most one, and let $X' = (X \setminus \{u\})$. At least three edges of the matching M , say $\mathcal{E} = \{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}\}$, survive in $H[V_H \setminus X]$. Without loss of generality, let $\{x_1, x_2, x_3\} \subseteq N(u)$. If $u \notin X$, then \mathcal{E} and the edges $\{u, x_1\}, \{u, x_2\}, \{u, x_3\}$ together form a T_2 in $H[V_H \setminus X]$, a contradiction. Hence $u \in X$, and so $|X'| = |X| - 1$. Clearly, removing X' from H' results in a graph of pathwidth one, and $|X'| \leq k - 1$.

For the other direction, if X' is a set of at most $k-1$ vertices of H' such that $H'[V'_H \setminus X'] = H[V'_H \setminus X']$ has pathwidth at most one, then clearly $X = X' \cup \{u\}$ is a set of at most k vertices of H such that $H[V_H \setminus X]$ has pathwidth at most one.

Running time. The rule can be applied in $\mathcal{O}(kn^{1.5})$ time as follows. Let $A = N(u)$, $B = N(A) \setminus \{u\}$ in H . Construct a new graph K from $H[A \cup B]$ by deleting all the edges in $H[B]$. By doing two levels of a breadth-first traversal starting from u , this can be done in $\mathcal{O}(|V_H| + |E_H|) = \mathcal{O}(n + (k+1)(n-1)) = \mathcal{O}(kn)$ time. Find a maximum matching \mathcal{M} in H in $\mathcal{O}(\sqrt{|V_K|}|E_K|) = \mathcal{O}(kn^{1.5})$ time [88]. \mathcal{M} is a largest matching of the kind specified in the rule, and so we only have to check if \mathcal{M} contains at least $k+3$ edges. \square

Rule 5. Let x, y be the end vertices of the spine $x, v_1, v_2, v_3 \dots, v_p, y$ of an induced caterpillar C in H such that (1) no $v_i; 1 \leq i \leq p$ is adjacent in H to any vertex outside C , and (2) every pendant vertex of C is a pendant vertex in H . If $p \geq 5$, then contract the edge (v_2, v_3) in H to obtain the graph H' . The resulting instance is $(H', k = k')$.

We need the following observations to show that [Rule 5](#) is sound.

Observation 2.

1. Let G be any graph that contains at least one cycle. Any graph G' obtained from G by contracting an edge of G also contains at least one cycle (possibly containing parallel edges).
2. Let G be a graph that contains a T_2 as a subgraph. If G' is a graph obtained from G by contracting an edge (u, v) where either u or v (or both) is not part of any T_2 in G , then G' also contains a T_2 as a subgraph.

Fact 4. [78] For any fixed non-negative integer p , the class of graphs of pathwidth at most p is closed under the operation of taking minors.

We can now prove that [Rule 5](#) is sound.

Claim 5. [Rule 5](#) is sound, and can be applied in $\mathcal{O}(kn)$ time.

Proof. We first prove the soundness, and then the bound on the running time.

Soundness. Let v_2v_3 be the vertex resulting from the edge contraction. Let X be a set of at most k vertices of H whose removal results in a graph K of pathwidth at most one. If $\{v_2, v_3\} \cap X = \emptyset$, then the graph $K' = H'[V'_H \setminus X]$ is a minor of K : K' can be obtained from K by contracting the edge $\{v_2, v_3\}$. If $\{v_2, v_3\} \cap X \neq \emptyset$, then let $X' = (X \cup \{v_2v_3\}) \setminus \{v_2, v_3\}$. Clearly $|X'| \leq |X|$, and $K' = H'[V'_H \setminus X']$ is a subgraph of K : if $\{v_2, v_3\} \subseteq X$, then K' is isomorphic to K , and if exactly one of v_2, v_3 is in X , then K' can be obtained from K by deleting the other vertex. In both cases, by [Fact 4](#), K' has pathwidth at most one, and so in all cases there is a vertex set of size at most k in H' whose removal gives a graph of pathwidth at most one.

For the other direction, suppose X' is a minimal set of at most k vertices of H' such that $K' = H'[V'_H \setminus X']$ has pathwidth at most one. If $v_2v_3 \notin X'$, then $X' \subseteq V_H$, and K' can be obtained from $K = H[V_H \setminus X']$ by contracting the edge $\{v_2, v_3\}$. By the contrapositive of [Observation 2](#), K contains neither a cycle nor a T_2 . Hence X' is a set of at most k vertices of H such that $H[V_H \setminus X']$ has pathwidth at most one. If $v_2v_3 \in X'$, then it is easy to see that $X = (X' \setminus \{v_2v_3\}) \cup \{v_2\}$ is a set of at most k vertices of H such that $H[V_H \setminus X]$ has pathwidth at most one.

Running time. The rule can be applied in $\mathcal{O}(kn)$ time as follows: first we delete all pendant vertices in the graph. This can be done in $\mathcal{O}(|V_H| + |E_H|)$ time. In the remaining graph, we check if there is a path of length 5 or more consisting of vertices of degree two. This can be done, by doing a BFS, in $\mathcal{O}(|V_H| + |E_H|)$ time. The total running time is thus $\mathcal{O}(|V_H| + |E_H|) = \mathcal{O}(kn)$. \square

These rules together bound $|V_3|$ and $|V_5|$, as we see by the next two lemmas.

Lemma 5.8. *Let $(G = (V, E), k)$ be an instance of the problem that is reduced with respect to [Rule 1](#) to [Rule 5](#), and let $S \subseteq V$ be such that $G[V \setminus S]$ has pathwidth at most one. Let $X \subseteq (V \setminus S)$ be the set of vertices in $(V \setminus S)$ that lie on the spines of nontrivial caterpillars in $G[V \setminus S]$. Then $|X| \leq 17k(k + 2)$.*

Proof. Let C_1, C_2, \dots, C_p be the nontrivial caterpillars in $G[V \setminus S]$, and for $1 \leq i \leq p$, let $P_i = \langle v_1, v_2, \dots, v_{r_i} \rangle$ be a path of the maximum length in C_i . It is sufficient to show that $\sum_{i=1}^p r_i \leq 17k(k + 2)$. Let $\mathcal{C}_s = \{C_i \mid |P_i| \leq 8\}$ (the “small” caterpillars), and let \mathcal{C}_l be the remaining, “large” caterpillars.

Each $C_i \in \mathcal{C}_s$ has at least one neighbour in S , or else [Rule 1](#) would apply. Any one $v \in S$ can have neighbours in at most $k + 2$ different elements of \mathcal{C}_s , or else [Rule 4](#) would apply to v and its neighbourhood. It follows that $|\mathcal{C}_s| \leq k(k + 2)$, and so the total number of vertices that lie on the spines of the elements of \mathcal{C}_s is at most $8k(k + 2)$.

Now we consider the caterpillars in \mathcal{C}_l . Without loss of generality, let $\mathcal{C}_l = \{C_1, C_2, \dots, C_{p'}\}$. For $1 \leq i \leq p'$, let $P'_i = \langle v_3, v_4, \dots, v_{r_i-2} \rangle$. P'_i can be thought of as containing blocks B_1, B_2, \dots , where each block consists of five consecutive vertices in the path. More specifically, $B_1 = \langle v_3, v_4, \dots, v_7 \rangle$, $B_2 = \langle v_8, v_9, \dots, v_{12} \rangle$, and so on, until there are fewer than 5 vertices left. Consider any block B_j on path P'_i . Let C_j be the set of pendant vertices, not belonging to P_i , adjacent to the vertices of B_j in $G[V \setminus S]$, and let $X_j = B_j \cup C_j$. Let x, y be the two vertices belonging to $P_i \setminus B_j$ that are adjacent to the two end vertices of B_j in $G[P_i]$, and let x', y' be the two vertices belonging to $P_i \setminus B_j$ that are adjacent to x, y , respectively. Thus, for example, for B_2 defined as above we have $x = v_7, x' = v_6, y = v_{13}, y' = v_{14}$. Note that x, y, x', y' as defined here are guaranteed to exist for each block B_j . Also note that for any B_j , (X_j, x, y, x', y') as defined here satisfy the requirements of [Rule 5](#) in $G[V \setminus S]$ with X_j as X . So, if none of the vertices of X_j is adjacent to any vertex of S , then (X_j, x, y, x', y') would satisfy these requirements in G as well, in which case [Rule 5](#) would apply to G , a contradiction. It follows that in G , at least one vertex of X_j has an edge to a vertex of S . By the same argument as above, there are at most $k(k+2)$ distinct blocks in $G[V \setminus S]$. It follows that the total number of vertices that lie on the spines of the elements of \mathcal{C}_l is at most $9k(k+2)$.

Putting these together, the bound in the lemma follows. \square

Lemma 5.9. *Let $(G = (V, E), k)$ be a YES instance of the problem that is reduced with respect to [Rule 1](#) to [Rule 5](#), and let $S \subseteq V; |S| \leq k$ be such that $G[V \setminus S]$ has pathwidth at most one. Let $P \subseteq (V \setminus S)$ be the set of pendant vertices in $G[V \setminus S]$ that have at least one neighbour in S . Then $|P| \leq 17(k+2)^2 k(2k-1)$.*

Proof. Let $T \subseteq (V \setminus S)$ be the set of vertices that lie on the spines of caterpillars in $G[V \setminus S]$. By [Lemma 5.8](#), $|T| \leq 17k(k+2)$. Partition T into l parts $T = T_1 \uplus T_2 \uplus \dots \uplus T_l$ where each $T_i; 1 \leq i < l$ contains exactly k vertices, and T_l contains the remaining at most k vertices. Clearly $l \leq 17(k+2)$. For $1 \leq i \leq l$, let $P_i = \cup_{v \in T_i} (N(v) \cap P)$; then $P = \cup_i P_i$. For $1 \leq i \leq l$, setting $X = S \cup T_i, A = P_i$ and applying [Lemma 5.7](#) we get $|P_i| \leq \binom{|S \cup T_i|}{2} (k+2) \leq \binom{2k}{2} (k+2) = k(2k-1)(k+2)$. Hence $|P| \leq l \cdot k(2k-1)(k+2) = 17(k+2)^2 k(2k-1)$. \square

From [Rule 1](#) to [Rule 5](#) it follows that $|V_3| \leq 17k(k+2)$ — [Lemma 5.8](#) — and that $|V_5| \leq 17(k+2)^2 k(2k-1)$ — [Lemma 5.9](#). Each vertex in G can have at most one pendant neighbour, or else [Rule 2](#) would apply. From this we get $|V_4| \leq |V_3| = 17k(k+2)$. Putting all the bounds together, $|V| \leq 34k^4 + 120k^3 + 103k^2 + k$, and so we have:

Rule 6. If none of [Rule 1](#) to [Rule 5](#) can be applied to the instance (H, k) , and $|V_H| > 34k^4 + 120k^3 + 103k^2 + k$, then set the resulting instance to be the trivial No instance (H', k') where H' is a cycle of length 3 and $k' = 0$.

From these claims, we get

Lemma 5.10. *On an input instance $(G = (V, E), k); |V| = n$ of PARAMETERIZED PATHWIDTH-ONE VERTEX DELETION, the kernelization algorithm ([Algorithm 3](#)) runs in $\mathcal{O}(n^4)$ time and outputs a kernel on $\mathcal{O}(k^4)$ vertices.*

Proof. From [Claim 1](#) to [Claim 5](#) it follows that [Rule 1](#) to [Rule 5](#) are sound, and that each can be applied in $\mathcal{O}(n^3)$ time. [Lemma 5.5](#) to [Lemma 5.9](#) show that [Rule 6](#) is sound, and it is easy to see that this rule can be applied in $\mathcal{O}(n)$ time. Each time a rule is applied, the number of vertices in the graph reduces by at least one (contracting an edge also reduces the vertex count by one). Hence the loop in lines 3 to 5 of [Algorithm 3](#) will run at most $|V| + 1 = n + 1$ times. The algorithm produces its output either at a step where [Rule 6](#) applies, or when none of the rules applies and the remaining instance has $\mathcal{O}(k^4)$ vertices. Thus the algorithm runs in $\mathcal{O}(n^4)$ time and outputs a kernel on $\mathcal{O}(k^4)$ vertices. \square

Hence we have

Theorem 5.2. *The PATHWIDTH-ONE VERTEX DELETION problem parameterized by solution size k has a kernel with $\mathcal{O}(k^4)$ vertices.*

5.3 Conclusion

In this chapter we defined the PARAMETERIZED PATHWIDTH-ONE VERTEX DELETION problem as a natural variant of the iconic PARAMETERIZED FEEDBACK VERTEX SET problem, and initiated the study of its algorithmic complexity. We established that the problem is NP-complete, and showed that the problem parameterized by the solution size k is fixed-parameter tractable. We gave an FPT algorithm for the problem that runs in $\mathcal{O}^*(7^k)$ time, and showed that the problem has a polynomial kernel on $\mathcal{O}(k^4)$ vertices. Recently, Cygan et al. have improved these bounds, to $\mathcal{O}^*(4.65^k)$ and $\mathcal{O}(k^2)$, respectively [29].

A more challenging problem is to try to solve the analogous problem for larger values of pathwidth: we know that for any positive integer c , the Pathwidth c Vertex Deletion problem, defined analogously to PARAMETERIZED PATHWIDTH-ONE VERTEX DELETION, is FPT parameterized by the solution size. This follows from the Graph Minor Theorem of Robertson and Seymour because, for each fixed c , the set of YES instances for this problem form a minor-closed class. However, for $c = 2$, the number of graphs in the

obstruction set is already a hundred and ten [76], and so our approach would probably be of limited use for $c \geq 2$. Thus the interesting open problems for $c \geq 2$ are: (i) Can we get an $\mathcal{O}^*(d^k)$ FPT algorithm for the problem for some constant d , and (ii) Does the problem have a polynomial kernel?

Connected Feedback Vertex Set

IN the FEEDBACK VERTEX SET problem the input consists of a graph G and a non-negative integer k , and the question is whether there is a set S of at most k vertices in G — a *feedback vertex set* of G — such that deleting S results in a graph which has no cycles. This is a classical NP-complete problem which has been extensively studied from many different algorithmic points of view. In particular, the parameterized complexity of this problem has been very well studied; when the parameter is k , the problem is fixed-parameter tractable (FPT) and has a polynomial kernel. The current fastest (deterministic) FPT algorithm for this parameterization runs in $\mathcal{O}^*(3.83^k)$ time, and a kernel of size $\mathcal{O}(k^2)$ is known.

Our focus in this chapter is on a *connected* variant of FEEDBACK VERTEX SET, namely, CONNECTED FEEDBACK VERTEX SET. Here, given a graph $G = (V, E)$ and a positive integer k , the objective is to check whether there exists a vertex-subset F of size at most k such that $G[V \setminus F]$ is a forest and $G[F]$ is connected. In contrast with FEEDBACK VERTEX SET, not many algorithmic results are known for the CONNECTED FEEDBACK VERTEX SET problem. In fact, to the best of our knowledge, the only work which addresses this problem is a recent paper by Sitters and Grigoriev [61] where they obtain a polynomial time approximation scheme (PTAS) for CONNECTED FEEDBACK VERTEX SET on *planar* graphs. This lack of results for CONNECTED FEEDBACK VERTEX SET is somewhat surprising, considering that the connected variants of other graph problems of similar vintage — such as CONNECTED DOMINATING SET and CONNECTED VERTEX COVER — are quite well-studied [55, 91]. In the paper on which this chapter is based, we initiate the algorithmic

study of CONNECTED FEEDBACK VERTEX SET from the view-point of parameterized algorithms. We investigate the parameterized complexity of this problem where the parameter is k , the size of the solution; we call this the PARAMETERIZED CONNECTED FEEDBACK VERTEX SET problem.

Our Results

We show that PARAMETERIZED CONNECTED FEEDBACK VERTEX SET can be solved in $\mathcal{O}^*(46.2^k)$ time on general graphs and in $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k} \log k)})$ time on graphs excluding a fixed graph H as a minor. In the latter expression, the hidden constant in the exponent depends only on the forbidden graph H . We also show that the problem does not have a polynomial kernel on general graphs unless the Polynomial Hierarchy collapses to the third level. Recent meta-results due to Fomin et al. [56] imply that when H is an *apex graph* — see Section 2.1 — the problem has a polynomial kernel on H -minor free graphs.

On the way to proving the FPT results for PARAMETERIZED CONNECTED FEEDBACK VERTEX SET, we establish that two variants of the well-studied PARAMETERIZED STEINER TREE problem (see Section 4.2), namely PARAMETERIZED DIRECTED STEINER OUT-TREE and PARAMETERIZED GROUP STEINER TREE, are FPT when parameterized by the number t of terminals (see Section 6.1 for definitions). Following the approach used by Nederlof [94] for the PARAMETERIZED STEINER TREE problem, we show that both these problems can be solved in $\mathcal{O}^*(2^t)$ time and polynomial space. We note that PARAMETERIZED GROUP STEINER TREE is known to be of interest to database theorists, and that Ding et al. [38] derived an algorithm for the problem which runs in $\mathcal{O}(3^t \cdot n + 2^t \cdot (n + m))$ time and using exponential space.

Many of the known FPT algorithms for connectivity problems enumerate *all* minimal solutions in FPT time, and then try to connect each solution using an FPT algorithm for the PARAMETERIZED STEINER TREE problem. For instance, this is the case with the existing FPT algorithms for PARAMETERIZED CONNECTED VERTEX COVER (See, e.g, the algorithm due to Molle et al. [91]) and for PARAMETERIZED CONNECTED DOMINATING SET on graphs of girth at least 5 (See Section 4.2). The crucial observation which such algorithms rely on is that there are only $f(k)$ -many *minimal* solutions of size at most k to enumerate, for some computable function f . Thus, there are at most 2^k minimal vertex covers of size at most k for *any* graph [91] and, after some preprocessing, $\mathcal{O}(k^{3k})$ minimal dominating sets of the relevant kind for the PARAMETERIZED CONNECTED DOMINATING SET instance (Section 4.2).

This approach fails for PARAMETERIZED CONNECTED FEEDBACK VERTEX SET, since the number of minimal feedback vertex sets of size at most k can be as large as $\Omega\left(\left(\frac{n}{k}\right)^k\right)$

— consider a graph which is a collection of k vertex-disjoint cycles, each of length approximately n/k . To circumvent this problem, we make use of “compact representations” of feedback vertex sets. A compact representation is a collection of families of mutually disjoint sets, where each family represents a number of different feedback vertex sets, and every feedback vertex set of size at most k is represented by some family. This notion was defined by Guo et al. [64] who showed that the compact representations of all minimal feedback vertex sets of size at most k of any graph can be enumerated in $\mathcal{O}^*(c^k)$ time, for a constant $c > 160$. We observe that using some results which were obtained after Guo et al.’s work, this enumeration can be done in $\mathcal{O}^*(23.1^k)$ time.

Organization of the rest of the chapter

We derive an FPT algorithm for PARAMETERIZED CONNECTED FEEDBACK VERTEX SET in [Section 6.1](#). In order to do this, we derive FPT algorithms for PARAMETERIZED GROUP STEINER TREE and PARAMETERIZED DIRECTED STEINER OUT-TREE in [Section 6.1.1](#) and a faster enumeration algorithm for compact representations of feedback vertex sets in [Section 6.1.2](#). We describe the — singly exponential — FPT algorithm for PARAMETERIZED CONNECTED FEEDBACK VERTEX SET in [Section 6.1.3](#). In [Section 6.2](#) we derive a faster FPT algorithm for the problem restricted to graphs which exclude some fixed graph H as a minor, which runs in sub-exponential FPT time. The first ingredient of this faster algorithm is an FPT algorithm for the problem where the parameter is the treewidth of the input graph, which we derive in [Section 6.2.1](#). We describe the sub-exponential FPT algorithm for H -minor free graphs in [Section 6.2.2](#). In [Section 6.3](#) we investigate the kernelization complexity of the problem, and show that it is unlikely to have a polynomial kernel on graphs in general. We conclude in [Section 6.4](#).

Notation

For a positive integer n , we use $[n]$ to denote the set $\{1, 2, \dots, n\}$. All the graphs in this chapter are finite. In general, we follow the graph terminology of [Section 2.1](#). We say that a graph G (undirected or directed) *contains* a graph H if H is a subgraph of G . Given a directed graph (digraph) $D = (V, A)$, we let $V(D)$ and $A(D)$ denote the vertex and arc set of D , respectively. A vertex $u \in V(D)$ is an *in-neighbour* (*out-neighbour*) of $v \in V(D)$ if $uv \in A$ ($vu \in A$, respectively). The in- and out-neighbourhood of a vertex v are denoted by $N^-(v)$ and $N^+(v)$, respectively. The *in-degree* $d^-(v)$ (resp. *out-degree* $d^+(v)$) of a vertex v is $|N^-(v)|$ (resp. $|N^+(v)|$). An *orientation* of an undirected graph G is a digraph obtained from G by replacing each edge with an arc between the same pair of vertices as the edge, in either direction. An oriented tree is an orientation of a tree. We say that a

sub-digraph T of D with vertex set $V_T \subseteq V(D)$ is an *out-tree* if T is an oriented tree with only one vertex r of in-degree zero (called the *root*). The vertices of T of out-degree zero are called *leaves* and every other vertex is called an *internal vertex*.

6.1 A Single-Exponential FPT Algorithm for General Graphs

In this section we derive an FPT algorithm for CONNECTED FEEDBACK VERTEX SET on general graphs which runs in $\mathcal{O}^*(46.2^k)$ time. As mentioned above, for this we derive and make use of an FPT algorithm for the — seemingly unrelated — PARAMETERIZED GROUP STEINER TREE problem, which is a variant of the well-studied PARAMETERIZED STEINER TREE problem. To solve the PARAMETERIZED GROUP STEINER TREE problem, we in turn use an FPT algorithm for PARAMETERIZED DIRECTED STEINER OUT-TREE, which is another variant of PARAMETERIZED STEINER TREE. We start by describing the FPT algorithms for these variants of PARAMETERIZED STEINER TREE.

6.1.1 Group Steiner Tree and Directed Steiner Tree

Recall that the PARAMETERIZED STEINER TREE problem is a parameterized version of the classical STEINER TREE problem, where the parameter is the number of *terminals* (See [Section 4.2](#) for a more detailed description):

PARAMETERIZED STEINER TREE

Input: An undirected graph H , a set $T \subseteq V(H)$ of designated “terminal” vertices, and a positive integer c .

Parameter: $|T|$

Question: Does H have a Steiner tree for the terminal set T , with at most c vertices?

In the PARAMETERIZED GROUP STEINER TREE problem, disjoint subsets of vertices take the place of terminals, and it is only required to construct a Steiner tree which contains at least one vertex from each “terminal set”.

PARAMETERIZED GROUP STEINER TREE

Input: An undirected graph $G = (V, E)$; disjoint subsets $S_1, \dots, S_t \subseteq V$; and an integer p .

Parameter: The integer t .

Question: Does G contain a tree on at most p vertices that includes at least one vertex from each S_i ?

To solve PARAMETERIZED GROUP STEINER TREE, we first reduce it to the following directed version of PARAMETERIZED STEINER TREE:

PARAMETERIZED DIRECTED STEINER OUT-TREE

Input: A directed graph $D = (V, A)$; a distinguished vertex $r \in V$; a set of terminals $S \subseteq V$; and an integer p .

Parameter: The integer $t = |S|$.

Question: Does D contain an out-tree on at most p vertices that is rooted at r and that contains all the vertices of S ?

Lemma 6.1. *The PARAMETERIZED GROUP STEINER TREE problem can be reduced in polynomial time to the PARAMETERIZED DIRECTED STEINER OUT-TREE problem, preserving the parameter.*

Proof. Let $(G = (V, E), S_1, \dots, S_t, p)$ be an instance of PARAMETERIZED GROUP STEINER TREE where $n = |V|$. Observe first that if $p \geq n$, then we can solve the problem in polynomial time by checking if there is at least one connected component of G which contains a vertex from each of the t sets S_i . So we assume, without loss of generality, that $p < n$. We construct a digraph from G as follows (See [Figure 6.1](#)):

1. Replace each edge $\{u, v\}$ in G by the two arcs uv, vu .
2. Add a new “root” vertex r and t “anchor” vertices $S = \{s_1, s_2, \dots, s_t\}$.
3. For $1 \leq i \leq t$ and for each vertex $x \in S_i$, add the arc xs_i . That is, add an arc from each vertex in a terminal set to the corresponding anchor vertex.
4. Add a directed path of length (number of edges) $n + 1$ from the root vertex r to each vertex v in V . That is, for each vertex $v \in V$, add n new vertices v_1, v_2, \dots, v_n and the arcs $rv_1, v_1v_2, v_2v_3, \dots, v_{n-1}v_n, v_nv$.

Let D be the resulting digraph. The reduced instance of PARAMETERIZED DIRECTED STEINER OUT-TREE is $(D, r, S, p + n + t + 1)$. Observe that the parameter is unchanged as $|S| = t$, and that the construction can be done in polynomial time.

Suppose G contains a tree T on at most p vertices that includes at least one vertex from each S_i . Let \tilde{T} be the sub-digraph of D which corresponds to T . That is, the vertex set of \tilde{T} is the same as that of T , and for each edge $\{u, v\}$ in T , \tilde{T} contains the two arcs uv, vu . Do a breadth-first search (BFS) on \tilde{T} starting from an arbitrary vertex v in \tilde{T} ; let T' be the BFS tree obtained. Observe that T' is an out-tree rooted at v which contains at least one vertex from each S_i . Add to T' the vertex r and the directed path of length

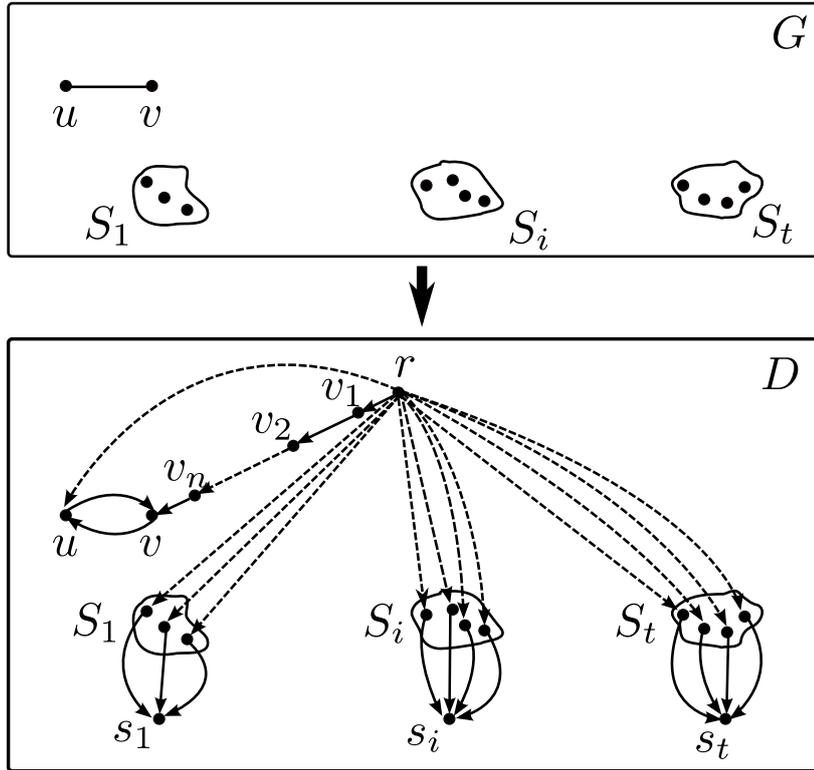


Figure 6.1: **Reduction from PARAMETERIZED GROUP STEINER TREE to PARAMETERIZED DIRECTED STEINER OUT-TREE.** Each edge $\{u, v\}$ in the input graph G is replaced with the two arcs uv, vu in the digraph D . r is a new “root” vertex, from which a directed path of length $n + 1$ is added to each vertex originally present in G ; n is the number of vertices in G . The dotted arcs from r represent such paths. A new “anchor” vertex s_i is added for each “terminal set” S_i , and an arc is added from each vertex in each set S_i to the corresponding anchor vertex s_i .

$n + 1$ from r to v . Now for $1 \leq i \leq t$, choose a vertex u_i in T' which is in S_i , and add the arc u_i, s_i to T' . This yields a out-tree rooted at r on at most $p + n + t + 1$ vertices which contains all the vertices of S , and this out-tree is a sub-digraph of D .

Conversely, let T be a directed out-tree in D on at most $(p + n + t + 1)$ vertices which is rooted at r and contains all the vertices $S = \{s_1, s_2, \dots, s_t\}$. We can assume, without loss of generality, that T is such a *minimal* tree, in that there are no “extra” vertices in T : deleting any vertex from T results in a structure which does not satisfy at least one of the properties of T stated above. Suppose T contains two distinct directed paths of length $n + 1$ from r to two vertices in V — these are vertices which are present in the original graph G — added by Step 4 of the construction. These paths and the set S together contain $2n + t + 3 > p + n + t + 1$ vertices, a contradiction. Thus T contains at most one directed path from r to a vertex in V . Observe that if we remove from D all the vertices in V , then there is no path left from r to any vertex s_i . It follows that T contains at least one

directed path from r to some vertex in V . Thus T contains exactly one directed path, say $r, v_1, v_2, \dots, v_n, v$, from r to some vertex v in V .

Suppose there is a vertex $u_1 \neq v_1$ such that r, u_1 is an arc in T . Since no vertex in V is reachable from u_1 in T , the subtree of T rooted at u_1 can be deleted from T to obtain a sub-digraph of T which satisfies all the properties of T stated above. This contradicts the minimality of T , and so the vertex r has degree exactly 1 in T .

This implies that deleting the vertices r, v_1, v_2, \dots, v_n from T results in a directed out-tree \tilde{T} on $p + t$ vertices which is rooted at v and contains all of S . Since no vertex $s_i \in S$ has an arc going out from it, every s_i is a leaf node in \tilde{T} . Therefore, deleting all the vertices of S from \tilde{T} yields a directed out-tree T' on p vertices which is rooted at v and contains at least one vertex from each set S_i . The corresponding tree in G — which has the same vertex set as T' , and has an edge in the place of each arc in T' — is a tree in G on at most p vertices that includes at least one vertex from each S_i . \square

We now show that **PARAMETERIZED DIRECTED STEINER OUT-TREE** can be solved in $2^t \cdot n^{O(1)}$ time and polynomial space. Our algorithm for **PARAMETERIZED DIRECTED STEINER OUT-TREE** is modelled after Nederlof’s algorithm [94] for **PARAMETERIZED STEINER TREE**.

First, recall the well-known Inclusion-Exclusion formula [73, Section 1.6]: Let U be a finite universe, $A_1, \dots, A_t \subseteq U$, $\overline{A_i} \equiv U \setminus A_i$, and define $\bigcap_{i \in \emptyset} \overline{A_i}$ to be the universe U . Then

$$\left| \bigcap_{i=1}^t A_i \right| = \sum_{X \subseteq [t]} (-1)^{|X|} \left| \bigcap_{i \in X} \overline{A_i} \right| \tag{6.1}$$

Given a problem, suppose we define the sets A_i in such a way that the value of $|\bigcap_{i=1}^t A_i|$ gives us the solution to the problem. One consequence of the above formula is that if, for each $X \subseteq [t]$,

1. we can evaluate $|\bigcap_{i \in X} \overline{A_i}|$ in time polynomial in the input size n , and,
2. the size of the universe U can be expressed using polynomially many bits in the input size n ,

then we can evaluate $|\bigcap_{i=1}^t A_i|$ in time $O(2^t \cdot n^{O(1)})$ using space polynomial in n . Essentially, we evaluate one by one the 2^t expressions $|\bigcap_{i \in X} \overline{A_i}|$ — one for each subset of $[t]$ — and keep a running “signed” sum of their values according to Equation 6.1. From the second condition above, we need only polynomially many bits in n to store these intermediate sums. This is the strategy we follow to solve **PARAMETERIZED DIRECTED STEINER OUT-TREE**.

Let G, H be two digraphs. A *digraph homomorphism** from G to H is a mapping $f : V(G) \rightarrow V(H)$ such that $uv \in A(G) \implies f(u)f(v) \in A(H)$. Given a directed graph $D = (V, A)$, we define a *directed branching walk* B in D to be a pair $(T_B = (V_B, A_B), \phi)$, where T_B is an out-tree and $\phi : V_B \rightarrow V$ is a digraph homomorphism from T_B to D . Note that T_B is an arbitrary out-tree, and is *not* necessarily a subgraph of D . The *length* of the directed branching walk B , denoted $|B|$, is defined to be $|A_B|$. For a node $s \in V$, we say that the directed branching walk B is *from* s if ϕ maps the root of T_B to s . We define $\phi(V_B) = \{\phi(u) \mid u \in V_B\}$ and $\phi(A_B) = \{\phi(a)\phi(b) \mid (a, b) \in A_B\}$.

It turns out that directed branching walks and out-trees stand or fall together.

Claim 1. *Let (D, r, S, p) be an instance of the PARAMETERIZED DIRECTED STEINER OUT-TREE problem. Then there exists an out-tree $T = (V', A')$ in D rooted at r such that $S \subseteq V'$ and $|V'| \leq p$ if and only if there exists a directed branching walk $B = (T_B = (V_B, A_B), \phi)$ in D from r such that $S \subseteq \phi(V_B)$ and $|B| \leq p - 1$.*

Proof. If $T = (V', A')$ is an out-tree in D rooted at r such that $S \subseteq V'$ and $|V'| \leq p$, then let $(V_B, A_B) = T_B = T = (V', A')$, and let ϕ be the identity map on the vertex set $V_B = V'$ of T_B . Then ϕ is a digraph homomorphism from T_B to D which maps the root r of T_B to the vertex r , $S \subseteq \phi(V_B) = V_B = V'$, and $|A_B| = |V_B| - 1 \leq p - 1$. Thus $B = (T_B = (V_B, A_B), \phi)$ is a directed branching walk in D from r such that $S \subseteq \phi(V_B)$ and $|B| \leq p - 1$.

Conversely, let $B = (T_B = (V_B, A_B), \phi)$ be a directed branching walk in D from r such that $S \subseteq \phi(V_B)$ and $|B| \leq p - 1$. Mark the vertex r in D . Starting from the root r_B of T_B , do a breadth-first traversal of T_B , respecting the directions of the arcs. For each new arc uv of T_B encountered in this traversal, if $\phi(v)$ is not already marked, then

- Mark the vertex $\phi(v)$, and
- Mark the arc $\phi(u)\phi(v)$.

Let $T = (V', A')$ be the marked sub-digraph of D obtained by this process. Observe that $V' = \phi(V_B)$, and so $S \subseteq V'$. Since a vertex in D is marked at most once, T has no cycles. Note that each time a new vertex is marked, an arc to that vertex from some previously marked vertex is also marked. It follows that the digraph T is connected, and is an out-tree rooted at r . Since each arc in A_B corresponds to at most one new vertex in T , and every vertex in T except the root vertex r_B corresponds to an arc in A_B , it follows that $|V'| \leq 1 + |A_B| \leq p$. Thus $T = (V', A')$ is an out-tree in D rooted at r such that $S \subseteq V'$ and $|V'| \leq p$. \square

* See the book by Hell and Nešetřil [70] for a detailed treatment of graph homomorphisms.

We take advantage of this relation between directed branching walks and rooted out-trees to solve the PARAMETERIZED DIRECTED STEINER OUT-TREE problem. The idea is to count the directed branching walks from a vertex instead of looking for the existence of an out-tree rooted at that vertex. As we see below, directed branching walks turn out to be easy to count using the Inclusion-Exclusion formula. Note that we are not interested in this count *per se*, but only in whether the count is zero or not. If the count is zero, then — by Claim 1 — there is no out-tree of the required kind in the digraph, and otherwise there is.

Let $(D = (V, A), r, S, p)$ be an instance of the PARAMETERIZED DIRECTED STEINER OUT-TREE problem. We now express, as an Inclusion-Exclusion formula, the number of directed branching walks in D from r such that $S \subseteq \phi(V_B)$ and $|B| \leq p - 1$. Let U be the set of *all* directed branching walks from r of length $p - 1$. For each $v \in S$, let B_v be the set of all elements of U that contain v . Then $\overline{B_v}$ is the set of all elements of U which do not contain v , and $\bigcap_{v \in S} B_v$ is the set of all directed branching walks that contain all the vertices of S . Substituting the sets $B_v; v \in S$ in Equation 6.1 we get

$$\left| \bigcap_{v \in S} B_v \right| = \sum_{X \subseteq S} (-1)^{|X|} \left| \bigcap_{u \in X} \overline{B_u} \right| \quad (6.2)$$

By Claim 1, $|\bigcap_{v \in S} B_v| > 0$ if and only if $(D = (V, A), r, S, p)$ is a YES instance.

For $X \subseteq S$ define $X' = X \cup (V \setminus S)$, and let $b_j^X(r)$ be the number of directed branching walks from r of length j in the graph $D[X']$. Note that $\bigcap_{u \in X} \overline{B_u}$ is the set of all elements of U which do not contain any vertex in X . That is, $\bigcap_{u \in X} \overline{B_u}$ is the set of all directed branching walks from r of length $p - 1$ in the graph $D[(V \setminus X)]$. Further, $b_{p-1}^{S \setminus X}(r)$ is the number of directed branching walks from r of length $p - 1$ in the graph $D[(S \setminus X) \cup (V \setminus S)] = D[V \setminus X]$. Thus $|\bigcap_{u \in X} \overline{B_u}| = b_{p-1}^{S \setminus X}(r)$.

Let Y be an arbitrary subset of S . The number of directed branching walks of length 0 rooted at r is 1: the tree $T_B = (V_B, A_B)$ in this case consists of just a single vertex, and the homomorphism ϕ maps this vertex to r . Thus $b_0^Y(r) = 1$. Any directed branching walk $(T_B = (V_B, A_B), \phi)$ of length $j > 0$ rooted at r maps some vertex in V_B to an out-neighbour s of r in D , and can be split into two parts: a directed branching walk (T_1, ϕ_1) of length $j' \geq 0$ rooted at s , and another directed branching walk (T_2, ϕ_2) of length $j - j' - 1$ rooted at r . The “loss” of 1 corresponds to the arc from r to s present in $\phi(A_B)$. Conversely, every pair of two such directed branching walks corresponds to a directed branching walk (T_B, ϕ) of length j rooted at r : the tree T_B is obtained by adding an arc from the

root of T_2 to the root of T_1 , and the homomorphism ϕ is the disjoint union of the two homomorphisms ϕ_1, ϕ_2 .

It follows that for any $j \geq 0, Y \subseteq S$, the number $b_j^Y(r)$ can be computed in time polynomial in n and j by a simple dynamic programming algorithm based on the following recurrence:

$$b_j^Y(r) = \begin{cases} 1 & \text{if } j = 0; \\ \sum_{s \in N^+(r) \cap Y} \sum_{j_1 + j_2 = j-1} b_{j_1}^Y(s) \cdot b_{j_2}^Y(r) & \text{otherwise.} \end{cases}$$

We now compute $|\bigcap_{v \in S} B_v|$ as per [Equation 6.2](#), by computing successive partial sums in the expression on the right hand side of this equation. Since $|\bigcap_{v \in X} \overline{B}_v| = b_{p-1}^{S \setminus X}(r)$ for each $X \subseteq S$, to compute the value $|\bigcap_{v \in X} \overline{B}_v|; X \subseteq S$ it is sufficient to compute $b_{p-1}^{S \setminus X}(r)$ instead. To compute each partial sum of the right hand side of [Equation 6.2](#), our algorithm computes the next value $b_{p-1}^Y(r)$ needed and adds (or subtracts, as the case may be) it to (or from) the partial sum computed so far. Since there are $2^{|S|} = 2^t$ values to be computed and each value can be computed in polynomial time, this can be done in $2^t \cdot n^{O(1)}$ time. The incremental process described above takes only polynomial space, since only the partial sum is stored between computations. We return YES if the final value is larger than zero, and No otherwise. Thus we have shown that

Lemma 6.2. *The PARAMETERIZED DIRECTED STEINER OUT-TREE problem can be solved in $2^t \cdot n^{O(1)}$ time using polynomial space.*

[Lemma 6.1](#) and [Lemma 6.2](#) together imply:

Lemma 6.3. *The PARAMETERIZED GROUP STEINER TREE problem can be solved in $(2^t \cdot n^{O(1)})$ time using polynomial space.*

6.1.2 Compact Representations of Feedback Vertex Sets

A second ingredient in our FPT algorithm for PARAMETERIZED CONNECTED FEEDBACK VERTEX SET is an algorithm which takes a graph G as input and enumerates “essentially all” minimal feedback vertex sets of G of size at most k in $\mathcal{O}^*(c^k)$ time for a constant c . Recall from the Introduction that one cannot hope, in general, to enumerate all minimal feedback vertex sets of size at most k of a graph in $f(k)$ time for any function f . For example, a graph which is a collection of k vertex-disjoint cycles, each of length approximately n/k , has $\Omega\left(\left(\frac{n}{k}\right)^k\right)$ minimal feedback vertex sets, each obtained by choosing one vertex from each cycle.

A way around this problem was found by Guo et al. [64], who introduced the notion of compact representations of feedback vertex sets. In the context of feedback vertex sets, a *compact representation* of a graph $G = (V, E)$ is a set \mathcal{C} of pairwise disjoint subsets of V such that choosing exactly one vertex from every set in \mathcal{C} results in a minimal feedback set for G . A compact representation is called a *k-compact representation* if the number of sets in the representation is at most k . It follows directly from this definition that for each connected feedback vertex set S of size at most k , there exists a minimal feedback vertex set $S' \subseteq S$ and *some* k -compact representation \mathcal{C} such that we can obtain S' by picking one vertex from each set in \mathcal{C} .

Guo et al. showed that given a graph G and a number k , one can enumerate *all* k -compact representations of G in $\mathcal{O}^*(c^k)$ time for some constant c .

Theorem 6.1. [64, Theorem 7] *Given a graph G and a positive integer k as input, all k -compact representations of G can be enumerated in $\mathcal{O}(c^k \cdot |G|)$ time where c is a constant.*

The constant c implied by the proof of this theorem is more than 160. This can be significantly improved using a result from the recent work of Cao et al. [22] where the authors present the current fastest deterministic FPT algorithm for PARAMETERIZED FEEDBACK VERTEX SET. We now sketch how this is accomplished. Both Guo et al. and Cao et al. use the following problem on the way to obtaining their respective results; our definition matches that of Cao et al:

PARAMETERIZED FOREST BIPARTITION

- Input:* An undirected graph $G = (V, E)$, possibly with multiple edges and loops, and a set $S \subseteq V$ such that $|S| = k + 1$ and $G \setminus S$ is acyclic.
- Parameter:* The integer k .
- Question:* Does G have a feedback vertex set of size at most k which is contained in $V \setminus S$?

Cao et al. describe a set of reduction rules such that if a YES instance of the PARAMETERIZED FOREST BIPARTITION problem is reduced with respect to this set of rules, then the instance has size at most $4k + 1$. In particular, in such a reduced instance we have $|V \setminus S| \leq 3k$. Guo et al. bound the number of k -compact representations by

$$\sum_{i=0}^k \binom{k+1}{i} \left(\sum_{j=0}^{k-i} \binom{X}{j} \binom{Y}{k-i-j} \right),$$

where X is the number of vertices in $V \setminus S$ that have degree at least three in G , and Y is the number of paths in $G[V \setminus S]$ whose endpoints are vertices of degree at least three and internal vertices have degree exactly two in G .

Suppose the instance (G, k) of PARAMETERIZED FOREST BIPARTITION is a YES instance, and has a feedback vertex set of size at most $r \leq k$ which is contained in $V \setminus S$. The reduction rules described by Cao et al. imply that in such a case, the reduced instance has $|X| \leq 3r$. Further, Guo et al. show that $|Y| \leq |X| + 2r$. Using these bounds in the arguments of Guo et al., we get that the number of k -compact representations is at most

$$\begin{aligned} \sum_{i=0}^k \binom{k+1}{i} \left(\sum_{j=0}^{k-i} \binom{3(k+1-i)}{j} \binom{5(k+1-i)}{k-i-j} \right) &\leq \\ \sum_{i=0}^k \binom{k+1}{i} \binom{8(k+1-i)}{k-i} &= \\ \sum_{i=0}^k \binom{9k-8i+9}{k} &\leq \\ k \binom{9k+9}{k} & \end{aligned}$$

This expression is upper bounded by $23.1^{k+1} \cdot k$. Guo et al. show that all these representations can be enumerated with $\mathcal{O}(|E(G)|)$ delay, and so we have

Theorem 6.2. *Given a graph $G = (V, E)$ and an integer k , all the k -compact representations of G can be enumerated in $\mathcal{O}(23.1^k \cdot k|E|)$ time.*

In contrast, the constant at the base of the expression for the running time in Guo et al.'s result — [Theorem 6.1](#) — is more than 160, because the best bound known for $|X|$ at that time was $14k$.

6.1.3 An FPT Algorithm for PARAMETERIZED CONNECTED FEEDBACK VERTEX SET

Recall that the PARAMETERIZED CONNECTED FEEDBACK VERTEX SET problem is defined as follows

PARAMETERIZED CONNECTED FEEDBACK VERTEX SET

Input: An undirected graph $G = (V, E)$, and a positive integer k .

Parameter: The integer k .

Question: Does there exist a set $S \subseteq V$ of at most k vertices of G such that (i) the graph $G[V \setminus S]$ has no cycles, and (ii) the graph $G[S]$ is connected?

Given the two FPT algorithms of the previous subsections — one for PARAMETERIZED GROUP STEINER TREE and the other for enumerating all k -compact representations — it is straightforward to derive an FPT algorithm for PARAMETERIZED CONNECTED FEEDBACK VERTEX SET. This algorithm enumerates all k -compact representations of the input graph. For each such representation $\mathcal{C} = \{S_1, S_2, \dots, S_r\}; 1 \leq r \leq k$, the algorithm solves the PARAMETERIZED GROUP STEINER TREE problem on the input $(G; S_1, S_2, \dots, S_r; k)$. If the answer to this latter problem is No for every k -compact representation, then the algorithm returns No : the graph G has no connected feedback vertex set of size at most k . Otherwise, it returns the first (group) Steiner tree found by the PARAMETERIZED GROUP STEINER TREE algorithm. The correctness of this procedure is evident from the discussion at the beginning of this subsection, and so from [Lemma 6.3](#) and [Theorem 6.2](#) we get

Theorem 6.3. *The PARAMETERIZED CONNECTED FEEDBACK VERTEX SET problem is fixed-parameter tractable. Given a graph $G = (V, E)$ and an integer k , one can decide whether G has a connected feedback set of size at most k in $46.2^k \cdot n^{O(1)}$ time.*

6.2 A Faster FPT Algorithm for H -minor free Graphs

In this section we show that for any fixed graph H , the PARAMETERIZED CONNECTED FEEDBACK VERTEX SET problem can be solved in $2^{O(\sqrt{k} \log k)} n^{O(1)}$ time on the class of H -minor-free graphs. This running time is achieved by doing dynamic programming (DP) on a tree decomposition of the input graph of suitably bounded tree width. See [Chapter 2](#) for the definitions of the various terms used in this section. We briefly recall the notion of a *nice* tree decomposition which is crucial in describing the DP algorithm.

A tree decomposition $(T = (V_T, E_T), \mathcal{X} = \{X_t\}_{t \in V_T})$ of a graph $G = (V, E)$ is called a *nice tree decomposition* [[14](#)] if the following conditions are satisfied:

- Every node of the tree T has at most two children. A node that has no children is called a *leaf* node. The non-leaf nodes are of three kinds:
 - If a node t has two children t_1 and t_2 , then $X_t = X_{t_1} = X_{t_2}$, and t is called a *join* node.
 - if a node t has one child t_1 , then either $|X_t| = |X_{t_1}| + 1$ and $X_{t_1} \subset X_t$ (t is called an *introduce* node), or $|X_t| = |X_{t_1}| - 1$ and $X_t \subset X_{t_1}$ (t is called a *forget* node).

It is possible to transform a given tree decomposition into a nice tree decomposition of the same width in time $\mathcal{O}(|V| + |E|)$ [14].

This section is divided into two subsections. In the first subsection we show that **PARAMETERIZED CONNECTED FEEDBACK VERTEX SET** can be solved in time $\mathcal{O}(w^{\mathcal{O}(w)} n^{\mathcal{O}(1)})$ on graphs with treewidth bounded by w . In the second subsection we bound the treewidth of the input graph by $\mathcal{O}(\sqrt{k})$ using well-known “grid theorems”. Together, these yield an algorithm with a running time of the stated kind.

6.2.1 CONNECTED FEEDBACK VERTEX SET Parameterized by Treewidth

In this section we show that the minimization version of the **CONNECTED FEEDBACK VERTEX SET** problem is FPT when the treewidth of the input graph is the parameter. That is, we show that the following problem is FPT:

- Input:* An undirected graph $G = (V, E)$; an integer k ; and a nice tree decomposition of G of width w .
- Parameter:* The treewidth w of the graph G .
- Question:* Find a set $S \subseteq V$ such that $G \setminus S$ is acyclic, $G[S]$ is connected, and for any connected feedback vertex set R of G , $|S| \leq |R|$.

For this problem, we design a dynamic programming algorithm on the nice tree decomposition which runs in time $w^{\mathcal{O}(w)} \cdot n^{\mathcal{O}(1)}$. See, for example, Hannes Moser’s Master’s thesis [92] for a detailed exposition of this algorithmic paradigm; in particular, our algorithm for **CONNECTED FEEDBACK VERTEX SET** is similar in spirit to the algorithm given by Moser for the **CONNECTED VERTEX COVER** problem. We start by giving an intuitive overview of the algorithm.

The general strategy used to obtain FPT — with the parameter being the treewidth — algorithms by dynamic programming on nice tree decompositions is to design a dynamic programming table (DP table) for each node of the nice tree decomposition such that:

1. The DP table for a leaf node can be computed in time which is FPT in the treewidth — that is, of the form $f(w) \cdot n^{\mathcal{O}(1)}$ where w is the treewidth, n the number of vertices in the graph, and f a computable function;
2. Starting at the leaves and going up to the root, given the DP table(s) of the child(ren) node(s) of a given node, the DP table for this node can be computed in time FPT in the treewidth; and,

3. The desired answer can be computed in FPT time (in many cases, read off directly) from the DP table for the root node.

One fruitful approach towards designing the DP table for solving “subset problems” (where the task is to find a subset of vertices which has some specified property) is to look at how a solution intersects the subgraph induced by all the vertices in a subtree rooted at an arbitrary node in the tree decomposition. Observe that any solution must intersect such a subgraph in *some* manner; the DP table must account for all possible ways in which this can happen, and must be designed in such a way that it is easy (in the sense listed above) to update.

For CONNECTED FEEDBACK VERTEX SET, it turns out that a correct way to design the DP table is to try to capture the *connectivity* properties of a partial solution when restricted to the subgraphs induced by subtrees of the tree decomposition. To be more precise: Let F be some fixed connected feedback vertex set of the input graph $G = (V, E)$ of size at most k , and let G_i be the subgraph induced by the vertices in the subtree rooted at an arbitrary node i in a nice tree decomposition of G . $G[F]$ restricted to the set X_i is, in general, a disconnected graph. So is the forest $G[V \setminus F]$ restricted to the set X_i . Our dynamic programming algorithm tries to guess the components of each of the two subgraphs $G[F \cap X_i]$ and $G[(V \setminus F) \cap X_i]$; it turns out that with this information in hand, it is not difficult to compute the DP tables of the nodes at the next higher level in the tree decomposition, be it an introduce, forget, or join node; the details follow.

Let $(T = (I, F), \{X_i | i \in I\})$ be a nice tree decomposition of the input graph G of width w and rooted at $r \in I$. We let T_i denote the subtree of T rooted at $i \in I$, and use $G_i = (V_i, E_i)$ to denote the subgraph of G induced on all the vertices of G in the subtree T_i , that is, $G_i = G[\bigcup_{j \in V(T_i)} X_j]$.

For each node $i \in I$ we compute a table A_i , the rows of which are 4-tuples $[S, P, Y, val]$. Table A_i contains one row for each combination of the first three components which denote the following:

- S is a subset of X_i .
- P is a partition of S into at most $|S|$ labelled pieces.
- Y is a partition of $X_i \setminus S$ into at most $|X_i \setminus S|$ labelled pieces.

We use $P(v)$ (resp. $Y(v)$) to denote the piece of the partition P (resp. Y) that contains the vertex v . We let $|P|$ (resp. $|Y|$) denote the number of pieces in the partition P (resp. Y). The last component val , also denoted as $A_i[S, P, Y]$, is the size of a smallest feedback vertex set $F_i \subseteq V(G_i)$ of G_i which satisfies the following properties:

- If $S = \emptyset$, then F_i is *connected* in G_i .
- If $S \neq \emptyset$, then
 - $F_i \cap X_i = S$.
 - All vertices of S that are in any one piece of P are in a single connected component of $G_i[F_i]$. Moreover $G_i[F_i]$ has exactly $|P|$ connected components.
 - All vertices of $X_i \setminus S$ that are in the same piece of Y are in a single connected component (a tree) of $G_i[V_i \setminus F_i]$. Moreover $G_i[V_i \setminus F_i]$ has at least $|Y|$ connected components.

We say that a feedback vertex set F_i which satisfies the above conditions is *feasible* for the triple (S, P, Y) .

If there is no such set F_i , then the last component of the row is set to ∞ .

We fix an arbitrary ordering of the vertices of X_i , and compute the table A_i for each node $i \in I$ of the tree decomposition. Since there are at most $w + 1$ vertices in each bag X_i , there are no more than

$$\sum_{i=0}^{w+1} \binom{w+1}{i} i^i \cdot (w+1-i)^{w+1-i} \leq (2w+2)^{2w+2} \quad (6.3)$$

rows in any table A_i . We compute the tables A_i starting from the leaf nodes of the tree decomposition and going up to the root.

Leaf Nodes. Let i be a leaf node of the tree decomposition. We compute the table A_i as follows. For each triple (S, P, Y) where S is a subset of X_i , P a partition of S , and Y a partition of $X_i \setminus S$:

- Set $A_i[S, P, Y] = \infty$ if at least one of the following holds:
 - $G_i \setminus S$ contains a cycle (i.e., S is *not* a feedback vertex set of G_i).
 - At least one piece of P is *not* connected in $G_i[S]$ or if $G_i[S]$ has less than $|S|$ connected components.
 - At least one piece of Y is *not* connected in $G_i[V_i \setminus S]$ or if $G_i[V_i \setminus S]$ has less than $|Y|$ connected components.
- In all other cases, set $A_i[S, P, Y] = |S|$.

It is easy to see that this computation correctly determines the last component of each row of A_i for a leaf node i of the tree decomposition.

Introduce Nodes. Let i be an introduce node and j its unique child. Let $x \in X_i \setminus X_j$ be the introduced vertex. For each triple (S, P, Y) , we compute the entry $A_i[S, P, Y]$ as follows.

Case 1. $x \in S$. If $N(x) \cap S \not\subseteq P(x)$, then set $A_i[S, P, Y] = \infty$. Otherwise, if $S = \{x\}$ and $0 < A_j[\emptyset, P, Y] < \infty$, then set $A_i[S, P, Y] = \infty$. Otherwise,

- Subcase 1. $P(x) = \{x\}$. Set $A_i[S, P, Y] = A_j[S \setminus \{x\}, P \setminus P(x), Y] + 1$.
- Subcase 2: $|P(x)| \geq 2$ and $N(x) \cap P(x) = \emptyset$. Set $A_i[S, P, Y] = \infty$, as no extension of S to an fvs for G_i can make $P(x)$ connected.
- Subcase 3: $|P(x)| \geq 2$ and $N(x) \cap P(x) \neq \emptyset$. Let \mathcal{A} be the set of all rows $[S', P', Y]$ of the table A_j that satisfy the following conditions:
 - $S' = S \setminus \{x\}$.
 - $P' = (P \setminus P(x)) \cup Q$, where Q is a partition of $P(x) \setminus \{x\}$ such that each piece of Q contains an element of $N(x) \cap P(x)$.

Set $A_i[S, P, Y] = \min_{[S', P', Y] \in \mathcal{A}} \{A_j[S', P', Y]\} + 1$.

To illustrate how the correctness of the different cases can be established, we now describe why the computation of $A_i[S, P, Y]$ in Subcase 1 is correct. Let F_i be a feedback vertex set of G_i . If all vertices of S that are in any one piece of P are in a single connected component of $G_i[F_i]$, and $N(x) \cap S \not\subseteq P(x)$, then the number of connected components of $G_i[F_i]$ is strictly less than $|P|$. This justifies setting $A_i[S, P, Y]$ to ∞ in this case. If $S = \{x\}$ and F_i is feasible for (S, P, Y) , then F_i can have no vertex in G_j . If $0 < A_j[\emptyset, P, Y] < \infty$, then there are cycles in G_j , and this justifies setting $A_i[S, P, Y]$ to ∞ in this case.

Now we consider Subcase 1. In this case the graph G_i can be obtained from the graph G_j by adding the vertex x and zero or more edges from x to vertices in $X_i \setminus S = X_j \setminus S$. Let $F'_j = F_j \setminus \{x\}$, $S' = S \setminus \{x\}$. Let P' be the natural restriction of P to S' . It is not difficult to see that (1) F'_j is a feedback vertex set of G_j , and, (2) if F_i is feasible for (S, P, Y) , then F'_j is feasible for (S', P', Y) . Conversely, if F_j is a nonempty feedback vertex set of G_j and F_j is feasible for (S', P', Y) , then $F'_i = F_j \cup \{x\}$ is a feedback vertex set of G_i which is feasible for (S, P, Y) . Since $|S| = |S'| + 1$, this justifies Subcase 1.

The correctness of the remaining cases can be argued in a similar fashion.

Case 2. $x \notin S$. If $N(x) \cap (X_i \setminus S) \not\subseteq Y(x)$, then set $A_i[S, P, Y] = \infty$. Otherwise,

- Subcase 1: $Y(x) = \{x\}$. Set $A_i[S, P, Y] = A_j[S, P, Y \setminus Y(x)]$.
- Subcase 2: $|Y(x)| \geq 2$ and $N(x) \cap Y(x) = \emptyset$. Set $A_i[S, P, Y] = \infty$, as no extension of S to an fvs F_i for G_i can make $Y(x)$ a connected component in $G_i[V_i \setminus F_i]$.
- Subcase 3: $|Y(x)| \geq 2$ and $N(x) \cap Y(x) \neq \emptyset$. Let \mathcal{A} be the set of all rows $[S, P, Y']$ of the table A_j where $Y' = (Y \setminus Y(x)) \cup Q$, and Q is a partition of $Y(x) \setminus \{x\}$ such that each piece of Q contains *exactly* one element of $N(x) \cap Y(x)$. Set $A_i[S, P, Y] = \min_{[S, P, Y'] \in \mathcal{A}} \{A_j[S, P, Y']\}$.

Forget Nodes. Let i be a forget node and j its unique child node. Let $x \in X_j \setminus X_i$ be the forgotten vertex. For each triple (S, P, Y) in the table A_i , let \mathcal{A} be the set of all rows $[S', P', Y]$ of the table A_j that satisfy the following conditions:

- $S' = S \cup \{x\}$, and
- $P'(x) = P(y) \cup \{x\}$ for some $y \in S$.

Let \mathcal{B} be the set of all rows $[S, P, Y']$ of the table A_j such that $Y'(x) = Y(z) \cup \{x\}$ for some $z \in S$. Set

$$A_i[S, P, Y] = \min \left\{ \min_{[S', P', Y] \in \mathcal{A}} A_j[S', P', Y], \min_{[S, P, Y'] \in \mathcal{B}} A_j[S, P, Y'] \right\}.$$

Join Nodes. Let i be a join node and j and l its children. For each triple (S, P, Y) we compute $A_i[S, P, Y]$ as follows.

- Case 1. $S = \emptyset$. If both $A_j[\emptyset, P, Y]$ and $A_l[\emptyset, P, Y]$ are positive finite, then set $A_i[\emptyset, P, Y] = \infty$. Else set $A_i[\emptyset, P, Y] = \max\{A_j[\emptyset, P, Y], A_l[\emptyset, P, Y]\}$.
- Case 2. $S \neq \emptyset$. Let \mathcal{A} denote the set of all pairs of triples $\langle (S, P_1, Y_1), (S, P_2, Y_2) \rangle$, where $(S, P_1, Y_1) \in A_j$ and $(S, P_2, Y_2) \in A_l$ with the following property: Starting with the partitions $Q_p = P_1$ and $Q_y = Y_1$ and repeatedly applying the following set of operations, we reach stable partitions that are identical to P and Y . The first operation that we apply is:

If there exist vertices $u, v \in S$ such that they are in different pieces of Q_p but are in the same piece of P_2 , delete $Q_p(u)$ and $Q_p(v)$ from Q_p and add $Q_p(u) \cup Q_p(v)$.

To describe the second set of operations, we need some notation. Let $Z = X_i \setminus S$ and let the connected components of $G_i[Z]$ be C_1, \dots, C_q . First contract each connected component C_i to a vertex c_i , the *representative* of that component,

and let $\mathcal{C} = \{c_1, \dots, c_q\}$. Note that for each $1 \leq i \leq q$, the component C_i is not split across pieces in either Y_1 or Y_2 . Denote by Y'_1 and Y'_2 the partitions obtained from Y_1 and Y_2 , respectively, by replacing each connected component C_i by its representative vertex c_i . Let $Q_y = Y'_1$. Repeat until no longer possible:

If there exist $c_a, c_b \in \mathcal{C}$ that are in different pieces of Q_y but in the same piece of Y_2 then delete $Q_y(c_a), Q_y(c_b)$ from Q_y and add $Q_y(c_a) \cup Q_y(c_b)$ provided the following condition holds: for all $c_e \in \mathcal{C} \setminus \{c_a, c_b\}$ either $Y_2(c_e) \cap Q_y(c_a) = \emptyset$ or $Y_2(c_e) \cap Q_y(c_b) = \emptyset$.

If this latter condition does not hold, move on to the next pair of triples. Finally expand each c_i to the connected component it represents.

Set

$$A_i[S, P, Y] = \min_{\langle (S, P_1, Y_1), (S, P_2, Y_2) \rangle \in \mathcal{A}} \{A_j[S, P_1, Y_1] + A_l[S, P_2, Y_2] - |S|\}.$$

The stated conditions ensure that $u, v \in S$ are in the same piece of P if and only if for each $\langle (S, P_1, Y_1), (S, P_2, Y_2) \rangle \in \mathcal{A}$, they are in the same piece of P_1 or of P_2 (or both). Similarly, the stated conditions ensure that merging solutions at join nodes do not create new cycles. Given this, it is easy to verify that the above computation correctly determines $A_i[S, P, Y]$.

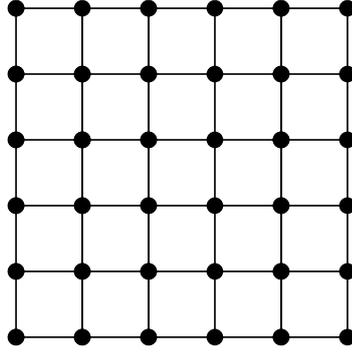
Root Node. We compute the size of a smallest connected feedback vertex set of G from the table A_r for the root node r as follows. Find the minimum of $A_r[S, P, Y]$ over all triples (S, P, Y) , where $S \subseteq X_r$, P a partition of S such that P consists of a single (possibly empty) piece and Y is a partition of $X_r \setminus S$. This minimum is the size of a smallest connected feedback vertex set of G .

This concludes the description of the dynamic programming algorithm for CONNECTED FEEDBACK VERTEX SET when the treewidth of the input graph is bounded by w . From the above description and the size of tables being bounded by $(2w + 2)^{2w+2}$ (See Equation 6.3), we obtain

Theorem 6.4. *Given a graph $G = (V, E)$ and a tree-decomposition of G of width w , one can compute the size of a smallest connected feedback vertex set of G (if it exists) in $(2w + 2)^{2w+2} \cdot n^{O(1)}$ time.*

6.2.2 A Sub-Exponential FPT Algorithm for H -Minor Free Graphs

As we describe in this subsection, Theorem 6.4 implies that for any fixed graph H , the PARAMETERIZED CONNECTED FEEDBACK VERTEX SET problem can be solved in sub-

Figure 6.2: A 6×6 grid graph.

exponential FPT time. We start by bounding the treewidth of the graph in a YES instance by $\mathcal{O}(\sqrt{k})$. We use $\text{tw}(G)$ to denote the treewidth of a graph G .

Lemma 6.4. *If (G, k) is a yes-instance of CONNECTED FEEDBACK VERTEX SET where G excludes a fixed graph H as a minor, then $\text{tw}(G) \leq c_H \sqrt{k}$, where c_H is a constant that depends only on the graph H .*

Proof. An $\ell \times \ell$ grid is the general case of the particular instance shown in Figure 6.2. Demaine and Hajiaghayi [35] have shown that for any fixed graph H , every H -minor-free graph G that does not contain a $(w \times w)$ -grid as a minor has treewidth at most $c'_H w$, where c'_H is a constant that depends only on the graph H . Any feedback vertex set of a $(w \times w)$ -grid contains at least $\frac{(w-1)^2+1}{3}$ vertices [84]. Therefore if the H -minor-free graph G has a connected feedback vertex set of size at most k , it cannot have a $(w \times w)$ -grid minor where $w \geq \sqrt{3k-1}+2$. Hence $\text{tw}(G) \leq c'_H(\sqrt{3k-1}+2) \leq c_H \sqrt{k}$, where $c_H = c'_H(2+\sqrt{2})$. \square

Demaine, Hajiaghayi, and Kawarabayashi [36] and Feige, Hajiaghayi and Lee [47] have shown that a constant-factor approximation to a tree decomposition of the minimum width can be computed in polynomial time for H -minor free graphs.

Theorem 6.5. [36, 47] *For any fixed graph H there exists a constant a_H which depends only on H , and a polynomial-time algorithm which computes a tree decomposition of width at most $a_H \text{tw}(G)$ for any graph G which excludes H as a minor.*

This approximation algorithm, together with the FPT algorithm of Theorem 6.4 and the bound on the treewidth given by Lemma 6.4, yields a sub-exponential FPT algorithm for PARAMETERIZED CONNECTED FEEDBACK VERTEX SET on H -minor free graphs.

Theorem 6.6. *For any fixed graph H , the PARAMETERIZED CONNECTED FEEDBACK VERTEX SET problem can be solved in $2^{\mathcal{O}(\sqrt{k} \log k)} n^{\mathcal{O}(1)}$ time on graphs which exclude H as a minor.*

Proof. Let a_H, c_H be the constants guaranteed to exist for H as per [Theorem 6.5](#) and [Lemma 6.4](#), respectively. Given an instance (G, k) of PARAMETERIZED CONNECTED FEEDBACK VERTEX SET where the graph G does not contain H as a minor, we first find an a_H -approximate minimum tree-decomposition \mathcal{T} of G using the polynomial-time constant-factor approximation algorithm of [Theorem 6.5](#). If this tree decomposition has width more than $a_H c_H \sqrt{k}$, then by [Theorem 6.5](#) $\text{tw}(G) > c_H \sqrt{k}$, and so by [Lemma 6.4](#) G has no connected feedback vertex set of size at most k . In this case the algorithm returns No as the answer.

In the other case, \mathcal{T} has width at most $a_H c_H \sqrt{k}$. Now the algorithm invokes the dynamic programming algorithm of [Theorem 6.4](#) on the tree decomposition \mathcal{T} to find the size s of a smallest connected feedback vertex set of G in $(2a_H c_H \sqrt{k} + 2)^{2a_H c_H \sqrt{k} + 2} \cdot n^{\mathcal{O}(1)}$ time. If $s \leq k$ then the algorithm returns YES. If $s > k$, or if G has no connected feedback vertex set, then the algorithm returns No. In the worst case this algorithm takes $(2a_H c_H \sqrt{k} + 2)^{2a_H c_H \sqrt{k} + 2} \cdot n^{\mathcal{O}(1)} < (c\sqrt{k})^{c\sqrt{k}} \cdot n^{\mathcal{O}(1)} = 2^{d\sqrt{k}\log k} \cdot n^{\mathcal{O}(1)}$ time where $c = 2a_H c_H + 1, d = \frac{c \log c}{2}$. \square

6.3 Kernelization Complexity

Although PARAMETERIZED CONNECTED FEEDBACK VERTEX SET is fixed-parameter tractable, it is unlikely to admit a polynomial kernel, as we show below. Contrast this with the closely related PARAMETERIZED FEEDBACK VERTEX SET problem which admits a quadratic kernel [107]. To prove this lower bound, we use notions and results from the recently developed theory of kernel lower bounds; these are described in detail in [Section 2.2.1](#).

Theorem 6.7. *The PARAMETERIZED CONNECTED FEEDBACK VERTEX SET problem does not admit a polynomial kernel unless the Polynomial Hierarchy (PH) collapses to the third level.*

Proof. We exhibit a polynomial parameter transformation — PPT — from PARAMETERIZED CONNECTED VERTEX COVER (defined below) to PARAMETERIZED CONNECTED FEEDBACK VERTEX SET.

PARAMETERIZED CONNECTED VERTEX COVER

Input: An undirected graph $G = (V, E)$, and a positive integer k .

Parameter: The integer k .

Question: Does there exist a set $S \subseteq V$ of at most k vertices of G such that (i) the graph $G[V \setminus S]$ has no edges, and (ii) the graph $G[S]$ is connected?

This problem does not admit a polynomial kernel unless PH collapses to the third level [39]. The derived classical problem corresponding to PARAMETERIZED CONNECTED VERTEX COVER, namely CONNECTED VERTEX COVER, is NP-complete even in planar graphs of maximum degree four [58]. The derived classical problem corresponding to PARAMETERIZED CONNECTED FEEDBACK VERTEX SET, namely CONNECTED FEEDBACK VERTEX SET, is clearly in NP. It follows from Theorem 2.3 that PARAMETERIZED CONNECTED FEEDBACK VERTEX SET does not admit a polynomial kernel unless PH collapses to the third level. We now describe the transformation.

Given an instance $(G = (V, E), k)$ of the PARAMETERIZED CONNECTED VERTEX COVER problem, construct a new graph G' from G as follows: For each edge $\{u, v\} \in E$, add a new vertex uv and the edges $\{u, uv\}, \{v, uv\}$. This completes the construction of G' ; G', k is the instance of PARAMETERIZED CONNECTED FEEDBACK VERTEX SET.

Let S be a connected vertex cover of size at most k in G . Since S is a vertex cover of G , no edge originally present in G is present in $G' \setminus S$. Thus all edges in any cycle in $G' \setminus S$ are “new” edges — those which are added by the construction. Each such new edge is incident on one “new” vertex, and so any cycle in $G' \setminus S$ contains at least one new vertex.

For any new vertex uv in G' , at least one of its two neighbours u, v is present in S , or else the edge $\{u, v\}$ is not covered by S . Thus any new vertex uv has degree at most one in $G' \setminus S$, and so cannot be part of any cycle in $G' \setminus S$.

Putting these together, it follows that $G' \setminus S$ contains no cycles. Thus S is a feedback vertex set of G' . Since $E(G') \supseteq E(G)$ and $G[S]$ is connected, so is $G'[S]$. Thus S is a connected feedback vertex set of G' , of size at most k .

Conversely, suppose S is a connected feedback vertex set of G' , of size at most k . Suppose S contains uv , a newly introduced vertex of degree two. If $S = \{uv\}$, then $S' = \{v\}$ is also a connected feedback vertex set of G' of the same size as S , since the vertex v is present in every cycle which passes through uv . If $S \neq \{uv\}$, then since $G'[S]$ is a connected graph, S must contain at least one of the two neighbours u, v of uv . If S contains both u and v , then $S' = S \setminus \{uv\}$ is a smaller connected feedback vertex set of G' , because (i) any cycle that passes through uv necessarily passes through both u and v , and (ii) the edge $\{u, v\}$ is present in G' . If S contains exactly one of u or v (say u), then for the same reasons as above, $S' = (S \setminus \{uv\}) \cup \{v\}$ is a connected feedback vertex set of G' .

We may therefore assume, without loss of generality, that the set S contains no newly introduced vertices; that is, $S \subseteq V$. Since $G'[V] = G$, S is connected in G' , and $S \subseteq V$, it follows that $G[S]$ is connected. If $G[V \setminus S]$ contains an edge (u, v) , then by construction, $G' \setminus S$ contains the triangle (u, v, uv) , which contradicts the assumption that S is a feedback vertex set of G' . Hence S is a vertex cover of G . Thus S is a connected vertex cover of G , of size at most k . \square

Interestingly, recent meta-theorems on kernelization due to Fomin et al. [56] imply that PARAMETERIZED CONNECTED FEEDBACK VERTEX SET has polynomial kernels on any graph class which excludes a fixed apex graph H as a minor.

6.4 Conclusion

In this chapter we described our study of the parameterized complexity of the CONNECTED FEEDBACK VERTEX SET problem, which is a connected variant of the well-studied FEEDBACK VERTEX SET problem. We showed that when parameterized by the size k of the solution, the problem is fixed-parameter tractable (FPT) and can be solved in $\mathcal{O}^*(46.2^k)$ time. On the way to obtaining this result, we showed that two parameterized variants of the classical STEINER TREE problem, namely PARAMETERIZED DIRECTED STEINER OUT-TREE and PARAMETERIZED GROUP STEINER TREE, are FPT when the parameter is the size t of the number of terminals (respectively, terminal sets) and can be solved in $\mathcal{O}^*(2^t)$ time. We feel that these results could be of use in showing that other connectivity problems are FPT. Very recently, Cygan et al. [30] showed that PARAMETERIZED CONNECTED FEEDBACK VERTEX SET can be solved in randomized $\mathcal{O}^*(3^k)$ time.

We then investigated the parameterized variant of CONNECTED FEEDBACK VERTEX SET where the parameter is the *treewidth* w of the input graph. We showed that this problem is FPT and can be solved in $\mathcal{O}^*((2w+2)^{2w+2})$ time. Using this fact and an $\mathcal{O}(\sqrt{k})$ bound on the treewidth of a YES instance which follows from an excluded-grid property, we showed that the PARAMETERIZED CONNECTED FEEDBACK VERTEX SET problem can be solved in $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k} \log k)})$ time on graphs which exclude some fixed graph H as a minor.

Finally, we examined the kernelization complexity of the PARAMETERIZED CONNECTED FEEDBACK VERTEX SET problem, and showed that the problem is unlikely to have a polynomial kernel on graphs in general, in stark contrast to the closely related PARAMETERIZED FEEDBACK VERTEX SET problem which has a quadratic kernel on general graphs. We note that some recent meta-theorems on kernelization imply that PARAMETERIZED CONNECTED FEEDBACK VERTEX SET has polynomial kernel on any graph class which excludes a fixed apex graph H as a minor.

There are not many results known for the CONNECTED FEEDBACK VERTEX SET problem, even from the point of view of classical complexity. The only classical results of which we are aware are for the problem restricted to planar graphs. Bodlaender et al. [16] derive many positive results for an edge-weighted planar variant of the problem, and Grigoriev and Sitters [61] derive a constant factor approximation algorithm and a polynomial time approximation scheme for the unweighted planar variant where the minimum degree is at least 3.

In particular, no non-trivial polynomial-time approximation algorithm is known for the CONNECTED FEEDBACK VERTEX SET problem on general graphs. Note that our FPT algorithm for the problem suggests a — trivial — polynomial-time approximation algorithm with approximation factor $n/c \log n$ for any constant c , where n is the number of vertices in the input graph. After doing the obvious sanity checks for connectivity, such an algorithm would run our FPT algorithm for $k = 1, 2, \dots, c \log n$, each time stopping after $\mathcal{O}^*(46.2^k)$ steps. This takes $n^{\mathcal{O}(1)}$ time. If the FPT algorithm finds a connected feedback vertex set S of the input graph, then the approximation algorithm returns the first such set as an — exact — answer. Otherwise, it returns the vertex set of the input graph, which is a solution and is no larger than $n/c \log n$ times the smallest connected feedback vertex set.

Thus an interesting open problem is to find a polynomial-time approximation algorithm for the CONNECTED FEEDBACK VERTEX SET problem with an approximation ratio asymptotically better than $n/\log n$, or to show that no such algorithm exists. Another question which we find interesting is whether PARAMETERIZED CONNECTED FEEDBACK VERTEX SET admits a polynomial kernel on graphs excluding an arbitrary fixed graph H as a minor. It will also be interesting to find other cases where an application of the $\mathcal{O}^*(2^t)$ algorithm for PARAMETERIZED GROUP STEINER TREE/PARAMETERIZED DIRECTED STEINER OUT-TREE yields fast FPT algorithms.

Total Vertex Cover and Total Edge Cover

GIVEN a graph G and a positive integer k as input, the VERTEX COVER problem asks whether G has a set of at most k vertices — a *vertex cover* of G — such that every edge of G has at least one of these vertices as an end point. The EDGE COVER problem is quite similar : given (G, k) as input, the question here is whether G has a set of at most k edges — an *edge cover* of G — such that every vertex in G is an end point of at least one of these edges. VERTEX COVER is a classical NP-hard problem [75] whose parameterized version with k as the parameter — PARAMETERIZED VERTEX COVER — is arguably the most investigated problem in parameterized algorithmics. The fastest known FPT algorithm for PARAMETERIZED VERTEX COVER runs in $\mathcal{O}^*(1.2738^k)$ time [25], and the problem has a kernel with at most $2k$ vertices [23]. In contrast, the EDGE COVER problem is solvable in polynomial time [96].

We investigate the parameterized complexity of variants of VERTEX COVER and EDGE COVER where additional connectivity constraints are imposed on the solution set S . More specifically, for each $1 \leq t \leq k$ we define variants of the two problems, named t -TOTAL VERTEX COVER and t -TOTAL EDGE COVER, respectively, as follows.

t -TOTAL VERTEX COVER

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

Question: Does G have a vertex cover S of size at most k such that each connected component of the subgraph of G induced on S contains at least t vertices?

t -TOTAL EDGE COVER

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

Question: Does G have an edge cover T of size at most k such that each connected component of the subgraph of G induced on T contains at least t edges from T ?

Observe that for $t = 1$, these problems are identical to VERTEX COVER and EDGE COVER, respectively. A vertex cover satisfying the conditions specified in the first problem is called a t -total vertex cover of the graph G ; an edge cover satisfying the conditions of the second problem is called a t -total edge cover of G .

These problems were introduced by Fernau and Manlove [50], who showed that t -TOTAL VERTEX COVER is NP-hard for all $1 \leq t \leq k$, and that t -TOTAL EDGE COVER is NP-hard for all $2 \leq t \leq k$. They also initiated the study of the following parameterized variants of these problems, which we investigate further in this chapter:

PARAMETERIZED t -TOTAL VERTEX COVER

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

Parameter: k

Question: Does G have a vertex cover S of size at most k such that each connected component of the subgraph of G induced on S contains at least t vertices?

PARAMETERIZED t -TOTAL EDGE COVER

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

Parameter: k

Question: Does G have an edge cover T of size at most k such that each connected component of the subgraph of G induced on T contains at least t edges?

Małafiejski and Żylinski studied 2-TOTAL EDGE COVER as a model of weak cooperation of guards in an art gallery problem [87]. Both Fernau and Manlove [50] and Małafiejski and Żylinski [87] derived a Gallai type identity which says that under certain conditions, the sum of (i) the cardinality of the largest possible packing of a graph with vertex-disjoint copies of a path of length two and (ii) the size of the smallest 2-total edge cover, equals the number of vertices of the graph. Fernau and Manlove also derived a generalization of this result to all $t \geq 2$ [50]. Combining this with the result of Kirkpatrick and Hell [77], who proved that finding a packing of vertex-disjoint copies of trees on t edges in a graph is NP-hard, they showed that t -TOTAL EDGE COVER is NP-complete for all $t \geq 2$.

Besides the art gallery problem mentioned above, further motivation for studying these problems can be drawn from certain models of fault-tolerant computing [81]. These problems are interesting from the point of view of computational biology as well, due to the close relation that these problems have to variants of the so-called TEST COVER PROBLEM [32].

Fernau and Manlove [50] derived a number of results for these problems. Apart from the NP-hardness results mentioned above, they showed that for $2 \leq t \leq k$ the t -TOTAL VERTEX COVER problem has a polynomial-time 2-approximation, but cannot be approximated to within less than a factor of $10\sqrt{5}-21$ (≈ 1.3606) in polynomial time unless $P=NP$. They further showed that for $2 \leq t \leq k$ the t -TOTAL EDGE COVER problem has a polynomial-time 2-approximation, and that there exists a constant $\delta > 1$ such that 2-TOTAL EDGE COVER cannot be approximated to within less than a factor of δ in polynomial time unless $P=NP$. As for the parameterized versions of these problems, they showed that (i) PARAMETERIZED 2-TOTAL VERTEX COVER is FPT and can be solved in $\mathcal{O}^*(2.3655^k)$ time, and that (ii) for $2 \leq t \leq k$, PARAMETERIZED t -TOTAL EDGE COVER is FPT and can be solved in $\mathcal{O}^*((2k)^{2k})$ time. They also claimed to prove that the PARAMETERIZED t -TOTAL VERTEX COVER problem has a kernel of size $\mathcal{O}(k(k+t))$ for $2 \leq t \leq k$. However, as we show in this chapter, such is not the case unless the Polynomial Hierarchy collapses to the third level, which is considered unlikely.

Our Results

We advance the study of the parameterized complexity of PARAMETERIZED t -TOTAL VERTEX COVER and PARAMETERIZED t -TOTAL EDGE COVER initiated by Fernau and Manlove [50]. We significantly improve their results and obtain several new results; in particular, we complete the picture on how even the slightest connectivity requirement dramatically changes the complexity of these problems. As noted above, EDGE COVER has been known to be solvable in polynomial time for over half a century, and it was recently shown [50] that the least possible connectivity requirement on the solution set T , namely that each connected component of the graph induced on T have at least 2 edges from T , makes the problem NP-hard.

We show a similar result for t -TOTAL VERTEX COVER, not in the context of classical complexity, but within the ambit of parameterized complexity. It is a well-known result in parameterized complexity that PARAMETERIZED VERTEX COVER has a kernel on at most $2k$ vertices [23]. We show that adding a connectivity constraint results in a dramatic change in kernelizability: We show that for any fixed $2 \leq t \leq k$, the t -TOTAL VERTEX COVER problem has no polynomial-size kernel unless the Polynomial Hierarchy (PH) collapses to the third

level, which is deemed unlikely in complexity theory. We complement this no-polynomial-kernel result with results on the fixed-parameter tractability of *PARAMETERIZED t -TOTAL VERTEX COVER* and *PARAMETERIZED t -TOTAL EDGE COVER*. Specifically, we show the following:

- *PARAMETERIZED t -TOTAL VERTEX COVER* can be solved in $\mathcal{O}(16.1^{k+\mathcal{O}(\log^2 k)} \times n^{\mathcal{O}(1)})$ time. To obtain this result we combine the classical result of Otter [97] on the number of unlabelled trees with a modification of the colour-coding technique of Alon et al. [4];
- *PARAMETERIZED t -TOTAL EDGE COVER* has a kernel on at most $\frac{t+1}{t}k$ vertices;
- *PARAMETERIZED t -TOTAL EDGE COVER* can be solved in $\mathcal{O}\left(2^{\frac{t+1}{t}k+\mathcal{O}(\sqrt{k})} \times n^{\mathcal{O}(1)}\right)$ time. To obtain this result, we combine kernelization techniques with a classical result of Hardy and Ramanujan [67] and the Fast Fourier Transform [79].

Organization of the rest of the chapter

We take up the *PARAMETERIZED t -TOTAL VERTEX COVER* problem in [Section 7.1](#). In [Section 7.1.1](#) we show that the problem does not admit polynomial kernels unless the Polynomial Hierarchy collapses to the third level. In [Section 7.1.2](#) we present our FPT algorithm for the problem. In [Section 7.2](#) we turn to the *PARAMETERIZED t -TOTAL EDGE COVER* problem. We present our improved kernel for this problem in [Section 7.2.1](#). In [Section 7.2.2](#) we describe our FPT algorithm for the problem. We conclude in [Section 7.3](#).

7.1 Computing Total Vertex Covers

We now consider the *t -TOTAL VERTEX COVER* problem. The problem is NP-complete for all values of t . For $t = 1$, *t -TOTAL VERTEX COVER* is the *VERTEX COVER* problem, and for $t = k$ it becomes the *CONNECTED VERTEX COVER* problem; these are two classical NP-complete problems [59, Problem GT1]. For $2 \leq t \leq k$, the *t -TOTAL VERTEX COVER* problem has been shown to be NP-hard by reduction from *VERTEX COVER* [50, Theorem 3]; we give an alternate proof of NP-hardness in [Claim 1](#) below. In this section we investigate the parameterized complexity of the *PARAMETERIZED t -TOTAL VERTEX COVER* problem.

7.1.1 Kernelization Complexity

Recall that for $t = 1$, *PARAMETERIZED t -TOTAL VERTEX COVER* is just *PARAMETERIZED VERTEX COVER*, and for $t = k$ it becomes *PARAMETERIZED CONNECTED VERTEX COVER*.

The former problem has a vertex kernel of size at most $2k$ [23], and the latter problem does not have polynomial kernels [39]. It turns out that this change in polynomial kernelizability occurs at the smallest possible value of t .

Theorem 7.1. *For each fixed $t \geq 2$, PARAMETERIZED t -TOTAL VERTEX COVER has no kernel of size bounded by k^c , for any fixed constant c , unless the Polynomial Hierarchy collapses to the third level.*

To prove this, we need a few notions and results from the recently developed theory of kernel lower bounds [15, 18, 39]; these are described in Section 2.2.1.

As we show later in this section, the derived classical problem — the “unparameterized” version — of PARAMETERIZED t -TOTAL VERTEX COVER is NP-complete for $2 \leq t \leq k$ (Claim 1). By Theorem 2.2, to show that PARAMETERIZED t -TOTAL VERTEX COVER has no polynomial kernel for $2 \leq t \leq k$, it is sufficient to exhibit composition algorithms for these problems. Unfortunately, this task turns out to be quite hard, and we have not been able to devise such composition algorithms. To get around this difficulty, we make use of the second tool for obtaining kernel lower bounds, namely polynomial parameter transformations (See Section 2.2.1) and Theorem 2.3. We introduce an intermediate problem named PARAMETERIZED RED BLUE DOMINATING SET. As we show below, the unparameterized version of this problem is NP-complete, and it is known that the problem does not admit polynomial kernels. For each $t; 2 \leq t \leq k$ we give a polynomial parameter transformation from PARAMETERIZED RED BLUE DOMINATING SET to PARAMETERIZED t -TOTAL VERTEX COVER, which implies, by Theorem 2.3, that the latter problem has no polynomial kernel. We start off by defining the intermediate problem:

PARAMETERIZED RED BLUE DOMINATING SET

Input: An undirected bipartite graph $G = (R \uplus B, E)$, and a positive integer k .

Parameter: $k + |B|$

Question: Does there exist a set $D \subseteq R$ of at most k vertices of G such that every $v \in B$ is adjacent to some $u \in D$ (i.e., D is a dominating set of B)?

The derived classical problem corresponding to PARAMETERIZED RED BLUE DOMINATING SET, defined below, is NP-complete.

RED BLUE DOMINATING SET

Input: An undirected bipartite graph $G = (R \uplus B, E)$, a positive integer k , and the number $k + |B|$ written in unary.

Question: Does there exist a set $D \subseteq R$ of at most k vertices of G such that every $v \in B$ is adjacent to some $u \in D$ (i.e., D is a dominating set of B)?

Fact 1. [42] *The RED BLUE DOMINATING SET problem is NP-complete.*

It has been shown that the parameterized version of this problem does not admit polynomial kernels:

Fact 2. [39, Theorem 2] *PARAMETERIZED RED BLUE DOMINATING SET does not admit a polynomial kernel unless the Polynomial Hierarchy collapses to the third level.*

We are now ready to show that PARAMETERIZED t -TOTAL VERTEX COVER has no polynomial kernel for any $2 \leq t \leq k$.

Theorem 7.2. *For each fixed $t; 1 \leq t \leq k$, PARAMETERIZED t -TOTAL VERTEX COVER has no kernel of size bounded by k^c , for any fixed constant c , unless the Polynomial Hierarchy collapses to the third level.*

Proof. As shown in Fact 1, the derived classical problem corresponding to PARAMETERIZED RED BLUE DOMINATING SET is NP-complete. Also, the derived classical problem corresponding to PARAMETERIZED t -TOTAL VERTEX COVER is evidently in NP. Now suppose PARAMETERIZED t -TOTAL VERTEX COVER has a polynomial kernel, and that there is a polynomial parameter transformation from PARAMETERIZED RED BLUE DOMINATING SET to PARAMETERIZED t -TOTAL VERTEX COVER. Then by Theorem 2.3, PARAMETERIZED RED BLUE DOMINATING SET has a polynomial kernel, and hence by Fact 2 PH collapses to the third level; it follows that PARAMETERIZED t -TOTAL VERTEX COVER does not have a polynomial kernel unless PH collapses to the third level. Hence to prove the theorem, it suffices to show that there is a polynomial parameter transformation from PARAMETERIZED RED BLUE DOMINATING SET to PARAMETERIZED t -TOTAL VERTEX COVER. We now proceed to give such a transformation.

Given an instance $(G = (R \uplus B, E), k)$ of PARAMETERIZED RED BLUE DOMINATING SET, we construct an instance of PARAMETERIZED t -TOTAL VERTEX COVER as follows: If B contains isolated vertices then (G, k) is a NO instance, and in this case we construct a trivial NO instance of PARAMETERIZED t -TOTAL VERTEX COVER. Otherwise, we add a distinct path of length (number of edges) $t-1$ starting from each vertex $v \in B$. Thus, if $t = 2$, then we attach a new, distinct pendant vertex w_i to each $v_i \in B$; if $t = 3$, then we add a path (v_i, u_i^1, w_i) to each $v_i \in B$. In general, for $2 \leq t \leq k$, we add a path $(v_i, u_i^1, u_i^2, \dots, u_i^{t-2}, w_i)$ to each $v_i \in B$, where the vertices u_i^j and w_i are all new and distinct: see Figure 7.1 for an illustration. We call the resulting graph H , and $(H, k + (t-1)|B|)$ is the constructed instance of PARAMETERIZED t -TOTAL VERTEX COVER. To complete the proof, we show:

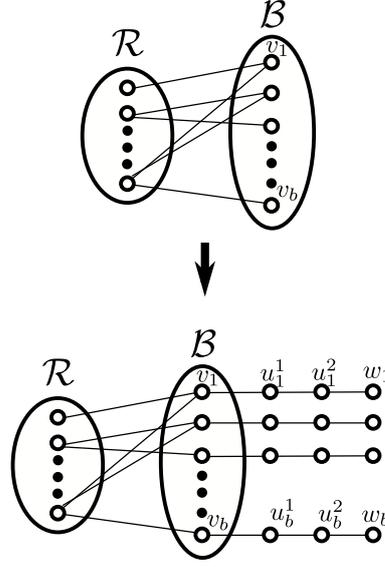


Figure 7.1: **Polynomial parameter transformation from PARAMETERIZED RED BLUE DOMINATING SET to PARAMETERIZED t -TOTAL VERTEX COVER, for the case $t = 4$.** The construction appends a path with $t - 1 = 3$ edges to each vertex in the set B ; the new parameter is $k + 3|B|$.

Claim 1. Let $(G = (R \uplus B, E), k)$ be an instance of PARAMETERIZED RED BLUE DOMINATING SET, and t a fixed positive integer. Let H be the graph constructed from G as described above. Then (G, k) is a YES instance of PARAMETERIZED RED BLUE DOMINATING SET if and only if $(H, k + (t - 1)|B|)$ is a YES instance of PARAMETERIZED t -TOTAL VERTEX COVER.

Proof. Let $(G = (R \uplus B, E), k)$ be a YES instance of PARAMETERIZED RED BLUE DOMINATING SET. Then there is an inclusion-minimal set $D \subseteq R$, $|D| \leq k$, that dominates B . Let S be the set of all new vertices added by the construction to H , *except* for the pendant vertex w_i in each new path. Thus, e.g., $S = \emptyset$ when $t = 2$, and in general $|S| = (t - 2)|B|$. Define $C = D \cup B \cup S$. Now,

1. $|C| = |D| + |B| + |S| \leq k + |B| + (t - 2)|B| \leq k + (t - 1)|B|$.
2. C is a vertex cover of H : $B \subseteq C$ covers all original edges, and all new edges adjacent to vertices in B ; S covers the rest of the new edges, if any.
3. Each connected component of $H[C]$ contains at least t vertices:
 - (a) Since D dominates B in G , any vertex $v_i \in B$ has at least one neighbour $w \in D \subseteq C$ in $H[C]$. Then v_i, w , and the $t - 2$ new vertices $\{u_i^1, u_i^2, \dots, u_i^{t-2}\} \subseteq S$ are all part of the same component in $H[C]$, as witnessed by the path

$w, v_i, u_i^1, u_i^2, \dots, u_i^{t-2}$. Thus each vertex $v_i \in B$ is part of a connected component of size at least t in $H[C]$.

- (b) D is an inclusion-minimal dominating set of B and $D \cup B \subseteq C$, and so each $w \in D$ has at least one neighbour $v_i \in B$ in the graph $H[C]$. It follows that each vertex $w \in D$ is part of a connected component of size at least t in $H[C]$, namely the component to which v_i belongs.
- (c) Each vertex $u_i^j \in S$ is in the same component in $H[C]$ as the vertex $v_i \in B$, and so u_i^j is part of a connected component of size at least t in $H[C]$, namely the component to which v_i belongs.

Thus C is a t -total vertex cover of H , of size at most $k + (t - 1)|B|$. This proves the forward direction.

To prove the reverse direction, suppose $(H, k + (t - 1)|B|)$ is a YES instance of t -TOTAL VERTEX COVER, and let C be a t -total vertex cover of H of size at most $k + (t - 1)|B|$. Consider any path $P = (u_i^0 = v_i, u_i^1, u_i^2, \dots, u_i^{t-2}, w_i)$ in H consisting of a vertex $v_i \in B$ and new vertices added by the construction. Since C is a vertex cover such that each connected component of $G[C]$ has at least t vertices, we have that $|P \cap C| \geq t - 1$. Now suppose there exists a vertex $x \in P \setminus C$. If $x = u_i^j$ for some $0 \leq j \leq t - 2$, then the vertices $u_i^{j+1}, u_i^{j+2}, \dots, u_i^{t-2}, w_i$ form a connected component of $H[C]$ of size strictly less than t , a contradiction. So one of the following holds:

1. All the t vertices of P are in C , or,
2. $\{v_i, u_i^1, u_i^2, \dots, u_i^{t-2}\} \in C$, and $w_i \notin C$.

Let p_t be the number of paths of the first kind in H , and let p_{t-1} be the number of such paths of the second kind. There is exactly one such path corresponding to each vertex of B , and so $p_t + p_{t-1} = |B|$. The total number of vertices contributed to C by these paths is $tp_t + (t - 1)p_{t-1}$, and so the number of vertices in $C \cap R$ is at most $k + (t - 1)|B| - (tp_t + (t - 1)p_{t-1}) = k - p_t$. Now let P be a path of the second kind. By definition, $|P \cap C| = (t - 1)$, and since each connected component of $G[C]$ has at least t vertices, there is at least one more vertex x in C that is adjacent in H to one of the vertices, say y , of $P \cap C$. The only possibility is that $x \in R$ and $y \in B$, and so at most $k - p_t$ vertices in $C \cap R$ dominate $p_{t-1} = |B| - p_t$ vertices in B . Since each vertex of B has a neighbour in R , at most k vertices in R dominate all of B . \square

This completes the proof of the theorem. \square

The reduction employed in the above argument also implies:

Corollary 7.1. *For each fixed $t \geq 2$ the t -TOTAL VERTEX COVER problem is NP-hard on bipartite graphs, and the PARAMETERIZED t -TOTAL VERTEX COVER problem does not admit a polynomial kernel on bipartite graphs unless the Polynomial Hierarchy collapses to the third level.*

7.1.2 Fixed Parameter Tractability

We now turn to the fixed-parameter tractability of PARAMETERIZED t -TOTAL VERTEX COVER. Two special cases of the problem, for the two extreme values namely $t = 1$ (PARAMETERIZED VERTEX COVER) and $t = k$ (PARAMETERIZED CONNECTED VERTEX COVER), have been studied extensively from the perspective of parameterized algorithms. The PARAMETERIZED VERTEX COVER problem is perhaps the most well-studied problem in parameterized algorithmics. After a long series of improvements, the current fastest FPT algorithm for this problem runs in time $\mathcal{O}^*(1.2738^k)$ [25]. Similarly PARAMETERIZED CONNECTED VERTEX COVER also has a history of improvements, and the current fastest FPT algorithm for this problem runs in time $\mathcal{O}^*(2.7606^k)$ [91]. We show in this section that PARAMETERIZED t -TOTAL VERTEX COVER is FPT parameterized by the solution size k for $2 \leq t \leq k$, by deriving an $\mathcal{O}^*(16.1^{k+\mathcal{O}(\log^2 k)})$ time algorithm for these problems.

Let $G = (V, E)$ be the input graph, and let $|V| = n$. Observe that the set V is a vertex cover of G . Therefore, if $|V| \leq k$, then we can solve the problem in polynomial time by checking whether each component of G has at least t vertices. Also, deleting isolated vertices does not affect the solution. Hence we assume without loss of generality that $|V| > k$, and that G has no isolated vertices. We start with a structural claim which is useful later.

Claim 2. *Let $G = (V, E)$, $|V| > k$ be a graph without isolated vertices. Then G has a t -total vertex cover of size at most k if and only if G has a t -total vertex cover of size exactly k .*

Proof. If G has a t -total vertex cover, say S , of size exactly k , then S itself is a t -total vertex cover of G of size at most k . For the other direction, let S be a t -total vertex cover of G size $l < k$. Consider any set of $k - l$ vertices $T \subseteq (V \setminus S)$. Since G has no isolated vertex, each $v \in T$ has at least one edge incident on it; since S is a vertex cover of G , the other end of this edge, say w , is in S . Now notice that every connected component of $G[S \cup T]$ has at least t vertices as each connected component of $G[S]$ has at least t vertices and every vertex of T gets attached to one of the components of $G[S]$. \square

The number of unlabelled trees on k vertices is known to be singly exponential in k , and all these trees can be enumerated with polynomial delay:

Fact 3. [12, 97] *The number of unlabelled trees on k vertices is at most 2.96^k . Moreover, all non-isomorphic unlabelled trees on k vertices can be enumerated in time $\mathcal{O}(2.96^k k^c)$ for some constant c independent of k .*

From this we get:

Lemma 7.1. *All unlabelled FORESTS on k vertices can be enumerated in $\mathcal{O}^*(2.96^k)$ time.*

Proof. Let F be a forest on k vertices. Add a new vertex v and one edge from v to an arbitrary vertex of each tree in F , to obtain a tree T on $k + 1$ vertices. Clearly, the forest F can be obtained by deleting one vertex (namely v) from T . It follows that a graph is a forest on k vertices if and only if it can be obtained by deleting a vertex from some tree on $k + 1$ vertices.

To enumerate all forests on k vertices, we first enumerate all trees on $k+1$ vertices. From **Fact 3**, this can be done in $\mathcal{O}(2.96^{k+1} (k+1)^c)$ time where c is a constant independent of k . For each tree T on $k + 1$ vertices obtained in this manner, we delete each of its $k + 1$ vertices, one at a time, to obtain a set of forests. By the above observation, this procedure yields every forest on k vertices (some of them perhaps many times). The procedure takes $\mathcal{O}((k+1) 2.96^{k+1} (k+1)^c) = \mathcal{O}^*(2.96^k)$ time. \square

We are now ready to prove the main result of this section.

Theorem 7.3. *For every $t \geq 1$ the PARAMETERIZED t -TOTAL VERTEX COVER problem is fixed-parameter tractable, and can be solved in time $\mathcal{O}^*(16.1^{k+\mathcal{O}(\log^2 k)})$.*

Proof. Observe that any t -total vertex cover, say S , of G is also a vertex cover of G and hence contains a minimal vertex cover $S' \subseteq S$ of G . The idea of our proof is to enumerate all the minimal vertex covers of G of size at most k and then try to expand each one to a t -total vertex cover of G . We will use **Fact 3** and the colour-coding technique of Alon et al. [4] to do the expansion phase of our algorithm. Our algorithm is based on the following claim.

Claim 3. *A graph $G = (V, E)$ has a t -total vertex cover of size k if and only if there exists a minimal vertex cover C of G of size at most k , and a subset $T \subseteq V \setminus C$ of size $k - |C|$, such that there exists a forest F on k vertices which is isomorphic to a spanning subgraph of $G[C \cup T]$, and in which each connected component has at least t vertices.*

Proof. If G has a t -total vertex cover S of size k , then by definition S is a vertex cover of G . Let C be any minimal vertex cover contained in S , and let $T = S \setminus C$. Then $|T| = k - |C|$, and each connected component of $G[C \cup T] = G[S]$ has, from the definition of a t -total

vertex cover, at least t vertices. Let F be a forest formed by picking one spanning tree from each connected component of $G[S]$. Then F satisfies the conditions of the claim.

Conversely, let there exist a minimal vertex cover C of G , a set $T \subseteq V \setminus C$ of size $k - |C|$, and a forest F on k vertices that is isomorphic to a spanning subgraph $G' = (S = C \cup T, E')$ of $G[C \cup T]$, and in which each connected component has at least t vertices. Since $C \subseteq S$, S is a vertex cover of G . Also $|S| = |C \cup T| = k$. Now since G' is a subgraph of $G[S]$, and each connected component of G' contains at least t vertices, each connected component of $G[S]$ has at least t vertices. It follows that S is a t -total vertex cover of G of size k . \square

If G has a t -total vertex cover of size at most k , then from [Claim 2](#) we know that G has a t -total vertex cover of size exactly k . Let S be a fixed t -total vertex cover of G of size exactly k , if there exists one. From [Claim 3](#) we get that S contains an inclusion-minimal vertex cover C of G , of size at most k . A “colouring” of the vertex set V of G is a function from V to some specified set of “colours”. A “good” colouring of V is a colouring in which the vertices in S are all distinctly coloured.

Our algorithm tries to find S by mimicking [Claim 3](#). First we enumerate all inclusion-minimal vertex covers of G of size at most k . This can be done in time $\mathcal{O}^*(2^k)$ by a simple 2-way branching on edges — for every edge at least one of its endpoints should be in any vertex cover. For each such vertex cover C , we do the following:

1. Colour each $v \in C$ with a distinct colour from $\{1, 2, \dots, |C|\}$.
2. Let $\ell = k - |C|$. Colour the vertices of the independent set $V \setminus C$ uniformly at random with ℓ new colours $\tilde{1}, \tilde{2}, \dots, \tilde{\ell}$.

Observe that $|S \setminus C| = \ell$. The number of ways of colouring the vertices in $S \setminus C$ with ℓ distinct colours is $\ell!$, and the total number of ways of colouring these vertices with these ℓ colours is ℓ^ℓ . The random colouring described above will therefore yield a good colouring of V with probability $\ell!/\ell^\ell \geq e^{-\ell}$.

We now check if the random colouring is a good colouring. For this, we iterate through all unlabelled forests on k vertices, and check if at least one of these forests is isomorphic to a spanning forest \mathcal{F} of $G[S]$, where each connected component of \mathcal{F} has at least t vertices. By [Lemma 7.1](#), we can iterate through all such forests in $\mathcal{O}^*(2.96^k)$ time. To check if a given forest F on k vertices is isomorphic with such a spanning forest \mathcal{F} of $G[S]$, we do the following:

1. We check if there is at least one tree in F that has less than t vertices. If yes, then we reject F .

2. Next we check if there is a colourful subgraph (one in which each vertex has a distinct colour) isomorphic to F in the coloured graph obtained above. Since F is of treewidth at most 1, this can be done in $\mathcal{O}(2^k \cdot k \cdot n^2)$ time [8, Corollary 6]. If such a subgraph is present, then F satisfies the requirements of Claim 3, and so we return YES; the underlying uncoloured graph of this colourful subgraph is a t -total vertex cover of G of size at most k . Otherwise we reject the forest F .

If the above check rejects all unlabelled forests on k vertices, then we return NO: this colouring is not a good colouring.

Observe that if the input graph G has a t -total vertex cover of size k , then the above algorithm will discover this t -total vertex cover with probability at least $e^{-\ell}$, and so will return YES with probability at least $e^{-\ell}$. If the input graph G is a NO instance, then the algorithm will always return NO. The expected number of times the algorithm has to be repeated before it finds a t -total vertex cover of size k of G , if it exists, is thus e^ℓ . The expected running time of this procedure is thus

$$\begin{aligned} \mathcal{O}^* \left(\sum_{\ell=0}^k 2^{k-\ell} \times e^\ell \times 2.96^k \times 2^k \right) &= \mathcal{O}^* \left((2 \times 2.96 \times 2)^k \times \sum_{\ell=0}^k \left(\left(\frac{e}{2} \right)^\ell \right) \right) \\ &= \mathcal{O}^* \left((5.92e)^k \right) = \mathcal{O}^* (16.1^k). \end{aligned}$$

To obtain a deterministic algorithm we have to replace the randomized step of the algorithm — where we colour the vertices of $G[V \setminus C]$ uniformly at random by ℓ colours — with a deterministic procedure. We do this using the so-called (n, ℓ, ℓ) -perfect hash families.

An (n, ℓ, ℓ) -perfect hash family \mathcal{H} is a set of functions from $\{1, \dots, n\}$ to $\{1, \dots, \ell\}$ such that for every subset $S \subseteq \{1, \dots, n\}$ of size ℓ there exists a function $f \in \mathcal{H}$ such that f is injective on S . That is, such that for all $i, j \in S$, $f(i) \neq f(j)$. There exists a construction of an (n, ℓ, ℓ) -perfect hash family of size $\mathcal{O}(e^\ell \cdot \ell^{\mathcal{O}(\log \ell)} \cdot \log n)$ and one can produce this family in time linear in the output size [93].

Let $\tilde{n} = |V \setminus C|$. To derandomize our algorithm, we construct an (\tilde{n}, ℓ, ℓ) -perfect hash family \mathcal{H} from $\{1, \dots, \tilde{n}\}$ to $\{\tilde{1}, \dots, \tilde{\ell}\}$. This has to be done only *once* during the algorithm. Now we replace the second step in the colouring process with the following:

- 2'. Instead of colouring the vertices of $V \setminus C$ uniformly at random with ℓ colours and checking if this yields a good colouring, we “colour” the vertices of $V \setminus C$ with *each* function in \mathcal{H} , in turn. For each such colouring, we check if it is a good colouring as before.

If G has a t -total vertex cover S of size k , then from the definition of an (\tilde{n}, ℓ, ℓ) -perfect hash family, it follows that at least one of the functions in \mathcal{H} will result in a good colouring of V . Thus, if G has a t -total vertex cover S of size k , then this algorithm will always discover S and return YES. If the input graph G is a No instance, then the algorithm will always return NO. Instead of the multiplicative factor of e^ℓ — which came from repeating the randomized algorithm these many times — in the running time of the randomized procedure, the derandomized version incurs the following costs:

1. An additive cost of $\mathcal{O}(e^\ell \cdot \ell^{\mathcal{O}(\log \ell)} \cdot \log \tilde{n})$ to compute the hash family, and
2. A multiplicative factor of $\mathcal{O}(e^\ell \cdot \ell^{\mathcal{O}(\log \ell)} \cdot \log \tilde{n})$ which arises from repeating the check for a good colouring once for each function in the hash family.

The running time of the derandomized algorithm is thus

$$\begin{aligned} & \mathcal{O}^* \left(\sum_{\ell=0}^k 2^{k-\ell} \times e^\ell \times \ell^{\mathcal{O}(\log \ell)} \times 2.96^k \times 2^k \right) = \\ & \mathcal{O}^* \left((2 \times 2.96 \times 2)^k \times \sum_{\ell=0}^k \left(\left(\frac{e}{2} \right)^\ell \times \ell^{\mathcal{O}(\log \ell)} \right) \right) = \\ & \mathcal{O}^* \left((11.84)^k \times k^{\mathcal{O}(\log k)} \times \left(\frac{e}{2} \right)^k \right) = \\ & \mathcal{O}^* \left(16.1^{k+\mathcal{O}(\log^2 k)} \right). \end{aligned}$$

This concludes the proof of the theorem. □

7.2 Computing Total Edge Covers

We now consider the t -TOTAL EDGE COVER problem. For $t = 1$ this problem becomes the EDGE COVER problem, which has long been known to be solvable in polynomial time [96]. The problem is NP-complete for all $t \geq 2$ [50, Theorem 3]. In this section we investigate the parameterized complexity of the PARAMETERIZED t -TOTAL EDGE COVER problem for $t \geq 2$. We first study the kernelization complexity of the problem, and improve the size of the kernels for all $t \geq 2$. Then we take up the fixed parameter tractability of this problem, and obtain an FPT algorithm with a significantly improved running time.

In our analysis we make use of a different formulation of the problem, other than the one presented at the beginning of this chapter. The following fact, culled from the proof of a theorem in the previous work due to Fernau and Manlove [50, Theorem 16], helps us show that the two formulations are equivalent.

Fact 4. [50] *In any connected graph G with n vertices, and for any $t < n$, there exists a minimal t -total edge cover, say S , of G such that the graph $G(S)$ induced by the edge set S is acyclic.*

7.2.1 Kernelization Complexity

Fernau and Manlove [50] observed the following simple vertex kernel of size at most $2k$ for t -TOTAL EDGE COVER [50]: any edge in a graph covers exactly 2 vertices, and a YES instance of the problem has an edge cover of size (number of edges) at most k , and so such an instance cannot have more than $2k$ vertices. In other words, if the input instance has more than $2k$ vertices, then the answer is NO. Otherwise, the input instance itself forms a kernel on at most $2k$ vertices. We can improve this bound on the kernel size for larger values of t by observing the following:

Lemma 7.2. *Given a graph $G = (V, E)$ and a non-negative integer k , solving the t -TOTAL EDGE COVER instance (G, k) is equivalent to solving the following problem: does there exist a partition of the vertex set V into q parts V_1, \dots, V_q , for some q , such that (i) $G[V_i]$ is connected, (ii) $|V_i| \geq t + 1$ for each $1 \leq i \leq q$, and (iii) $\sum_{i=1}^q (|V_i| - 1) \leq k$?*

Proof. Let $(G = (V, E), k)$ be an instance of PARAMETERIZED t -TOTAL EDGE COVER and let S be an edge-minimal t -total edge cover of G . Let V_1, \dots, V_q be the vertex sets of the connected components of $G(S) = (V, S)$. It directly follows from Fact 4 and the properties of S given in the definition of PARAMETERIZED t -TOTAL EDGE COVER that V_1, \dots, V_q satisfy all the conditions in the statement of the lemma. For the reverse direction, observe first that the edges of a spanning tree of any connected graph form an edge cover of the graph. Now, if V_1, \dots, V_q satisfy all the conditions in the statement of the lemma, then let S_i be the edges of a spanning tree of $G[V_i]$, for $1 \leq i \leq q$, and let $S = \bigcup_{i=1}^q S_i$. Observe now that S has the properties stated in the definition of PARAMETERIZED t -TOTAL EDGE COVER. \square

We use this reformulation to get smaller bounds on the kernel size for PARAMETERIZED t -TOTAL EDGE COVER.

Theorem 7.4. *PARAMETERIZED t -TOTAL EDGE COVER admits a vertex kernel of size $\frac{t+1}{t}k$.*

Proof. Let $(G = (V, E), k)$ be a YES instance of PARAMETERIZED t -TOTAL EDGE COVER. By Lemma 7.2, there exists a partition of V into q parts of the kind stated in Lemma 7.2. Now $|V_i| \geq t + 1 \implies |V_i| - 1 \geq t \implies \sum_{i=1}^q (|V_i| - 1) \geq qt$. By Lemma 7.2, $\sum_{i=1}^q (|V_i| - 1) \leq k$, and so $qt \leq k$, and $q \leq \frac{k}{t}$. Also, $\sum_{i=1}^q (|V_i| - 1) \leq k \implies \sum_{i=1}^q |V_i| \leq k + q \leq k + \frac{k}{t} = \frac{t+1}{t}k$, and so G has at most $\frac{t+1}{t}k$ vertices. \square

Thus, if the input graph G has more than $\frac{t+1}{t}k$ vertices, then the answer is No. Therefore we can assume without loss of generality that the input graph has at most $\frac{t+1}{t}k$ vertices, and so *any* exact algorithm for the problem is in fact an FPT algorithm. In particular we have:

Corollary 7.2. *If PARAMETERIZED t -TOTAL EDGE COVER has an exact exponential time algorithm that runs in $\mathcal{O}^*(c^{f(|V|)})$ time on an input instance $(G = (V, E), k)$ for some constant c and function $f(\cdot)$, then the problem has an FPT algorithm that runs in $\mathcal{O}^*\left(c^{f\left(\frac{t+1}{t}k\right)}\right)$ time.*

7.2.2 Fixed Parameter Tractability

We now present an exact exponential-time algorithm for PARAMETERIZED t -TOTAL EDGE COVER which runs in $\mathcal{O}^*(2^{n+\mathcal{O}(\sqrt{n})})$ time where n is the number of vertices in the input graph. By [Corollary 7.2](#) this yields an FPT algorithm for the problem which runs in $\mathcal{O}^*(c^k)$ time for some fixed constant c . This is a significant improvement over the previous best upper bound of $\mathcal{O}^*((2k)^{2k})$ [\[50\]](#).

Let $(G = (V, E), k)$ be an input instance of PARAMETERIZED t -TOTAL EDGE COVER, and let $|V| = n$. We start by enumerating all unordered partitions of n . An *unordered partition* of a positive integer n is a way of writing n as a sum of positive integers, where the order of the summands is ignored. By the Hardy-Ramanujan asymptotic formula, n has at most $2^{\mathcal{O}(\sqrt{n})}$ unordered partitions [\[67\]](#). The partitions of n can be generated with constant average delay [\[112\]](#), and so we can enumerate all unordered partitions of n in $2^{\mathcal{O}(\sqrt{n})}$ time.

We consider those partitions of n of the form $n = n_1 + n_2 + \dots + n_q$ which satisfy the following conditions of [Lemma 7.2](#):

1. $\sum_{i=1}^q (|n_i| - 1) \leq k$
2. $|n_i| \geq t + 1$

For each such partition of n , we check if there exists a partition of the vertex set V into q parts V_1, \dots, V_q such that

1. $|V_i| = n_i$ for $1 \leq i \leq q$, and
2. each induced subgraph $G[V_i]$ is connected.

To do these latter checks, we construct the q lists

$$L_i = \{V' \subseteq V \mid |V'| = n_i \text{ and } G[V'] \text{ is connected}\}$$

for $1 \leq i \leq q$. For $1 \leq i \leq q$ we compute the polynomials

$$P_i = \sum_{V' \in L_i} z^{\chi(V')}$$

where z is a formal variable and $\chi(V')$ is the (binary number represented by the) characteristic vector of $V' \subseteq V$. That is, let $V = \{v_1, v_2, \dots, v_n\}$. Then $\chi(V')$ is a bit vector with $|V|$ bits where, for $1 \leq j \leq |V|$, the j th bit of $\chi(V')$ is 1 if and only if $v_j \in V'$. We treat $\chi(V')$ as a binary number in all computations. The lists L_i and the polynomials P_i can be computed in $\Theta^*(2^n)$ time, by enumerating all subsets of V . We now compute the product

$$Q = P_1 \times P_2 \times \dots \times P_q$$

in the given order, with a small modification: at each step, given the partial product Q_i of the first i terms, we first compute $Q_i \times P_{i+1}$. Then we delete all those terms αz^β in $Q_i \times P_{i+1}$ where (the binary representation of) β does not contain exactly $\sum_{j=1}^{i+1} n_j$ 1s, and set Q_{i+1} to be the resulting polynomial. This pruning operation ensures that the partial product Q_i , for $1 \leq i \leq q$, represents exactly those sets of size $\sum_{j=1}^i n_j$ that can be obtained by taking the union of one set each from L_1, L_2, \dots, L_i . Observe that the product $Q_q = Q$ is non-zero if and only if there exists a partition of V into q parts satisfying the conditions stated above.

The degree of each polynomial involved in the multiplications is at most $2^{|V|} - 1 = 2^n - 1$, and so, using the Fast Fourier Transform, we can multiply two of these polynomials in $\mathcal{O}(2^n \log 2^n) = \mathcal{O}(n2^n)$ time [26, Chapter 30]. We have to perform at most $q \leq n$ such multiplications to compute Q , and so given the P_i s we can compute Q in $\mathcal{O}(n^2 2^n) = \mathcal{O}^*(2^n)$ time. The running time of this algorithm is thus $2^{\mathcal{O}(\sqrt{n})} \times (\mathcal{O}^*(2^n) + \mathcal{O}^*(2^n)) = \mathcal{O}^*(2^{n+\mathcal{O}(\sqrt{n})})$, and so we have:

Theorem 7.5. *t -TOTAL EDGE COVER can be solved in $\mathcal{O}^*(2^{n+\mathcal{O}(\sqrt{n})})$ time, where n is the number of vertices in the input graph.*

From this theorem and [Corollary 7.2](#) we get:

Theorem 7.6. *PARAMETERIZED t -TOTAL EDGE COVER can be solved in $\mathcal{O}^*\left(2^{\frac{t+1}{t}k+\mathcal{O}(\sqrt{k})}\right)$ time.*

The above algorithm uses exponential space, for constructing the lists L_i . We can use an approach similar to the one used in [Section 7.1.2](#) to get an FPT algorithm which runs in polynomial space. Specifically, we enumerate all unordered partitions (n_1, \dots, n_q) of n where $n - k \leq q \leq \frac{n}{t+1}$, such that $n_i \geq t + 1$ and $\sum_{i=1}^q (n_i - 1) \leq k$. As mentioned above,

this can be done in $\mathcal{O}^*(2^{\sqrt{n}})$ time. For each such partition, we enumerate all trees with number of vertices $n_i; 1 \leq i \leq q$. As mentioned above, this can be done in $\mathcal{O}^*(2.96^n)$ time. Then, for each enumerated q -tuple of trees (T_1, \dots, T_q) , we test whether the forest $T_1 \uplus T_2 \uplus \dots \uplus T_q$ is a subgraph of G . Since the forest has treewidth one and has the same number of vertices as G , this test for subgraph isomorphism can be done in $\mathcal{O}^*(2^n)$ time and polynomial space [7, Theorem 5]. Combining this with [Corollary 7.2](#) we get:

Theorem 7.7. *PARAMETERIZED t -TOTAL EDGE COVER can be solved in $\mathcal{O}^*\left(2^{\frac{t+1}{t}k + \mathcal{O}(\sqrt{k})}\right)$ time and using polynomial space.*

7.3 Conclusion

We study the effect of imposing some natural connectivity constraints on the subgraph induced by the solution set for two classical problems, namely VERTEX COVER and EDGE COVER. These problems exhibit contrasting behaviour with respect to classical complexity: VERTEX COVER is NP-hard while EDGE COVER is solvable in polynomial time. For both these problems, the additional constraint imposed is that each connected component of the subgraph induced by the solution set be at least as large as some specified number t ; these problems were introduced by Fernau and Manlove [50], who named the problems t -TOTAL VERTEX COVER and t -TOTAL EDGE COVER, respectively. They showed that these problems are NP-hard for $t \geq 2$, and initiated the study of the parameterized complexity of these problems when the parameter is the solution size k . We take this study further, and improve on their results.

In both cases we see that adding a connectivity constraint (each component of the solution must have at least a certain number of vertices/edges from the solution) causes a drastic change in the computational complexity of the problem. In the case of t -TOTAL EDGE COVER, the shift is from polynomial-time computability to NP-hardness, as had been observed earlier [50]. We show that a similar shift occurs in the case of the parameterized version PARAMETERIZED t -TOTAL VERTEX COVER of the NP-hard problem t -TOTAL VERTEX COVER. As is well known, for $t = 1$ the problem has a linear vertex kernel [23]. In contrast, we showed that for any $t \geq 2$ the PARAMETERIZED t -TOTAL VERTEX COVER problem has no polynomial-size kernel unless the Polynomial Hierarchy collapses, which is considered unlikely. We also show that both these problems have FPT algorithms that run in time $\mathcal{O}^*(c^k)$ for different constants c . These results improve known bounds for these problems [50].

One interesting direction of future research would be to examine the effect of such connectivity constraints on other parameterized graph problems. Another would be

to try to improve the base c of the exponent of the running times that we obtained for PARAMETERIZED t -TOTAL VERTEX COVER and PARAMETERIZED t -TOTAL EDGE COVER. Recall — [Theorem 7.3](#) — that our algorithm for the PARAMETERIZED t -TOTAL VERTEX COVER problem runs in $\mathcal{O}^*(16.1^k)$ time. For the three special values $t = 1$, $t = 2$ and $t = k$, the PARAMETERIZED t -TOTAL VERTEX COVER problem is known to be solvable in $\mathcal{O}^*(c^k)$ time for have much smaller values of c , namely $c = 1.2738$, $c = 2.3655$ and $c = 2.7606$, respectively [[25](#), [50](#), [91](#)]. It will be interesting to see if the value of c for the general case can be brought closer to these smaller values.

Part IV

Conclusion

CHAPTER 8

Conclusion

POLYNOMIAL-time preprocessing is a time-tested way of dealing with instances of hard problems. It has often been known to result in surprising gains in the speed with which the problem can be solved. The following quote from Alber et al.'s early work [1] on polynomial-time data reduction for DOMINATING SET portrays a notable example of such an occurrence, from Weihe's practical work [108, 109] on RED BLUE DOMINATING SET:

Weihe ... gave a striking example when dealing with the NP-complete RED BLUE DOMINATING SET problem appearing in the context of the European railroad network. In a preprocessing phase, he applied two simple data reduction rules again and again until no further application was possible. The impressive result of his empirical study was that each of his real-world instances was broken into very small pieces such that for each of these a simple brute-force approach was sufficient to solve the computationally hard problems efficiently and optimally.

In this thesis we saw some examples of polynomial-time preprocessing applied to different hard graph problems. In each case the efficiency of the method was analysed using the mathematical framework provided by the notion of *kernelization* from Parameterized Complexity Theory. We also proved some lower bounds on kernel size. In this concluding chapter we look back on the results obtained, and ponder on possible directions of future work.

We first focused our attention on the PARAMETERIZED DOMINATING SET problem. While this problem is known to be $W[2]$ -complete and thus unlikely to have kernels of any size on general graphs, we showed that the problem has polynomial kernels on larger classes of graphs than was known before our work. Specifically, we showed that for every fixed $j \geq i \geq 1$, the PARAMETERIZED DOMINATING SET problem restricted to graphs that do not have $K_{i,j}$ as a subgraph is FPT and has a polynomial kernel. We described a polynomial-time algorithm that, given a $K_{i,j}$ -free graph G and a non-negative integer k , constructs a graph H (the “kernel”) and an integer k' such that (1) G has a dominating set of size at most k if and only if H has a dominating set of size at most k' , (2) H has $\mathcal{O}((j+1)^{i+1}k^{i^2})$ vertices, and (3) $k' = \mathcal{O}((j+1)^{i+1}k^{i^2})$.

Since d -degenerate graphs do not have $K_{d+1,d+1}$ as a subgraph, this result directly implies a polynomial kernel on $\mathcal{O}((d+2)^{d+2}k^{(d+1)^2})$ vertices for the PARAMETERIZED DOMINATING SET problem on d -degenerate graphs, solving an open problem posed by Alon and Gutner [2, 65].

Dom et al. [39] have shown that PARAMETERIZED DOMINATING SET restricted to d -degenerate graphs has no kernel of size polynomial in *both* k and d unless the Polynomial Hierarchy collapses to the third level. This implies that our kernel size is nearly the best possible for the problem on this class of graphs.

The most general class of graphs for which a polynomial kernel was previously known for PARAMETERIZED DOMINATING SET is the class of K_h -topological-minor-free graphs [65]. Graphs of bounded degeneracy are the most general class of graphs for which an FPT algorithm was previously known for this problem [3]. K_h -topological-minor-free graphs are $K_{i,j}$ -free for suitable values of i, j (but not vice versa), and so our results show that PARAMETERIZED DOMINATING SET has both FPT algorithms and polynomial kernels on strictly more general classes of graphs.

Using the same techniques, we also obtained an $\mathcal{O}(jk^i)$ vertex-kernel for the PARAMETERIZED INDEPENDENT DOMINATING SET problem on $K_{i,j}$ -free graphs.

One interesting direction of future work is to try to demonstrate (no) kernels of size $f(d) \cdot k^c$ for the PARAMETERIZED DOMINATING SET problem on d -degenerate graphs, where c is independent of d . Note that the result of Dom et al. mentioned above does *not* suggest that such kernels are unlikely. Dell and van Melkebeek [34] have recently developed a lower-bound technique which allows them to show, *inter alia*, that the vertex deletion problem for any nontrivial hereditary graph class has no kernel of size $\mathcal{O}(k^{2-\epsilon})$ for any $\epsilon > 0$. A second interesting open problem is whether this new machinery can be extended to show that the PARAMETERIZED DOMINATING SET problem does not have kernels of size $f(d) \cdot k^c$ on d -degenerate graphs. Cygan et al. [28] have recently shown that PARAMETERIZED CONNECTED DOMINATING SET has no polynomial kernels on graphs of

degeneracy $d \geq 2$ unless the Polynomial Hierarchy collapses to the third level. A third set of interesting open problems consists of finding if the natural parameterized versions of other NP-hard variants of DOMINATING SET such as EFFICIENT DOMINATING SET, ANNOTATED DOMINATING SET, ROMAN DOMINATION, MAXIMUM MINIMAL DOMINATING SET, and many others* have polynomial kernels and/or FPT algorithms on d -degenerate graphs and beyond.

The second graph domination problem which we took up for study was the PARAMETERIZED CONNECTED DOMINATING SET problem. This problem is also known to be $W[2]$ -complete on general graphs. We investigated the effect of excluding short cycles, as subgraphs, on the kernelization complexity of PARAMETERIZED CONNECTED DOMINATING SET. It turned out that the PARAMETERIZED CONNECTED DOMINATING SET problem is hard on graphs with small cycles, and becomes progressively easier as the girth increases. More precisely, we obtained the following kernelization landscape: PARAMETERIZED CONNECTED DOMINATING SET

- does not have a kernel of *any* size on graphs of girth 3 or 4 (since the problem is $W[2]$ -hard);
- admits a kernel of size $2^k k^{3k}$ on graphs of girth at least 5;
- has *no* polynomial kernel (unless the Polynomial Hierarchy collapses to the third level) on graphs of girth at most 6, and,
- has a cubic ($\mathcal{O}(k^3)$) vertex kernel on graphs of girth at least 7.

The most important technical contribution in this work is the third result, which says that the PARAMETERIZED CONNECTED DOMINATING SET problem has no polynomial kernels on graphs with girth at most 6. On the way to proving this result we introduced a new problem, namely PARAMETERIZED FAIR CONNECTED COLOURS, and showed that this problem has no polynomial kernels. This intermediate result is interesting in its own right; we feel that it could be used to show kernelization lower bounds for other connectivity problems on graphs that exclude small cycles. Most known kernelization and FPT results for W -hard problems are for graph classes characterized by excluded *minors*. Our results add to the small but growing collection of such results for graph classes characterized by excluded *subgraphs*. One interesting direction of future work is to study how the exclusion of small cycles — or other small graphs — as subgraphs affects the kernelization complexity of other W -hard parameterized graph problems.

* Please see the Compendium [27] for the definitions of these and other related problems.

We now turned our attention to graph covering problems. In these the objective is to find a small set of vertices or edges of a graph whose removal deletes some structure from the graph. We first looked at the PARAMETERIZED PATHWIDTH-ONE VERTEX DELETION problem, in which the objective is to obtain a graph of pathwidth at most one by vertex deletion. We initiated the study of the parameterized complexity of this problem, parameterized by the solution size k . We showed that the problem has a quartic vertex-kernel: We showed that, given an input instance $(G = (V, E), k)$ where $|V| = n$, we can construct, in polynomial time, an instance (G', k') such that (i) (G, k) is a YES instance if and only if (G', k') is a YES instance, (ii) G' has $\mathcal{O}(k^4)$ vertices, and (iii) $k' \leq k$. We also derived an FPT algorithm for the problem that runs in $\mathcal{O}(7^k k \cdot n^2)$ time. These results are based on two different characterizations of graphs of pathwidth at most one: the first, as a collection of *caterpillars*, and the second, as the set of graphs which exclude the graphs K_3 and T_2 as minors.

Cygan et al. recently improved these bounds: they found an FPT algorithm which runs in $\mathcal{O}^*(4.65^k)$ time, and a kernel of size $\mathcal{O}(k^2)$ [29].

A challenging open problem is to try to solve the analogous problem for larger values of pathwidth. We know by results from graph minor theory that for any positive integer c , the Pathwidth c Vertex Deletion problem, defined analogously to PARAMETERIZED PATHWIDTH-ONE VERTEX DELETION, is FPT parameterized by the solution size. This is so because for each fixed c , the set of YES instances for this problem form a minor-closed class. However, for $c = 2$, the number of graphs in the obstruction set is already a hundred and ten [76], and so our approach would probably be of limited use for $c \geq 2$. Thus the interesting open problems for $c \geq 2$ are: (i) Can we get an $\mathcal{O}^*(d^k)$ FPT algorithm for the problem for some constant d , and (ii) Does the problem have a polynomial kernel?

The second graph covering problem which we looked at was the CONNECTED FEEDBACK VERTEX SET problem, where the question is whether the input graph has a small feedback vertex set which induces a connected subgraph. The related PARAMETERIZED FEEDBACK VERTEX SET problem is one of the best studied problems in parameterized complexity. In contrast, the PARAMETERIZED CONNECTED FEEDBACK VERTEX SET had not been investigated prior to our work, in which we derived the first FPT algorithms for the problem. We showed that the problem can be solved in $\mathcal{O}^*(46.2^k)$ time on general graphs, and in $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k} \log k)})$ time on graphs excluding a fixed graph H as a minor. These results imply that the problem has kernels of size 46.2^k on general graphs and $2^{\mathcal{O}(\sqrt{k} \log k)}$ on H -minor free graphs. We further showed that the problem is unlikely to have polynomial kernels on general graphs.

On the way to proving the FPT results for PARAMETERIZED CONNECTED FEEDBACK VERTEX SET, we establish that two variants of the well-studied PARAMETERIZED STEINER

TREE problem, namely PARAMETERIZED DIRECTED STEINER OUT-TREE and PARAMETERIZED GROUP STEINER TREE, are FPT when parameterized by the number t of terminals. We show that both these problems can be solved in $\mathcal{O}^*(2^t)$ time and polynomial space. We find these FPT algorithms to be of independent interest, and believe that it could be useful for obtaining parameterized algorithms for other connectivity problems.

When H is an apex graph, recent meta-results due to Fomin et al. [56] imply that the problem has a polynomial kernel on H -minor free graphs. Very recently, Cygan et al. [30] achieved a significant breakthrough in solving connectivity problems using which they show, *inter alia*, that both PARAMETERIZED FEEDBACK VERTEX SET and PARAMETERIZED CONNECTED FEEDBACK VERTEX SET can be solved in *randomized* $3^k n^{\mathcal{O}(1)}$ time.

Note that our FPT algorithm for the problem directly implies a polynomial-time approximation algorithm with approximation factor $n/c \log n$ for any constant c , where n is the number of vertices in the input graph. Somewhat surprisingly, this is the best polynomial-time approximation algorithm known for this problem. A very interesting open problem is thus to find a polynomial-time approximation algorithm for the CONNECTED FEEDBACK VERTEX SET problem with an approximation ratio asymptotically better than $n/\log n$, or to show that no such algorithm exists. It would also be interesting to find whether PARAMETERIZED CONNECTED FEEDBACK VERTEX SET admits a polynomial kernel on graphs excluding an arbitrary fixed graph H as a minor.

Finally, we looked at variants of two well-studied graph covering problems — VERTEX COVER and EDGE COVER — with additional “partial” connectivity constraints. Specifically, for each fixed $2 \leq t \leq k$ we imposed the additional requirement that each connected component of the graph induced by the solution have

- at least t vertices for VERTEX COVER, and called the problem t -TOTAL VERTEX COVER;
- at least t edges from the solution for EDGE COVER, and called the problem t -TOTAL EDGE COVER.

Both these problems are known to be NP-hard for each $2 \leq t \leq k$. We studied the parameterized complexity of these problems when the parameter is the solution size k . We showed that

- both problems remain fixed-parameter tractable with these restrictions, with running times of the form $\mathcal{O}^*(c^k)$ for some constant $c > 0$ in each case;
- for $2 \leq t \leq k$, t -TOTAL VERTEX COVER has no polynomial kernel unless the Polynomial Hierarchy collapses to the third level, and,

- for $2 \leq t \leq k$, t -TOTAL EDGE COVER has a linear vertex kernel of size $\frac{t+1}{t}k$.

These results significantly improve earlier work on these problems. Our no-poly-kernel result for t -TOTAL VERTEX COVER, and the known NP-hardness result for t -TOTAL EDGE COVER, are in stark contrast to the fact that VERTEX COVER has a $2k$ vertex kernel, and that EDGE COVER is solvable in polynomial time. These illustrate how even the slightest connectivity requirement results in a drastic change in the tractability of problems.

Our algorithm for the PARAMETERIZED t -TOTAL VERTEX COVER problem runs in $\mathcal{O}^*(16.1^k)$ time. For the three special values $t = 1$, $t = 2$ and $t = k$, the PARAMETERIZED t -TOTAL VERTEX COVER problem is known to be solvable in $\mathcal{O}^*(c^k)$ time for have much smaller values of c , namely $c = 1.2738$, $c = 2.3655$ and $c = 2.7606$, respectively [25, 50, 91]. One interesting open problem is to find whether the value of c for the PARAMETERIZED t -TOTAL VERTEX COVER problem can be brought closer to these smaller values. Another, more general direction of future research would be to examine the effect of partial connectivity constraints on other parameterized graph covering problems.



Bibliography

1. Jochen Alber, Michael R. Fellows, and Rolf Niedermeier. Polynomial-time data reduction for Dominating Set. *Journal of the ACM*, 51(3):363–384, 2004.
2. Noga Alon and Shai Gutner. Kernels for the Dominating Set Problem on Graphs with an Excluded Minor. Technical Report TR08-066, The Electronic Colloquium on Computational Complexity (ECCC), 2008. URL <http://eccc.hpi-web.de/eccc-reports/2008/TR08-066/index.html>. A revised version [65] has been published in the proceedings of IWPEC 2009.
3. Noga Alon and Shai Gutner. Linear time algorithms for finding a dominating set of fixed size in degenerated graphs. *Algorithmica*, 54(4):544–556, 2009.
4. Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995.
5. Noga Alon, Lajos Rónyai, and Tibor Szabó. Norm-graphs: Variations and applications. *Journal of Combinatorial Theory, Series B*, 76(2):280–290, 1999.
6. Carme Álvarez and Maria J. Serna. The Proper Interval Colored Graph problem for caterpillar trees. *Electronic Notes in Discrete Mathematics*, 17:23 – 28, 2004.
7. Omed Amini, Fedor V. Fomin, and Saket Saurabh. Counting subgraphs via homomorphisms. In *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, volume 5555 of LNCS, pages 71–82. Springer, 2009.

8. Omed Amini, Fedor V. Fomin, and Saket Saurabh. Counting subgraphs via homomorphisms. Full version manuscript, available online at <http://www.ii.uib.no/~fomin/articles1/si.pdf>, 2009.
9. Stefan Arnborg, Andrzej Proskurowski, and Detlef Seese. Monadic Second Order Logic, Tree Automata and Forbidden Minors. In *Computer Science Logic, 4th Workshop, CSL '90, Heidelberg, Germany, October 1-5, 1990, Proceedings*, volume 533 of LNCS, pages 1–16. Springer, 1990.
10. S. F. Assmann, G. W. Peck, Maciej M. Sysło, and J. Zak. The bandwidth of caterpillars with hairs of length 1 and 2. *SIAM Journal on Algebraic and Discrete Methods*, 2(4): 387–393, 1981.
11. Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and approximation — Combinatorial optimization problems and their approximability properties*. Springer-Verlag, 1999.
12. Terry Beyer and Sandra Mitchell Hedetniemi. Constant time generation of rooted trees. *SIAM Journal on Computing*, 9(4):706–712, 1980.
13. Hans L. Bodlaender. On disjoint cycles. *International Journal of Foundations of Computer Science*, 5(1):59–68, 1994.
14. Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.
15. Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009.
16. Hans L. Bodlaender, Corinne Feremans, Alexander Grigoriev, Eelko Penninkx, René Sitters, and Thomas Wolle. On the minimum corridor connection problem and other generalized geometric problems. *Computational Geometry*, 42(9):939–951, 2009.
17. Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) Kernelization. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 629–638. IEEE, 2009.

18. Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel Bounds for Disjoint Cycles and Disjoint Paths. In *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of LNCS, pages 635–646, 2009.
19. Béla Bollobás. *Extremal graph theory*. Dover Publications, 2004.
20. Béla Bollobás and Andrew Thomason. Proof of a conjecture of Mader, Erdős and Hajnal on topological complete subgraphs. *European Journal of Combinatorics*, 19(8):883–887, 1998.
21. R. L. Bryant, Nancy G. Kinnarsley, Michael R. Fellows, and Michael A. Langston. On Finding Obstruction Sets and Polynomial-Time Algorithms for Gate Matrix Layout. In *Proceedings of the 25th Allerton Conference on Communication, Control and Computing*, pages 397–398, 1987.
22. Yixin Cao, Jianer Chen, and Yang Liu. On Feedback Vertex Set: New Measure and New Structures. In Haim Kaplan, editor, *Algorithm Theory - SWAT 2010, 12th Scandinavian Symposium and Workshops on Algorithm Theory, Bergen, Norway, June 21-23, 2010*, volume 6139 of LNCS, pages 93–104. Springer, 2010.
23. Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex Cover: Further observations and further improvements. *Journal of Algorithms*, 41(2):280–301, 2001.
24. Jianer Chen, Henning Fernau, Iyad A. Kanj, and Ge Xia. Parametric Duality and Kernelization: Lower Bounds and Upper Bounds on Kernel Size. *SIAM Journal on Computing*, 37(4):1077–1106, 2007.
25. Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010.
26. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, second edition, 2001.
27. Pierluigi Crescenzi, Viggo Kann, Magnús M. Halldórsson, Marek Karpinski, and Gerhard Woeginger. A compendium of NP optimization problems, 2000. URL <http://www.nada.kth.se/~viggo/wwwcompendium/wwwcompendium.html>.
28. Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. Kernelization Hardness of Connectivity Problems in d -Degenerate Graphs. In Dimitrios M. Thilikos, editor, *Graph Theoretic Concepts in Computer Science - 36th*

- International Workshop, WG 2010, Zarós, Crete, Greece, June 28-30, 2010. Revised Papers*, volume 6410 of LNCS, pages 147–158, 2010.
29. Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. An improved FPT algorithm and quadratic kernel for Pathwidth One Vertex Deletion. In Venkatesh Raman and Saket Saurabh, editors, *Parameterized and Exact Computation - 5th International Symposium, IPEC 2010, Chennai, India, December 13-15, 2010. Proceedings*, volume 6478 of LNCS, pages 95–106, 2010.
 30. Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. Accepted at the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2011), 2011.
 31. Anuj Dawar and Stephan Kreutzer. Domination problems in nowhere-dense classes. In Ravi Kannan and K Narayan Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2009, December 15-17, 2009, IIT Kanpur, India*, volume 4 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 157–168, Dagstuhl, Germany, 2009. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
 32. Koen M. J. De Bontridder, Bjarne V. Halldórsson, Magnús M. Halldórsson, Cor A. J. Hurkens, Jan Karel Lenstra, R. Ravi, and Leen Stougie. Approximation algorithms for the Test Cover problem. *Mathematical Programming, Series B*, 98:477–491, 2003.
 33. Frank Dehne, Michael R. Fellows, Michael A. Langston, Frances A. Rosamond, and Kim Stevens. An $\mathcal{O}(2^{\mathcal{O}(k)} n^3)$ FPT-Algorithm for the Undirected Feedback Vertex Set problem. *Theory of Computing Systems*, 41(3):479–492, 2007.
 34. Holger Dell and Dieter van Melkebeek. Satisfiability Allows No Nontrivial Sparsification Unless The Polynomial-Time Hierarchy Collapses. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 251–260. ACM, 2010.
 35. Eric D. Demaine and Mohammadtaghi Hajiaghayi. Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica*, 28(1):19–36, 2008.
 36. Eric D. Demaine, Mohammadtaghi Hajiaghayi, and Ken-ichi Kawarabayashi. Algorithmic graph minor theory: Decomposition, approximation, and coloring. In

- 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005)*, 23-25 October 2005, Pittsburgh, PA, USA, *Proceedings*, pages 637–646. IEEE Computer Society, 2005.
37. Reinhard Diestel. *Graph Theory*. Springer-Verlag, Heidelberg, third edition, 2005.
 38. Bolin Ding, Jeffrey Xu Yu, Shan Wang, Lu Qin, Xiao Zhang, and Xuemin Lin. Finding top-k min-cost connected trees in databases. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, April 15-20, 2007, Istanbul, Turkey*, pages 836–845, 2007.
 39. Michael Dom, Daniel Lokshantov, and Saket Saurabh. Incompressibility through Colors and IDs. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikolettseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, volume 5555 of LNCS, pages 378–389. Springer, 2009.
 40. Rodney G. Downey and Michael R. Fellows. Fixed Parameter Tractability and Completeness. In *Complexity Theory: Current Research*, pages 191–225. Cambridge University Press, 1992.
 41. Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999.
 42. Rodney G. Downey, Michael R. Fellows, Alexander Vardy, and Geoff Whittle. The parametrized complexity of some fundamental problems in coding theory. *SIAM Journal on Computing*, 29(2):545–570, 1999.
 43. S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972.
 44. Gideon Ehrlich. Loopless algorithms for generating permutations, combinations, and other combinatorial configurations. *Journal of the ACM*, 20(3):500–513, 1973.
 45. John A. Ellis, Hongbing Fan, and Michael R. Fellows. The Dominating Set problem is fixed parameter tractable for graphs of bounded genus. *Journal of Algorithms*, 52(2):152–168, 2004.
 46. Luis Fanjul-Peyro and Rubén Ruiz. Size-reduction heuristics for the unrelated parallel machines scheduling problem. *Computers & Operations Research*, 38(1):301–309, 2011.

47. Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee. Improved approximation algorithms for minimum-weight vertex separators. *SIAM Journal on Computing*, 38(2):629–657, 2008.
48. Michael R. Fellows and Michael A. Langston. On Search, Decision and the Efficiency of Polynomial-time Algorithms. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 501–512. ACM Press, 1989.
49. Henning Fernau. *Parameterized Algorithmics: A Graph-Theoretic Approach*. Habilitationsschrift, Universität Tübingen, Germany, 2005.
50. Henning Fernau and David F. Manlove. Vertex and edge covers with clustering properties: Complexity and algorithms. *Journal of Discrete Algorithms*, 7:149–167, 2009.
51. Henning Fernau, Fedor V. Fomin, Geevarghese Philip, and Saket Saurabh. The Curse of Connectivity: t -Total Vertex(Edge) Cover. In My T. Thai and Sartaj Sahni, editors, *Computing and Combinatorics, 16th Annual International Conference, COCOON 2010, Nha Trang, Vietnam, July 19-21, 2010. Proceedings*, volume 6196 of LNCS, pages 34–43. Springer, 2010.
52. Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006.
53. Fedor V. Fomin and Dimitrios M. Thilikos. Fast Parameterized Algorithms for Graphs on Surfaces: Linear Kernel and Exponential Speed-Up. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*, volume 3142 of LNCS, pages 581–592. Springer, 2004.
54. Fedor V. Fomin and Dimitrios M. Thilikos. Dominating Sets in Planar Graphs: Branch-Width and Exponential Speed-Up. *SIAM Journal on Computing*, 36(2): 281–309, 2006.
55. Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. Solving connected dominating set faster than 2^n . *Algorithmica*, 52(2):153–166, 2008.
56. Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and Kernels. In Moses Charikar, editor, *Proceedings of the Twenty-*

First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010, pages 503–510, 2010.

57. Gianni Franceschini, Fabrizio Luccio, and Linda Pagli. Dense trees: a new look at degenerate graphs. *Journal of Discrete Algorithms*, 4:455–474, 2006.
58. Michael R. Garey and David S. Johnson. The Rectilinear Steiner Tree Problem is NP-Complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.
59. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.
60. Petr A. Golovach and Yngve Villanger. Parameterized Complexity for Domination Problems on Degenerate Graphs. In Hajo Broersma, Thomas Erlebach, Tom Friedetzky, and Daniël Paulusma, editors, *Graph-Theoretic Concepts in Computer Science, 34th International Workshop, WG 2008, Durham, UK, June 30 - July 2, 2008*, volume 5344 of LNCS, 2008.
61. Alexander Grigoriev and René Sitters. Connected feedback vertex set in planar graphs. In Christophe Paul and Michel Habib, editors, *Graph-Theoretic Concepts in Computer Science, 35th International Workshop, WG 2009, Montpellier, France, June 24-26, 2009. Revised Papers*, volume 5911 of LNCS, pages 143–153, 2009.
62. Qianping Gu and Navid Imani. Connectivity Is Not a Limit for Kernelization: Planar Connected Dominating Set. In López-Ortiz Alejandro, editor, *LATIN 2010: Theoretical Informatics, 9th Latin American Symposium, Oaxaca, Mexico, April 19-23, 2010. Proceedings*, volume 6034 of LNCS, pages 26–37, 2010.
63. Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, 38(1):31–45, 2007.
64. Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *Journal of Computer and System Sciences*, 72(8):1386–1396, 2006.
65. Shai Gutner. Polynomial kernels and faster algorithms for the dominating set problem on graphs with an excluded minor. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation: 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, volume 5917 of LNCS, pages 246–257, Berlin, Heidelberg, 2009. Springer-Verlag.

66. Michel Habib, Christophe Paul, and Laurent Viennot. A Synthesis on Partition Refinement: A Useful Routine for Strings, Graphs, Boolean Matrices and Automata. In Michel Morvan, Christoph Meinel, and Daniel Krob, editors, *STACS 98, 15th Annual Symposium on Theoretical Aspects of Computer Science, Paris, France, February 25-27, 1998, Proceedings*, volume 1373 of LNCS, pages 25–38. Springer, 1998.
67. Godfrey H. Hardy and Srinivasa Ramanujan. Asymptotic formulae in combinatory analysis. In *Proceedings of the London Mathematical Society*, volume 17, 1918.
68. Teresa W. Haynes, Stephen T. Hedetniemi, and Peter J. Slater. *Domination in graphs: advanced topics*, volume 209 of *Pure and Applied Mathematics : A Series of Monographs and Textbooks*. Marcel Dekker, Inc, 1998.
69. Teresa W. Haynes, Stephen T. Hedetniemi, and Peter J. Slater. *Fundamentals of Domination in Graphs*, volume 208 of *Pure and Applied Mathematics : A Series of Monographs and Textbooks*. CRC Press, 1998.
70. Pavoll Hell and Jaroslav Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.
71. Danny Hermelin, Matthias Mnich, Erik Jan van Leeuwen, and Gerhard J. Woeginger. Domination when the stars are out. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, volume 6755 of LNCS, pages 462–473. Springer, 2011.
72. Raymond R. Hill, Yong Kun Cho, and James T. Moore. Problem reduction heuristic for the 0-1 multidimensional knapsack problem. *Computers & Operations Research*, 39(1):19 – 26, 2012. Special Issue on Knapsack Problems and Applications.
73. Stasys Jukna. *Extremal Combinatorics: With Applications in Computer Science*. Texts in Theoretical Computer Science. an Eatcs. Springer, 2011.
74. Iyad A. Kanj, Michael J. Pelsmayer, and Marcus Schaefer. Parameterized Algorithms for Feedback Vertex Set. In Rodney G. Downey, Michael R. Fellows, and Frank K. H. A. Dehne, editors, *Parameterized and Exact Computation, First International Workshop, IWPEC 2004, Bergen, Norway, September 14-17, 2004, Proceedings*, volume 3162 of LNCS, pages 235–247. Springer, 2004.
75. Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Communications*, pages 85–103, 1972.

76. Nancy G. Kinnersley and Michael A. Langston. Obstruction Set Isolation for the Gate Matrix Layout problem. *Discrete Applied Mathematics*, 54(2-3):169–213, 1994.
77. David G. Kirkpatrick and Pavol Hell. On the completeness of a generalized matching problem. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 240–245. ACM, 1978.
78. Ton Kloks. *Treewidth – computations and approximations*, volume 842 of LNCS. Springer-Verlag, 1994.
79. Donald E. Knuth. *The Art of Computer Programming*. Addison-Wesley, third edition, 1998. Vol. 3: Seminumerical Algorithms.
80. János Komlós and Endre Szemerédi. Topological cliques in graphs II. *Combinatorics, Probability and Computing*, 5:79–90, 1996.
81. Adrian Kosowski, Michał Małafiejski, and Pavel Žyliński. Parallel processing subsystems with redundancy in a distributed environment. In Roman Wyrzykowski, Jack Dongarra, Norbert Meyer, and Jerzy Wasniewski, editors, *Parallel Processing and Applied Mathematics, 6th International Conference, PPAM 2005, Poznan, Poland, September 11-14, 2005, Revised Selected Papers*, volume 3911 of LNCS, pages 1002–1009. Springer, 2006.
82. Mingen Lin, Zhiyong Lin, and Jinhui Xu. Graph bandwidth of weighted caterpillars. *Theoretical Computer Science*, 363(3):266 – 277, 2006.
83. Daniel Lokshantov, Matthias Mnich, and Saket Saurabh. Linear Kernel for Planar Connected Dominating Set. *Theoretical Computer Science*, 412(23):2536–2543, 2011.
84. Flaminia L. Luccio. Almost exact minimum feedback vertex set in meshes and butterflies. *Information Processing Letters*, 66(2):59–64, 1998.
85. Weizhong Luo, Jianxin Wang, Qilong Feng, Jiong Guo, and Jianer Chen. An improved kernel for planar connected dominating set. In Mitsunori Ogihara and Jun Tarui, editors, *Theory and Applications of Models of Computation*, volume 6648 of LNCS, pages 70–81. Springer Berlin / Heidelberg, 2011.
86. John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219 – 230, 1980.

87. Michał Małafiejski and Pavel Žyliński. Weakly cooperative guards in grids. In Osvaldo Gervasi, Marina L. Gavrilova, Vipin Kumar, Antonio Laganà, Heow Pueh Lee, Youngsong Mun, David Taniar, and Chih Jeng Kenneth Tan, editors, *Computational Science and Its Applications - ICCSA 2005, International Conference, Singapore, May 9-12, 2005, Proceedings, Part I*, volume 3480 of *LNCS*, pages 647–656. Springer, 2005.
88. Silvio Micali and Vijay V. Vazirani. An $\mathcal{O}(\sqrt{|V||E|})$ Algorithm for Finding Maximum Matching in General Graphs. In *21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, 13-15 October 1980*, pages 17–27. IEEE Computer Society, 1980.
89. Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, and Saket Saurabh. The effect of girth on the kernelization complexity of Connected Dominating Set. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, volume 8 of *LIPICs*, pages 96–107. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.
90. Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, Saket Saurabh, and Somnath Sikdar. FPT algorithms for connected feedback vertex set. *Journal of Combinatorial Optimization*, 2011. URL <http://dx.doi.org/10.1007/s10878-011-9394-2>. Published online. A preliminary version was published in Md. Saidur Rahman and Satoshi Fujita, editors, *WALCOM: Algorithms and Computation, 4th International Workshop, WALCOM 2010, Dhaka, Bangladesh, February 10-12, 2010. Proceedings, LNCS volume 5942*, pages 269–280, 2010.
91. Daniel Mölle, Stefan Richter, and Peter Rossmanith. Enumerate and expand: Improved algorithms for connected vertex cover and tree cover. *Theory of Computing Systems*, 43(2):234–253, 2008.
92. Hannes Moser. Exact algorithms for generalizations of vertex cover. Master's thesis, Institut für Informatik, Friedrich-Schiller-Universität, 2005.
93. Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23-25 October 1995*, pages 182–193, Los Alamitos, 1995. IEEE Computer Society Press.
94. Jesper Nederlof. Fast polynomial-space algorithms using möbius inversion: Improving on steiner tree and related problems. In Susanne Albers, Alberto Marchetti-

- Spaccamela, Yossi Matias, Sotiris E. Nikolettseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, volume 5555 of LNCS, pages 713–725. Springer, 2009.
95. Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2006. ISBN 978-0-19-856607-6; 0-19-856607-7.
 96. Robert Z. Norman and Michael O. Rabin. An algorithm for a minimum cover of a graph. *Proceedings of the American Mathematical Society*, 10:315–319, 1959.
 97. Richard Otter. The number of trees. *Annals of Mathematics*, 49(3):583–599, 1948.
 98. Christos H. Papadimitriou. The NP-Completeness of the bandwidth minimization problem. *Computing*, 16(3):263–270, 1976.
 99. Geevarghese Philip, Venkatesh Raman, and Somnath Sikdar. Polynomial kernels for Dominating Set in graphs of bounded degeneracy and beyond. Accepted for publication in the journal *ACM Transactions on Algorithms*. A preliminary version was published with the title *Solving Dominating Set in Larger Classes of Graphs: FPT Algorithms and Polynomial Kernels* in Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, LNCS volume 5757, pages 694–705, 2009.
 100. Geevarghese Philip, Venkatesh Raman, and Yngve Villanger. A Quartic Kernel for Pathwidth-One Vertex Deletion. In Dimitrios M. Thilikos, editor, *Graph Theoretic Concepts in Computer Science - 36th International Workshop, WG 2010, Zarós, Crete, Greece, June 28-30, 2010. Revised Papers*, volume 6410 of LNCS, pages 196–207, 2010.
 101. Willard V. Quine. The problem of simplifying truth functions. *The American Mathematical Monthly*, 59(8):521–531, 1952.
 102. Venkatesh Raman and Saket Saurabh. Short Cycles Make W-hard Problems Hard: FPT Algorithms for W-hard Problems in Graphs with no Short Cycles. *Algorithmica*, 52(2):203–225, 2008.
 103. Venkatesh Raman, Saket Saurabh, and C.R. Subramanian. Faster fixed parameter tractable algorithms for finding feedback vertex sets. *ACM Transactions on Algorithms*, 2(3):403–415, 2006.

104. Neil Robertson and Paul D. Seymour. Graph minors I. Excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1):39–61, 1983.
105. Neil Robertson and Paul D. Seymour. Graph Minors. II. Algorithmic Aspects of Tree-Width. *Journal of Algorithms*, 7(3):309–322, 1986.
106. Horst Sachs. Regular graphs with given girth and restricted circuits. *Journal of the London Mathematical Society*, s1-38(1):423–429, 1963.
107. Stéphan Thomassé. A quadratic kernel for feedback vertex set. In Claire Mathieu, editor, *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 115–119. Society for Industrial and Applied Mathematics, 2009.
108. Karsten Weihe. Covering trains by stations or the power of data reduction. In Roberto Battiti and Alan Bertossi, editors, *Algorithms and Experiments (ALEX 1998)*, pages 1–8, 1998. URL <http://rtm.science.unitn.it/alex98/proceedings.html>.
109. Karsten Weihe. On the differences between “practical” and “applied”. In Stefan Näher and Dorothea Wagner, editors, *Algorithm Engineering, 4th International Workshop, WAE 2000, Saarbrücken, Germany, September 5-8, 2000, Proceedings*, volume 1982 of LNCS, pages 1–10. Springer, 2000.
110. Howard J. Weiss and J. Yael Assous. Reduction in problem size for ranking alternatives in group decision-making. *Computers & Operations Research*, 14(1):55 – 65, 1987.
111. Takeo Yamada and Mayumi Futakawa. Heuristic and reduction algorithms for the knapsack sharing problem. *Computers & Operations Research*, 24(10):961 – 967, 1997.
112. A. Zoghbi and I. Stojmenovic. Fast algorithms for generating integer partitions. *International Journal of Computer Mathematics*, 70:319–332, 1998.

Colophon

This thesis was typeset with $\text{\LaTeX 2}_{\epsilon}$ and [TeX Live](#) on an Acer Aspire 4736 notebook running [Arch Linux](#). It was typed in using [GNU Emacs](#) and [AUCTeX](#), and compiled into PDF using [pdfTeX](#) invoked as `pdflatex`. The figures were drawn and exported to PDF using [Inkscape](#).

The main content is typeset using the [Minion Pro](#) font designed by Robert Slimbach for Adobe Systems. Its format follows the style of [Lev S. Bishop's doctoral thesis](#), whose [\$\text{\LaTeX}\$ sources](#) he has generously shared online. The front matter format is based on the one used in André Miede's [classicthesis](#) package.