KERNELS FOR THE $\mathcal{F}$-DELETION PROBLEM

By

Neeldhara Misra

The Institute of Mathematical Sciences, Chennai.

A thesis submitted to the
Board of Studies in Mathematical Sciences

In partial fulfillment of the requirements

For the Degree of

DOCTOR OF PHILOSOPHY

of

HOMI BHABHA NATIONAL INSTITUTE

September 2011

Homi Bhabha National Institute

Recommendations of the Viva Voce Board

As members of the Viva Voce Board, we recommend that the dissertation prepared by **Neeldhara Misra** entitled "Kernels for the $\mathcal{F}$-Deletion Problem" may be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

---------- **Date :**
Chairman : Chair of committe

---------- **Date :**
Convener : Conv of Committe

---------- **Date :**
Member : Member 1 of committe

---------- **Date :**
Member : Member 2 of committe

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to HBNI.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it may be accepted as fulfilling the dissertation requirement.

---------- **Date :**
Guide : Venkatesh Raman

DECLARATION

I, hereby declare that the investigation presented in the thesis has been carried out by me. The work is original and the work has not been submitted earlier as a whole or in part for a degree/diploma at this or any other Institution or University.

Neeldhara Misra

ACKNOWLEDGEMENTS

I have been extremely fortunate in having enjoyed the guidance and care of my advisor, Prof. Venkatesh Raman. His presence has been a source of inspiration and strength like no other, and every opportunity of interaction has been a treasured learning experience. More importantly than leading me up to successes, he found me the motivation that I needed to last me through the inevitably larger number of failures. I will never cease be amazed at his expertise and teaching, and my own stroke of luck for having had the chance to witness it from close quarters.

I am also deeply obliged to Prof. Saket Saurabh for his relentless efforts in advancing my comprehension of a number of things, technical and non-technical alike. He has always been incredibly accessible and it is thanks to his infinite patience that I eventually got around to messing with things that I would otherwise keep at a safe distance. I also learned from him to value the pursuit as much as, and irrespective of, the associated discovery. The energy and intensity that he brings to research is nothing short of magical, and the experience has been a privilege.

Prof. Mike Fellows and Frances Rosamond have my eternal gratitude for taking care of me at various stages, and ensuring that I have nothing less than the time of my life! They have been very generous in their enthusiastic support, and I have certainly received from them much more than I deserve.

I would like to thank Prof. Fedor Fomin, whom I had an opportunity to visit at the University of Bergen — the stay was intensely enriching. He was very generous in with sharing his profound experience, and the discussions were delightful. Special thanks are due to Prof. Pinar Heggernes, Pim van't Hof, Daniel Marx, and Yngve Villanger for a particularly memorable collaboration. I learned a good deal from Daniel Lokshtanov, and I am thankful for his time and valuable insights.

Thanks to Somnath Sikdar, Geevarghese Philip, and M. S. Ramanujan, the typical day was always something to look forward to!

I would like to thank Abhimanyu M. Ambalath, Radheshyam Balasundaram, Chintan Rao H., Venkata Koppula, Matthias Mnich, N. S. Narayanaswamy, and Bal Sri Shankar, for many exciting discussions. I also enjoyed very much a number of discussions with Prof. Ronojoy Adhikari - thank you!

I will be forever grateful to all the faculty at the Department of Theoretical Computer Science. Any understanding that I might have developed about Computer Science certainly is thanks to their patient, creative, and inspiring teaching. I am equally thankful to all my friends and colleagues for being tirelessly supportive.

The work leading to this thesis executed at IMSc and the University of Bergen. I would like to thank both institutions for the financial support, excellent infrastructure and wonderful research environments. Special thanks to the administrative staff, who have always been especially helpful.

My parents happen to be my life, universe, and everything. For chasing my dreams with me, in true no-matter-what style, thanks very much indeed. I will add that more can be said that goes without saying, and is therefore left unsaid.

Abstract

In this thesis, we use the parameterized framework for the design and analysis of algorithms for NP-complete problems. This amounts to studying the parameterized version of the classical decision version. Herein, the classical language appended with a secondary measure called a "parameter". The central notion in parameterized complexity is that of *fixed-parameter tractability*, which means given an instance (x, k) of a parameterized language L, deciding whether $(x, k) \in L$ in time $f(k) \cdot p(|x|)$, where f is an arbitrary function of k alone and p is a polynomial function. The notion of kernelization formalizes preprocessing or data reduction, and refers to polynomial time algorithms that transform any given input into an equivalent instance whose size is bounded as a function of the parameter alone.

The center of our attention in this thesis is the $\mathcal{F}$-Deletion problem, a vastly general question that encompasses many fundamental optimization problems as special cases. In particular, we provide evidence supporting a conjecture about the kernelization complexity of the problem, and this work branches off in a number of directions, leading to results of independent interest. We also study the Colorful Motifs problem, a well-known question that arises frequently in practice. Our investigation demonstrates the hardness of the problem even when restricted to very simple graph classes.

The $\mathcal{F}$-Deletion Problem Let $\mathcal{F}$ be a finite family of graphs. The $\mathcal{F}$-Deletion problem takes as input a graph G on n vertices, and a positive integer k. The question is whether it is possible to delete at most k vertices from G such that the remaining graph contains no graph from $\mathcal{F}$ as a minor. This question encompasses fundamental problems such as Vertex Cover (consider $\mathcal{F}$ consisting of the graph with two vertices and one edge) or Feedback Vertex Set (the set $\mathcal{F}$ consists of a cycle with three vertices). A number of other deletion-based optimization problems also turn out to be special cases of Planar $\mathcal{F}$-Deletion, for instance: Diamond Hitting Set, Outerplanar Deletion Set, Pathwidth r-Deletion Set and Treewidth r-Deletion Set.

The general $\mathcal{F}$-Deletion problem is NP-complete. From the parameterized perspective, by one of the most well-known consequences of the celebrated Graph Minor theory of Robertson and Seymour, the $\mathcal{F}$-Deletion problem is fixed-

parameter tractable for every finite set of forbidden minors. It is conjectured that the $\mathcal{F}$-Deletion problem admits a polynomial kernel if, and only if, $\mathcal{F}$ contains a planar graph. We refer to the $\mathcal{F}$-Deletion question when restricted to the case when $\mathcal{F}$ contains a planar graph as the Planar $\mathcal{F}$-Deletion problem.

We obtain a number of kernelization results for the Planar $\mathcal{F}$-Deletion problem. We give a linear vertex kernel on graphs excluding t-claw $K_{1,t}$, the star with t leves, as an induced subgraph, where t is a fixed integer, and a obtain polynomial kernels for the case when $\mathcal{F}$ contains graph θ_c as a minor for a fixed integer c. The graph θ_c consists of two vertices connected by c parallel edges. Even though this may appear to be a very restricted class of problems it already encompasses well-studied problems such as Vertex Cover, Feedback Vertex Set and Diamond Hitting Set. The generic kernelization algorithm is based on a non-trivial application of protrusion techniques, previously used only for problems on topological graph classes. We also demonstrate an approximation algorithm achieving an approximation ratio of $O(\log^{3/2} OPT)$, where OPT is the size of an optimal solution on general undirected graphs. The approximation algorithm is used crucially in the kernelization routines.

We also show an algorithm for Planar $\mathcal{F}$-Deletion that runs in time $2^{O(k \log k)} n^2$, improving substantially from what was known previously, $2^{2^{O(k \log k)}} n^{O(1)}$. Since Planar $\mathcal{F}$-Deletion is a generalization of Vertex Cover, we have that it cannot be solved in time $2^{o(k)} n^{O(1)}$, unless Exponential Time Hypothesis fails, so our algorithm is also quite close to being optimal. This algorithm uses the technique of iterative compression, and the instance encountered at every iteration is subject to kernelization. The intermediate kernelization algorithm is quite non-trivial and requires reformulating some of the fundamental machinery that was used in all the other situations. Also, towards the kernel, we show a "decomposition lemma" that asserts that any graph of constant treewidth can essentially be partitioned into a number of protrusions. We believe this result to be of independent interest.

We also consider the complementary "packing" question: we wish to maximize the number of vertex (or edge) disjoint minor models of graphs in $\mathcal{F}$. We show that the packing number is closely related to the size of the optimal $\mathcal{F}$-hitting set in the special case when $\mathcal{F} = \{\theta_c\}$. Independently, but on a related note, we show two lower bounds, demonstrating that the problem of finding if there are at least k *vertex-disjoint* minor models of θ_c is unlikely to admit a polynomial kernel

parameterized by k, and this is also true for the problem of checking if there are at least k *vertex-disjoint* cycles of odd length (again parameterized by k).

Colorful Motifs We study the problem of Colorful Motifs on various graph classes. We prove that the problem of Colorful Motifs restricted to superstars is NP-complete. Further, we show NP-completeness on graphs of diameter two. We apply this result towards settling the classical complexity of Connected Dominating Set on graphs of diameter two — specifically, we show that it is NP-complete. Further, we show that on graphs of diameter two, the problem is NP-complete *and* is unlikely to admit a polynomial kernel.

Next, we show that obtaining polynomial kernels for Colorful Motifs on comb graphs is infeasible, but we show the existence of n polynomial kernels. Further, we study the problem of Colorful Motifs on trees, where we observe that the natural strategies for many polynomial kernels are not successful. For instance, we show that "guessing" a root vertex, which helped in the case of comb graphs, fails as a strategy because the Rooted Colorful Motif problem has no polynomial kernels on trees.

Contents

List of Figures

List of Tables

Every great and deep difficulty bears in itself its own solution.
It forces us to change our thinking in order to find it.

Niels Bohr

"The time has come," the walrus said, "to talk of many things:
Of shoes and ships - and sealing wax - of cabbages and kings."

Lewis Carroll

One of the greatest achievements of theoretical computer science is the development of the theory of NP-completeness. This theory offers an explanation as to why some problems are unlikely to admit fast algorithms. It also provides a solid and convincing foundation for the classification of computationally hard problems. Unfortunately, most problems that arise in practical applications turn out to be intractable. Thus, a significant part of research in theoretical computer science is devoted to finding algorithms with a *provably* good running time and solution quality. We do not expect to achieve these apparently contradictory goals simultaneously: hence, in real-life situations one often resorts to *heuristics* that abandon one or both these demands.

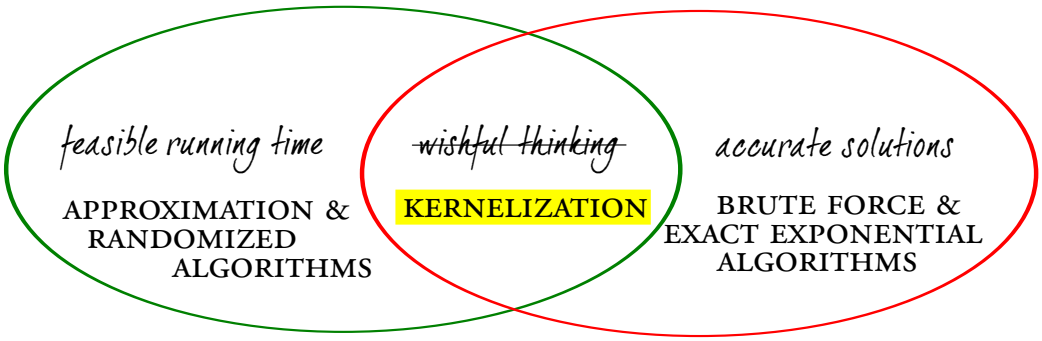


Figure 1.1: Algorithms for computationally intractable problems

Our ability of formally comparing heuristic measures and evaluating them (both in absolute and relative terms), depends on developing well-defined measures of efficiency and extent of simplification. For a long time, the mathematical analysis of

polynomial-time preprocessing algorithms was neglected. The basic reason for this anomalous development was the following: if we start with an instance I of an NP-hard problem and seek an efficient (polynomial-time) subroutine that replaces I with an equivalent instance I' such that $|I'| < |I|$ then we can use the said subroutine to actually *solve* the problem in polynomial time, which in turn would imply $P = NP$ — discouraging efforts in this research direction. The situation has changed drastically with advent of parameterized complexity where the issue of replacing an instance of a *parameterized* problem by an equivalent one of smaller size can be framed in a natural manner.

1.1 Parameterized Complexity and Kernelization

It is almost always the case that in addition to the overall input size (the sole "dimension" considered in the classical NP-completeness theory) there is a secondary measurement that crucially affects the computational complexity of real-world problems. The basic idea of parameterized complexity is to capitalize on these additional measures and incorporate them into the design and analysis of algorithms. The theory introduces a formal secondary measurement known as the *parameter* to capture the secondary measure. The parameter can represent an aggregate of bounds (e.g., the number of sequences *and* the alphabet size *and* the approximation performance). It is in some sense a natural two-dimensional sequel to the classical theory of NP-completeness.

More precisely, the subject deals with the study of parameterized problems. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet and $\mathbb{N}$ is the set of non-negative integers. The second component is called the *parameter* of the problem. The central notion in parameterized complexity is that of *fixed-parameter tractability*, which means given an instance (x, k) of a parameterized language L, deciding whether $(x, k) \in L$ in time $f(k) \cdot p(|x|)$, where f is an arbitrary function of k alone and p is a polynomial function. Such an algorithm is called a fixed-parameter tractable algorithm and we call a problem that admits an algorithm of this kind fixed-parameter tractable (FPT).

We now turn to the formal notion that captures the notion of simplification, which is what most heuristics do when applied to a problem. A *data reduction rule* for a parameterized language L is a function $\phi : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ that maps an instance (x, k) of L to an equivalent instance (x', k') of L such that

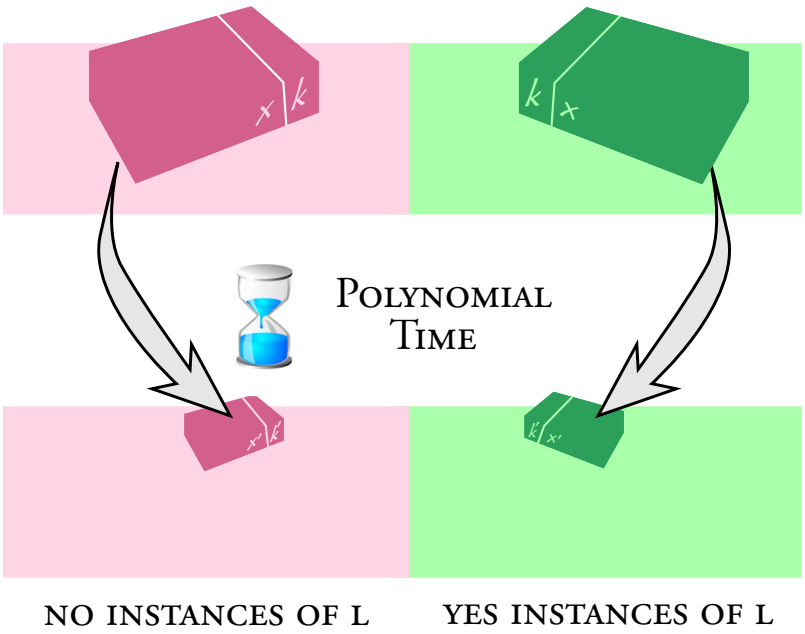


Figure 1.2: Data Reduction: A Schematic View

1. ϕ is computable in time polynomial in $|x|$ and k;
2. $|x'| \leqslant |x|$.

Two instances of L are equivalent if $(x, k) \in L$ if and only if $(x', k') \in L$.

We exploit the notion of a parameter to further our program of formalizing preprocessing algorithms. We might now understand preprocessing to be a polynomial algorithm that returns as output an instance equivalent to the input instance, but in size only a function of the parameter.

Formally, a *kernelization algorithm* consists of a finite set of data reduction rules such that by applying the rules to an instance (x, k) (in some specified order) one obtains an instance (x', k') with the property that $|x'| \leqslant g(k)$ and $k' \leqslant g(k)$, for some function g only depending on k. Such a "reduced" instance is called a *problem kernel* and $g(k)$ is called the *kernel size*.

It is folklore that a parameterized problem Π is in FPT if and only if there exists a computable function g such that Π admits a kernel of size $g(k)$. However, kernels obtained by this theoretical result are usually of exponential (or even worse) size, while problem-specific data reduction rules often achieve quadratic ($g(k) = O(k^2)$) or even linear-size ($g(k) = O(k)$) kernels. So a natural question for any concrete FPT problem is whether it admits polynomial-time kernelization to a problem kernel that is bounded by a polynomial function of the parameter ($g(k) = O(k^{O(1)})$).

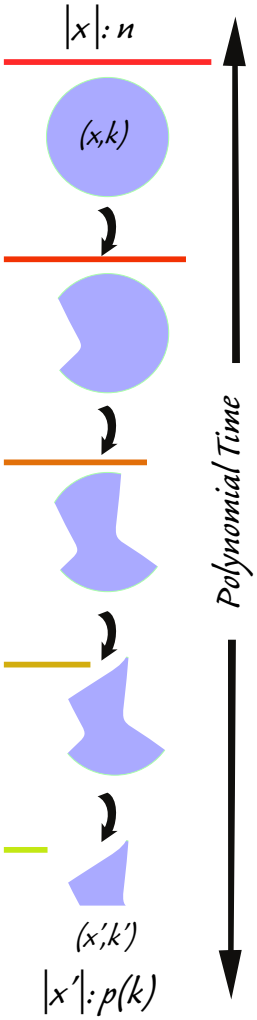


Figure 1.3: Kernelization

Polynomial kernels form our basic notion of efficient kernelization. As we noted above, there is no reason to believe that every problem in FPT admits a polynomial kernel. In fact, recent developments demonstrate that under reasonable complexity theoretic assumptions some problems in FPT do not admit polynomial kernels.

For formal definitions, the reader is referred to Chapter 2 (Technical Preliminaries), and for a comprehensive study of fixed-parameter tractability and kernelization, we refer to the books [DF99, FG06, Nie06] and the surveys [GN07, MRS11]. In this thesis, we develop kernelization algorithms for general classes of problems, while demonstrating lower bounds on some occasions. In the rest of this chapter, we introduce the problems, and describe the organization of the thesis.

1.2 Specific Problems: A Brief Overview

A large class of graph optimization problems are of the following kind: we are interested in the smallest subset of vertices whose removal makes the remaining graph possess a given property. For example, the Vertex Cover problem involves finding the smallest number of vertices whose removal makes the graph independent.

A very generic form of this question, that would account for several special cases,

would be to ask the following:

> Let H be a fixed graph. What is the smallest number of vertices after whose removal the graph does not contain any copy of H?

Whenever the property desired of the graph can be "phrased" in terms of avoiding copies of a fixed graph, the particular question we are interested in becomes a special case of the one above. For example, when we look for a vertex cover, we want to "avoid" edges: thus, setting H to be the single edge makes the problem above exactly the question of finding the optimal vertex cover.

We may make the question above somewhat more general, by requiring that we avoid not only a single graph H, but an entire set of specified graphs:

> Let $\mathcal{F}$ be a fixed collection of graphs. What is the smallest number of vertices after whose removal the graph does not contain any copy of H, for every $H \in \mathcal{F}$?

A popular special case of this question is the Feedback Vertex Set problem: recall that a feedback vertex set is a set of vertices whose removal makes the graph a forest. In the above, setting $\mathcal{F}$ to be the family of all cycles, gives us exactly the question of finding an optimal feedback vertex set.

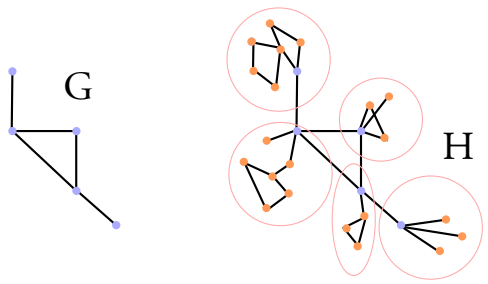


Figure 1.4: G is a *minor* of H, or equivalently, H is a *minor model* of G.

A "copy of H" is usually understood to be the presence of H as a subgraph. As we will see, requiring that we avoid the presence of H as a *minor* makes the question more succinct. When we avoid H as a minor, we not only avoid copies of H as a subgraph, but also copies of all graphs that can be contracted to H as subgraphs. We note that if H can be obtained from H' by a series of contractions, then H' is called a *minor model* of H (see Figure 1.4). So the question that we are now suggesting is the following:

Let $\mathcal{F}$ be a fixed collection of graphs. What is the smallest number of vertices after whose removal the graph does not contain any copy of any minor model of H, for every $H \in \mathcal{F}$?

Thus, the question of finding an optimal Feedback Vertex Set can be now obtained as a special case of the problem above by simply setting $\mathcal{F}$ to be a single graph, the triangle (a cycle on three vertices). Since any cycle can be contracted to a triangle, a graph that is free of minor models of triangles is a graph that is free of cycles (unless the graph is a *multigraph*, in which case we simply set $\mathcal{F}$ to be a double edge instead of a triangle).

As we might imagine, a substantial number of questions may turn out to be special cases of the problem above. Thus, it is tempting to pursue algorithmic solutions for such a problem, since it provides answers for multiple questions "in one shot". This goal lies at the heart of this thesis.

Let us state the $\mathcal{F}$-Deletion question formally. We first define the notion of a $\mathcal{F}$-*hitting set*.

Definition 1.1 ($\mathcal{F}$-hitting set). *Let* G *be a graph and let* $\mathcal{F}$ *be a collection of graphs. A* $\mathcal{F}$*-hitting set is a subset* $S \subseteq V(G)$ *such that* $G \setminus S$ *does not contain* H *as a minor, for all* $H \in \mathcal{F}$.

The question of $\mathcal{F}$-Deletion is the following:

$\mathcal{F}$-Deletion
Input: A graph G, a family of graphs $\mathcal{F}$.
Parameter: k
Question: Does there exist $\mathcal{F}$-hitting set S such that $|S| \leqslant k$?

As we have already observed, for particular choices of $\mathcal{F}$, this question corresponds to well-studied problems: For instance, it is the Vertex Cover problem when $\mathcal{F}$ consists of an edge. When $\mathcal{F} = \{\triangle\}$, a triangle, this is the feedback vertex set problem. Other famous cases are $\mathcal{F} = \{K_{2,3}, K_4\}$, $\mathcal{F} = \{K_{3,3}, K_5\}$ and $\mathcal{F} = \{K_3, T_2\}$, which correspond to removing vertices to obtain outerplanar graphs, planar graphs, and graphs of pathwidth one respectively (see Figure 1.5). Here, $K_{i,j}$ denotes the complete bipartite graph with bipartitions of sizes i and j, and K_i denotes the complete graph

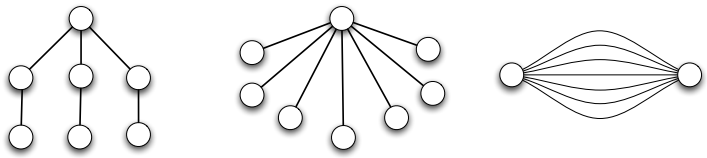


Figure 1.5: Graphs T_2, t-claw $K_{1,t}$ with $t = 7$, and θ_c with $c = 7$

on i vertices. Further, a T_2 is a star on three leaves, each of whose edges has been subdivided exactly once.

> KERNELS FOR $\mathcal{F}$-DELETION: THE CONJECTURE
>
> The general $\mathcal{F}$-deletion problem is NP-complete. From the parameterized perspective, by one of the most well-known consequences of the celebrated Graph Minor theory of Robertson and Seymour, the $\mathcal{F}$-deletion problem is fixed-parameter tractable for every finite set of forbidden minors. It is conjectured that the $\mathcal{F}$-DELETION problem admits a polynomial kernel if, and only if, $\mathcal{F}$ contains a planar graph.

This conjecture is addressed in this thesis, and we make progress by showing that many special cases of the PLANAR $\mathcal{F}$-DELETION problem do admit polynomial kernels. These results require several intermediate results that are of independent interest. We first show, in Chapter 8, that PLANAR $\mathcal{F}$-DELETION admits an approximation algorithm with ratio $O(\log^{3/2} OPT)$. Then, in Chapter 9 we restrict ourselves to a subclass of graphs, called t-claw-free graphs (those that exclude $K_{1,t}$ as an induced subgraph), and show that PLANAR $\mathcal{F}$-DELETION admits a polynomial kernel on this graph class. In an independent endeavor, we show in Chapter 10 that when $\mathcal{F}$ contains a particular graph, called θ_c (a pair of vertices with c edges between them, see Figure 1.5), then $\mathcal{F}$-DELETION admits a polynomial kernel on general graphs.

Finally, in Chapter 11, we arrive at general result, namely that the disjoint version of the PLANAR $\mathcal{F}$-DELETION admits a polynomial kernel on general graphs (Chapter 11). The input to the disjoint version is a graph with a partition of the vertex set into two parts such that the graph induced on either partition does not contain $\mathcal{F}$ as a minor.

One of the parts is declared forbidden, and we are required to find an optimal $\mathcal{F}$-hitting set from the other part. An important consequence of the polynomial kernel for this problem is an uniform FPT algorithm for the Planar $\mathcal{F}$-Deletion problem that runs in time $2^{O(k \log k)} n^{O(1)}$. No algorithm running in time $(2^{k^{O(1)}} n^{O(1)})$ was known previously.

As we pointed out above, Feedback Vertex Set is a well-studied special case of $\mathcal{F}$-Deletion — and we investigate the variant Independent Feedback Vertex Set, where we require that the vertices of the feedback vertex set induce an independent subgraph. In Chapter 5, we study the minimization version, and demonstrate polynomial kernels for Independent Feedback Vertex Set under the standard parameterization (solution size). This also serves the purpose of demonstrating some of the techniques used in the more general settings. While a polynomial kernel for Feedback Vertex Set was known before [Tho10], we note that a kernel for Independent Feedback Vertex Set would imply a kernel of the same size for Feedback Vertex Set because of a parameter-preserving reduction from the latter to the former.

Further, the $\mathcal{F}$-Deletion problem triggers interest in various related problems that are also examined, and are also of independent interest. One natural complementary question is to do with packing: note that $\mathcal{F}$-Deletion can be thought of as a "covering" problem (attempting to "hit" all minor models of graphs in $\mathcal{F}$ with the smallest possible subset), and the corresponding packing question involves maximizing the number of vertex (or edge) disjoint minor models of graphs in $\mathcal{F}$. In Chapter 12, we provide an Erdős-Pósa style result for the case when $\mathcal{F} = \{\theta_c\}$, demonstrating a relationship between the sizes of the optimal covering number and hitting set size. This result borrows techniques from the approximation algorithm for the $\mathcal{F}$-Deletion problem and also makes use of known connections between *brambles* and treewidth [ST93] (see Chapter 2 for definitions).

In this context, we apply the "Erdős-Pósa property" of θ_c minor models to make partial progress towards finding polynomial kernels for packing edge disjoint minor models of θ_c. In particular, in polynomial time, we are able to reduce any instance of the problem to one where the maximum degree is bounded by a poylnomial in k. Independently, but on a related note, we show two lower bounds, demonstrating that the problem of finding if there are at least k *vertex-disjoint* minor models of θ_c is unlikely to admit a polynomial kernel parameterized by k, and this is also true for the problem of checking if there are at least k *vertex-disjoint* cycles of odd length (again parameterized by k). These results are presented in Chapter 14.

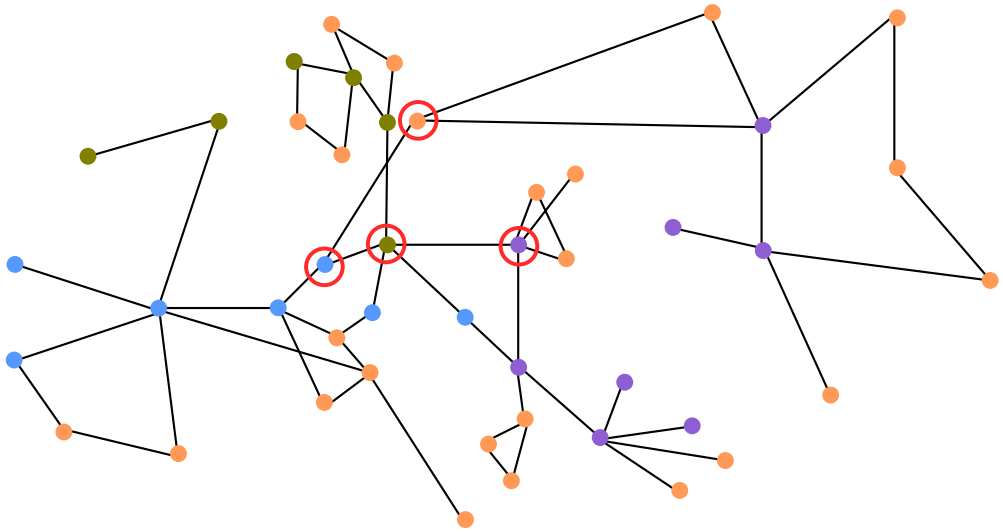


Figure 1.6: A Colorful Motif

The second central problem that we investigate in this thesis is the Colorful Motif problem. The Graph Motif problem concerns a vertex-colored undirected graph G and a *multiset* M of colors. We are asked whether there is a set S of vertices of G such that the subgraph induced on S is connected and there is a color-preserving bijective mapping from S to M. That is, the problem is to find if there is a connected subgraph H of G such that the multiset of colors of H is identical to M.

The Graph Motif problem has immense utility in bioinformatics, especially in the context of metabolic network analysis (eg. motif search in metabolic reaction graphs with vertices representing reactions and edges connecting successive reactions) [BHK+09, LFS06] . The problem is NP-complete even in very restricted cases, such as when G is a tree with maximum degree 3, or when G is a bipartite graph with maximum degree 4 and M is a multiset over just two colors. When parameterized by $|M|$, the problem is FPT, and it is W[2]-hard when parameterized by the *number of colors* in M, even when G is a tree [FFHV07].

The Colorful Motif problem is a simpler version of the Graph Motif problem, where M is a set (and not a multiset). Even this problem is NP-hard on simple classes of graphs, such as when G is a tree with maximum degree 3 [FFHV07]. The problem is FPT on general graphs when parameterized by $|M|$, and the current fastest FPT algorithm, by Guillemot and Sikora, runs in $O^*(2^{|M|})$ time and polynomial

space [GS10].

Formally, the Colorful Motif problem is the following:

Colorful Motif

Input:	A graph $G = (V, E)$, $k \in \mathbb{N}$, and a coloring function $c : V \to [k]$.
Parameter:	k
Question:	Does G contain a subtree T on k vertices such that c restricted to T is bijective?

In Chapter 15, we study the kernelization complexity of Colorful Motif on various graph classes and obtain the second example of "many polynomial kernels" in the literature (see Chapter 13, Section 13.2.2 for a formal definition, and Chapter 15, Lemma 15.5 for a simple example). The results we obtain in the context of Colorful Motif contribute to the rapidly growing collection of problems for which polynomial kernels do not exist under reasonable complexity-theoretic assumptions. Given that many of our results pertain to very special graph classes, we hope these hardness results — which make these special problems available as starting points for further reductions — will be useful in settling the kernelization complexity of many other problems.

1.3 Organization of the Thesis

The structure of the thesis is, briefly, as follows. Chapter 2 introduces the necessary technical framework in which the problems above are studied. Chapters 3, 4, 6 and 13 survey important techniques that are used in the rest of the thesis. Chapter 5 is about Independent Feedback Vertex Set [MPRS10], which asks for a feedback vertex set of size at most k that induces an independent graph (the problem is parameterized by solution size). We use this problem to illustrate techniques based on crown decompositions that will be useful in subsequent chapters.

Chapters 8,9,10,11 are about the $\mathcal{F}$-Deletion problem based on work in [FLM+11]. Chapter 12 covers an Erdős-Pósa-style result for a certain class of graphs that generalize cycles [FLM+10a]. We also describe how this result can be applied to reduce the maximum degree of instances of the edge-disjoint θ_c packing problem. Chapter 14 shows that the vertex-disjoint packing variant of a special case of the $\mathcal{F}$-Deletion

problem is unlikely to admit a polynomial kernel [FLM+10b]. Finally, in Chapter 15, we study the Colorful Motif problem [ABH+10].

The secret of getting ahead is getting started.

Mark Twain

In the beginning the Universe was created.
This has made a lot of people very angry and has been widely regarded as a bad move.

Douglas Adams

This chapter is an overview of the notation and definitions used in the rest of the document, and a collection of technical results that will be useful in future discussions.

2.1 Notational Conventions

The letters G, H, etc. are used to refer to graphs, and $\mathcal{F}$, $\mathcal{H}$, $\mathcal{J}$, etc. are used to refer to finite or infinite collections of graphs. The small letters k and l are generally reserved for referring to problem parameters.

We use $V(G)$ to denote the vertex set of a graph G, and $E(G)$ to denote the edge set. The notation M_G refers to minor models of G, that is, it is the set of graphs that contain G as a minor (for the notion of a minor, cf. Section 2.2.1). Small letters u, v, x etc. are used to denote vertices of graphs. The degree of a vertex v in G is the number of edges incident on v, and is denoted by $d(v)$. We use $\Delta(G)$ to denote the maximum degree of G, and $\delta(G)$ to refer to the minimum degree of a graph G.

For $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, \ldots, n\}$.

2.2 Definitions

2.2.1 Graphs

A graph G is a pair (V, E), where V is a finite set and $E \subseteq V \times V$. The elements of V are referred to as the *vertices* of the graph and the elements of E are called its *edges*. A

graph G' is a *subgraph* of G if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. The subgraph G' is called an *induced subgraph* of G if $E(G') = \{\{u, v\} \in E(G) \mid u, v \in V(G')\}$. Given a subset $S \subseteq V(G)$ the subgraph induced by S is denoted by $G[S]$. The subgraph induced by $V(G) \setminus S$ is denoted by $G \setminus S$. We denote by $N(S)$ the open neighbourhood of S, i.e. the set of vertices in $V(G) \setminus S$ adjacent to S.

Contraction and Minors By *contracting* an edge (u, v) of a graph G, we mean identifying the vertices u and v, keeping all the parallel edges and removing all the loops. A *minor* of a graph G is a graph H that can be obtained from a subgraph of G by contracting edges. We keep parallel edges after contraction since the graph θ_c which we want to exclude as a minor itself contains parallel edges. We use the following equivalent characterization of minors for ease of presentation.

Proposition 2.1 ([Die05]). *A graph H is a minor of G if and only if there is a map $\phi : V(H) \to 2^{V(G)}$ such that for every vertex $v \in V(H)$, $G[\phi(v)]$ is connected, for every pair of vertices $v, u \in V(H)$, $\phi(u) \cap \phi(v) = \emptyset$, and for every edge $(u, v) \in E(H)$, there is an edge $(u', v') \in E(G)$ such that $u' \in \phi(u)$ and $v' \in \phi(v)$.*

Definition 2.1. *Let G, H be two graphs. A subgraph G' of G is said to be a* minor model *of H in G if G' contains H as a minor. The subgraph G' is a* minimal minor model *of H in G if no proper subgraph of G' is a minor model of H in G.*

From Proposition 2.1 we get:

Corollary 2.2. *For any $c \in \mathbb{N}$, a subgraph M of a graph G is a minimal minor model of θ_c in G if and only if M consists of two trees, say T_1 and T_2, and a set S of c edges, each of which has one end vertex in T_1 and the other in T_2.*

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 $H \notin \mathcal{C}$.

Chromatic Number A k-coloring of a graph G is a vertex coloring that is an assignment of one of k possible colors to each vertex of G (i.e., a vertex coloring) such that no two adjacent vertices receive the same color. The *chromatic number* of a graph G is the smallest number of colors $\gamma(G)$ needed to color the vertices of G so that no two adjacent vertices share the same color, i.e., the smallest value of k possible to obtain a k-coloring. For example, the chromatic number of an edgeless graph is 1 and the complete graph on n vertices is n.

Brambles. We say that two subsets of $V(G)$ *touch* if they either have at least one vertex in common or if there is at least one edge with one endpoint in each subset.

Definition 2.2 (Brambles). *A* bramble $\mathcal{B}$ *of a graph* G *is a collection of mutually touching connected subgraphs, called the* elements *of the bramble.*

The canonical example of a bramble is the set of crosses (union of a row and a column) of an $(l \times l)$-grid (see Figure 2.1).

Further, the following terms are also useful:

- A *hitting set* of a bramble is a subset of vertices S whose intersection with every element of the bramble is non-trivial.
- For a bramble $\mathcal{B}$, the *order* of $\mathcal{B}$ is the minimum cardinality of a hitting set of the bramble.
- For a graph G, *bramble number* of G is the maximum order of a bramble of G.

For instance, for the example of the set of crosses of the $(l \times l)$-grid above, the vertices of any row (or column), would intersect all elements of the bramble and would therefore constitute a hitting set.

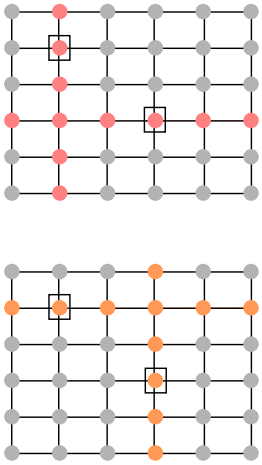


Figure 2.1: A pair of elements that share vertices, from the bramble that consists of the set of crosses of a grid.

Trees and the Helly Property

A *tree* is a connected graph with the property that the removal of any edge disconnects the graph. Equivalently, a connected graph without cycles is a tree. We will need the *Helly property* of trees, which states that if we have a collection of subtrees of a tree that pairwise intersect, then there is at least one vertex that is common to *all* the subtrees considered. We prove this below.

Lemma 2.1 (Helly Property for trees). *Let* $T_1, T_2, \ldots, T_n$ *be subtrees of a tree* T. *Suppose*

$$V(T_i) \cap V(T_j) \neq \emptyset \quad \forall i, j.$$

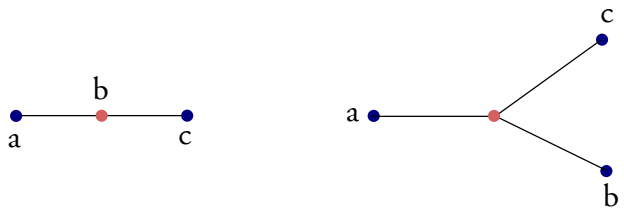


Figure 2.2: Possible paths among vertices a, b, c in tree T

Then

$$V(T_1) \cap V(T_2) \cap \ldots \cap V(T_n) \neq \emptyset.$$

Proof. Let a, b, c be any vertices of T. Let

$$S = \{i : T_i \text{ contains at least 2 of } a, b, c\}.$$

Claim 2.1. $\bigcap_{s \in S} V(T_s) \neq \emptyset$.

Let P_1 be unique (a, b)-path in T, let P_2 be unique (b, c)-path in T, and let P_3 be unique (a, c)-path in T. Since T is a tree, the intersection $V(P_1) \cap V(P_2) \cap V(P_3)$ is nonempty; additionally, the intersection must be a single vertex, say x. (See Figure 2.2.)

Every T_s for $s \in S$ must contain at least one of P_1, P_2, and P_3. Therefore, $x \in V(T_s)$ for all $s \in S$, and the claim is proven.

We now prove the lemma with a proof by induction on the number of subtrees n. The basis cases $n = 1, 2$ are trivial. Suppose the result is true for all collections of less than n subtrees of tree T (that intersect pairwise). Consider n subtrees $T_1, T_2, \ldots, T_n$ of T that intersect pairwise. By the induction hypothesis, we know that there exist vertices a, b, c such that

$$a \in V(T_1) \cap V(T_2) \cap \cdots \cap V(Tn1),$$

$$b \in V(T_2) \cap V(T_3) \cap \cdots \cap V(T_n)$$

$$c \in V(T_1) \cap V(T_n).$$

Note that every subtree T_i contains two of the vertices a, b, c, and by our previous claim, it follows that

$$V(T_1) \cap V(T_2) \cap \ldots \cap V(T_n) \neq \emptyset.$$

□

2.2.2 Kernelization

A parameterized problem L is a subset of $\Sigma^* \times \mathbb{N}$ for some finite alphabet Σ. An instance of a parameterized problem consists of (x, k), where k is called the parameter. A central notion in parameterized complexity is *fixed parameter tractability (FPT)* which means for a given instance (x, k) solvability in time $f(k) \cdot p(|x|)$, where f is an arbitrary function of k and p is a polynomial in the input size. The notions of *kernelization* is formally defined as follows.

Definition 2.3. *A* kernelization algorithm*, or in short, a kernel for a parameterized problem* $Q \subseteq \Sigma^* \times \mathbb{N}$ *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)* $(x, k) \in Q$ *if and only if* $(x', k') \in Q$ *and (b)* $|x'| + k' \leqslant g(k)$*, where* g *is an arbitrary computable function. The function* g *is referred to as the size of the kernel. If* g *is a polynomial function then we say that* Q *admits a polynomial kernel.*

For further details about kernelization, we refer the reader to Chapter 3.

2.2.3 Treewidth and Tree Decompositions

The concept of treewidth was discovered independently by several different researchers and given several different names. The actual term 'treewidth' and its definition in terms of tree decompositions were introduced by Robertson and Seymour [RS86].

Let G be a graph. A *tree decomposition* of a graph G is a pair $(T, \mathcal{X} = \{X_t\}_{t \in V(T)})$ such that

- $\cup_{t \in V(T)} X_t = V(G)$,
- for every edge $\{x, y\} \in E(G)$ there is a $t \in V(T)$ such that $\{x, y\} \subseteq X_t$, and
- for every vertex $v \in V(G)$ the subgraph of T induced by the set $\{t \mid v \in X_t\}$ is connected.

The *width* of a tree decomposition is $\left(\max_{t \in V(T)} |X_t|\right) - 1$ and the *treewidth* of G is the minimum width over all tree decompositions of G. A tree decomposition $(T, \mathcal{X})$ is called a *nice tree decomposition* if T is a tree rooted at some node r where $X_r = \emptyset$, each node of T has at most two children, and each node is of one of the following kinds:

1. *Introduce node*: a node t that has only one child t' where $X_t \supset X_{t'}$ and $|X_t| = |X_{t'}| + 1$.
2. *Forget node*: a node t that has only one child t' where $X_t \subset X_{t'}$ and $|X_t| = |X_{t'}| - 1$.
3. *Join node*: a node t with two children t_1 and t_2 such that $X_t = X_{t_1} = X_{t_2}$.
4. *Base node*: a node t that is a leaf of t, is different than the root, and $X_t = \emptyset$.

Notice that, according to the above definition, the root r of T is either a forget node or a join node. It is well known that any tree decomposition of G can be transformed into a nice tree decomposition in time $O(|V(G)| + |E(G)|)$ maintaining the same width [Klo94]. We use G_t to denote the graph induced on the vertices $\cup_{t'} X'_t$, where t' ranges over all descendants of t, including t. We use H_t to denote $G_t[V(G_t) \setminus X_t]$.

For a vertex $v \in G$, we denote the subtree of T induced by the set $\{t \mid v \in X_t\}$ by T_v.

Tree decompositions have a number of useful properties. We make note of an observation that will be useful in subsequent chapters.

Lemma 2.2. *If* $B \subseteq V(G)$ *induces a connected subgraph of* G*, then* $T_B := \cup\{T_v \mid v \in V(B)\}$ *is a subtree of* T.

Proof. We prove this by contradiction. Suppose T_B, as defined in the statement of the lemma, is not a subtree of T, and is instead an union of at least two disjoint subtrees. Without loss of generality, let T be the union of two disjoint subtrees T_1 and T_2. For $i = 1, 2$ let B_i denote $\cup\{v \in B \mid v \in X(T_i)\}$, where $X(T_i)$ denotes the union of the bags corresponding to the vertices in T_i. Since B induces a connected subgraph of G, there exists an edge (u, v) with $u \in B_1$ and $v \in B_2$. By the definition of a tree decomposition, there is a bag X_{uv} that contains both u and v, and this bag must belong to both T_1 and T_2. But this would imply that the two subtrees T_1 and T_2 share a common vertex, and are not disconnected, contradicting our starting assumption. □

It is well-known that the chromatic number of a graph is a lower bound for its treewidth. We will find this bound useful in Chapter 9 (Section 9.2.1). For the sake of completeness, we provide a proof here.

Proposition 2.3. *For any graph* G, $\chi(G) \leqslant tw(G) + 1$, *where* $\chi(G)$ *is the chromatic number of* G.

Proof. We show the inequality by induction on $|V(G)|$; $|V(G)| = n$. Fix $n \geqslant 2$ and assume that $\chi(G) \leqslant tw(G) + 1$ holds true for any graph with at most n vertices. Let G be a graph with $|V(G)| = n+1$. We show that $|V(G)| \geqslant 2$, then there are is a vertex v in G such that $d(v) \leqslant tw(G)$. Consider T, a nice tree decomposition of G such that for any other nice tree decomposition T', $V(T) \leqslant V(T')$. Let t be a leaf node in this decomposition, and let t' be the parent node of t. Note that there is a vertex $v \in X_t \setminus X_{t'}$ (we use X_t to denote the bag associated with t). Indeed, if not, we have that the bags X_t and $X_{t'}$ are identical, and we may obtain a nice tree decomposition on fewer vertices by deleting t, contradicting the assumption of vertex-minimality of T. Note that all neighbors of v are contained in X_t — this follows directly from the definition of a tree decomposition and the fact that v lies in a leaf node whose parent does not contain v. Therefore, we have that that $d(v) \leqslant tw(G)$.

So consider a vertex v of G such that $d(v) \leqslant tw(G)$. By induction,

$$\chi(G \setminus \{v\}) \leqslant tw(G \setminus \{v\}) + 1,$$

and that is clearly at most $tw(G) + 1$. Since $d(v) \leqslant tw(G)$, there is always a color class X in the $(tw(G) + 1)$-coloring of $G \setminus \{v\}$ which is such that v is not adjacent to any vertex in X (note that X may not contain any vertices at all). So by extending the $(tw(G) + 1)$-coloring of $G \setminus \{v\}$ by adding v to the color class X, we have that $\chi(G) \leqslant tw(G) + 1$. □

Given a graph G and $S \subseteq V(G)$, we define $\partial_G(S)$ as the set of vertices in S that have a neighbor in $V(G) \setminus S$. For a set $S \subseteq V(G)$ the neighborhood of S is $N_G(S) = \partial_G(V(G) \setminus S)$. When it is clear from the context, we omit the subscripts. We now define the notion of a *protrusion*, introduced in [BFL+09].

Definition 2.4 (r-**protrusion**). *Given a graph* G, *we say that a set* $X \subseteq V(G)$ *is an* r-protrusion *of* G *if* $tw(G[X]) \leqslant r$ *and* $|\partial(X)| \leqslant r$.

For more about protrusions, we refer the reader to Chapter 6.

2.2.4 Monadic Second Order Logic (MSO)

The syntax of MSO on graphs includes the logical connectives $\vee$, $\wedge$, $\neg$, $\Leftrightarrow$, $\Rightarrow$, variables for vertices, edges, sets of vertices and sets of edges, the quantifiers $\forall$, $\exists$ that can be applied to these variables, and the following five binary relations:

1. $u \in U$ where u is a vertex variable and U is a vertex set variable;
2. $d \in D$ where d is an edge variable and D is an edge set variable;
3. $\mathrm{inc}(d, u)$, where d is an edge variable, u is a vertex variable, and the interpretation is that the edge d is incident on the vertex u;
4. $\mathrm{adj}(u, v)$, where u and v are vertex variables u, and the interpretation is that u and v are adjacent;
5. equality of variables representing vertices, edges, set of vertices and set of edges.

Many common graph-theoretic notions such as vertex degree, connectivity, planarity, being acyclic, and so on, can be expressed in MSO, as can be seen from introductory expositions [BPT92, Cou97]. Of particular interest to us are p-MIN-MSO problems. In a p-MIN-MSO graph problem Π, we are given a graph G and an integer k as input. The objective is to decide whether there is a vertex/edge set S of size at most k such that the MSO-expressible predicate $P_\Pi(G, S)$ is satisfied.

The following well known result states that every optimization problem expressible in MSO has a linear time algorithm on graphs of bounded treewidth, and we will find this very useful in various situations.

Theorem ([ALS91, Bod96, BPT92, Cou90, CM93]). *Let ϕ be a property that is expressible in Monadic Second Order Logic. For any fixed positive integer t, there is an algorithm that, given a graph G of treewidth at most t as input, finds a largest (alternatively, smallest) set S of vertices of G that satisfies ϕ in time $f(t, |\phi|)|V(G)|$.*

This result is revisited in Chapter 7.

The ability to simplify means to eliminate the unnecessary
so that the necessary may speak.

Hans Hofmann

Simplify, simplify.

Henry David Thoreau

In attacking computationally hard problems, it is common (especially in practice) to attempt "reducing" the input instance using efficient pre-processing. Historically, as an algorithmic technique, preprocessing was somewhat underrated in the theoretical context, since their correctness and performance was tricky to analyze. In recent times, ideas from parameterized complexity have offered a natural but formal framework for the analysis of many well-known heuristics. These ideas also provided the opportunity for a new kind of algorithm design, and developments in this direction have evolved into what is now a well known specialization when it comes to algorithms for NP-hard problems: *kernelization*.

Informally, preprocessing involves obtaining equivalent instances that are "simpler" than the original. To make the notion precise, we will need well-defined measures of efficiency and simplicity. In our discussions, for the former, we use the conventional benchmark of *polynomial time computability*. As for simplicity, we restrict ourselves to considerations of *size*. Thus our attempts will be concentrated on making the instance size as small as possible, where the size of an instance is defined according to the problem under consideration.

Many input instances have the property that they consist of some parts that are relatively easy to handle, and other parts that form the "really hard" core of the problem. The data reduction paradigm aims to efficiently "cut away easy parts" of the given problem instance and to produce a new and size-reduced instance where exhaustive search methods and other cost-intensive algorithms can be applied.

For a long time, the mathematical analysis of polynomial time preprocessing algorithms was neglected. The basic reason for this was the following anomaly: if we start

with an instance I of an NP-hard problem and can show that in polynomial time we can replace this with an equivalent instance I' with $|I'| < |I|$ then that would imply P=NP in classical complexity. The situation changed drastically with advent of parameterized complexity. Combining tools from parameterized complexity and classical complexity it has become possible to derive upper and lower bounds on the sizes of reduced instances, or so called *kernels*. The importance of preprocessing and the mathematical challenges it poses is beautifully expressed in the following quote by Fellows [Felo6]:

> *It has become clear, however, that far from being trivial and uninteresting, that pre-processing has unexpected practical power for real world input distributions, and is mathematically a much deeper subject than has generally been understood.*

In the framework of parameterized complexity, each problem instance comes with a parameter k and the parameterized problem is said to admit a *polynomial kernel* if there is a polynomial time algorithm (the degree of polynomial is independent of k), called a *kernelization* algorithm, that reduces the input instance down to an instance with size bounded by a polynomial $p(k)$ in k, while preserving the answer. This reduced instance is called a $p(k)$ *kernel* for the problem. If $p(k) = O(k)$, then we call it a *linear kernel*. Kernelization has been extensively studied in the realm of parameterized complexity, resulting in polynomial kernels for a variety of problems. Notable examples include a $2k$-sized vertex kernel for Vertex Cover [CKJo1], a $67k$ kernel for Dominating Set on planar graphs [CFKXo7], and an $O(k^2)$ kernel for Feedback Vertex Set [Thoio] parameterized by the solution size.

3.1 Examples of Kernels

We illustrate the method of kernelization using the parameterized version of Max3Sat where given a boolean 3-CNF formula and an integer parameter k, we would like to know whether there is an assignment to the variables that satisfies at least k of the clauses. Our other examples in this Section include a kernel for d-Hitting Set using the *Sunflower Lemma*, a $4k$ sized kernel for Vertex Cover using *crown decomposition* and a 2^k kernel for (Edge) Clique Cover.

3.1.1 Max3Sat

We begin with a simple example: the classical satisfiability problem on propositional formulas asks if there is an assignment that satisfies a given formula. The following is an optimization version of the problem, and asks for an assignment that satisfies the largest number of clauses. We state the decision version:

> Max-3-SAT: Given a propositional formula ϕ in 3CNF, does there exist an assignment of the variables that satisfies at least k clauses?

Let ϕ be the given boolean CNF 3-SAT formula with n variables and m clauses. It is well known that in any boolean CNF formula, there is an assignment that satisfies at least half of the clauses (given any assignment that doesn't satisfy half the clauses, its bitwise complement will). So if the parameter k is less than $m/2$, then there is an assignment to the variables that satisfies at least k of the clauses. In this case, we conclude that the given input is a YES-instance of the problem, and the kernel is a trivial YES-instance. Otherwise, $m \leqslant 2k$, and so $n \leqslant 6k$, and the instance given as input itself is the kernel.

3.1.2 d-Hitting Set

In this Section we give a kernelization algorithm for the d-Hitting Set problem which is defined as follows:

> d-Hitting Set (d-HS) : Given a collection $\mathcal{C}$ of d element subsets of an universe U and a positive integer k, the problem is to determine whether there exists a subset $U' \subseteq U$ of size at most k such that U' contains at least one element from each set in $\mathcal{C}$.

Our kernelization algorithm is based on the following widely used *Sunflower Lemma*. We first define the terminology used in the statement of the lemma. A *sunflower* with k *petals* and a *core* Y is a collection of sets $S_1, S_2 \cdots S_k$ such that $S_i \cap S_j = Y$ for all $i \neq j$; the sets $S_i \setminus Y$ are petals and we require that none of them be empty. Note that a family of pairwise disjoint sets is a sunflower (with an empty core).

Lemma 3.1 ([FG06]). **[Sunflower Lemma]** *Let $\mathcal{F}$ be a family of sets over an universe U each of cardinality s. If $|\mathcal{F}| > s!(k-1)^s$ then $\mathcal{F}$ contains a sunflower with k petals and such a sunflower can be computed in time polynomial in the size of $\mathcal{F}$ and U.*

Now we are ready to prove the following theorem about kernelization for d-HS.

Theorem 3.1. *d-HS has a kernel of size $O(k^d d! d^2)$. That is, given an instance $(U, \mathcal{C}, k)$ of d-HS, we can replace it with an equivalent instance $(U, \mathcal{C}', k')$ with $|\mathcal{C}'| \leqslant O(k^d d! d)$ in polynomial time.*

Proof. The crucial observation is that if $\mathcal{C}$ contains a sunflower $S = \{S_1, \cdots, S_{k+1}\}$ of cardinality $k+1$ then every hitting set of $\mathcal{C}$ of size at most k must intersect with the core Y of the sunflower S, otherwise we will need hitting set of size more than k. Therefore if we let $\mathcal{C}' = \mathcal{C} \setminus (S \cup Y)$ then the instance $(U, \mathcal{C}, k)$ and $(U, \mathcal{C}', k)$ are equivalent.

Now we apply the Sunflower Lemma for all $d' \in \{1, \cdots, d\}$, repeatedly replacing sunflowers of size at least $k+1$ with their cores until the number of sets for any fixed $d' \in \{1, \cdots, d\}$ is at most $O(k^{d'} d'!)$. Summing over all d we obtain the desired kernel of size $O(k^d d! d)$. □

3.1.3 Crown Decomposition : Vertex Cover

In this Section we introduce a *crown decomposition* based kernelization for Vertex Cover. It is based on a connection between matchings and vertex cover which is that the maximum size of a matching is a lower bound for the minimum cardinality vertex cover. We first define Vertex Cover precisely as follows.

> Vertex Cover (VC): Given a graph $G = (V, E)$ and a positive integer k, does there exist a subset $V' \subseteq V$ of size at most k such that for every edge $(u, v) \in E$ either $u \in V'$ or $v \in V'$.

Vertex Cover can be modelled as 2-HS with universe $U = V$ and $\mathcal{C} = \{\{u, v\} \mid (uv) \in E\}$ and hence using Theorem 3.1 we get a kernel with at most $4k^2$ edges and $8k^2$ vertices. Here we give a kernel with at most $4k$ vertices.

Now we define crown decomposition.

Definition 3.1. *A crown decomposition of a graph $G = (V, E)$ is a partitioning of V as C, H and R, where C and H are nonempty and the partition satisfies the following properties.*

1. *C is an independent set.*

2. *There are no edges between vertices of* C *and* R*, that is* $N[C] \cap R = \emptyset$.
3. *Let* E' *be the set of edges between vertices of* C *and* H. *Then* E' *contains a matching of size* $|H|$*, that is the bipartite subgraph* $G' = (C \cup H, E')$ *has a matching saturating all the vertices of* H.

We need the following lemma by Chor et. al. [CFJ05] which makes it possible to find a crown decomposition efficiently.

Lemma 3.2. *If a graph* $G = (V, E)$ *has an independent set* $I \subseteq V$ *such that* $|N(I)| < |I|$*, then a crown decomposition* (C, H, R) *of* G *such that* $C \subseteq I$ *can be found in time* $O(m + n)$*, given* G *and* I.

The crown-decomposition gives us a global method to reduce the instance size. Its importance is evident from the following simple lemma.

Lemma 3.3. *Let* (C, H, R) *be a crown decomposition of a graph* $G = (V, E)$. *Then* G *has a vertex cover of size* k *if and only if* $G' = G[R]$ *has a vertex cover of size* $k' = k - |H|$.

Proof. Suppose G has a vertex cover V' of size k in G. Now, we have a matching of size $|H|$ between C and H that saturates every vertex of H. Thus $|V' \cap (H \cup C)| \geqslant |H|$, as any vertex cover must pick one vertex form each of the matching edge. Hence the number of vertices in V' covering the edges not incident to $H \cup C$ is at most $k - |H|$, proving one direction of the result.

For the other direction, it is enough to observe that if V'' is a vertex cover of size $k - |H|$ for G' then $V'' \cup H$ is a vertex cover of size k for G. □

Theorem 3.2. *Vertex Cover has a kernel of size* $4k$.

Proof. Given an input graph $G = (V, E)$ and a positive integer k, we do as follows. We first find a maximal matching M of G. Let $V(M)$ be the set of endpoints of edges in M. Now if $|V(M)| > 2k$, we answer NO and stop as any vertex cover must contain at least one vertex from each of the matching edges and hence has size more than k. Now we distinguish two cases based on the size of $|V - V(M)|$. If $|V - V(M)| \leqslant 2k$, then we stop as we have obtained a kernel of size at most $4k$. Else $|V - V(M)| > 2k$. In this case we have found an independent set $I = V - V(M)$ such that $|N(I)| \leqslant |V(M)| < |I|$ and hence we can apply Lemma 3.2 to obtain a crown decomposition (C, H, R) of G. Given a crown decomposition (C, H, R), we

apply Lemma 3.3 and obtain a smaller instance for a vertex cover with $G' = G[R]$ and parameter $k' = k - |H|$. Now we repeat the above procedure with this reduced instance until either we get a NO answer or we have $|V - V(M)| \leqslant 2k$ resulting in a kernel of size $4k$. □

The bound obtained on the kernel for Vertex Cover in Theorem 3.2 can be further improved to $2k$ with much more sophisticated use of crown decomposition. An independently developed method to obtain a $2k$ size kernel for Vertex Cover is through a linear programming formulation of Vertex Cover. On close inspection, however, it can inferred that the mechanics of the linear programming approach amount to performing crown-based reductions. See [FG06] and [Nie06] for further details about the linear programming based kernelization of Vertex Cover.

3.1.4 Clique Cover

Unfortunately, not all known problem kernels are shown to have polynomial size. Here, we present some data reduction results with exponential-size kernels. Clearly, it is a pressing challenge to find out whether these bounds can be improved to polynomial ones.

In this section, we study the (Edge) Clique Cover problem, where the input consists of an undirected graph $G = (V, E)$ and a nonnegative integer k and the question is whether there is a set of at most k cliques in G such that each edge in E has both its endpoints in at least one of the selected cliques.

Given an n-vertex and m-edge graph G, we use $N(v)$ to denote the neighborhood of vertex v in G, namely, $N(v) := \{u \mid \{u, v\} \in E\}$. The *closed* neighborhood of vertex v, denoted by $N[v]$, is equal to $N(v) \cup \{v\}$.

We formulate data reduction rules for a generalized version of (Edge) Clique Cover in which already some edges may be marked as "covered". Then, the question is to find a clique cover of size k that covers all uncovered edges. We apply the following data reduction rules [GGHN06]:

1. Remove isolated vertices and vertices that are only adjacent to covered edges.
2. If an uncovered edge $\{u, v\}$ is contained in exactly one maximal clique C, that is, if the common neighbors of u and v induce a clique, then add C to the solution, mark its edges as covered, and decrease k by one.

3. If there is an edge $\{u, v\}$ whose endpoints have exactly the same closed neighborhood, that is, $N[u] = N[v]$, then mark all edges incident to u as covered. To reconstruct a solution for the non-reduced instance, add u to every clique containing v.

The correctness of the rules is easy to prove. To show the following problem kernel, only the first and third rule are needed.

Theorem 3.3 ([GGHN06]). (Edge) Clique Cover *admits a problem kernel with at most* 2^k *vertices.*

Proof. Consider any graph $G = (V, E)$ with more than 2^k vertices that has a clique cover $C_1, \ldots, C_k$ of size k. We assign to each vertex $v \in V$ a binary vector b_v of length k where bit i, $1 \leqslant i \leqslant k$, is set to 1 if and only if v is contained in clique C_i. Since there are only 2^k possible vectors, there must be $u \neq v \in V$ with $b_u = b_v$. If b_u and b_v are zero, the first rule applies; otherwise, u and v are contained in the same cliques. This means that u and v are connected and share the same neighborhood, and thus the third rule applies. □

The essence of mathematics is not to make simple things complicated,
but to make complicated things simple.

S Gudder

The further a mathematical theory is developed,
the more harmoniously and uniformly does its construction proceed,
and unsuspected relations are disclosed
between hitherto separated branches of the science.

David Hilbert

In this chapter, we introduce the q-expansion lemma, a variation of Hall's Theorem. The q-expansion lemma is a generalization of a result due to Thomassé [Tho10, Theorem 2.3], and captures a certain property of neighborhood sets in graphs that has been used implicitly by several authors to obtain polynomial kernels for many graph problems. When $q = 1$, the application of this lemma is exactly the well-known Crown Reduction Rule [AFLS07].

Consider a bipartite graph with partitions $(A \uplus B)$. One direction of Hall's theorem guarantees the existence of a matching saturating A if every vertex subset S of A has a neighborhood at least as large as itself. Further, if every vertex subset has a neighborhood that is at least q times as large as itself (for some positive integer $q \geqslant 1$) then we can argue, along similar lines, the existence of q matchings saturating A, all vertex-disjoint in B. In the context of applying this result in kernelization, it is useful to rephrase it with a weaker hypothesis, and a weaker conclusion. Specifically, the version of the statement that is useful requires only the size of B to be at least $q \cdot |A|$, and guarantees the existence of q matchings, vertex-disjoint in B, that saturate some non-empty subset of A.

We first present a tighter version of the previous statement, obtaining the same conclusion with the requirement that $|B|$ is at least $q \cdot m$, where m is the size of the maximum matching in the bipartite graph. Then we demonstrate how the lemma may be used in kernelization algorithms by providing three examples: Vertex Cover, $(n - k)$ Coloring, and Edge Dominating Set. We hope that a general strategy emerges

from these examples that would be useful in the context of other problems. Indeed, the q-expansion lemma is crucial in the kernelization for Independent Feedback Vertex Set, and a special case of the $\mathcal{F}$-Deletion problem. For all the problems that are discussed in this chapter, the sizes of the kernels obtained differ from the sizes of the best-known kernels by only constant factor.

4.1 The q-expansion lemma

Consider a bipartite graph G with vertex bipartition $A \uplus B$. Given subsets $S \subseteq A$ and $T \subseteq B$, we say that S has $|S|$ q-stars in T if to every $x \in S$ we can associate a subset $F_x \subseteq N(x) \cap T$ such that (a) for all $x \in S$, $|F_x| = q$; (b) for any pair of vertices $x, y \in S$, $F_x \cap F_y = \emptyset$. Observe that if S has $|S|$ q-stars in T then every vertex x in S could be thought of as the center of a star with its q leaves in T, with all these stars being vertex-disjoint. Further, a collection of $|S|$ q-stars is also a family of q edge-disjoint matchings, each saturating S.

We are now ready to state and prove the main lemma of this chapter.

Lemma 4.1. [**The q-expansion lemma**] *Let q be a positive integer, and let m be the size of the maximum matching in a bipartite graph G with vertex bipartition $A \uplus B$. If there are no isolated vertices in B, and $|B| > mq$, then there exist nonempty vertex sets $S \subseteq A, T \subseteq B$ such that S has $|S|$ q-stars in T and no vertex in T has a neighbor outside S.*

Furthermore, the sets S, T can be found in time polynomial in the size of G.

Proof. Consider the graph H obtained from $G = (A \uplus B, E)$ by adding $(q-1)$ copies of all the vertices in A, and giving all copies of a vertex v the same neighborhood in B as v. Formally, let $\{u_1, u_2, \ldots, u_p\}$ denote the vertices of A, and let $A_1, \ldots, A_q$ denote q vertex sets, with q vertices each:

$$A_i := \{u_1^{(i)}, \ldots u_p^{(i)}\}.$$

Further, we use X to denote $A_1 \cup \cdots \cup A_q$. The graph H is the bipartite graph $(X \uplus B, E^*)$, where E^* is given by:

$$\bigcup_{1 \leqslant j \leqslant q} \{(u_i^{(j)}, v) \mid u_i^{(j)} \in A_j, v \in B \text{ such that } (u_i, v) \in E\}. \text{ (See Figure 4.1)}$$

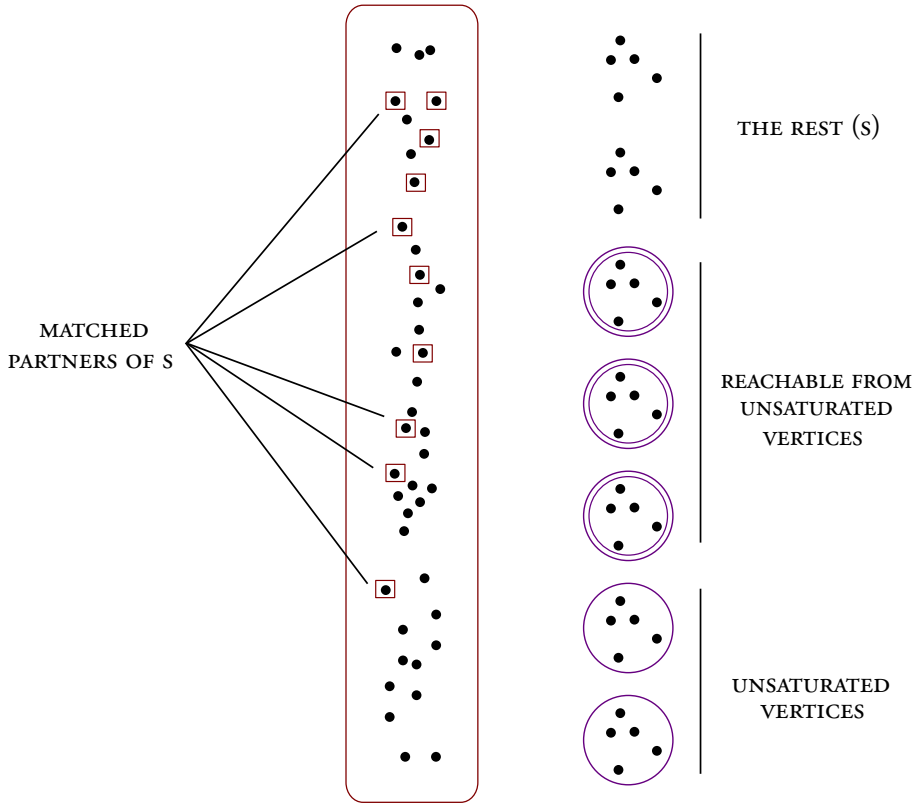


Figure 4.1: An informal schematic of the auxiliary bipartite graph H constructed in the proof.

Let M be a maximum matching in H. For the rest of this discussion, vertices are saturated and unsaturated with respect to this fixed matching M.

Let U_X be the vertices in X that are unsaturated, and R_X be those that are reachable from U_X via alternating paths. We let $S_A = X \setminus (U_X \cup R_X)$. Let U_B be the set of unsaturated vertices in B, and let T denote the set of partners of S_A in the matching M, that is, $T = \{x \in B \mid \{u, x\} \in M \text{ and } u \in S_A\}$ (see Figure 4.1).

Note that S_A is non-empty: since $|B| > mq$, the set U_B of unsaturated vertices of B in H is non-empty. Further, by the assumption that B admits no isolated vertices, the neighbors of U_B form a non-trivial subset of A. Now, notice that neighbors of U_B cannot lie in either U_X or R_X (in both cases we obtain augmenting paths, contradicting the fact that M is a maximum matching). Therefore, the neighbors of U_B must lie in S_A, and therefore S_A is non-empty.

For every $v \in A$, let $C(v)$ be the set of all copies of v (including v). We claim that either $C(v) \cap S_A = C(v)$, or $C(v) \cap S_A = \emptyset$ (see Figure 4.2). Suppose that $v \in S_A$ but a copy of v, say u, is in U_X. Let $\{v, w\} \in M$. Then v is reachable from u because $\{u, w\} \in E(H)$, contradicting the assumption that $v \in S_A$. In the case when $v \in S_A$ but a copy of u is in R_X, let $\{w, u\}$ be the last edge on some alternating path from U_X to u. Since $\{w, v\} \in E(H)$, we have that there is also an alternating path from U_X to v, contradicting the fact that $v \in S_A$. Now, let $S = \{v \in A | C(v) \subseteq S_A\}$. Then the subgraph $G[S \cup T]$ contains q edge-disjoint matchings, each of which saturates S in G — this is because in H, M saturates each copy of $v \in S$ separately.

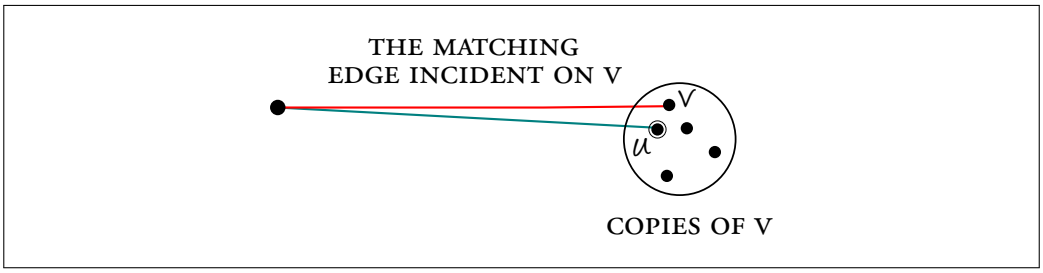


Figure 4.2: A vertex $v \in S_A$ with a copy $u \notin S_A$. If $u \notin S_A$, then it is either unsaturated or reachable from an unsaturated vertex (say w). Thus, v is reachable from either u or w, implying it should be in R_X by construction, not S_A - thus we arrive at a contradiction.

We now show that no vertex in T has a neighbor outside S in G. Notice that if no vertex in T has a neighbor outside S_A in H, then from the construction no vertex in T has a neighbor outside S in G, thus it suffices to prove that no vertex in T has a neighbor outside S_A in H. For the purpose of contradiction, let us assume that for some $v \in T$, $u \in N(v)$, but $u \notin S_A$. Suppose $u \in R_X$. We know that $u \in R_X$ because there is some unsaturated vertex (say w) that is connected by an alternating path to u. This path can be extended to a path to v using the edge $\{u, v\}$, and can be further extended to v', where $\{v, v'\} \in M$. However, $v' \in S_A$, and by construction, there is no path from $w \in U_X$ to v', a contradiction. If $u \in U_X$, then we arrive at a contradiction along the same lines (in fact, the paths from w to a vertex in S will be of length two in this case). This proves the claim that no vertex in T has a neighbor outside S_A in H.

This concludes the proof.

□

4.2 Applications of q–Expansion Lemma in Kernelization

Lemma 4.1 provides us an uniform way to view several known kernel results in the literature and is easy to apply. In this section we derive several known kernel results for Vertex Cover, $(n-k)$-Coloring, and Edge-Dominating Set. By similar arguments, kernels for many problems including Maximum Satisfiability, Set Splitting, Minimum Maximal Set, Irredundant Set can be derived almost directly from q–Expansion Lemma.

4.2.1 Vertex Cover

The Vertex Cover question asks for a subset of vertices S such that $G \setminus S$ is an independent set. Such a subset is called a vertex cover. In the optimization setting, we seek a vertex cover of minimum cardinality, and the parameterized version of the question asks if there exists a vertex cover of size at most k, where k is the parameter:

Vertex cover
Input: A graph G.
Parameter: k
Question: Does there exist a subset $S \subseteq V(G)$, such that $G \setminus S$ is an independent set, and $|S| \leqslant k$?

This problem has a problem kernel on $2k$ vertices, obtained using arguments based on a theorem of Nemhauser-Trotter, or using LP relaxation [HN94]. We present a simple argument that yields a $3k$-vertex kernel for this problem, using the q-Expansion Lemma with $q = 1$.

Remove all isolated vertices from G: this does not affect the size of any minimal vertex cover of G. Find a maximal matching M of G. If M contains more than k edges, then G does not have a vertex cover of size at most k, and the input is a NO instance. Otherwise, let A be the set of (at most $2k$) vertices that constitute the endpoints of the edges in M, and let $B = V(G) \setminus A$. Now $G[B]$ is an independent set, and every vertex in B has a neighbor in A. Consider the bipartite graph $H = (A \uplus B, E')$

obtained from G by deleting all edges with both endpoints in A. Now we apply the q-Expansion Lemma to H with $q = 1$.

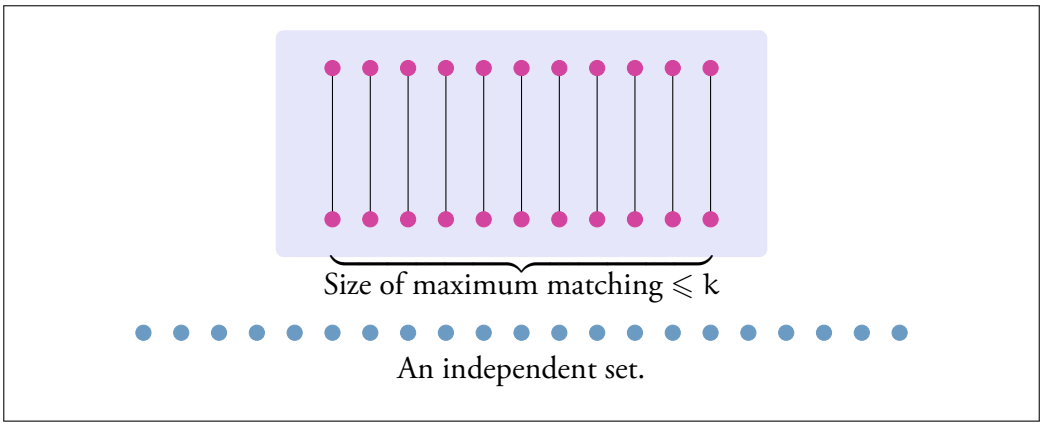


Figure 4.3: Any maximal matching has at most k edges in a YES instance of Vertex Cover. The remaining vertices form an independent set.

We find a maximum matching M_1 of H. As before, if $|M_1| > k$ then we can immediately answer NO, and so we can assume that $|M_1| \leqslant k$. If $|B| \geqslant |M_1|$, then by the q-Expansion Lemma (with $q = 1$), we can find in polynomial time nonempty vertex sets $S \subseteq A, T \subseteq B$ such that $H[S \cup T]$ contains a matching saturating S, and in H (and hence in G), no vertex in T has a neighbor outside S. It follows that there exists a minimum-size vertex cover of G that contains all of S and none of T, and so we remove $S \cup T$ from G and set $k := k - |S|$.

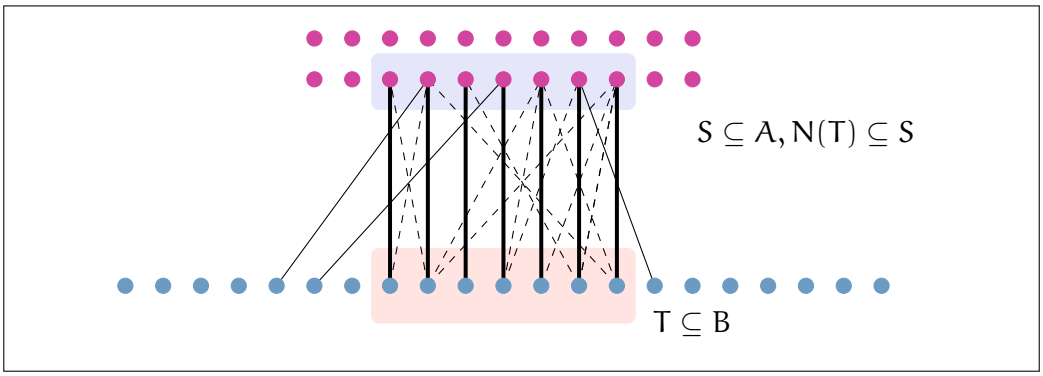


Figure 4.4: The subsets S and T obtained from an application of the expansion lemma.

By repeating this procedure, we finally obtain a graph where $|B| \leqslant |M| = k$. Thus $G[A \cup B]$ is the kernel for the problem, and $|A \cup B| \leqslant 3k$. This concludes the description of a $3k$-vertex kernel:

Lemma 4.2. *Vertex Cover admits a kernel on* $3k$ *vertices when parameterized by solution size.*

4.2.2 $(n-k)$ Coloring

Given a graph G and a positive integer k as input, the $(n-k)$ Graph Coloring problem asks whether the vertices of G can be properly colored using at most $(n-k)$ colors. If this is indeed feasible, then the *chromatic number* of G is at most $(n-k)$ and this is denoted by $\chi(G) \leqslant (n-k)$. We consider this problem parameterized by k. Note that the version of the problem that asks if a given graph can be colored with at most k colors, when parameterized by k, is clearly not solvable in FPT time. This follows from the fact that the question is NP-complete for fixed values of k.

$(n-k)$ Graph Coloring

Input:	A graph $G = (V, E)$.
Parameter:	k
Question:	Does there exist a partition of $V(G)$ into $(n-k)$ parts such that each part induces an independent subgraph?

This problem has a kernel on $3k-3$ vertices, obtained using the fact that the minimum number of colors needed to properly color G is equal to the minimum number of cliques needed to cover the complement $\overline{G}$ of G [CFJ05]. In this section, present a simple argument that yields a $(3k-3)$-vertex kernel for this problem, using the q-Expansion Lemma with $q = 1$.

If a vertex v in G is adjacent to every other vertex of G, then in any proper coloring of G, v must get a unique color. Therefore, if there are l vertices in G that are each adjacent to all other (except itself) vertices in G, and G' is the graph obtained from G by removing all these "global" vertices, then the chromatic number of G (the smallest number of colors required to properly color G), $\chi(G)$, is at most $(n-k)$ if and only if $\chi(G')$ is at most $(n-k-l)$. So we can assume without loss of generality that G does not contain such global vertices. It follows that the complement $\overline{G}$ of G does not contain any isolated vertex.

Construct $\overline{G}$, and find a maximal matching M' of $\overline{G}$. If M' contains at least k edges, then $\chi(G) \leqslant (n-k)$, and the input is a YES instance, as follows: The two endpoints of each edge in M' are *not adjacent* in G, and so can be given the same color. This uses up $|M'|$ colors. The remaining $n - 2|M'|$ vertices of G can be properly colored using at most $n - 2|M'|$ colors, and this gives a proper coloring of G using at most $n - 2|M'| + |M'| = n - |M'| = (n - k)$ colors. So we can assume without loss of generality that $|M'| < k$.

Let A be the set of (at most $2k-2$) vertices that constitute the endpoints of the edges in M', and let $B = V(\overline{G}) \setminus A$. Now $\overline{G}[B]$ is an independent set, and every vertex in B has a neighbor in A. Consider the bipartite graph $H = (A \uplus B, E')$ obtained from $\overline{G}$ by deleting all edges both of whose end vertices are in A. Now we apply the q-Expansion Lemma to H, with $q = 1$.

We find a maximum matching M of H. As before, if $|M| \geqslant k$ then we can immediately answer YES, and so we can assume that $|M| < k$. If $|B| > |M|$, then by q-Expansion Lemma we can find, in polynomial time, vertex sets $S \subseteq A, T \subseteq B; S, T \neq \emptyset$ such that $H[S \cup T]$ contains a matching saturating S, and in H (and hence in $\overline{G}$), no vertex in T has a neighbor outside S. We claim that $\chi(G) \leqslant (n - k)$ if and only if $\chi(G \setminus (S \cup T)) \leqslant (n - k - |T|)$. Note that the number of vertices in $G \setminus (S \cup T)$ is $n' = n - |S| - |T|$, and so the parameter has dropped from k to $k - |S|$ in the new instance. To see this, suppose $\chi(G) \leqslant (n - k)$, and consider any coloring $\mathcal{C} : V \to [n-k]$ of G with at most $n-k$ colors. In G, any vertex u in T is adjacent to every other vertex in B, and to every vertex in $A \setminus S$. We now recolor the vertices of S as follows: for each $v \in S$, we assign to v the color $\mathcal{C}(u)$, where u is the vertex to which v is matched by M. Clearly u is not adjacent to v in G, and after this recoloring, u and v are the only two vertices in G that have the color $\mathcal{C}(u)$. G is thus properly colored by at most $(n - k)$ colors after the recoloring, and the new colors of the vertices in S form a subset of the colors of the vertices in T. It follows that $\chi(G \setminus (S \cup T)) \leqslant (n-k-|T|)$. Essentially the same argument yields the converse direction as well.

So we remove $S \cup T$ from G and set $k := k - |S|$. By repeating this procedure, we finally obtain a graph where $|B| \leqslant |M| \leqslant k-1$. The kernel is $G[A \cup B]$, with $|A \cup B| \leqslant (3k - 3)$:

Lemma 4.3. *The* $(n - k)$ *Graph Coloring problem admits a kernel on* $(3k - 3)$ *vertices when parameterized by* k.

4.2.3 Edge-Dominating Set

An edge $e = (u, v) \in E(G)$ is *adjacent* to another edge $e' = (u', v') \in E(G)$ if $\{u, v\} \cap \{u', v'\} \neq \phi$. Given a graph G and a positive integer k as an input, the Edge Dominating Set problem asks whether G has a edge dominating set — a set $S \subseteq E(G)$ of edges such that any edge in G is adjacent to least one edge in S — of size at most k.

Edge Dominating Set
Input: A graph $G = (V, E)$.
Parameter: k
Question: Does there exist a subset of edges, $S \subseteq E(G)$, such that every edge $e \in E(G)$ is adjacent to some edge in S, and $|S| \leqslant k$?

We now present a simple argument that yields a $2k(k + 3)$-vertex kernel for this problem, using the q-Expansion Lemma with q set to $(k + 1)$.

Remove all isolated vertices from G: this does not affect the size of any minimal edge dominating set of G. Find a maximal matching M of G. Note that any edge in an edge dominating set can dominate at most two edges from a matching. Thus, if M contains more than $2k$ edges, then G does not have a edge dominating set of size at most k, and the input is a NO instance. Otherwise, let A be the set of (at most $4k$) vertices that constitute the endpoints of the edges in M, and let $B = V(G) \setminus A$. Now $G[B]$ is an independent set, and every vertex in B has a neighbor in A. Consider the bipartite graph $H = (A \uplus B, E')$ obtained from G by deleting all edges with both endpoints in A. Now we apply the q-Expansion Lemma to H with $q = (k + 1)$.

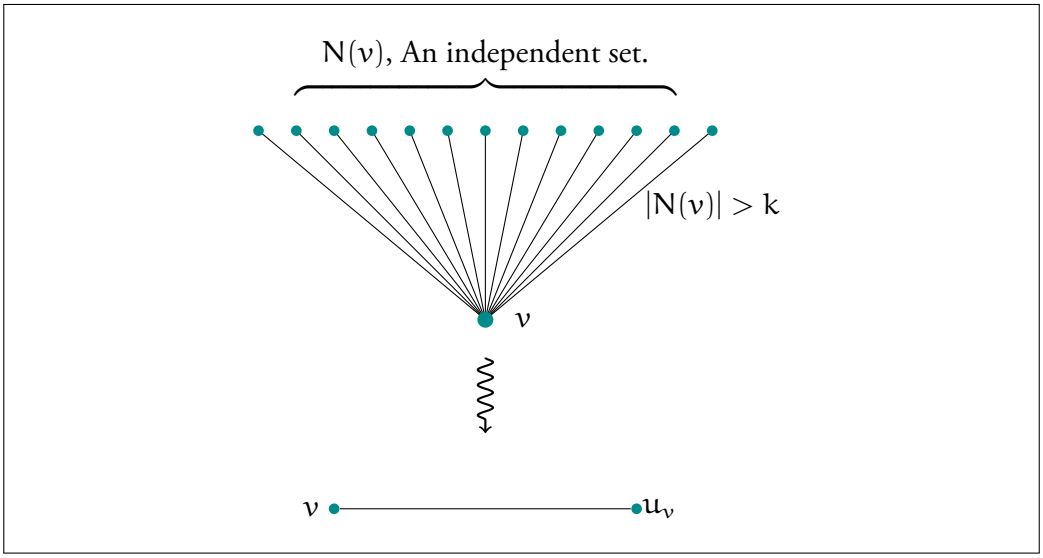


Figure 4.5: If a vertex has an independent neighborhood of size at least k, then at least one of the edges incident on the vertex is forced in any solution. This is recorded with a new vertex, and the independent neighborhood is deleted.

We find a maximum matching M_1 of H. As before, if $|M_1| > 2k$ then we can immediately answer NO, and so we can assume that $|M_1| \leqslant 2k$. If $|B| > |M_1| \cdot (k+1)$, then by the q-Expansion Lemma (with $q = (k+1)$), we can find in polynomial time nonempty vertex sets $S \subseteq A, T \subseteq B$ such that $H[S \cup T]$ contains $|S|$ $(k+1)$-stars, saturating S in H (and hence in G), and no vertex in T has a neighbor outside S. For every $u \in S$, any edge dominating set of size at most k picks at least one edge incident on u. Indeed, since the other endpoints of all edges incident on u form an independent set, an edge dominating set that picks no edges incident on u must pick at least $(k+1)$ distinct edges to dominate the edges of the star centered at u, and such a edge dominating set is no longer of size at most k. Notice that the presence of any edge incident on u in the edge dominating set also accounts for dominating all the edges on the star centered at u. Noting that T is an independent set, and there are no edges from T outside S, it can be verified that we may obtain an equivalent and smaller instance by removing all vertices in T and edges incident on them, and adding the edges (u, v_u) for all $u \in S$, where v_u is a new vertex. Note that the parameter remains unchanged.

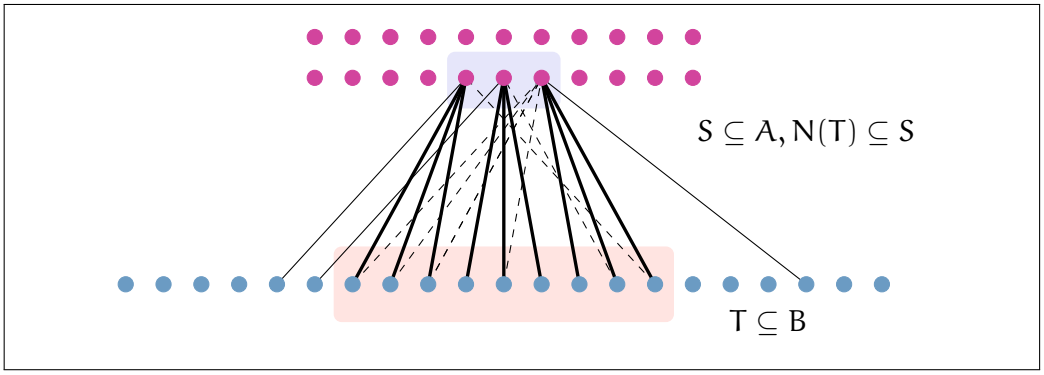


Figure 4.6: S and T obtained from the q-expansion lemma with $q = (k + 1)$. Here $k = 2$.

By repeating this procedure, we finally obtain a graph where $|B| < |M_1| \cdot (k + 1) = (2k \cdot k+1)$. Thus $G[A \cup B]$ is the kernel for the problem, and $|A \cup B| \leqslant 2k(k+1)+4k = 2k(k + 3)$:

Lemma 4.4. *Edge Dominating Set admits a kernel on* $2k(k + 3)$ *vertices when parameterized by solution size.*

The art of doing mathematics is finding that special case
that contains all the germs of generality.

David Hilbert

Each problem that I solved became a rule
which served afterwards to solve other problems.

Rene Descartes

The question of finding a feedback vertex set of size at most k is a well-studied problem in the kernelization context. Note that this is a special case of the problem of finding a $\mathcal{F}$-hitting set of size at most k. Indeed, if we use θ_2 to denote the graph with two vertices and a single edge between them, then note that with $\mathcal{F} = \{\theta_2\}$, a $\mathcal{F}$-hitting set corresponds exactly to a feedback vertex set. It turns out that a certain technique used in the kernelization algorithm for the feedback vertex set problem can be generalized into an useful subroutine in the kernelization program that we develop for the more general problem of finding a $\mathcal{F}$-hitting set of size at most k. The objective of this chapter is to introduce the generalized version of this technique in an isolated but interesting context. We note that the technique involves an application of Hall's theorem to simplify vertices of "high" degree in the input instance.

The particular context in which we wish to explore this technique is the Independent Feedback Vertex Set problem. It is a natural variant of the standard feedback vertex set problem, wherein we require that the graph induced on the solution be independent. The formal definition of the problem is the following:

Independent feedback vertex set

Input:	A graph G.
Parameter:	k
Question:	Does there exist a subset $S \subseteq V(G)$, such that $G \setminus S$ is a forest, $G[S]$ is an independent set, and $\lvert S\rvert \leqslant k$?

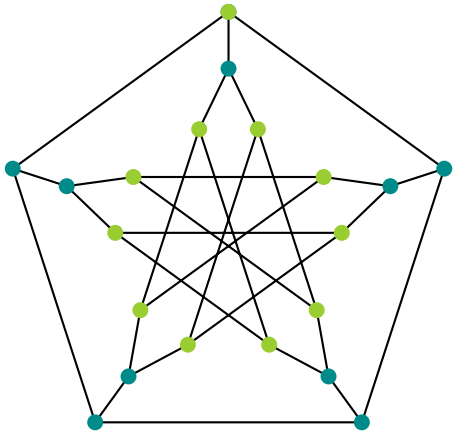


Figure 5.1: An example of an independent feedback vertex set of size eleven in a graph. Notice that there is a feedback vertex set of size five.

The problem is NP-complete by a simple reduction from feedback vertex set (see Lemma 5.1). In this chapter, we show that the problem admits a cubic kernel, and this also demonstrates that the problem is fixed-parameter tractable.

Remark. Note that any YES-instance of IFVS cannot contain a K_4 (a complete graph on four vertices) as a subgraph. More generally, the W_n (a wheel graph on n vertices) is a forbidden subgraph for this problem for odd values of n. We provide a brief explanation. Let $v_1, \ldots, v_n$ be the vertices on the outer rim of the wheel. Note that for any consecutive pair $\{v_i, v_{(i+1)}\}$, $1 \leqslant i < n$, at least one of the vertices $\{v_i, v_{(i+1)}\}$ must belong to any independent feedback vertex set. Further, because of the fact that the feedback vertex set is an independent set and the vertices $\{v_i, v_{(i+1)}\}$ share an edge, *exactly* one of them belong to any independent feedback vertex set. With these restraints, it is easy to check that if n is odd, then no feedback vertex set can be formed that is also independent.

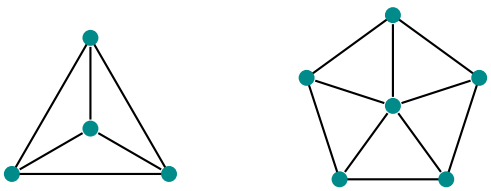


Figure 5.2: Forbidden subgraphs for YES instances of independent feedback vertex set.

Before we discuss further advances in the fixed-parameter tractability context, we present the proof that the problem is NP-complete.

Lemma 5.1. *Independent Feedback Vertex Set is* NP-complete.

Proof. The proof is by a reduction from Feedback Vertex Set. Let (G, k) be a feedback vertex set instance. Let H be the graph obtained from G by subdividing every edge. Formally,

- $V(H) = V(G) \cup \{v_e \mid e \in E(G)\}$ and
- $E(H) = \{(x, v_e), (v_e, y) \mid e = (x, y) \in E(G)\}$.

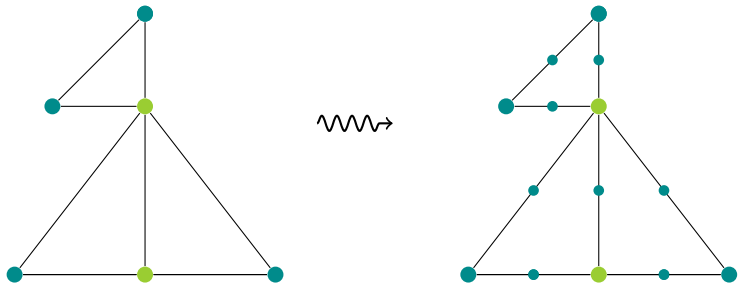


Figure 5.3: Reduction from feedback vertex set to independent feedback vertex set

We claim that the independent feedback vertex set instance (H, k) is equivalent to (G, k):

$\diamondsuit$ If S is a feedback vertex set of G, then notice that the vertices corresponding to S in H form an independent feedback vertex set of H, since no two vertices of G share an edge in H.

$\diamondsuit$ Let S be a feedback vertex set of H. For every $x \in S \subseteq V(H)$, consider $v_x \in V(G)$ given as follows:

$$v_x = \begin{cases} x & \text{if } x \in V(G) \\ z & \text{where } z \in N(x) \text{ is an arbitrarily chosen neighbor of } x \text{ in } H \end{cases}$$

Notice that if $x \notin V(G)$, then $N_H(x)$ consists of exactly two vertices, both of which correspond to vertices of G. In this case, every cycle passing through x must pass through both its neighbors. Therefore, x may be replaced by either of its neighbors in any feedback vertex set. This proves that the set $\{v_x \mid x \in S\}$ is a feedback vertex set of size at most k for G.

□

We now turn to the question of obtaining polynomial kernels. A natural starting point would be to "imitate" the procedures in place for Feedback Vertex Set. In what follows, we provide a detailed description of this approach — which turns out to be successful after some appropriate modifications.

5.1 A Cubic Kernel

The kernelization algorithm is described in the following two phases:

▷ First, the instance is simplified with respect to a *degree reduction procedure*. This involves the recursive application of reduction rules that transform an instance (G, k) of Independent Feedback Vertex Set into an equivalent instance (H, l) where $l \leqslant k$, and the maximum degree of H is at most $\mathcal{O}(k^2)$.

▷ Subsequently, we argue that if the maximum degree of a graph is bounded by Δ, then any YES-instance of Independent Feedback Vertex Set has at most $\mathcal{O}(k\Delta)$ vertices. Together with the degree reduction procedure, this argument leads us to a cubic kernel.

For each reduction rule described below, the input instance is an arbitrary but fixed instance denoted by (G, k).

We refer to a vertex $v \in G$ as *irrelevant* if it does not belong to any cycle, and *relevant* otherwise. Notice that isolated vertices and pendant vertices are irrelevant.

Reduction Rule 5.1 (Irrelevant Vertex Rule). *Let* X *denote the set of all irrelevant vertices in* G. *The reduced instance is* $(G \setminus X, k)$.

The soundness of Rule 5.1 is evident, since it is clear that an irrelevant vertex would never participate in a minimal solution.

We now describe an annnotation procedure that will be useful subsequently. At various stages during kernelization, we would like to encode the fact that a vertex v of the reduced instance never participates in a solution of the reduced instance. Towards this, we modify the instance to include $(k + 1)$ triangles that intersect precisely at one vertex, and are otherwise isolated from the remaining vertices of the

graph G. Formally, we add the vertices $\{x_1, y_1, z\}, \ldots, \{x_{k+1}, y_{k+1}, z\}$ to the graph, along with the adjacencies (x_i, y_i), (x_i, z) and (y_i, z) for all $1 \leqslant i \leqslant (k+1)$. Let $\triangle_i$ denote the triangle induced by the vertices (x_i, y_i, z). Notice that the instance $((G \cup \triangle_1 \cup \ldots \cup \triangle_{k+1}), k+1)$ is equivalent to the instance (G, k), and that any solution of $((G \cup \triangle_1 \cup \ldots \cup \triangle_{k+1}), k+1)$ contains the vertex z. We say *a vertex v is colored black* to mean that we introduce the edge (v, z). An *uncolored vertex* is a vertex that has not been colored black. Notice that if v is a vertex colored black, then v is not a part of any independent feedback vertex set of size at most k.

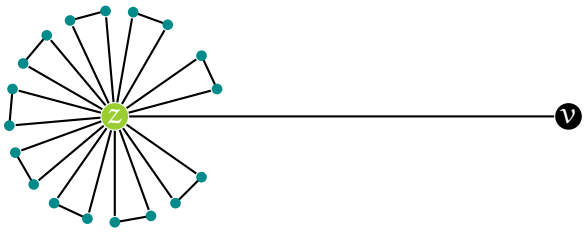


Figure 5.4: The operation of "coloring" a vertex v black.

Reduction Rule 5.2. *If G has a vertex x of degree two adjacent to vertices y and z, $y \neq x$ and $z \neq x$, and at least one of y or z also have degree two, then short circuit by removing x and joining y and z by a new edge (even if y and z were adjacent earlier).*

The proof of the soundness of Rule 5.2 is deferred to Lemma 5.2.

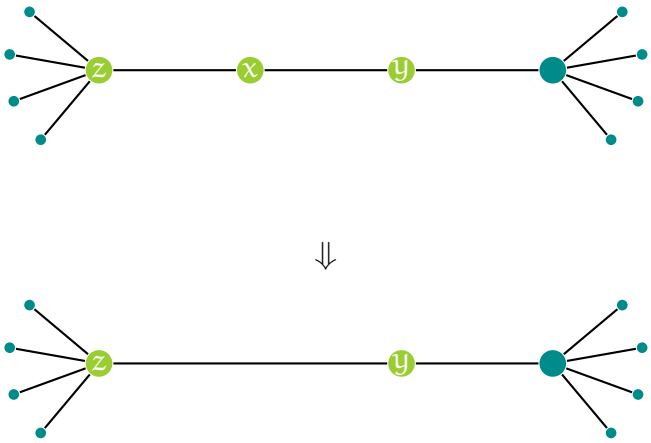


Figure 5.5: A reduction rule for adjacent degree two vertices.

Notice that after the application of Reduction Rule 5.2, we are left with a graph where every vertex of degree two is adjacent to vertices of degree greater than two. We now

ensure that the number of vertices of degree two "between" any pair of vertices of degree greater than two is also bounded, with the help of the following reduction rule.

Reduction Rule 5.3 (Intermediate Degree Two Vertices). *Let* (G, k) *be an instance of Independent Feedback Vertex Set. Let* $u_1, u_2, \ldots, u_r$ *be vertices such that*

$$N(u_1) = N(u_2) = \cdots = N(u_r) = \{x, y\},$$

and let $r > (k+2)$. *Let* H *be the graph obtained from* G *after the removal of vertices* $\{u_1, u_2, \ldots, u_r\}$ *and the inclusion of two new vertices* $\{u_a, u_b\}$, *and the edges*

$$(x, u_a), (x, u_b), (y, u_a), (y, u_b).$$

The vertices u_a *and* u_b *in* H *are colored black. The reduced instance is* (H, k).

Again, the proof of the soundness of Rule 5.3 is deferred to Lemma 5.2.

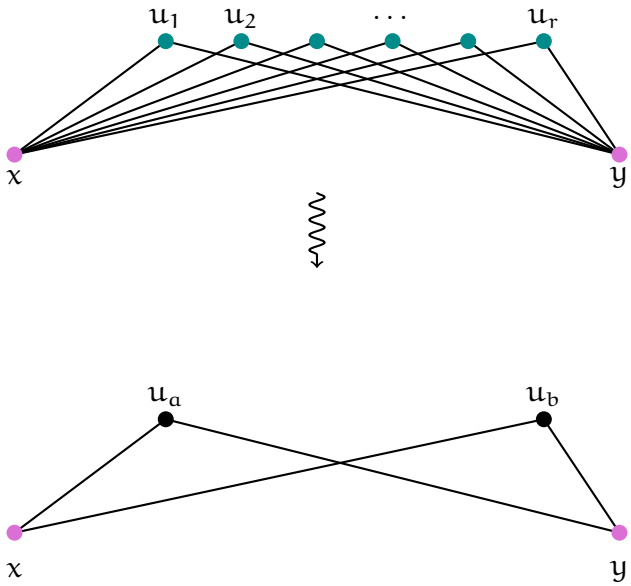


Figure 5.6: Removing more than $(k+2)$ vertices of degree two between a fixed pair of vertices of higher degree.

Our next reduction rule ensures that the total number of vertices of degree two in a reduced instance is bounded.

Reduction Rule 5.4 (Detecting Vertex Disjoint Cycles). *Consider the graph* H *constructed as follows:*

- $V(H) = V(G)$
- $(u, v) \in V(H)$ *if in* G*, there exist at least two vertices* x *and* y *of degree two with neighbors* u *and* v*.*

If the size of a maximum matching in H *is greater than* k*, then return that* G *does not have an independent feedback vertex set of size at most* k*.*

The proof of the soundness of Rule 5.4 is deferred to Lemma 5.2.

Next, we apply the "flower" rule, which ensures that vertices that are at the intersection of more than k cycles that are otherwise vertex-disjoint are "forced" into an independent feedback vertex set.

Definition 5.1. *Given a graph* G *and a vertex* $v \in V(G)$*, an* ℓ-flower passing through v *is a set of* ℓ *distinct cycles in* G*, each containing* v *and no two sharing any vertex other than* v*.*

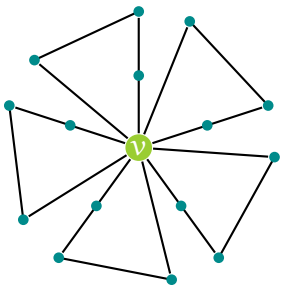


Figure 5.7: A 5-flower passing through the vertex v.

Reduction Rule 5.5 (Flower Rule). *Let* v *be a vertex at the center of a* $(k+1)$*-flower. Let* H *be the graph obtained by deleting* v *from* G *and coloring all neighbors of* v *black. The reduced instance is then* (H, k)*.*

Rule 5.5 is standard in the literature of feedback vertex set and is known to be sound (this is also easily verified) and in polynomial time [Tho10].

Our final rules are quite straightforward, we ensure that vertices with self-loops are forced in solutions and infeasible instances are detected to the extent possible:

Reduction Rule 5.6. *Let* (G, k) *be an instance of Independent Feedback Vertex Set.*

→ *If* G *has an uncolored vertex* x *with a self-loop then remove* x*, color all neighbors of* x *black, and decrease the parameter* k *by one. That is, let the resulting instance be* $(G \setminus \{x\}, k-1)$.

→ *If* G *has a black vertex* x *with a self-loop, then return that* G *does not have an independent feedback vertex set of size at most* k.

Reduction Rule 5.7. *Let* (G, k) *be an instance of Independent Feedback Vertex Set. If* $k = 0$ *and* G *is not a forest then return that* G *does not have an independent feedback vertex set of size at most* k.

Reduction rules 5.6 and 5.7 are trivially checked to be sound, and applicable in polynomial time.

We now turn to the proofs of soundness of rules 5.2,5.3 and 5.4. We note that rule 5.2 is a modification of a traditional "short circuiting" reduction rule usually applied to Feedback Vertex Set instances. Rules 5.3 and 5.4 are newly introduced for Independent Feedback Vertex Set to control the number of degree two vertices, since we are not able to delete them. That these reduction rules are polynomial time applicable is easy to check. Below, we show the correctness of these two rules, and also establish that there are at most $O(k^3)$ vertices of degree two in a reduced instance.

Lemma 5.2. *Reduction rules 5.2,5.3 and 5.4 are sound, and a graph reduced with respect to the rules 5.1 – 5.5 is either a* NO *instance or has* $O(k^3)$ *vertices of degree two.*

Proof. In this proof, we use G to refer to the graph corresponding to the input instance and H to refer to the graph corresponding to the reduced instance.

Reduction Rule 5.2: Let $v_1, \ldots, v_r$ denote a "chain" of degree two vertices, that is, (v_i, v_{i+1}) is an edge for every $i \in \{1, 2, \ldots, r-1\}$ and every v_i has degree two in G. Let x and y denote the neighbors of v_1 and v_r respectively. Recall that this reduction rule replaces $\{v_1, \ldots, v_r\}$ with a single vertex v that is adjacent to x and y.

Observe that any minimal independent feedback vertex set of G intersects $\{v_1, \ldots, v_r\}$ in at most one vertex. If the intersection is empty, then notice that there is nothing to prove, since the same subset is clearly an independent feedback vertex set of H. Further, if a minimal independent feedback vertex set contains a vertex v_i and neither x or y, then again there is nothing to prove (the same subset, with v_i replaced by

v is an independent feedback vertex set of H — independence is ensured with the premise that x and y do not belong to the independent feedback vertex set under consideration). Also observe that a *minimal* independent feedback vertex set of G will not contain any v_i if it contains at least one of x and y, since both x and y necessarily belong to any cycle passing through v_i, for any $1 \leqslant i \leqslant r$. This completes the argument for the soundness of Reduction Rule 5.2.

Reduction Rule 5.3: Note that H is a subgraph of G except that u_a and u_b are colored black, and the vertices $\{u_a, u_b, x, y\}$ form a cycle of length four. Thus, any independent feedback vertex set of G that contains x or y is also an independent feedback vertex set of H. We now show that *any* independent feedback vertex set of G of size at most k contains at least one of x or y. Trivially, any independent feedback vertex set S of size at most k can contain at most k vertices from the subset:

$$\{u_1, u_2, \ldots, u_r\}.$$

Thus, if $r > (k+2)$, then there exist i and j for which $G \setminus S$ contains u_i and u_j. Since the vertices

$$x - u_i - y - u_j - x$$

form a cycle, it is clear that S must contain at least one of x or y.

Now consider S, and independent feedback vertex set of H. It is straightforward that S contains at least one of x or y. We only need to demonstrate that any cycle passing through the vertices $\{u_1, u_2, \ldots, u_r\}$ necessarily contains both x and y. This is evident: since each u_i is a vertex of degree two with neighbors x and y, and any vertex on a cycle is of degree two, it is clear that any cycle passing through u_i contains x and y. Therefore, every cycle passing through any u_i is hit by S, since S contains at least one of x or y. This concludes the proof of soundness for Reduction Rule 5.3.

Reduction Rule 5.4 We show that a matching of size r in the auxiliary graph H (see Rule 5.4) corresponds to r vertex-disjoint cycles in G. It then follows that if $r > k$, then there is no independent feedback vertex set of size at most k, and the instance is a NO instance — this also implies the correctness of the rule.

Notice that every edge (u, v) in H corresponds to a four-cycle in G containing u and v. The other two vertices on the cycle have degree two in G, and therefore do not participate in any other cycles. Now, consider the collection of cycles corresponding to the edges of a matching. Clearly, since the edges of a matching do not share any

endpoints, the cycles do not share any of the edge endpoints, and since we argued that remaining participants of these cycles can belong to at most one cycle, it is clear that all these cycles are mutually vertex disjoint, as desired.

A Cubic Bound on the Number of Degree Two Vertices Consider again the auxiliary graph H obtained in Reduction Rule 5.4. Using arguments similar to those in the proof of the soundness of Reduction Rule 5.4, it is easily verified that any r-star in H corresponds to a r-flower in G. Therefore, since G is reduced with respect to the flower rule (Reduction Rule 5.5), the maximum degree of any vertex in H is at most k.

Since the maximum matching of H is at most k, the total number of edges in H is clearly bounded by

$$k + 2k(k),$$

where k counts the edges in the matching, and $2k$ is the number of matched vertices, so that the second term counts all remaining edges (recall that the maximum degree is at most k).

Since the instance is reduced with respect to Reduction Rule 5.3, every edge in G corresponds to at most $k + 2$ vertices of degree two. Therefore, the total number of degree two vertices in G is at most:

$$(k + 2k(k))(k + 2) = O(k^3).$$

This completes the proof of the bound on the number of degree two vertices of a reduced instance.

□

Lemma 5.3. *Given an undirected multi graph G on m edges and a positive integer k, in $O(mk)$ time we can*

1. *produce a multi-graph H where at least $(n-t)$ vertices have degree at least three, where $t = O(k^3)$, and no vertex is at the center of a r-flower for $r > k$*

2. *and compute a positive integer l,*

such that (G, k) and (H, l) are equivalent instances of Independent Feedback Vertex Set.

Remark 5.1. *Our inability to completely eliminate vertices of degree two is one of the key points of difference between the Independent Feedback Vertex Set and Feedback Vertex Set problems.*

At this point, we abuse notation and continue to use (G, k) to indicate the instance obtained from G after it is reduced with respect to reduction rules 5.1 – 5.5. For our next argument we need to find, for every vertex v, a subset of vertices that hit all cycles passing through v.

Further, we require this subset to exclude v and be of size polynomial in k. For a given $v \in V(G)$, such a hitting set $H_v \subseteq V(G) \setminus \{v\}$ can be found in polynomial time due to the following Theorem:

Theorem 5.1 ([Tho10] Corollary 2.1). *Let v be a vertex of a graph G, and let there be no self-loop at v. If there is no $(k+1)$-flower passing through v, then there exists a set $X \subseteq V(G) \setminus \{v\}$ of size at most $2k$ which intersects every cycle that passes through v, and such a set can be found in polynomial time.*

Reduction Rule 5.8. *Having established the choice of H_v for all v, we are now ready to describe the q-expansion Rule with $q = (k+2)$.*

Given an instance (G, k), and a family of sets H_v, we show that if there is a vertex v with degree more than $2k+(k+2)k$, then we can reduce its degree to at most $2k+(k+2)k$ by repeatedly applying the q-expansion lemma with $q = (k+2)$. Consider the graph $\texttt{Greduction} \setminus H_v$. Let the components of this graph that contain a neighbor of v be $C_1, C_2, \ldots, C_r$. Note that v cannot have more than two neighbors into any component, since this would imply a cycle passing through v that lies outside H_v. Also note that if (G, k) is a YES-instance, then at most k of the C_i's can contain cycles, and the rest induce trees. Without loss of generality, let $C_1, C_2, \ldots, C_p$ be the components that contain cycles, and let $C_{p+1}, \ldots, C_r$ be the remaining. We also have the understanding that $p \leqslant k$. In the discussion that follows, all statements about components are made in the context of those that do not contain cycles; and we will ignore the first p components. We say that a component C_i is adjacent to H_v if there exists a vertex $u \in C_i$ and $w \in H_v$ such that $(u, w) \in E(G)$. Next we show that components that are trees are adjacent to H_v by demonstrating that if they are not, then all vertices in such a component are irrelevant. Recall a vertex is irrelevant if there is no cycle that contains it. Consider a vertex u in a component C that is not adjacent to H_v. Since $G[C \cup \{v\}]$ does not contain any cycles, clearly any $u \in C$ cannot

belong to a cycle if C is not adjacent to H_v. This renders the vertex u irrelevant. Since the graph is assumed to be reduced with respect to the irrelevant vertex rule, we have shown that tree component is adjacent to H_v.

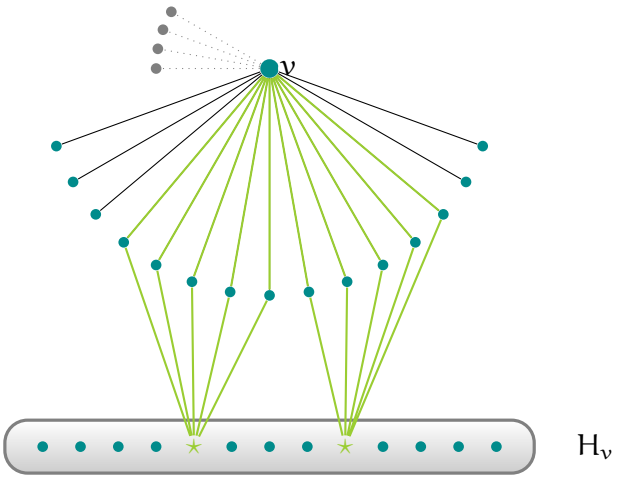


Figure 5.8: A picture of two $(k+2)$ stars for $k = 3$.

Now, consider a bipartite graph $\mathcal{G}$ with vertex bipartitions H_v and C. Here $C = \{c_1, \ldots, c_s\}$ contains a vertex c_i corresponding to each tree component C_i (notice that s would amount to $(r - p)$). We add an edge (v, c_i) if there is a vertex $w \in C_i$ such that $\{v, w\} \in E(G)$.

Now, v has at most h_v edges to vertices in H_v (since G is a simple graph). Since v has at most one edge to each C_i, it follows that if $d(v) > h_v + (k+2)h_v + p$, then the number of components that are trees is more than $(k+2)h_v$. Now by applying q-expansion lemma with $q = (k+2)$, $A = H_v$, and $B = C$, we find a subset $S \subseteq H_v$ and $T \subseteq D$ such that S has $|S|$ $(k+2)$-stars in T and $N(T) = S$.

The reduction rule involves deleting edges of the form (v, u) for all $u \in C_i$ such that $c_i \in T$ (see Figure 5.9), and adding a pair of subdivided edges between v and w for all $w \in S$ (notice that this creates a cycle of length four involving v and w) whenever such a cycle does not already exist (see Figure 5.10).

Formally, consider $w \in S$ for which there is no four-cycle involving w, v, and a pair of vertices that have been annotated black. For all such w, we add vertices $\{w_a, w_b\}$ and the edges (v, w_a), (v, w_b), (w, w_a) and (w, w_b). Finally, for every $w \in S$, we color the vertices w_a and w_b black, and for all $c_i \in T$, we color all neighbors of v in the component C_i black.

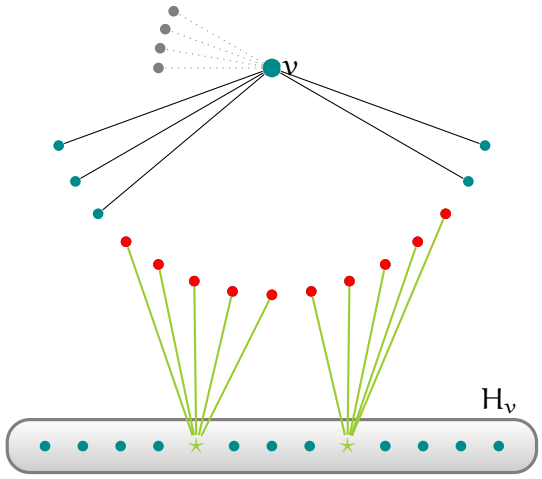


Figure 5.9: The first part of Reduction Rule 5.8. Involves deleting edges incident on v with their other endpoints in T. Vertices labeled $\star$ belong to S and vertices colored red are in T.

We note that this is done to ensure that no independent feedback vertex set in the reduced instance intersects non-trivially with $\{w_a, w_b\}$ for any $w \in S$. All neighbors of v are also blacklisted, for reasons that will be clear presently.

This completes the description of the q-expansion reduction rule with $q = (k+2)$. Let G_R be the graph obtained after applying the reduction rule. The following lemma shows the correctness of the rule.

Lemma 5.4 (Soundness). *Let* G, S *and* v *be as above and* G_R *be the graph obtained after applying the* q*-expansion rule. Then* (G, k) *is an yes instance of Independent Feedback Vertex Set if and only if* (G_R, k) *is an yes instance of Independent Feedback Vertex Set.*

Proof. We first show that if G_R has an independent feedback vertex set Z of size at most k, then the same vertex subset Z when considered in G hits all cycles in G and is independent. We first argue that — by construction — either $v \in Z$ or $S \subseteq Z$. Indeed, let $w \in S$, and assume that $v \notin Z$. Observe that $w_a \notin Z$, and $w_b \notin Z$ (recall that w_a, w_b are colored black). Thus, if $v \notin Z$, then $w \in Z$, since Z must contain at least one vertex from the four-cycle $\{v, w_a, w_b, w\}$. Since this is true of all $w \in S$, we have that if $v \notin Z$, then $S \subseteq Z$.

Suppose $v \in Z$. Notice that $G_R \setminus \{v\}$ is the same as $G \setminus \{v\}$. Therefore $Z \setminus \{v\}$, an independent feedback vertex set of $G_R \setminus \{v\}$ is also a feedback vertex set of $G \setminus \{v\}$. Further, Z is also an *independent* feedback vertex set of G: all we need to establish is

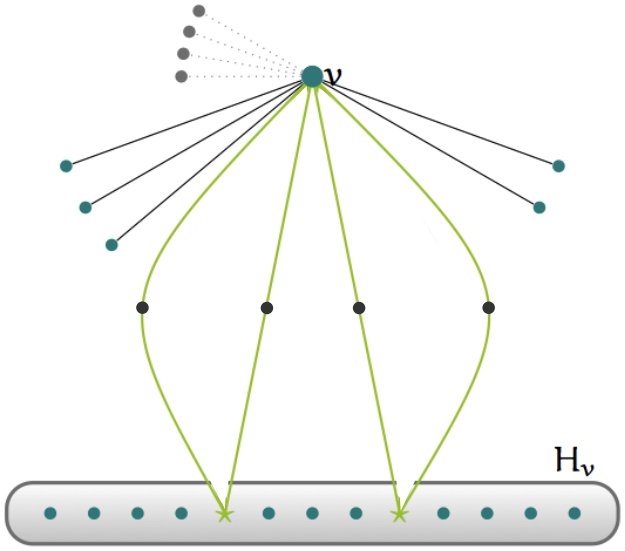


Figure 5.10: The first part of Reduction Rule 5.8. We add a cycle of length four between vertices in S and v.

that Z does not contain any neighbors of v. Notice that this is true, since every u that is a neighbor of v in G is either a neighbor of v in G_R, or is colored black. Therefore, if $v \in Z$, then Z does not contain any neighbor of v. This shows that Z is an independent feedback vertex set of size at most k of G. The case when $S \subseteq Z$ is similar.

To prove that an independent feedback vertex set of size at most k in G implies an independent feedback vertex set of size at most k in G_R, it suffices to prove that whenever G admits an independent feedback vertex set of size at most k, G also has an independent feedback vertex set of size at most k that contains either v or all of S, and none of the neighbors of v in components corresponding to vertices in T.

We first establish that *any* independent feedback vertex set of size at most k contains either v or all of S. Note that this will account for our first requirement. Consider an independent feedback vertex set W that does not contain v, and omits at least one vertex from S. Let x be a vertex in S that is not in W. Let $x_1, \ldots, x_{k+2}$ denote the neighbors of x in distinct components that formed the $(k+2)$-star incident on x. Since $|W| \leqslant k$, there exist i, j such that $x_i \notin W$ and $x_j \notin W$. This implies that in $G \setminus W$, the vertices $\{x, v, x_i, x_j\}$ induce a four-cycle. This contradicts the assumption that W is a feedback vertex set.

Now, we show that if there exists an independent feedback vertex set of size at most k,

there exists one that contains none of the neighbors of v in components corresponding to vertices in T. Let W an independent feedback vertex set of size at most k, and let $X \subseteq W$ be vertices in W that are neighbors of v in components corresponding to vertices in T. Clearly, if $v \in W$, then $X = \emptyset$, because W is an *independent* feedback vertex set. The other situation is when $v \notin W$, and therefore, $S \subseteq W$. Consider $W \setminus X$. We claim that $W \setminus X$ is also an independent feedback vertex set of size at most k. That the size of $W \setminus X$ is at most k is clear, since $|W| \leqslant k$ by assumption, and it also obvious that $W \setminus X$ is independent. To show that $W \setminus X$ is a feedback vertex set, observe that any cycle that passes through the vertices in X must necessarily intersect S. This follows from the fact that the neighborhood of X is contained in $\{v\} \cup S$, and if the cycle does not intersect S at all, then it is easy to see that v is a cut vertex of the proposed cycle: this is a contradiction since it is impossible for a cycle to admit a cut vertex. This concludes the proof. □

Notice that the reduction rule for bounding the degree applies as long as there are vertices of degree more than $2k+(k+2)k$, since $h_v+(k+2)h_v+p \leqslant k+(k+2)k+k$ (recall that $h_v \leqslant k$ and p is the number of components that contain cycles, and thus $p \leqslant k$). This concludes the first part of the description. We now turn to the claim that if Δ is a bound on the maximum degree of the graph, then there are no more than $k\Delta$ vertices in G if (G, k) is a YES-instance of the problem.

Lemma 5.5. *Let G be a graph on n vertices that has at least $(n-t)$ vertices of degree at least δ, and maximum degree Δ. Then the size of the minimum feedback vertex set for G is greater than $\frac{n(\delta-2)-t\delta}{2(\Delta-1)}$.*

Proof. Let F be a minimum feedback vertex set for G, and let E_F be the set of edges with at least one end point in F. Since $G \setminus F$ is a forest, there are at most $n - |F| - 1$ edges in $G \setminus F$ (See Figure 5.11).

Thus

$$\Delta|F| \geqslant |E_F| \geqslant |E(G)| - n + |F| + 1 > \frac{(n-t)\cdot(\delta)}{2} - n + |F|$$

which implies

$$(\Delta - 1)|F| > \frac{n(\delta-2)-t\delta}{2},$$

and further,

$$|F| > \frac{n(\delta-2)-t\delta}{2(\Delta-1)}.$$

□

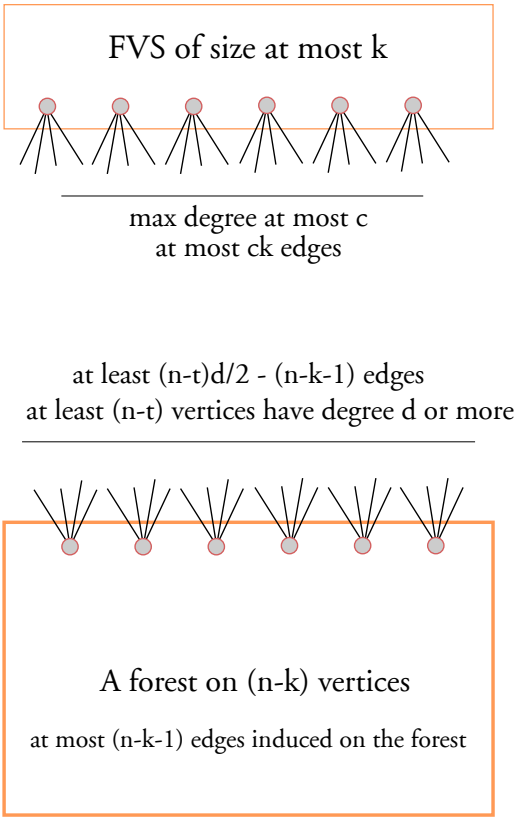


Figure 5.11: Two ways of counting the edges that have one endpoint each in an independent feedback vertex set, and the remaining forest.

Now, consider a graph G is a graph for which Δ is bounded above by an arbitrary but fixed positive integer c. This gives us following result.

Lemma 5.6. *Let $\mathcal{G}$ be the class of graphs such that the maximum degree of every graph $G \in \mathcal{G}$ is bounded by Δ. Then Independent Feedback Vertex Set has a kernel on $O(k\Delta)$ vertices and edges.*

Proof. Given a graph $G \in \mathcal{G}$, we first apply Lemma 5.3 and obtain a graph H where at least $(n - t)$ vertices have degree at least three. Using Lemma 5.5 we know that every minimum feedback vertex set of H must have size at least

$$\frac{n(\delta - 2) - t\delta}{2(\Delta - 1)},$$

where δ is at least three. This simplifies to:

$$\frac{(n - t\delta)}{2(\Delta - 1)}.$$

Hence if $k < (n - t\delta)/2(\Delta - 1)$ then we return that G does not have a feedback vertex set of size at most k. Thus $k \geqslant (n - t\delta)/2(\Delta - 1)$ and hence $n \leqslant 2k(\Delta - 1) + t\delta$. Again, from Lemma 5.5, recall that $t = (k + 2k(k))(k + 2)$, thus we have that:

$$n \leqslant 2k(\Delta - 1) + (k + 2k(k))(k + 2)\delta,$$

or equivalently:

$$n \leqslant 2k(\Delta - 1) + 3(k + 2k(k))(k + 2),$$

with our understanding that $\delta \geqslant 3$.

Furthermore, the number of edges in the graph with feedback vertex set of size at most k and with maximum degree Δ is bounded by $k\Delta + (n - k - 1)$ — this expression accounts for, respectively, edges incident on the vertices in the feedback vertex set and the edges induced by the forest. Hence, if the number of edges in G' is more than $3k\Delta - 3k + (k + 2k(k))(k + 2)$, we return that G does not have a feedback vertex set of size at most k. In all other cases we have that $|V(G')| = O(k\Delta + k^3)$ and $|E(G')| = O(k\Delta + k^3)$. Thus the lemma follows. □

Lemma 5.6 combined with the degree reduction procedure described previously lead us to the conclusion that we can, in polynomial time, either bound the number of vertices *and* edges of the given Independent Feedback Vertex Set instance by $\mathcal{O}(k^3)$, or report that the input is a NO instance of the problem. Thus, we have:

Theorem 5.2. *Independent Feedback Vertex Set has a kernel on* $\mathcal{O}(k^3)$ *vertices and edges.*

The sculptor produces the beautiful statue by chipping away
such parts of the marble block as are not needed — it is a process of elimination.

Elbert Hubbard

That's the secret to life — replace one worry with another.

Charlie Brown

The notion of protrusions and finite integer index is central to all the kernelization algorithms in this thesis. In this chapter, we explore in brief the theorems that we will find useful in subsequent discussions. We refer the reader to [BFL+09, dF97] for a comprehensive exposition.

6.1 The Blueprint

In the context of problems on graphs, there are three central ideas that form the undercurrent of all protrusion-based reduction rules:

▷ describing an equivalence that classifies all instances of a problem in an useful manner,

▷ the ability to easily identify, given a problem, whether the said equivalence has a finite number of equivalence classes (finite index),

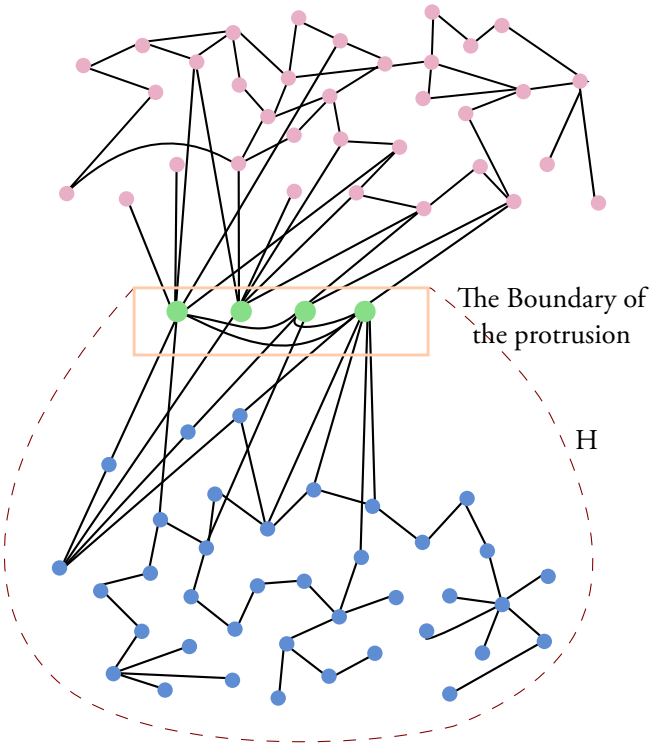
▷ given an instance of a problem, finding large subgraphs that "can be replaced" with smaller subgraphs that are equivalent to the original.

It is quite non-trivial to describe the exact circumstances in which a subgraph may be replaced. This was formally captured by the notion of a *protrusion*: specifically, a r-protrusion in a graph G is a subgraph H such that:

♢ the number of vertices in H that have neighbors in $G \setminus H$ is at most r, and,

♢ the treewidth of H is at most r.

The *size* of the protrusion is the number of vertices in it, that is, $|V(H)|$. The vertices in H that have neighbors in $G \setminus H$ comprise the *boundary* of H (See Figure 6.1). The notion of a protrusion was introduced in [BFL+09].



A Protrusion of size 40,
with a boundary of four vertices and treewidth two.

Figure 6.1: An example of a protrusion.

Informally, the protrusion H may be thought of as a part of the graph that is separated from the "rest of the graph" by a small-sized separator, and everything about H may be understood in terms of the graph induced on H itself and the limited interaction it has with $G \setminus H$ via its boundary vertices. If the size of the protrusion is large, we may want to replace it with another graph X that is much smaller, and whose behavior with respect to $G \setminus H$ is identical to H in the context of the problem that we are studying. Specifically, we would like that the solution to the problem in question does not change after we have made the replacement (or changes in a controlled manner that can be tracked as we make these replacements).

This motivates us to define an equivalence that captures the essence of what we hope to do in terms the replacement. We would like to declare H equivalent to X if the size of the solution of G and $(G \setminus H) \cup^* X$ is exactly the same, where $\cup^*$ is some notion of replacement that we have not defined precisely yet. Notice, however, that a natural notion of replacement would leave the boundary vertices intact and perform a cut-and-paste on the rest of H. This is precisely what the protrusion based reduction rules do. These, combined with some combinatorial properties of planar graphs or more generally on graphs excluding a fixed graph H as a minor, result in polynomial and in most cases linear kernel for variety of problems. In what follows we give the required definitions and set up the machinery for the later use.

6.2 Some Definitions

Given a graph $G = (V, E)$ and $S \subseteq V$, we define $\partial_G(S)$ as the set of vertices in S that have a neighbor in $V \setminus S$. For a set $S \subseteq V$ the neighborhood of S is $N_G(S) = \partial_G(V \setminus S)$. When it is clear from the context, we omit the subscripts. We now define the notion of a *protrusion*.

Definition 6.1. [r-**protrusion**] *Given a graph* $G = (V, E)$*, we say that a set* $X \subseteq V$ *is an* r-protrusion *of* G *if* $|\partial(X)| \leqslant r$ *and* $tw(G[X]) \leqslant r$.

For an r-protrusion X, the vertex set $X' = X \setminus \partial(X)$ is a *restricted* r*-protrusion*. The set X' is the restricted protrusion of X and X is the protrusion of X'.

We now define the notion of t-*boundaried graphs* and various operations on them.

Definition 6.2. [t-**Boundaried Graphs**] *A* t*-boundaried graph is a graph* $G = (V, E)$ *with* t *distinguished vertices, uniquely labeled from* 1 *to* t*. The set* $\partial(G)$ *of labeled vertices is called the boundary of* G*. The vertices in* $\partial(G)$ *are referred to as boundary vertices or terminals.*

For a graph $G = (V, E)$ and a vertex set $S \subseteq V$, we will sometimes consider the graph $G[S]$ as the $|\partial(S)|$-boundaried graph with $\partial(S)$ being the boundary.

Definition 6.3. [**Gluing by** $\oplus$] *Let* G_1 *and* G_2 *be two* t*-boundaried graphs. We denote by* $G_1 \oplus G_2$ *the* t*-boundaried graph obtained by taking the disjoint union of* G_1 *and* G_2 *and identifying each vertex of* $\partial(G_1)$ *with the vertex of* $\partial(G_2)$ *with the same label; that is, we glue them together on the boundaries. In* $G_1 \oplus G_2$ *there is an edge between two labeled vertices if there is an edge between them in* G_1 *or in* G_2*. (See Figure 6.2.)*

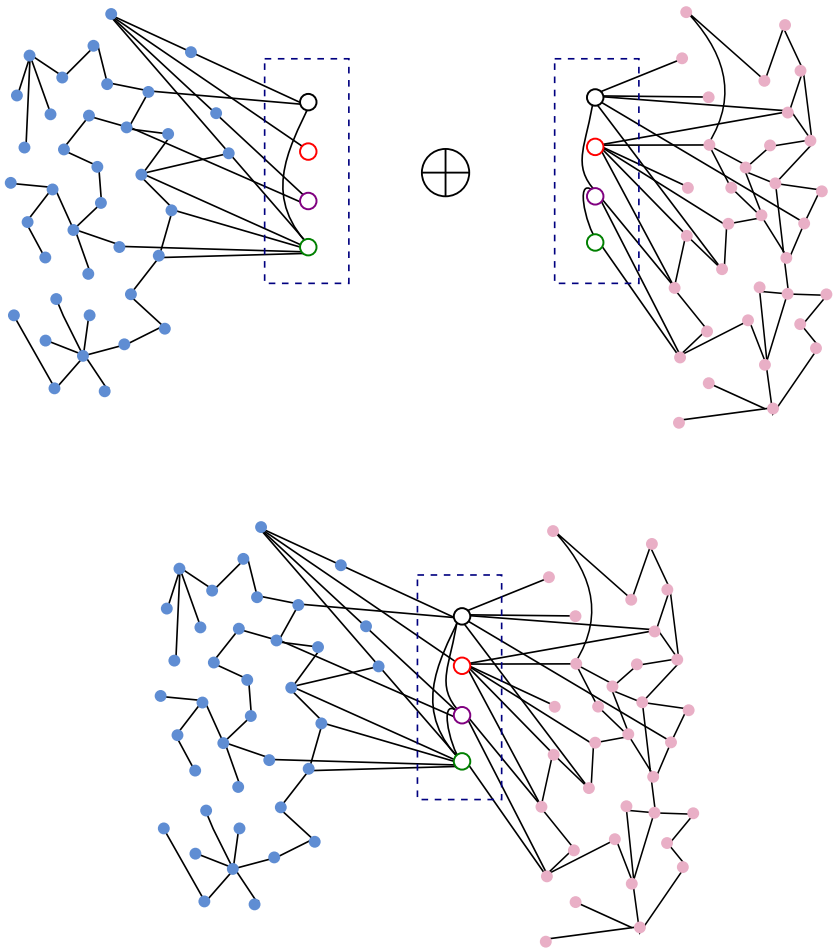


Figure 6.2: Gluing two t-boundaried graphs by $\oplus$. Note how the graph induced on the boundary has edges from both the graphs being glued together.

Definition 6.4. [**Legality**] *Let $\mathcal{G}$ be a graph class, G_1 and G_2 be two t-boundaried graphs, and $G_1, G_2 \in \mathcal{G}$. We say that $G_1 \oplus G_2$ is legal with respect to $\mathcal{G}$ if the unified graph $G_1 \oplus G_2 \in \mathcal{G}$. If the class $\mathcal{G}$ is clear from the context we do not say with respect to which graph class the operation is legal.*

Definition 6.5. [**Replacement**] *Let $G = (V, E)$ be a graph containing an r-protrusion X. Let X' be the restricted protrusion of X and let G_1 be an r-boundaried graph. The act of replacing X' with G_1 corresponds to changing G into $G[V \setminus X'] \oplus G_1$. Replacing $G[X]$ with G_1 corresponds to replacing X' with G_1. (See Figure 6.3.)*

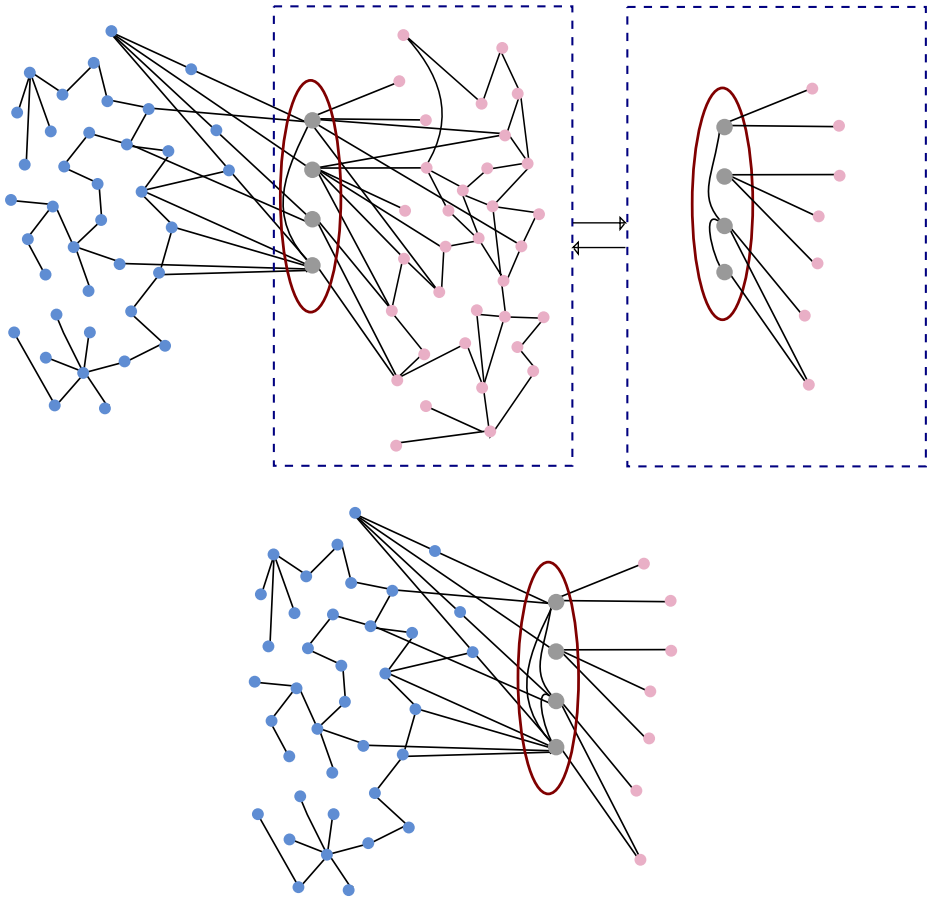


Figure 6.3: An example of replacing a protrusion with a new t-boundaried graph.

Definition 6.6. *For a parameterized problem Π on a graph class $\mathcal{G}$ and two t-boundaried graphs G_1 and G_2, we say that $G_1 \equiv_\Pi G_2$ if there exists a constant c such that for all t-boundaried graphs G_3 and for all k: (a) $G_1 \oplus G_3$ is legal if and only if $G_2 \oplus G_3$ is legal; (b) $(G_1 \oplus G_3, k) \in \Pi$ if and only if $(G_2 \oplus G_3, k + c) \in \Pi$.*

Definition 6.7. [**Finite Integer Index**] *We say that a parameterized problem* Π *has finite integer index in a graph class* $\mathcal{G}$ *if for every* t *there exists a finite set* $\mathcal{S}$ *of* t*-boundaried graphs such that* $\mathcal{S} \subseteq \mathcal{G}$ *and for any* t*-boundaried graph* G_1 *there exists* $G_2 \in \mathcal{S}$ *such that* $G_2 \equiv_\Pi G_1$*. Such a set* $\mathcal{S}$ *is called a set of representatives for* (Π, t)*.*

Note that for every t, the relation $\equiv_\Pi$ on t-boundaried graphs is an equivalence relation. A problem Π is finite integer index (FII), if and only if for every t, $\equiv_\Pi$ is of finite index, that is, has a finite number of equivalence classes.

The parameterized versions of many fundamental optimization problems have finite integer index, including problems like Dominating Set, r-Dominating Set, q-Threshold Dominating Set, Efficient Dominating Set, Vertex Cover, Connected r-Dominating Set, Connected Vertex Cover, Minimum Maximal Matching, Connected Dominating Set, Almost Outerplanar, Feedback Vertex Set, Cycle Domination, Edge Dominating Set, Clique Transversal, Independent Set, r-Scattered Set, Min Leaf Spanning Tree, Induced Matching, Triangle Packing, Cycle Packing, and Maximum Full-Degree Spanning Tree [BFL+09, dF97]. Both generic Vertex-$\mathcal{H}$-Packing and Vertex-$\mathcal{S}$-Packing also have finite integer index [BFL+09].

Examples of problems not having finite integer index are Longest Path, Longest Cycle, Maximum Cut, Minimum Covering by Cliques, Independent Dominating Set, and Minimum Leaf Out-branching [dF97].

6.3 Examples of proving FII

In this section we give couple of examples that will demonstrate how one can show a problem to be FII [BFL+09, dF97].

6.3.1 An Example: Independent Set

Lemma 6.1. Independent Set *has FII.*

Proof. Throughout the proof we fix t, that is, the number of vertices in the boundary of the graph. Given a t-boundaried graph G with boundary set X, we define a function

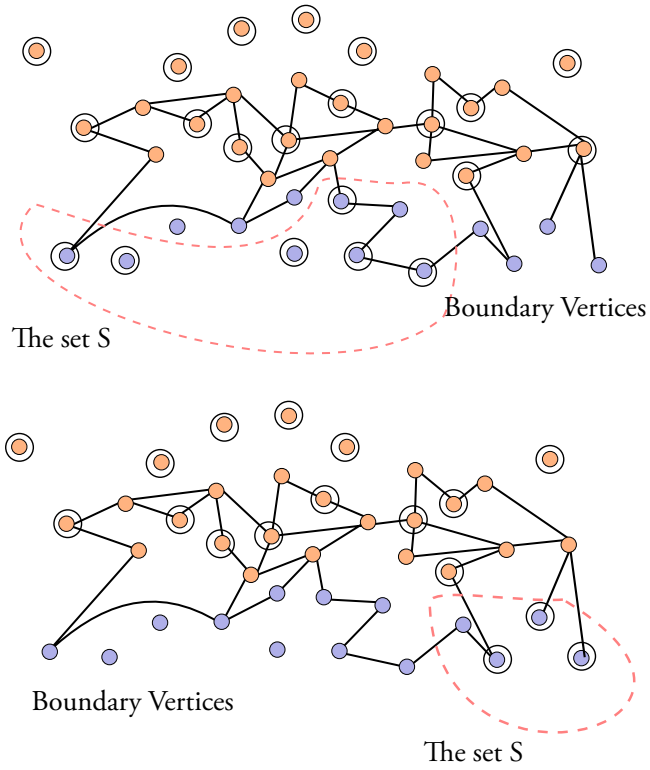


Figure 6.4: Illustrating the behavior of the function f_G. The number of doubly-circled vertices vertices correspond to the value of $f_G(S)$, where S is the subset of the boundary indicated by the dotted periphery.

$f_G : 2^X \to \mathbb{N}$ as follows (see Figure 6.4):

$$f_G(S) := \max_{\substack{I \text{ is an independent set of } G \\ I \cap X \subseteq S}} |I|.$$

For a fixed constant c, let 1_c be a function from 2^X to $\{c\}$ which uniformly defines c for every element in the domain. We make a couple of remarks about the function f_G which will be useful later in the proof.

1. For any $G, S, S' \subseteq X$, $|f_G(S) - f_G(S')| \leqslant t$.

2. For any $S \subseteq X$ we have that $f_G(\emptyset) \leqslant f_G(S) \leqslant f_G(\emptyset) + t$. (Monotonicity of subsets.)

Note that the properties above follow directly from the definition of f_G. Now, let $\mathbb{T} = \{g \mid g : 2^X \to \{0, 1, \ldots, t\}\}$. Now observe that for any t-boundaried graph G with boundary set X, there exists a function $g \in \mathbb{T}$ such that $f_G := 1_c + g$, where c is the size of the maximum independent set in $G \setminus X$. We say that two t boundaried graphs G_1 and G_2 are equivalent with respect to $\equiv_{\mathbb{T}}$ if there exists a function $g \in \mathbb{T}$ such that $f_{G_1} := 1_{c_1} + g$ and $f_{G_2} := 1_{c_2} + g$. Here, c_1 and c_2 are the sizes of the maximum independent set of $G_1 \setminus X$ and $G_2 \setminus X$, respectively. It is easy to observe that $\equiv_{\mathbb{T}}$ is an equivalence relation. Furthermore, since the size of $\mathbb{T}$ is upper bounded by $2^{2^{t+1}}$, we have that $\equiv_{\mathbb{T}}$ has finite index.

Recall that a problem Π is finite integer index, if and only if for every t, $\equiv_\Pi$ is of finite index, that is, has a finite number of equivalence classes. For $\Pi =$Independent Set, we will show this by showing that $\equiv_{\mathbb{T}}$ refines $\equiv_\Pi$. That is, if $G_1 \equiv_{\mathbb{T}} G_2$ then $G_1 \equiv_\Pi G_2$.

Claim 6.1. *If* $G_1 \equiv_{\mathbb{T}} G_2$ *then* $G_1 \equiv_\Pi G_2$.

Proof. For two t-boundaried graphs G_1 and G_2, recall that $G_1 \equiv_\Pi G_2$ if there exists a constant c such that for all t-boundaried graphs G_3 and for all k: $(G_1 \oplus G_3, k) \in \Pi$ if and only if $(G_2 \oplus G_3, k + c) \in \Pi$.

Let c_1 and c_2 be the sizes of the maximum independent set of $G_1 \setminus X$ and $G_2 \setminus X$, respectively. Without loss of generality assume that $c_1 \leqslant c_2$. For the purpose of proof choose $c := c_2 - c_1$. Now $(G_1 \oplus G_3, k) \in \Pi$ means that $G_1 \oplus G_3$ has an independent set of size k. Let this set be I_1 and its intersection with X be $X \cap I_1$. Now we know that G_2 has an independent set I_2 of size $g(X \cap I_1) + c_2$ such that its intersection with the boundary X is contained in $X \cap I_1$. This implies that $I_2 \cup (I_1 \cap (V(G_3) \setminus X))$ is an independent set in $G_2 \oplus G_3$. The size of this independent set is:

$$\begin{aligned} |I_2 \cup (I_1 \cap (V(G_3) \setminus X))| &\geqslant g(X \cap I_1) + c_2 + k - (|I_1 \cap V(G_1)| \\ &\geqslant k + c_2 + g(X \cap I_1) - (g(X \cap I_1) + c_1) \\ &= k + c_2 - c_1 \\ &= k + c \end{aligned}$$

This shows that $(G_2 \oplus G_3, k + c) \in \Pi$. We can similarly show the reverse direction, that is, if $(G_2 \oplus G_3, k + c) \in \Pi$ then $(G_1 \oplus G_3, k) \in \Pi$. This proves the claim. (see Figure 6.5) □

The above claim shows that $\equiv_{\mathbb{T}}$ indeed refines $\equiv_\Pi$ and thus has finite index. This proves that Indepndent Set has FII. □

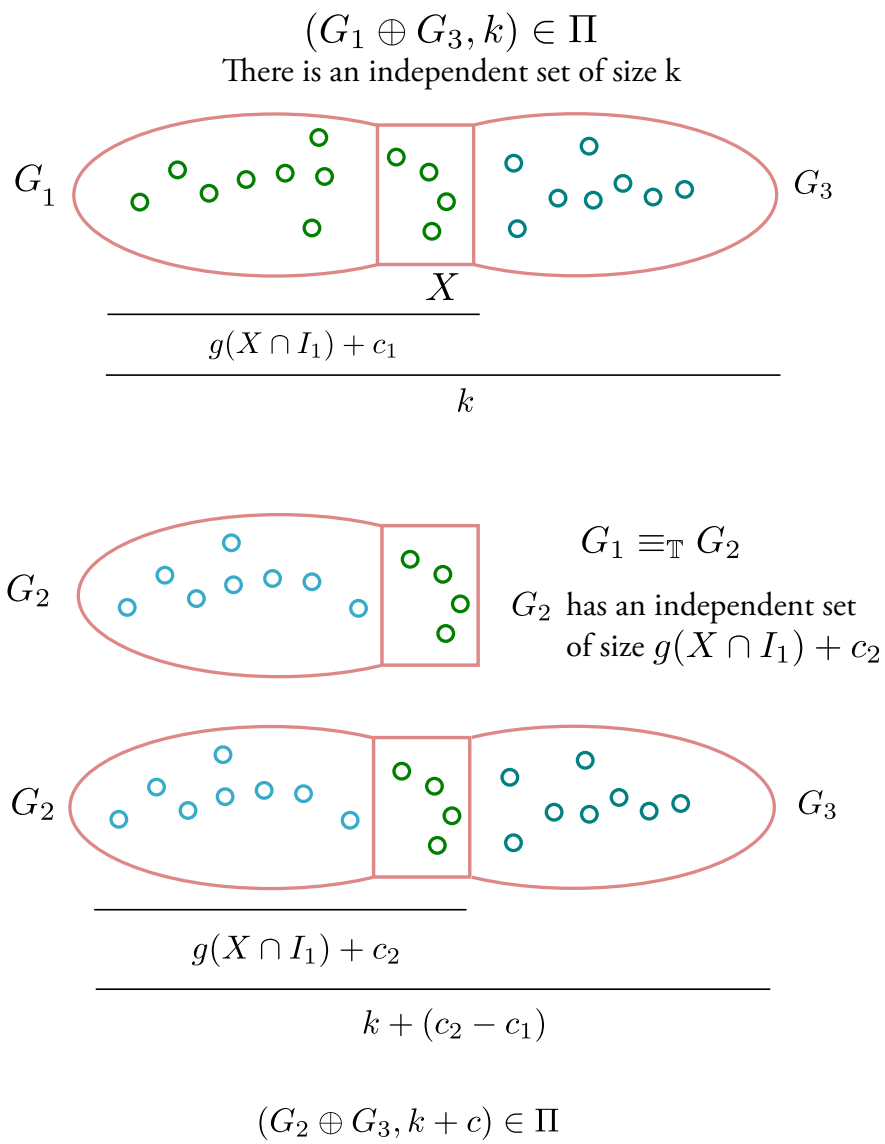


Figure 6.5: A schematic of the proof of Claim 6.1

6.4 One example of proving not FII

In this section we work out an example that demonstrates how we could show a problem does not have FII. We illustrate this with an example of Longest Path.

Theorem 6.1. Longest Path *does not have FII.*

Proof. Let Π= Longest Path in this proof. For each $n \geqslant 1$, let G_n be the two terminal graph defined by the vertex set $V(G_n) = \{x_1, x_2\} \cup \{a_1, \ldots, a_n\}$ and the edge set $E(G_n) = \{(x_1, a_1)\} \cup \{(a_i, a_{i+1}) \mid 1 \leqslant i \leqslant n-1\}$. Similarly, for each $p \geqslant 1$, let H_p be the two terminal graph defined by the vertex set $V(H_p) = \{y_1, y_2\} \cup \{b_1, \ldots, b_p\}$ and the edge set $E(H_p) = \{(y_2, b_1)\} \cup \{(b_i, b_{i+1}) \mid 1 \leqslant i \leqslant p-1\}$.

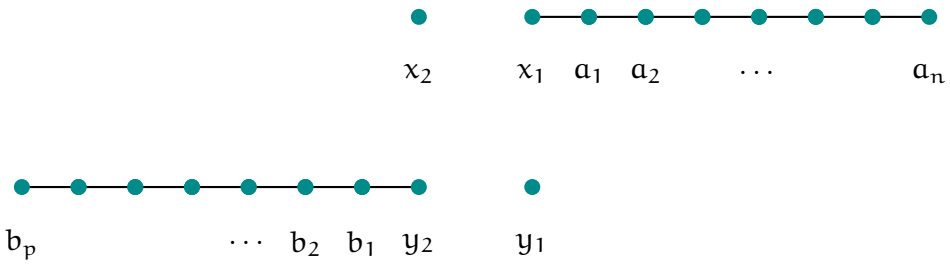


Figure 6.6: The graphs G_n and H_p.

In these two terminal graphs the first terminals are x_1 and y_1 and the second terminals are x_2 and y_2. Also observe that the length of a maximum path of $G_n \oplus H_p$ is $\max\{n, p\}$.

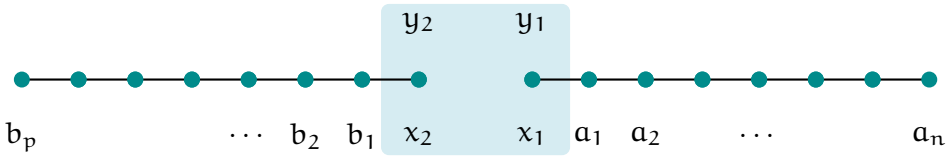


Figure 6.7: The graphs $G_n \oplus H_p$.

We will show that every *even* $n < m$, the graphs G_n and G_m are not equivalent, that is, $G_n \not\equiv_\Pi G_m$. For contradiction assume that $G_n \equiv_\Pi G_m$ and $n < m$. This implies

that there exists a c such that for all k and for all two terminal graph G, we have that $(G_n \oplus G, k) \in \Pi$ if and only if $(G_m \oplus G, k + c) \in \Pi$. We obtain our contradiction by considering the following three cases based on the value of c.

- Suppose $c > 0$, now consider $k = m$ and $G = H_m$. Clearly $G_n \oplus G$ has a path of length at least $m := \max\{n, m\}$ and hence $(G_n \oplus G, k) \in \Pi$. However, $G_m \oplus G$ has a path of length $m := \max\{m, m\}$ and hence does not have a path of length $k + c$ and thus $(G_m \oplus G, k) \notin \Pi$.
- Now assume that $c < 0$. Now consider $k = m - c$ and $G = H_m$. Clearly $G_n \oplus G$ has a path of length $m := \max\{n, m\}$ and hence does not have a path of length k as $n < m$ and $c < 0$. Thus $(G_n \oplus G, k) \notin \Pi$. However, $G_m \oplus G$ has a path of length $m := \max\{m, m\}$ and hence has a path of length $k + c = m - c + c$ and thus $(G_m \oplus G, k) \in \Pi$.
- Finally, consider the case of $c = 0$. Let $k = m$ and $G = H_{n+1}$. Clearly $G_n \oplus G$ has a path of length $n + 1 := \max\{n, n + 1\}$ and hence does not have a path of length k as $n \leqslant m - 2$. Thus $(G_n \oplus G, k) \notin \Pi$. However, $G_m \oplus G$ has a path of length $m := \max\{m, n + 1\}$ and hence it has a path of length $k = m-$ and thus $(G_m \oplus G, k) \in \Pi$.

Thus the above argument shows that $\equiv_\Pi$ does not have finitely many equivalence classes for $t = 2$. This completes the proof that Longest Path does not have FII. □

6.4.1 A Simple Sufficient Condition for Showing FII

The notion of *strong monotonicity* is an easily checked sufficient condition for a p-min-MSO problem to have finite integer index. We first introduce the concept of a *signature*, followed by the definition of strong monotonicity.

Definition 6.8. [**Signatures**] *Let* Π *be a* p-min-MSO *problem. For a* t*-boundaried graph* G *we define the* signature function $\zeta_G^\Pi : \mathcal{H}_t \to \mathbb{N} \cup \{\infty\}$ *as follows. For a pair* $(G', S') \in \mathcal{H}_t$*, if there is no set* $S \subseteq V(G)$ *such that* $P_\Pi(G \oplus G', S \cup S')$ *holds, then* $\zeta_G^\Pi((G', S')) = \infty$*. Otherwise* $\zeta_G^\Pi((G', S'))$ *is the size of the smallest* $S \subseteq V(G)$ *such that* $P_\Pi(G \oplus G', S \cup S')$ *holds.*

Definition 6.9. [**Strong Monotonicity**] *A* p-min-MSO *problem* Π *is said to be* strongly monotone *if there exists a function* $f : \mathbb{N} \to \mathbb{N}$ *such that the following condition is*

satisfied. For every t*-boundaried graph* G*, there is a subset* $S \subseteq V(G)$ *such that for every* $(G', S') \in \mathcal{H}_t$ *such that* $\zeta_G^{\Pi}((G', S'))$ *is finite,* $P_\Pi(G \oplus G', S \cup S')$ *holds and* $|S| \leqslant \zeta_G^{\Pi}((G', S')) + f(t)$.

The next lemma provides the application of strong monotonicity in demonstrating finite integer index.

Lemma 6.2 ([BFL+09]). *Every strongly monotone* p-MIN-MSO *problem has finite integer index.*

In practice, it is much easier to show finite integer index by establishing strong monotonicity. We will encounter a simple application of Lemma 6.2 in Chapter 9.

6.5 The Reduction Based on FII

In this section we provide reduction rules for graph problems that have finite integer index. The main reduction lemma is the following, and this will be the crux of several of our reduction rules in subsequent chapters.

Lemma 6.3 ([BFL+09]). *Let* Π *be a problem that has finite integer index. There exists a constant* c *and an algorithm that given as input a graph* G*, an integer* k*, and a* r*-protrusion* X *in* G *with* $|X| > c$*, outputs in* $O(|X|)$ *steps a graph* $G^* = (V^*, E^*)$ *and an integer* k^**, such that* $|V(G^*)| < |V(G)|$*,* $k^* \leqslant k$*, and* $(G^*, k^*) \in \Pi$ *if and only if* $(G, k) \in \Pi$.

Proof. Let $\mathcal{S}$ be a set of representatives for $(\Pi, 2r)$ and let $c = \max_{Y \in \mathcal{S}} |Y|$. Let ζ be a mapping from 2r-boundaried graphs with at most 2c vertices to $\mathcal{S}$ such that for any 2r-boundaried graph H on at most 2c vertices, $H \equiv_\Pi \zeta(H)$. Also, let η be a mapping from 2r-boundaried graphs with at most 2c vertices to $\mathbb{N}$ such that for any 2r-boundaried graph H' and k',

$$(H \oplus H', k + \eta(H)) \in \Pi \iff (\zeta(H) \oplus H', k) \in \Pi.$$

If $|X| > 2c$, then we find a 2r-protrusion $X' \subseteq X$ such that $c < |X'| \leqslant 2c$ and work on X' instead of X. This can be done in time $O(|X|)$ since $G[X]$ has treewidth at most r. In particular, consider a nice tree-decomposition of $G[X]$ and pick a lowermost bag b such that the total number of vertices appearing in bags below b is more than c

(see Figure 6.8). Let X' be the set of vertices appearing in b or in bags below b. The choice of b ensures that $c < |X'| \leqslant 2c$. From now on, we assume that $|X| \leqslant 2c$.

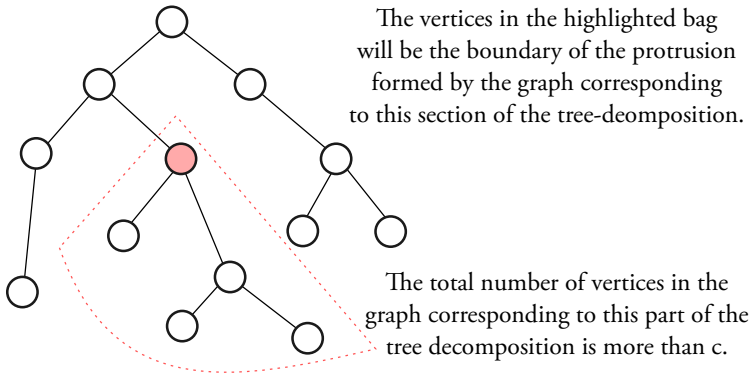


Figure 6.8: Finding a protrusion on more than c but at most $2c$ vertices by using the tree decomposition of a protrusion on more than $2c$ vertices.

Let $H = \zeta(G[X])$ and $k^* = k - \eta(G[X])$. We make G^* from G by replacing the $2r$-protrusion $G[X]$ with H. Since $|X| > c$ and H has at most c vertices, we have that $|V(G^*)| < |V(G)|$. By the choice of H and k^* we have that $(G^*, k^*) \in \Pi$ if and only if $(G, k) \in \Pi$. The running time of the algorithm is $O(|X|)$. This concludes the proof. □

I cannot tell my story without reaching a long way back.

Herman Hesse

It is a mistake to think you can solve
any major problems just with potatoes.

Douglas Adams

This chapter acts as a prelude the next four chapters, that focus entirely on the Planar $\mathcal{F}$-Deletion problem. In this chapter we collect important auxiliary results which we will find useful subsequently (often on more than one occasion). Many of the results in this chapter have no specific context with the Planar $\mathcal{F}$-Deletion problem, and we believe that they may be of independent interest, and applicable in other circumstances as well.

7.1 Partitioning Property of Graphs of Bounded Treewidth

We begin with a decomposition lemma which shows that any "large enough" graph can be broken down into a multitude of r-protrusions of "large" size, where r is proportional to the treewidth of the graph. This is applied in resolving the disjoint version of the Planar $\mathcal{F}$-Deletion problem in Chapter 11.

We informally remark that the size of the protrusions detected by this lemma is proportional to the size of the graph, making them viable for kernelization. However, while the lemma makes a powerful statement, we note that it does not *directly* imply that any graph of substantial size contains a protrusion that can be reduced — this is because the protrusion-based reduction rule requires the treewidth of the protrusions in question to be a constant, while this lemma implies protrusions whose treewidth is proportional to the treewidth of the graph, which in general is not necessarily constant.

We begin by defining the notion of a dissolution, which is essentially a partition into protrusions (see Figure 7.1).

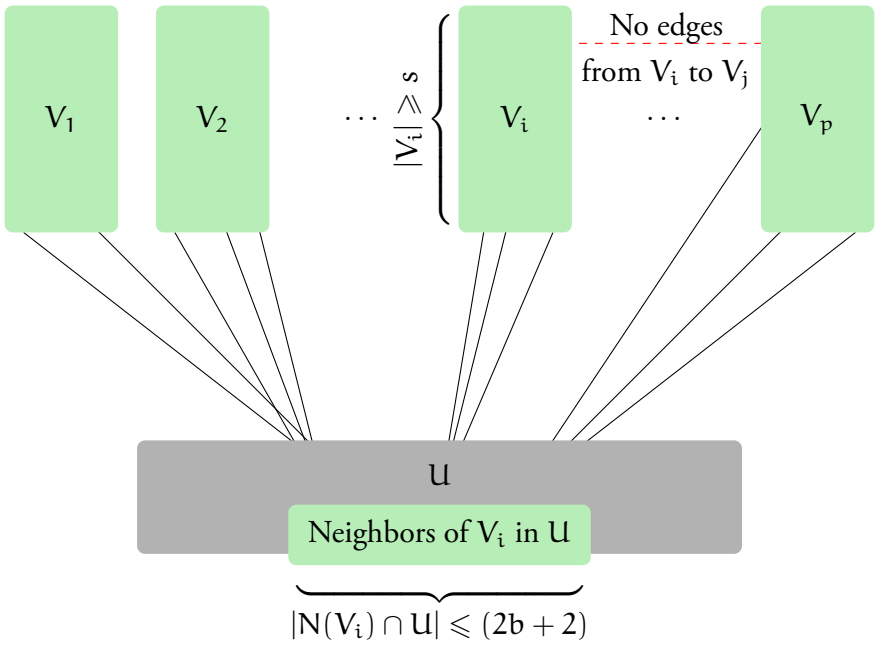


Figure 7.1: The definition of a (s, p)-dissolution for a graph of treewidth b.

Definition 7.1. *Let* G *be a graph of treewidth* b. *An* (s, p)-dissolution *of* G *is defined to be a partition of the vertex set of* G *into* $(p + 1)$ *parts* $V_1, \ldots, V_p$ *and* U *such that:*

- ♢ *Among the first* p *partitions, there are no edges with endpoints in different partitions, that is, for all* $1 \leqslant r \neq s \leqslant p$, *if* $u \in V_r$ *and* $v \in V_s$, *then* $(u, v) \notin E(G)$.
- ♢ *Each of the first* p *parts has size at least* s, *that is,* $|V_i| \geqslant s$, *for every* i, $1 \leqslant i \leqslant p$.
- ♢ *The neighborhood of* V_i *in* U *is at most* $(2b + 2)$*:*

$$|N(V_i) \cap U| \leqslant 2b + 2,$$

for every i, $1 \leqslant i \leqslant p$.

Note that we described a dissolution as a partition into protrusions because each part V_i along with its neighborhood in U is clearly a $(2b + 2)$-protrusion in G. We now claim that there exists a constant d such that any graph G with more than $d \cdot (bsp)$ vertices has a (s, p)-dissolution.

Lemma 7.1 (Partitioning Lemma). *Let* G *be a graph of treewidth* b. *There is an integer constant* d *such that if* G *has at least* $(d \cdot bsp)$ *vertices (for some integers* s *and* p*),* G *admits a* (s, p)*-dissolution.*

Proof. Let G be a graph on at least $(d \cdot bsp)$ vertices and let $(\mathcal{X}, T)$ be a *nice* tree-decomposition of G of width b. Given an arbitrary tree-decomposition of G of width b, a nice tree-decomposition can be constructed in polynomial time as described in [DST02], and the only property of nice tree-decompositions (as opposed to normal tree-decompositions) we will use in this proof is that the maximum degree of the decomposition tree T is three. For $T' \subseteq V(T)$ we use $X(T')$ to denote $\cup_{q \in T'} X_q$.

Claim 7.1. *There is a set* $S \subseteq V_T$ *of size* $2p$ *such that there are* $2p$ *connected components* $T_1, \ldots, T_{2p}$ *of* $T \setminus S$ *such that for every* i, $|X(T_i)| \geqslant 3s + b$.

A word of caution — observe that we do not claim that $T_1, \ldots, T_{2p}$ are the only components of $T \setminus S$.

Proof. We begin by introducing terminology that will be used in this proof. For a rooted tree T, and a vertex $v \in T$, a component C of $T \setminus \{v\}$ is said to be below v if all vertices of C are descendants of v in T.

We construct the set S with a simple greedy procedure. We root the tree T at some root r. In the beginning $S = \emptyset$ and $T^r = T$. We maintain a loop invariant that T^r is the connected component of $T \setminus S$ that contains r. Now, at step i of the greedy procedure we pick a lowermost vertex v_i in $V(T^r)$ such that there is a connected component T_i of $T^r \setminus \{v_i\}$ *below* v_i such that $|X(T_i)| \geqslant 3s + b$. Now we add v_i to S and update T^r accordingly. The procedure terminates when no vertex v in T_r has this property. In particular, if for any $v \in T^r$, every component C of $T^r \setminus v$ (below v) has fewer than $(3s + b)$ vertices, the procedure terminates.

Note that if this procedure does not terminate during the first $2p$ steps, then it produces a set S and components $T_1, \ldots, T_{2p}$ with the desired property. Thus it only remains to prove that the procedure does not terminate during the first $2p$ steps.

In the first step, observe that since $T^r = T$ and a nice tree decomposition has at least n bags,

$$|X(T^r)| \geqslant n \geqslant dbsp.$$

Recall that T is a binary tree and we always pick a lowermost vertex v_i in $V(T^r)$ such that there is a connected component T_i of $T^r \setminus \{v_i\}$ below v_i such that $|X(T_i)| \geqslant 3s + b$. Since T^r is a ternary tree, v_i has at most two children in T^r — say x and y. Note that the connected components of $T^r \setminus \{x\}$ and $T^r \setminus \{y\}$ have less than $(3s + b)$ vertices each (else we contradict the assumption that v_i was the "lowermost" vertex of T^r with the said property).

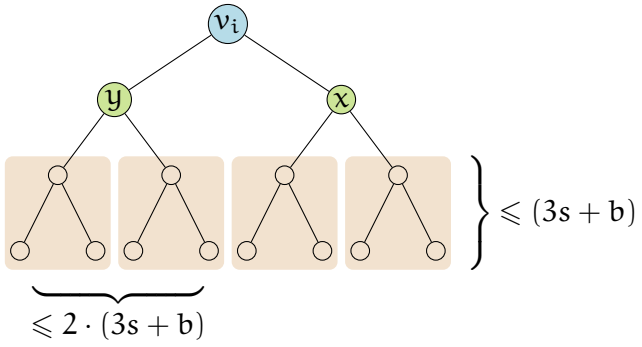


Figure 7.2: A bound on the number of vertices in the components below v_i.

Therefore, the total number of vertices in all components below v_i put together is bounded by:

$$\overbrace{((3s + b) \cdot 2)}^{\text{Vertices below } x} + \overbrace{((3s + b) \cdot 2)}^{\text{Vertices below } y} + 2,$$

the last summand accounting for x and y (see also Figure 7.2). Thus, at every step, $|X(T^r)|$ decreases by at most

$$12s + 6b \leqslant 12sb$$

at each step. Hence the procedure does not terminate during the first $2p$ steps for a suitable choice of d, such as 32. We first note that it is useful to rewrite 32 as $(12 \cdot 2) + 8$.

Now note that

$$n > (12sb)(2p) + 8(sbp)$$

and after $(2p - 1)$ steps of our procedure we will have that

$$n > (12sb) \cdot 1 + 8(sb)p,$$

which gives us enough vertices to find one with the desired property. Indeed, the removal of the root clearly leaves us with two components, at least one of which must

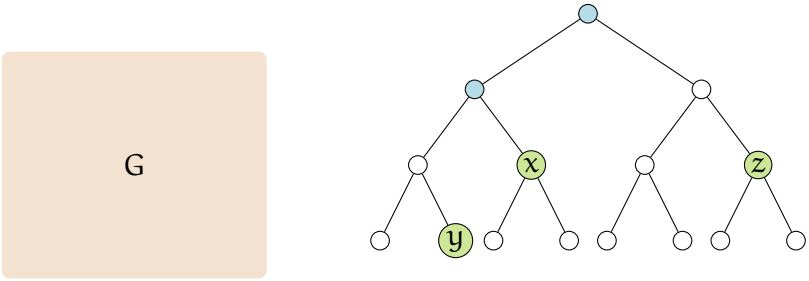


Figure 7.3: The procedure for constructing S' from $S = \{x, y, z\}$.

have at least $4(sb)$ vertices, and therefore at least $(3s+b)$ vertices; and it now remains only to find the lowermost vertex that continues to have this property.

Clearly, with the above analysis, the claim follows.

□

Having constructed S and $T_1, \ldots, T_{2p}$ as described in Claim 7.1 we construct S' from S as follows. In the beginning $S' = S$ and then, whenever two vertices u and v are in S' and their lowest common ancestor t in V_T is not, we add t to S'.

From the fact that any binary tree on l leaves has at most l internal nodes, we deduce that $|S'| \leqslant 2|S|$. Notice that any node v in a tree can be the least common ancestor only of nodes in the subtree rooted at v, and the subtree rooted at a leaf is empty. Thus an unmarked leaf can never be the least common ancestor of any pair of nodes from S. Therefore, from the given tree decomposition and a subset of its nodes, S, we may obtain an "equivalent" one where every leaf belongs to S — simply by recursively removing leaves that are not in S. In this tree, the number of leaves is at most $|S|$, and the total number of nodes chosen in S' is at most twice the number of leaves. Therefore, $|S'| \leqslant 2|S|$. Let $S^* = S' \setminus S$.

Since $|S| > 2p$ and $|S'| \leqslant 2|S|$, it follows that

$$|S^*| \leqslant 2p.$$

Note that therefore, at most p of the components $T_1, \ldots, T_{2p}$ contain at least two vertices of S^*. Hence, at least p of the components $T_1, \ldots, T_{2p}$ contain at most one vertex of S^*. Without loss of generality $T_1, \ldots, T_p$ contain at most one vertex of S^* each. For every $i \leqslant p$, if T_i contains no vertex of S^* then $T_i' = T_i$ is a component of

$T \setminus S'$ with $|X(T_i')| \geqslant s$. If T_i contains one vertex v of S^*, since v has degree at most 3 and $|X(T_i)| \geqslant 3s + b$, $T_i \setminus \{v\}$ has at least one component T_i' with $|X(T_i')| \geqslant s$.

Now, let $V_i = X(T_i')$ and U be the vertices not appearing in any V_i. Clearly $|V_i| \geqslant s$. Also, every T_i' has at most two neighbors in S'. Let these neighbours be u and v (with $u = v$ if T_i' has one neighbor in S'). Since $N(V_i) \subseteq X_u \cup X_v$, it follows that $|N(V_i)| \leqslant 2b + 2$. This concludes the proof. □

For the sake of completeness, we state a more general version of Lemma 7.1. Let f be a function that associates a natural number with every vertex, that is, let $f : V(G) \to \mathbb{N}$. Let the function $\hat{f} : 2^V \to \mathbb{N}$ be defined as follows:

$$\hat{f}(S) = \sum_{v \in S} f(v).$$

We define a (f, s, p)-dissolution as follows:

Definition 7.2. *Let G be a graph of treewidth b, and let $\hat{f}$ be defined as above. An* (f, s, p)-dissolution *of G is defined to be a partition of the vertex set of G into $(p + 1)$ parts $V_1, \ldots, V_p$ and U such that:*

- ♢ *Among the first p partitions, there are no edges with endpoints in different partitions, that is, for all $1 \leqslant r \neq s \leqslant p$, if $u \in V_r$ and $v \in V_s$, then $(u, v) \notin E(G)$.*
- ♢ *Each of the first p parts is such that $\hat{f}(V_i) \geqslant s$, for every i, $1 \leqslant i \leqslant p$.*
- ♢ *The neighborhood of V_i in U is at most $(2b + 2)$:*

$$|N(V_i) \cap U| \leqslant 2b + 2,$$

for every i, $1 \leqslant i \leqslant p$.

Then, we have the following generalization of Lemma 7.1. The proof is exactly along the lines of the proof of Lemma 7.1 with only minor modifications.

Lemma 7.2 (Generalized Partitioning Lemma). *Let G be a graph of treewidth b, let $f : V(G) \to \mathbb{N}$ be a function, and let $\hat{f} : 2^V \to \mathbb{N}$ be defined as follows:*

$$\hat{f}(S) = \sum_{v \in S} f(v).$$

There is an integer constant d such that if G has at least $(d \cdot bsp)$ vertices (for some integers s and p), G admits a (f, s, p)-dissolution.

7.2 Inferring Protrusions from Subgraphs of Constant Treewidth

Our aim in this section is to demonstrate a $(2\tau+1)$-protrusion within X, where τ is a constant and X is a subgraph of G that has treewidth τ. This technique is applied in Chapters 9 and 10. Notice that the graph induced on all of X is already half-way through to being a protrusion, since it has constant treewidth. On the other hand, the graph induced on X may not have a constant-sized boundary, hence we face the need to identify a suitable subgraph that does have a constant-sized boundary.

A good starting point appears to be the tree decomposition of $G[X]$, since given that X has constant treewidth, we will have bags of constant size. Notice that if we can identify a collection of bags that:

- correspond to a large enough subgraph of X, and
- have a limited number of vertices from $\partial(X)$,

then we will be done. This is achieved by a neat trick that involves eliminating a carefully chosen set of bags such that the components that remain all have a limited number of vertices from $\partial(X)$, and at least one of the components corresponds to a large enough subgraph. Keeping this goal in mind, let us turn to the formal proof.

Lemma 7.3. *There is a linear time algorithm that:*

→ *given an n-vertex graph G and a subset of vertices X such that $tw(G[X]) \leqslant \tau$,*
← *outputs a $2(\tau+1)$-protrusion of G of size at least*

$$\left(\frac{|X|}{4|\partial(X)|+1}\right).$$

Proof. The algorithm begins by computing a nice tree decomposition of $G[X]$ of width at most τ. Since τ is a constant, this can be done in linear time [Bod96].

The nice tree decomposition of $G[X]$ is a pair $(T, \mathcal{B} = \{B_\ell\}_{\ell \in V(T)})$, where T is a rooted binary tree. We will now *mark* some of the nodes of T (see Figure 7.4). For every $v \in \partial(X)$, we mark the topmost node ℓ in T such that $v \in B_\ell$. In this manner, at most $|\partial(X)|$ nodes are marked. Now we mark more nodes of T by exhaustively applying the following rule: if u and v are marked, mark their least common ancestor in T. Let M be the set of all marked nodes of T.

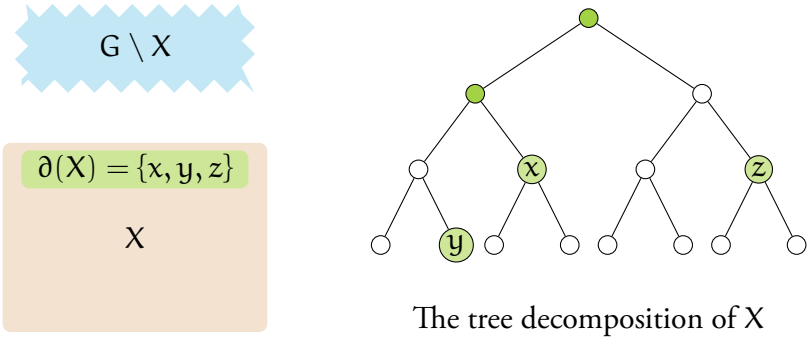


Figure 7.4: Marking nodes from the boundary, and more.

From the fact that any binary tree on l leaves has at most l internal nodes, we deduce that $|M| \leqslant 2|\partial(X)|$. Notice that any node v in a tree can be the least common ancestor only of nodes in the subtree rooted at v, and the subtree rooted at a leaf is empty. Thus an unmarked leaf can never be the least common ancestor of any pair of marked nodes. Therefore, from the given marked tree decomposition, we may obtain an "equivalent" one where every leaf is a marked node — simply by removing, recursively, leaves that are not marked. In this tree, the number of leaves is at most $\partial(X)$, and the total number of marked nodes is at most twice the number of leaves. This shows that $|M| \leqslant 2|\partial(X)|$.

Since T is a binary tree, it follows that $T \setminus M$ has at most $2|M| + 1$ connected components. Let the vertex sets of these connected components be $C_1, C_2 \ldots C_\eta$, $\eta \leqslant 2|M| + 1$.

We now show that for all i, C_i is adjacent to at most two bags in M. We will repeatedly make use of the fact that least common ancestors of marked nodes were marked in our construction of M. Let us consider the subtree C_i, and imagine it rooted at the vertex that is closest to the root of T. Let r_i denote this root. Further, let A_i and B_i denote the two subtrees rooted at the two children of r_i. Notice that if one of the leaves of A_i (respectively, B_i) is adjacent to a vertex in M, then no leaf of B_i (respectively, A_i) can be adjacent to a vertex of M — because if leaves of both subtrees are adjacent to marked vertices, then r_i would be marked as well, and would not be in one of the components of $T \setminus M$. By a similar reasoning, at most one of the leaves of A_i (or B_i) is adjacent to marked nodes — if more than one of the leaves of, say, A_i is adjacent to marked nodes, then that would require us to mark an internal node of C_i, again a contradiction. Finally, notice that no leaf is adjacent to *two* marked nodes, since this would cause the leaf to be marked. Thus, at most one of the leaves of the subtree C_i

is adjacent to a marked node, and at most one marked node is adjacent to any of the leaves.

The only other nodes that might be adjacent to a marked node are the root, or internal nodes that have degree two. We distinguish two cases, in the context of the root — and we remark that the case for the internal nodes is similar.

• If the root has degree two in C_i, we are done, as the root can have at most one other neighbor in T, and as we have already argued, at most one of the leaves of C_i is adjacent to at most one marked vertex, and thus, in this case, C_i is adjacent to at most two marked nodes.

• If the root has degree one in C_i, then notice that if both children of the root are marked, none of the leaves of the subtree rooted at the only child of r_i (in C_i) might be adjacent to a marked node (if it were, again, r_i would be compelled to belong to M). So either both neighbors of the root are marked and none of the leaves of C_i are adjacent to a marked node, or exactly one of the neighbors of the root is marked, and at most one of the leaves of C_i is adjacent to at most one marked vertex. In either case, C_i is adjacent to at most two marked nodes.

For every $i \leqslant \eta$, let D_i denote the set of bags C_i and the bags from M that were adjacent to bags in C_i, that is:

$$D_i := C_i \cup N_T(C_i).$$

Further, let P_i denote the vertices of G that appear in all the bags of D_i, that is:

$$P_i = \bigcup_{u \in D_i} B_u.$$

We claim that the subgraphs induced on any of the P_i form protrusions. We only need to show that P_i contains a limited number of vertices from $\partial(X)$. Given our setup, note that any vertex from $\partial(X)$ that appears in C_i must also appear in $N_T(C_i)$. This is because we are dealing with a tree decomposition, and if a vertex makes an appearance in bags A and B, then it must present itself in all bags that are on the path from A and B. Recall that every vertex v from $\partial(X)$ is in *some* marked bag B. If v is in a bag A from C_i, then failure to appear in both marked bags that are neighbors

of C_i would imply that not all bags between A and the marked bag that contained v contain v, contradicting the fact that we are working with a tree decomposition.

Given that any vertex from $\partial(X)$ that appears in C_i must also appear in $N_T(C_i)$, it is easy to see that $P_i \cap \partial(X)$ is a constant: since $N_T(C_i)$ comprises at most two bags, each of size at most $(\tau+1)$, C_i can accommodate at most $2(\tau+1)$ vertices from $\partial(X)$. Thus every P_i is a $2(\tau+1)$-protrusion of G.

Since $\eta \leqslant 2|M|+1 \leqslant 4|\partial(X)|+1$, the pigeon-hole principle yields that there is a protrusion P_i with at least $\frac{|X|}{4|\partial(X)|+1}$ vertices. The algorithm constructs M and $P_1 \ldots P_\eta$ and outputs the largest protrusion P_i. It is easy to implement this procedure to run in linear time. This concludes the proof. □

7.3 Facts concerning minor models of θ_c

In this section, we collect some results about minor models of θ_c. These results will be relevant to the kernelization algorithm for $\mathcal{F}$-Deletion when $\mathcal{F}$ contains the graph θ_c (see Chapter 10). In particular, we show the following:

- ⋆ Minor models of θ_c do not contain any cut vertices.
- ⋆ On graphs of constant treewidth, an optimal θ_c-hitting set may be found in polynomial time.
- ⋆ On graphs of constant treewidth, a maximum collection of subgraphs may be found in polynomial time, such that each of them contain a minor model of θ_c, and are vertex disjoint except at a single vertex.
- ⋆ In polynomial time, we can either classify an instance of Θ_c-Deletion as a NO instance, or determine a hitting set T_v of size $k^{O(1)}$ for all minor models that pass through a specified vertex v. We note that $v \notin T_v$.

7.3.1 Minor Models of θ_c do not have Cut Vertices

We will need the following description of minimal minor models of θ_c for the proof of our first observation. The proof of this proposition is immediate from the definition of a minimal minor model and the graph θ_c.

Proposition 7.1. *For any $c \in \mathbb{N}$, a subgraph M of graph G is a minimal minor-model of θ_c in G if and only if M consists of two trees, say T_1 and T_2, and a set S of c edges, each of which has one end vertex in T_1 and the other in T_2.*

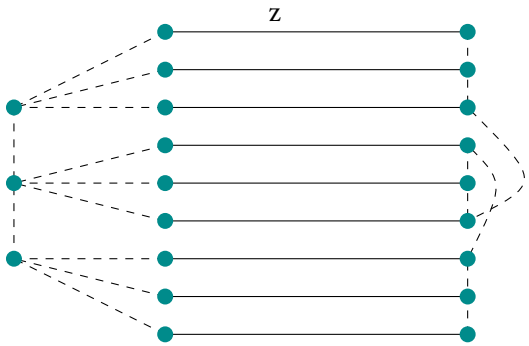


Figure 7.5: The form of a minimal minor model of θ_c. In this example, the dashed edges may be contracted to obtain the θ_9 graph.

Observation 7.2. *For $c \geqslant 2$, any minimal θ_c minor-model M of a graph G is a connected subgraph of G, and does not contain a vertex whose degree in M is less than 2, or a vertex whose deletion from M results in a disconnected graph (a cut vertex of M).*

Proof. From Proposition 7.1, whose terminology we use in this proof, M is connected and contains no isolated vertex. Suppose x is a vertex of degree exactly one in M. Then x is a leaf node in one of the two trees in M, say T_1, and no edge in S is incident on x. Removing x from T_1 results in a smaller θ_c minor-model, contradicting the minimality of M. It follows that every vertex of M has degree at least two.

Now suppose x is a cut vertex in M which belongs to, say, the tree T_1. Let $T_1^1, T_1^2, \ldots, T_1^l$ be the subtrees of T_1 obtained when x is deleted from T_1. Let M' be the graph obtained by deleting x from M. If $l > 0$, then each T_1^i has a leaf node, which, by the above argument, has at least one neighbor in T_2. If $l = 0$, then $M' = T_2$. Thus M' is connected in all cases, and so x is not a cut vertex, a contradiction. □

7.4 Some MSO Formulations

The following well known result states that every optimization problem expressible in MSO has a linear time algorithm on graphs of bounded treewidth.

Proposition 7.3 ([ALS91, Bod96, BPT92, Cou90, CM93]). *Let ϕ be a property that is expressible in Monadic Second Order Logic. For any fixed positive integer t, there is an algorithm that, given a graph G of treewidth at most t as input, finds a largest (alternatively, smallest) set S of vertices of G that satisfies ϕ in time $f(t, |\phi|)|V(G)|$.*

In the kernelization procedure, we will need to find an optimal θ_c-hitting set on graphs of constant treewidth. To this end, we demonstrate that the property of being an optimal θ_c-hitting set can be expressed in MSO, and appeal to Proposition 7.3 to conclude that an optimal θ_c-hitting set can be found in polynomial time on graphs of constant treewidth.

We will also encounter the need to identify, the largest θ_c "flower" in polynomial time: this involves finding a maximum-size collection M of minimal θ_c minor-models, all of which pass through v and no two of which share any vertex other than v. Our strategy is again to describe a MSO formulation and apply Proposition 7.3.

MSO formulation of H minor-models. For a graph G, we use $\phi_H(G)$ to denote an MSO formula which states that G contains H as a minor — equivalently, that G contains a H minor model. Let $V(H) = \{h_1, \ldots, h_c\}$. Then, $\phi_H(G)$ is given by:

$$\begin{aligned}\phi_H(G) \equiv \exists X_1, \ldots, X_c \subseteq V(G)[& \\ & \bigwedge_{i \neq j}(X_i \cap X_j = \emptyset) \wedge \bigwedge_{1 \leqslant i \leqslant c} \mathsf{Conn}(G, X_i) \wedge \\ & \bigwedge_{(h_i,h_j)\in E(H)} \exists x \in X_i \wedge y \in X_j[(x,y) \in E(G)] \\ ] & \end{aligned} \tag{7.1}$$

Here, $\mathsf{conn}(G, X)$ is the standard MSO formulation for expressing the fact that $G[X]$ is a connected subgraph. Formula 7.1 states that the graph G contains c vertex-disjoint connected subgraphs, $X_1, \ldots, X_c$, such that whenever $(h_i, h_j) \in E(H)$, there is at least one edge in G with one endpoint each in the subgraphs X_i and X_j. Clearly, if we contract the subgraphs $X_1, \ldots X_c$ to vertices $x_1, \ldots x_c$, then $G[x_1, \ldots, x_c]$ contains H as a subgraph, and thus H is a minor of G whenever $\phi_H(G)$ is satisfiable. It is also easy to see, from the definition of a minor, that if G contains H as a minor, then $\phi_H(G)$ is satisfiable.

Minimum-size $\mathcal{F}$-hitting set. Let $\mathcal{F}$ be a finite forbidden set. A minimum-size $\mathcal{F}$-hitting set of graph G can be expressed as:

$$\text{Minimize } S \subseteq V(G)[\bigwedge_{H\in\mathcal{F}} \neg\phi_H(G \setminus S)] \tag{7.2}$$

Formula 7.2 seeks the smallest subset of vertices S for which $G \setminus S$ does not contain H as a minor, for every $H \in \mathcal{F}$. Clearly, the subset S that is identified by formula 7.2 is an optimal $\mathcal{F}$-hitting set.

Largest θ_c "flower". We will need the notion of a θ_c flower to proceed further.

Definition 7.3. *Given a graph* G *and a vertex* $v \in V(G)$*, an* θ_c*-flower of size* ℓ passing through v *is a set of* ℓ *different* θ_c *minor-models in* G*, each containing* v *and no two sharing any vertex other than* v*.*

Let v be a fixed vertex. We describe a MSO representation of a witness for a θ_c-flower passing through v of the largest size. A maximum-size set M of θ_c minor-models in G, all of which pass through v and no two of which share any vertex other than v, can be represented as follows:

Maximize $S \subseteq V(G)$:

$$\exists F \subseteq E(G) \quad \left[\forall x \in S \left(\exists X \subseteq V' \left\{ \begin{array}{cc} & \text{MaxConn}(G', X) \\ \wedge & x \in X \\ \wedge & \forall y \in S[y \neq x \implies y \notin X] \\ \wedge & \phi_{\theta_c}(X \cup \{v\}) \end{array} \right\} \right) \right] \tag{7.3}$$

Formula 7.3 identifies a subset S of vertices with the property that each vertex in S belongs to a minor model of θ_c, and each of these minor models are vertex-disjoint except for v. We let F denote the subset of edges that participate in this collection of minor models, and let G' denote the graph with vertex set $V(G)$ and edge set F, and $V' = V(G) \setminus \{v\}$. Notice that for every vertex in $x \in S$, the formula above identifies a subgraph X of G that contains x and is such that $X \cup \{v\}$ contains a minor model of θ_c. The witness subsets are vertex-disjoint because of the criteria that no two vertices of S belong to the same witness subgraph, and because the subgraphs are required to be maximally connected in G'. Indeed, let $x, y \in S$, and let X and Y be the corresponding witness subgraphs. For the sake of contradiction, assume that X

and Y are not vertex-disjoint. Let $u \in X$ and $u \in Y$. Then, there is a path from u to y because Y is a connected subgraph of G', and therefore $y \in X$, because X is a *maximally* connected subgraph of G': this contradicts the fact that no two vertices of S belong to the same witness subgraph.

Proposition 7.3 together with MSO formulations 7.2 and 7.3 given above allow us to find optimal $\mathcal{F}$-hitting sets and θ_c flowers in polynomial time on graphs of constant treewidth. Note that we can also find θ_c-hitting sets in polynomial time on graphs of constant treewidth, by simply using the set $\{\theta_c\}$ as $\mathcal{F}$ in the MSO formula 7.2. We make a note of these observations in the following lemmas:

Lemma 7.4. *Let G be a graph on n vertices and v a vertex of G. Given a tree decomposition of width $t \in O(1)$ of G, we can, in $O(n)$ time, find both (1) a smallest set $S \subseteq V$ of vertices of G such that the graph $G \setminus S$ does not contain θ_c as a minor, and (2) a largest collection $\{M_1, M_2, \ldots, M_l\}$ of θ_c minor models of G such that for $1 \leqslant i < j \leqslant l$, $(V(M_i) \cap V(M_j)) = \{v\}$.*

Lemma 7.5. *Let $\mathcal{F}$ be a collection of graphs, let G be a graph on n vertices and v a vertex of G. Given a tree decomposition of width $t \in O(1)$ of G, we can, in $O(n)$ time, find the smallest set $S \subseteq V$ of vertices of G such that the graph $G \setminus S$ does not contain H as a minor, for any $H \in \mathcal{F}$.*

7.5 A Bound on the treewidth of YES instances of Planar $\mathcal{F}$-Deletion

In this section, we establish that if G is a YES-instance of Planar $\mathcal{F}$-Deletion, then the treewidth of G is bounded by a function of k. Although this is a relatively simple observation to make from certain known results, it is a crucial starting point for our subsequent endeavors, both in the context of kernelization and approximation.

Lemma 7.6. *Let $\mathcal{F}$ be an obstruction set containing a planar graph of size h. If G has an $\mathcal{F}$-hitting set S of size at most k, then $tw(G \setminus S) \leqslant d$ and $tw(G) \leqslant k + d$, where $d = 20^{2(14h-24)^5}$.*

Proof. By assumption, $\mathcal{F}$ contains at least one planar graph. Let h be the size of the smallest planar graph H contained in $\mathcal{F}$. By a result of Robertson et al. [RST94], H is a minor of the $(\ell \times \ell)$-grid, where $\ell = 14h - 24$. In the same paper, Robertson

et al. [RST94] have shown that any graph with treewidth greater than $20^{2\ell^5}$ contains a $(\ell \times \ell)$-grid as a minor. Let S be a $\mathcal{F}$–hitting set of G of size at most k. Since the $(\ell \times \ell)$-grid contains H as a minor, we have that $\mathrm{tw}(G \setminus S) \leqslant 20^{2\ell^5}$. Therefore, $\mathrm{tw}(G) \leqslant k + d$, where $d = 20^{2\ell^5}$ — indeed, a tree decomposition of width $(k + d)$ can be obtained by adding the vertices of S to every bag in an optimal tree decomposition of $G \setminus S$. This completes the proof of the lemma. □

It turns out that the bound we obtain in Lemma 7.6 can be tightened significantly in the special case when $\mathcal{F} = \theta_c$. This is established in a result from [BTTvL95] that establishes a linear bound on the treewidth of a graph that does not contain minor models of θ_c:

Lemma 7.7 ([BTTvL95]). *Every graph not containing a θ_c as a minor has tree-width at most $(2c - 1)$.*

This has the following immediate corollary:

Corollary 7.4. *Let $\mathcal{F}$ be an obstruction set containing θ_c. If G has an $\mathcal{F}$-hitting set S of size at most k, then $tw(G) \leqslant k + (2c - 1)$.*

Proof. Let S be a $\mathcal{F}$–hitting set of G of size at most k. Since $\theta_c \in \mathcal{F}$, due to Lemma 7.7, we have that $\mathrm{tw}(G \setminus S) \leqslant (2c - 1)$. Therefore, $\mathrm{tw}(G) \leqslant k + (2c - 1)$ — indeed, a tree decomposition of width $(k + d)$ can be obtained by adding the vertices of S to every bag in an optimal tree decomposition of $G \setminus S$. This completes the proof.

□

The intellect has little to do on the road to discovery.
There comes a leap in consciousness, call it intuition or what you will,
and the solution comes to you and you don't know why or how.

Albert Einstein

It's talking, Merry. The tree is talking.

The Lord of the Rings

This chapter marks our first encounter with the $\mathcal{F}$-Deletion problem. Since this question is at the heart of this thesis, we will first spend some time acquainting ourselves with the problem. This will be followed by an approximation algorithm with an approximation ratio of $\mathcal{O}(\mathsf{OPT}^2\sqrt{\log \mathsf{OPT}})$. In the final section, we present an algorithm with an improved approximation ratio of $\mathcal{O}(\log^{3/2} \mathsf{OPT})$.

8.1 An Introduction to the $\{\mathcal{F}\}$-deletion problem

We use $\mathcal{F}$ to denote a family of graphs. Imagine that we are interested in the class of graphs that exclude H as a minor, for every $H \in \mathcal{F}$. For example, if $\mathcal{F}$ were to consist of just a triangle, then we are talking about *forests* when we consider the class of simple graphs that exclude a triangle as a minor. In the setting of multigraphs, we might think of forests as the class of graphs that exclude the double edge (a pair of vertices with two edges between them) as a minor. Note that the Feedback Vertex Set question asks if there exists a small subset of vertices S such that $G \setminus S$ induces a forest.

Generalizing this setting, it is natural to ask if, given a graph G and a family $\mathcal{F}$, there exists a small set of vertices S such that $G \setminus S$ induces a graph that excludes H as a minor, for every $H \in \mathcal{F}$. For particular choices of $\mathcal{F}$, this corresponds to well-studied problems: we have already seen feedback vertex set to be a special case of this question, and it is easy to see that so is Vertex Cover. Other famous cases are $\mathcal{F} = \{K_{2,3}, K_4\}$, $\mathcal{F} = \{K_{3,3}, K_5\}$ and $\mathcal{F} = \{K_3, T_2\}$, which correspond to removing vertices

to obtain outerplanar graphs, planar graphs, and graphs of pathwidth one respectively. Here, $K_{i,j}$ denotes the complete bipartite graph with bipartitions of sizes i and j, and K_i denotes the complete graph on i vertices. Further, a T_2 is a star on three leaves, each of whose edges has been subdivided exactly once.

We are now ready to state the $\mathcal{F}$-DELETION question formally. We first define the notion of a *$\mathcal{F}$-hitting set*.

Definition 8.1 ($\mathcal{F}$-hitting set)**.** *Let G be a graph and let $\mathcal{F}$ be a collection of graphs. A $\mathcal{F}$-hitting set is a subset $S \subseteq V(G)$ such that $G \setminus S$ does not contain H as a minor, for all $H \in \mathcal{F}$.*

The question of $\mathcal{F}$-DELETION is the following:

$\mathcal{F}$-DELETION
Input: A graph G, a family of graphs $\mathcal{F}$.
Parameter: k
Question: Does there exist $\mathcal{F}$-hitting set S such that $|S| \leqslant k$?

It is quite evident that the general $\mathcal{F}$-deletion problem is NP-complete, since a number of NP-complete problems are special cases of the $\mathcal{F}$-deletion question. From the parameterized perspective, by one of the most well-known consequences of the celebrated Graph Minor theory of Robertson and Seymour, the $\mathcal{F}$-deletion problem is fixed-parameter tractable for every finite set of forbidden minors, in a non-uniform sense — that is, we have a fixed-parameter tractable algorithm whenever $\mathcal{F}$ is given explicitly.

In this chapter, we discuss an approximation algorithm for $\mathcal{F}$-DELETION, when $\mathcal{F}$ contains at least one planar graph. Even with this restraint, notice that all the special cases we described previously are accommodated for, so the question retains its generality. We refer to the problem we address as PLANAR $\mathcal{F}$-DELETION:

PLANAR $\mathcal{F}$-DELETION
Input: A graph G, a family of graphs $\mathcal{F}$ that contains at least one planar graph.
Parameter: k
Question: Does there exist $\mathcal{F}$-hitting set S such that $|S| \leqslant k$?

We are now ready to describe the approximation algorithm.

8.2 A First Approximation Algorithm

The approximation algorithm for Planar $\mathcal{F}$-Deletion is based on a divide-and-conquer strategy. We begin with an informal discussion leading us up to a situation where the strategy presents itself. Let H be an arbitrary planar graph in $\mathcal{F}$ of the smallest size, and let h denote $|H|$.

Imagine that we have found some way of identifying a section of the graph — say G_H — that is guaranteed to have at least one minor-model of H, and has a simple enough local structure for us to solve the problem exactly in polynomial time. Further, suppose there is a small subset of vertices S such that G_H is one of the connected components of $G \setminus S$. If we are able to obtain these ingredients in polynomial time, then a recursive strategy is quite natural: we pick S in our solution, find a locally optimal solution on G_H and append it to S. Then it only remains to recursively solve the problem on each of the remaining connected components. The base case of the recursion is reached when we have graphs where the problem can be solved optimally.

Once the above is formalized, it is not difficult to "verify" that the strategy would yield a *valid* $\mathcal{F}$-hitting set. However, we wish to emphasize the intuition for why the solution might be an approximation to the optimal. Let OPT denote the size of the optimal $\mathcal{F}$-hitting set. Notice that:

▷ The number of recursive calls is a lower bound on OPT. This is because every recursive call guarantees the existence of at least one minor-model of H that is vertex-disjoint from any discovered in previous calls. This is why it is important that S separates G_H from all the other pieces on which we will recurse.

▷ The size of the solution thus obtained is at most $(\mathsf{OPT} \cdot |S|)$. This establishes our incentive for finding a *small* separator whose removal gives us at least one component with a minor model of H.

Before we present the algorithm precisely, let us recall and collect some relevant facts that will be required.

Guaranteeing the existence of a minor model of H The first thing we need to formalize is the idea of finding a section of the graph that contains at least one minor-model of H. We will make use of the following:

♢ Any planar graph H of size h is a minor of the $(t \times t)$-grid, where $t = (14h - 24)$. [RST94]

♢ Any graph with treewidth greater than 20^{2t^5} contains a $t \times t$ grid as a minor. [RST94]

Clearly, we are able to conclude from the above that if h is the size of H and $tw(G) > 20^{2(14h-24)^5}$, then G has a minor model of H, and this is the criteria that we will make use of. Notice here our dependence on the fact that $\mathcal{F}$ has at least one planar graph. Let d denote the constant $(20^{2(14h-24)^5} + 1)$. So now our task is reduced to finding a section of the graph that has treewidth at least d.

Finding an optimal solution on G_H The other property we desire of G_H is that it be simple enough for us to solve the problem in polynomial time. To this end, we recall that Planar $\mathcal{F}$-Deletion is polynomial time on graphs of constant treewidth (see Lemma 7.5). So if we can ensure that the treewidth of G_H is, for instance, $(d + 1)$, then we can identify an optimal $\mathcal{F}$-hitting set in polynomial time. This will also be the situation in the base case of the recursion — we will be able to stop when we encounter graphs of constant treewidth, and as we will soon see, we will either be able to find a suitable G_H, or conclude that the entire graph has constant treewidth.

A Bound on the treewidth of G We also need, for a starting point, that if G is a YES-instance of Planar $\mathcal{F}$-Deletion, then the treewidth of G is bounded by a function of k. This enables us to reject instances of large treewidth, and if the treewidth is small, we will have a tree decomposition of bounded width to work on. We recall Lemma 7.6 from Chapter 7, which gives us exactly what we need:

Lemma. *Let $\mathcal{F}$ be an obstruction set containing a planar graph of size h. If G has an $\mathcal{F}$-hitting set S of size at most k, then $tw(G \setminus S) \leqslant d$ and $tw(G) \leqslant k + d$, where $d = 20^{2(14h-24)^5}$.*

8.2.1 The Algorithm

We are now ready to present a first approximation algorithm for Planar $\mathcal{F}$-Deletion. We first describe an algorithm that solves the decision version, where k denotes the size of the solution sought, and we will see subsequently that this algorithm produces an approximate solution with the desired ratio when run at most n times with different values of k.

If $\mathrm{tw}(G) \leqslant d$, then we find an optimum $\mathcal{F}$-hitting set of G in linear time using Lemma 7.5. If the treewidth of the input graph is more than d then we find an approximate tree decomposition of width ℓ using an algorithm of Feige et al. [FHL08] such that

$$\mathrm{tw}(G) \leqslant \ell \leqslant d'\mathrm{tw}(G)\sqrt{\log \mathrm{tw}(G)},$$

where d' is a fixed constant. If $\ell > (k+d)d'\sqrt{\log(k+d)}$, then by Lemma 7.6, we know that the size of a minimum $\mathcal{F}$-hitting set of G is at least $(k+1)$, and thus the algorithm aborts at this point and returns a NO answer. Else, we know that the treewidth of G is bounded:

$$\mathrm{tw}(G) \leqslant \ell \leqslant (k+d)d'\sqrt{\log(k+d)}.$$

In the next step we compute S and G_H, keeping in mind that our goal is to identify a section of the graph that has treewidth exactly $(d+1)$. This is done by a bottom-up calculation performed on the tree decomposition. We begin by converting the given tree decomposition to a nice tree decomposition of the same width in linear time [Klo94]. Given a nice tree decomposition $(T, \mathcal{X} = \{X_t\}_{t \in V(T)})$ of G, recall that H_t refers to the subgraph induced on the union of vertices appearing in all bags present in the subtree rooted at t, except the vertices at the bag corresponding to node t itself. Notationally,

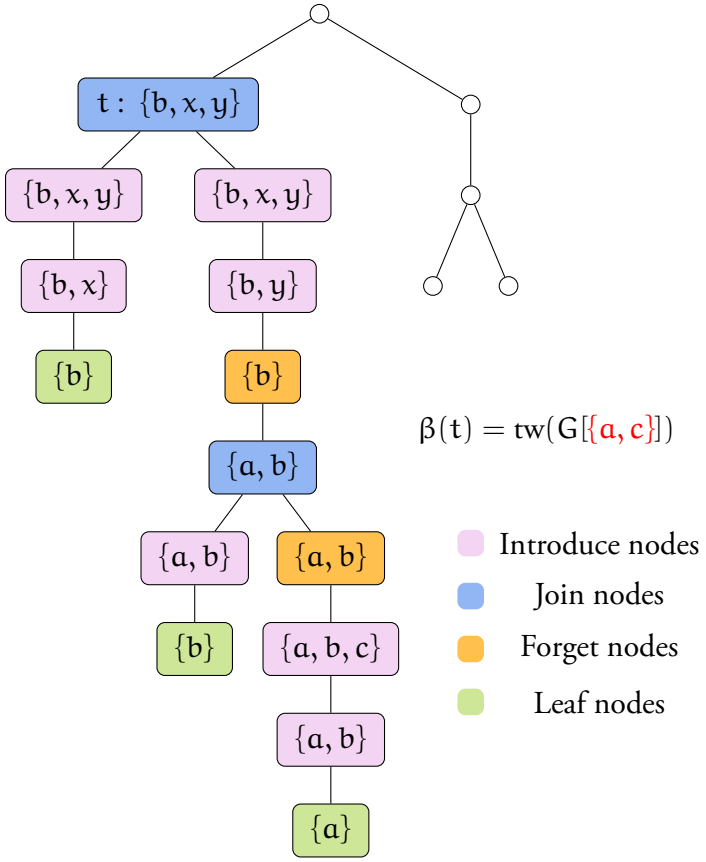
$$H_t := G_t[V(G_t) \setminus X_t].$$

We define the function $\beta : V(T) \to \mathbb{N}$ as follows:

$$\beta(t) = \mathrm{tw}(H_t).$$

Notice that the value of β is zero at the leaves, and as we move our way up the tree starting from the leaves, β is non-decreasing — that is, it either increases or stays the same:

- If t is a leaf node, H_t is empty, and $\mathrm{tw}(H_t)$ is zero.
- If t is an introduce node, and s is the child of t, then H_t and H_s correspond to the same graph and the value of β does not change.
- If t is a forget node, and s is the child of t, then H_t has at most one vertex more than H_s, and the treewidth of H_s is at most one more than H_t.



A cross-section of a nice tree-decomposition

Figure 8.1: An example of $\beta()$ at a join node

- If t is a join node with children r and s, it is evident that the treewidth of H_t is at most

$$\max\{\text{tw}(H_s), \text{tw}(H_r)\},$$

  thus, relative to one of r or s, β does not change, and relative to the other, it increases.

Notice that we do not expect to be able to compute β precisely at each node in polynomial time, since computing the optimal treewidth of a general graph is NP-hard. However, we will see that the extent to which we will need to compute $\beta()$ will be computable in polynomial time.

We compute β in a bottom up fashion starting from base nodes and moving towards

the root: at every node we check if $\beta(t) = tw(H_t) > d$. We abort this computation the first time that we find a node t such that $\beta(t) = tw(H_t) > d$.

Let $P = V(H_t)$, $Q = (V(G) \setminus P) \setminus X_t$ and $S = X_t$, where X_t is the subset of vertices in the bag corresponding to node t. Let G_H and G^* denote, respectively, the graphs induced on P and Q. Notice that G_H has exactly the property we need: 1. it has treewidth at least d by choice, and this guarantees a minor model of H, and 2. has treewidth at most $(d + 1)$, because it is easily checked that if t is the first node at which the $\beta(t) > d$, then the increase in $\beta()$ is at most one from its child node, and thus the problem of finding a $\mathcal{F}$-hitting set is polynomial time solvable on G_H.

We can now recursively solve the problem on the graphs induced on G_H and the connected components of G^*, after including S in our solution. Notice that the base case is achieved when the treewidth of the entire graph is at most d. An outline of the algorithm can be found in Algorithm 1 and a schematic view can be found in 8.2.

Algorithm 1 HIT-SET-I-(G, k)

if $tw(G) \leqslant d$ **then**
  Find S, an optimal $\mathcal{F}$-hitting set of G using Lemma 7.5.
  if $|S| > k$ **then**
    Return NO.
  else
    Return S.
  end if
end if
Compute an approximate tree decomposition $(T, \mathcal{X} = \{X_t\}_{t \in V(T)})$ of width ℓ.
if $\ell > (k + d)\sqrt{\log(k + d)}$, where d is as in Lemma 7.6 **then**
  Return that G does not have $\mathcal{F}$–hitting set of size at most k.
end if
Convert $(T, \mathcal{X} = \{X_t\}_{t \in V(T)})$ to a nice tree decomposition of the same width.
Find a partitioning of vertex set $V(G)$ into G_H, G^* and X_t (a bag corresponding to a node in T) such that $tw(G_H) = (d + 1)$ as explained in the description of the algorithm.
Compute an optimal solution on G_H, and let Y denote this solution.
Let Z denote HIT-SET-I-$(G^*, k - OPT(G_H))$.
if (Z is NO) or ($|Y| > k$) **then**
  Return NO
end if
Return $X \cup Y \bigcup$ HIT-SET-I-$(G^*, k - OPT(G_H))$

8.2.2 Analysis: Correctness and Running Time

We now turn to an analysis of the algorithm. We first reason that the set that is output by the algorithm is indeed a $\mathcal{F}$-hitting set, and then argue that the algorithm requires polynomial time. Finally, we show that with input (G, k), the size of the $\mathcal{F}$-hitting set returned by the algorithm is bounded by $O(k^2\sqrt{\log k})$. The proof is by induction on the depth of recursion.

Lemma 8.1 (Correctness). *If Algorithm 1 returns a NO answer, then there is no hitting set of size at most k in the input graph G. Else, let S denote the output of Algorithm 1. Then, the following holds:*

(i) S *is a $\mathcal{F}$-hitting set for the input graph G,*
(ii) $|S| = O(k^2\sqrt{\log k})$, *and*
(iii) S *is computed in polynomial time.*

Proof. Suppose the Algorithm returns a NO answer because the approximate treewidth ℓ exceeds $(k + d)\sqrt{\log(k + d)}$, where d is as in Lemma 7.6. In this case, correctness follows from Lemma 7.6 and the correctness of the approximation algorithm that is used to compute the treewidth.

Other than the above, the algorithm returns a NO answer when the subroutine designed to find an optimal hitting set on a subgraph of the input graph returns a solution of size greater than k. The correctness of this is evident: if the size of the optimal hitting set in a subgraph of G exceeds k, then the size of the optimal hitting set of G also exceeds k. This, together with the correctness of the optimal algorithm (Lemma 7.5) implies the correctness of the NO answer in this situation.

Finally, the algorithm returns a NO answer when a recursive call returns a NO answer. The correctness of this follows by induction on the depth of recursion. We may assume, by the induction hypothesis, that if a recursive call to G^* returns a NO answer, then the size of the optimal hitting set of G^* is greater than k. Since G^* is a subgraph of G, clearly G has no hitting set of size at most k. The base case of this induction is described in the paragraphs above.

Thus we may conclude that when the algorithm returns a NO answer, then the input graph G indeed does not admit any $\mathcal{F}$-hitting set of size at most k.

We begin with a proof of (i). Let $\mathfrak{d}$ denote the depth of recursion in Algorithm 1. If $\mathfrak{d}$ is one, then the problem is solved optimally, and the correctness follows from Lemma 7.5.

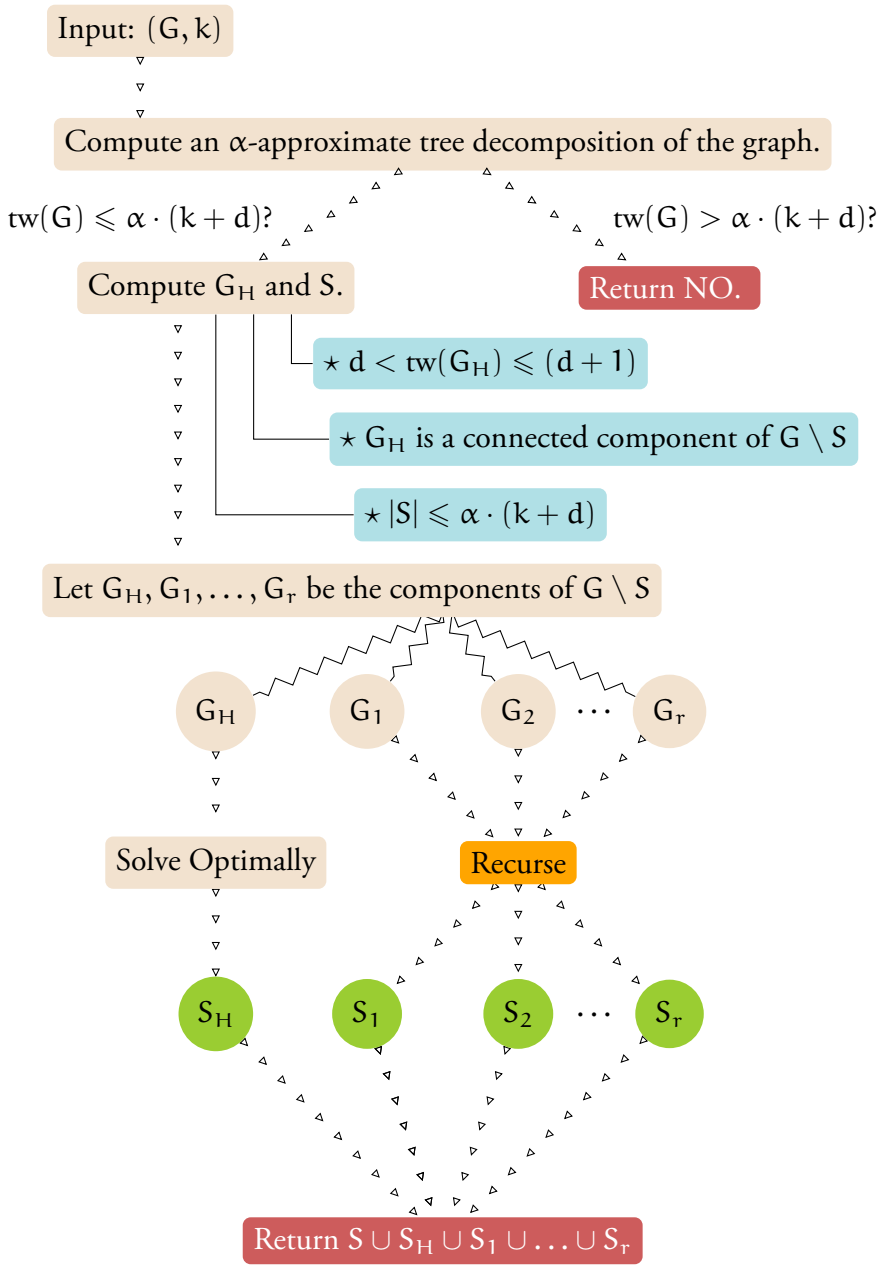


Figure 8.2: The first approximation algorithm: a schematic view.

Our induction hypothesis is that the algorithm returns a $\mathcal{F}$-hitting set when the depth of recursion is $\mathfrak{d}$. Consider the situation when the depth of recursion is $(\mathfrak{d}+1)$. At this point, the algorithm returns as the solution:

$$S(G_H) \cup X_t \bigcup \left(\textsc{Hit-Set-I-}(G^*)\right),$$

where:

- G_H is a subgraph of constant treewidth,
- $S(G_H)$ is an optimal $\mathcal{F}$-hitting set on the graph G_H,
- X_t separates G_H from the rest of the graph,
- G^* is the rest of the graph, $(G \setminus G_H) \setminus X_t$, and
- $|X_t| \leqslant (k+d)d'\sqrt{\log(k+d)}$ (since X_t is the vertices of a bag picked from a tree decomposition of this width).

Notice that for any minor model M_H of H in G, we have the following:

- M_H is contained in G_H,
- M_H is contained in G^*,
- M_H intersects X_t (since M_H is a connected subgraph, and X_t separates G_H and G^*, any M_H that is not entirely contained in one of the these parts is forced to go accross through the separator X_t).

Notice that any M_H is thus "hit", respectively, by:

- $S(G_H)$, which is a hitting set of G_H by Lemma 7.5,
- $\textsc{Hit-Set-I-}(G^*)$: the depth of recursion for G^* is at most $\mathfrak{d}$ (by assumption) — and notice that the output of the algorithm is a $\mathcal{F}$-hitting set of G^* by the induction hypothesis,
- X_t, the separator that is included in the solution.

This proves that the output of $\textsc{Hit-Set-I-}(G)$ is a $\mathcal{F}$-hitting set for G. We now turn to (ii), to show that the size of the hitting set is at most $O(k^2\sqrt{\log k})$. Let $S(G,k)$ denote the output of the algorithm on input (G,k). Let G^*_i and G^i_H denote the graphs that is obtained as G^* and G_H, respectively, at the i^{th} level of recursion. Also, let X^i_t denote the separator that is chosen at the i^{th} level of recursion. Then we have the general relation:

$$S(G^*_i, i) = \mathsf{OPT}(G_H^{(i-1)}) + |S(G^*_{(i-1)}, i-1)| + |X^i_t|.$$

If the algorithm does not return a NO answer, then G_0^* has no minor models of H, and hence $S(G_0^*, 0)$ is the empty set.

$$|S(G_0^*, 0)| = 0.$$

Note that $S(G)$ is given by:

$$OPT(G_H^k) + |S(G_k^*, k-1)| + |X_t^k|.$$

Expanding this using the recurrence above, we get:

$$S(G) = (OPT(G_H^k) + OPT(G_H^{k-1}) + \cdots + OPT(G_H^1)) + k \cdot \max_i\{X_t^i\}.$$

Note that the algorithm returns NO if $OPT(G_H^i) > k$ for any $1 \leqslant i \leqslant k$, so we may assume that $OPT(G_H^i) \leqslant k$. Further, notice that $X_t^i = O(k\sqrt{\log k})$, for all $1 \leqslant i \leqslant k$. Thus, we have:

$$S(G) = k^2 + k \cdot O(k\sqrt{\log k}),$$

or

$$S(G) = O(k^2\sqrt{\log k}).$$

Finally, to show that the algorithm runs in polynomial time, observe that the algorithm involves:

- k computations of an approximate tree decomposition of the input graph (or a subgraph of it),
- k computations of the sets G_H, G^* and X_t,
- k computations of optimal hitting set of G_H.

The approximate tree decompositions can be obtained in polynomial time [FHL08], and the optimal hitting sets for G_H, which is a graph of constant treewidth by design, is also a polynomial time computation (Lemma 7.5). To see that the computation of the sets G_H, G^* and X_t is polynomial time, observe that it involves checking if the treewidth of a certain subgraph is at most a constant d, and these checks are executed

at most polynomially many times (as any nice tree decomposition of a graph on n vertices can be shown to have at most a polynomial number nodes).

Thus, in summary, it is easily checked that the algorithm runs in polynomial time, and this concludes the proof.

□

8.2.3 The Approximation Ratio

To get an approximation algorithm out of Algorithm 1, we run it with different values of k, $1 \leqslant k \leqslant n$, starting from 1, and stop at the earliest point when we are successful. By our choice of k we know that G does not have $\mathcal{F}$-hitting set of size at most $k-1$ and hence $\mathsf{OPT} \geqslant k$. This implies that the size of S returned by Algorithm 1 is at most $O(k^2\sqrt{\log k})$ (see Lemma 8.1). This gives us our first approximation algorithm:

Lemma 8.2. *Let $\mathcal{F}$ be an obstruction set containing a planar graph, and let OPT be the size of the smallest $\mathcal{F}$-hitting set. Given a graph G, in polynomial time we can find a subset $S \subseteq V(G)$ such that $G[V \setminus S]$ contains no element of $\mathcal{F}$ as a minor and $|S| = O(\mathsf{OPT}^2 \cdot \sqrt{\mathsf{OPT}})$.*

8.3 Bootstrapping: An Improved Algorithm

In this section, we improve the approximation ratio further from $O(\mathsf{OPT}\cdot\sqrt{\log \mathsf{OPT}})$ to $O(\log \mathsf{OPT}^{3/2})$. However, this does not render Algorithm 1 obsolete, because we *use the output of Algorithm 1* to obtain the algorithm with an improved approximation ratio. For convenience, we will again resolve the decision version of the problem approximately, and then run this algorithm repeatedly to get the desired approximation ratio relative to OPT. For ease of presentation, we will discuss the ideas of the algorithm with respect to the decision version, and we use (G, k) denote the input instance of Planar $\mathcal{F}$-Deletion.

The central theme is to define a function μ on graphs that respects the following:

- ▷ the value of μ on the graph corresponding to the input instance is polynomial in k,
- ▷ we can find a small separator X that separates the input graph into two parts G_A and G_B such that the values $\mu(G_A)$ and $\mu(G_B)$ are at most a constant fraction of $\mu(G)$,

▷ and the problem is easily solved when $\mu(G)$ is a small constant.

Then the idea is again to recurse on the parts G_A and G_B and picking X in the solution. In the first approximation algorithm, at any recursion depth we had to deal with only two instances. In this situation, the recursion tree unfolds as a binary tree (not necessarily complete). So this time, with some additional observations, we maintain that the *total size* of the separators that are picked at any given depth of recursion is "small" - let's say λ. Thus, the size of the hitting set returned will be:

$$\lambda \cdot \text{depth of recursion},$$

Note that the depth of recursion will be logarthmic in the value of the function on the entire graph: this is due to the "constant fraction drop" in the value of the function whose value is polynomial in k to begin with, and the fact that we will never recurse beyond the point when the value of the function is, say, zero. So overall, this implies a depth of recursion that is proportional to $\log(k^{O(1)})$, or $O(\log k)$; and the size of the hitting set obtained will be:

$$\lambda \cdot O(\log k).$$

Thus, if we can guarantee that the total size of the separators picked at every stage is $O(k\sqrt{\log k})$, as before, then we will realize the improvement in the approximation ratio from $O(OPT \cdot \sqrt{\log OPT})$ to $O(\log OPT^{3/2})$.

We now proceed with an informal description of the function that meets our requirements. Let us begin obtaining a hitting set of size at most $O(k^2\sqrt{\log k})$ using Algorithm 1. If the algorithm says NO we simply repeat the negative answer. Else, consider the solution produced by the algorithm. For ease of discussion, let us refer to the vertices of the graph that belong to the hitting set as marked vertices, and we use m to refer to the total number of marked nodes ($m = O(k^2\sqrt{\log k})$). Let us define $\mu(G)$ to be the number of marked vertices in G:

$$\mu(G) := \text{ Number of marked vertices in } G.$$

Note that the set of marked vertices stays fixed throughout the algorithm, even on subproblems, and is not recomputed at any point. Thus, Algorithm 1 is run only once, and the entire analysis is with respect to this fixed set of marked vertices.

Now, to find a small separator, the natural starting point is again a tree decomposition of G; because the vertices in bags of the tree decompositions are good candidate separators, and they will be of the desired size given the fact that the treewidth is at most $(k+d)\sqrt{\log(k+d)}$ for any feasible instance. It turns out that the treewidth "spreads out" over all the instances at any given depth of recursion. This has the consequence that the total size of the separators picked at any depth of recursion continues to be of the desired size. However, we defer the details of this calculation to the next section.

Thus it is now only a matter of finding a bag that separates the graph into parts where the number of marked nodes is a constant fraction of the total number of marked nodes in the graph. For any bag t, let us consider $\mu(H_t)$, where H_t is the subgraph induced on the union of vertices appearing in all bags present in the subtree rooted at t, except the vertices at the bag corresponding to node t itself.

For the given tree decomposition, let us analyze the behavior of μ on the graphs H_t. It is easy to verify the following:

- the value of $\mu(H_t)$ at an introduce node t is the same as the value of $\mu(H_s)$ if s is the child of t.
- the value of $\mu(H_t)$ at a forget node t is at most one more than value of $\mu(H_s)$ if s is the child of t (it is possible that the forgotten vertex v was marked, and this vertex appears in the graph H_t but not in H_s.
- the value of $\mu(H_t)$ at a join node t is $\mu(H_r)+\mu(H_s)$, where r and s are the children of t.

Consider the node at the greatest distance from the root at which $\mu(H_t)$ is more than, say, (m/α): that is, a node t such that for some constant α, $\mu(H_t) > (m/\alpha)$ and $\mu(H_s) \leqslant (m/\alpha)$, where s is the child of t (if t has two children, then the aforesaid is required to be true of *each* of the children). Notice that any node t which realizes this property is either a join node or a forget node, because the value of $\mu()$ does not "jump" at an introduce node.

Notice that if t is a forget node with child s, the subgraphs H_s and $G \setminus (H_t \cup X_t)$ are exactly the pieces we need, separated by X_s. Notice that $\mu(G \setminus H_t)$ is at most $m(1-1/\alpha)$, since the total number of marked vertices in the graph is at most m and the number of marked vertices of H_t is at least (m/α) by choice of t. Thus, $\mu(G \setminus (H_t \cup X_t))$ is also at most $(m(1-1/\alpha))$, since $G \setminus (H_t \cup X_t)$ is a subgraph of $G \setminus H_t$. So for any choice of α strictly greater than 1, we have the constant factor drop that we seek.

The case when t is a join node is slightly more involved. Let r and s be the child nodes of t. Notice that $X_t = X_s = X_r$, and the natural candidates for G_A and G_B (the subgraphs to recurse on) would be some combinations of the pieces obtained after the deletion of X_t. Let's consolidate what we know:

$$\mu(H_s) \leqslant m/\alpha \text{ and } \mu(H_r) \leqslant m/\alpha,$$

$$\mu(H_s) + \mu(H_r) \leqslant m$$

$$\mu(H_t) > m/\alpha,$$

and

$$\mu(G \setminus (H_t \cup X_t)) \leqslant (1 - 1/\alpha)m.$$

Let us first get an easy case out of the way: if at least one of $\mu(H_s)$ or $\mu(H_r)$ was at most $m/2\alpha$, then the choice of G_A and G_B is immediate: suppose, WLOG, that $\mu(H_s) \leqslant m/2\alpha$, then let G_A be $H_s \cup G \setminus (H_t \cup X_t)$, and let G_B be H_r. Notice that $\mu(G_B) \leqslant m/\alpha$, and

$$\mu(G_A) \leqslant m/2\alpha + m(1 - 1/\alpha) = (1/2\alpha + 1 - 1/\alpha)m = \frac{(2\alpha - 1)}{2\alpha} \cdot m,$$

and notice that this a constant fraction of m for any choice of α.

On the other hand, if neither of $\mu(H_s)$ or $\mu(H_r)$ is at most $m/2\alpha$, then we have:

$$\mu(H_s) > m/2\alpha \text{ and } \mu(H_r) > m/2\alpha.$$

In particular, let:

$$\mu(H_s) = m(1/2\alpha + \delta_s) \text{ and } \mu(H_r) = m(1/2\alpha + \delta_r).$$

Since we know that $\mu(H_s) + \mu(H_r) \leqslant m$, we have that:

$$(1/\alpha + \delta_s + \delta_r \leqslant 1) \Rightarrow \delta_s + \delta_r \leqslant 1 - 1/\alpha,$$

and this means that at least one of δ_s or δ_r is at most $(1 - 1/\alpha)/2$: let

$$\delta_s \leqslant \frac{\alpha - 1}{2\alpha},$$

WLOG. Now we are tempted to recurse on $H_s \cup G \setminus (H_t \cup X_t)$ and H_r. As before, $\mu(G_B) \leqslant m/\alpha$. Let us now examine $\mu(H_s \cup G \setminus (H_t \cup X_t))$:

$$\mu(H_s \cup G \setminus (H_t \cup X_t)) \leqslant m(1/2\alpha + (\alpha - 1)/2\alpha) + m(1 - 1/\alpha),$$

which works out to:

$$\mu(H_s \cup G \setminus (H_t \cup X_t)) \leqslant m \cdot \frac{(3\alpha - 2)}{2\alpha}.$$

For this to be a constant fraction of m, we require that $(3\alpha - 2) < 2\alpha$, or equivalently, that $\alpha < 2$. So by what we had in the case of the forget node, and the conclusion we have drawn here, we are successful in finding suitable choices of G_A and G_B in all cases for any choice of α in the range $(1, 2)$. For ease of presentation, we chose α to be $3/2$, or we look for nodes in the tree decomposition where the value of μ flips to being more than $(2/3)m$. With these ideas in mind, we now turn to a formal description of the algorithm.

8.3.1 The Second Algorithm

As before, we first describe an algorithm that solves the decision version, where k denotes the size of the solution sought, and we will see subsequently that this algorithm produces an approximate solution with the desired ratio when run at most n times with different values of k. The input to this algorithm is (G, k), and a hitting set S of size at most $O(k^2\sqrt{\log k})$. We use m to denote $V(G) \cap S$.

If $|S| = 1$, there is a $\mathcal{F}$-hitting set of size one. This solution can be found in $O(n)$ time by experimenting with every vertex as a candidate solution, and we return the solution thus obtained. If $|S| = 0$, there is nothing to do, we return the empty set.

We find an approximate tree decomposition of width ℓ using an algorithm of Feige et al. [FHL08] such that

$$\mathrm{tw}(G) \leqslant \ell \leqslant d'\mathrm{tw}(G)\sqrt{\log \mathrm{tw}(G)},$$

where d' is a fixed constant. As before, if $\ell > (k+d)d'\sqrt{\log(k+d)}$, then by Lemma 7.6, we know that the size of a minimum $\mathcal{F}$-hitting set of G is at least $(k+1)$, and thus the algorithm aborts at this point and returns a NO answer. Else, we know that the treewidth of G is bounded:

$$tw(G) \leqslant \ell \leqslant (k+d)d'\sqrt{\log(k+d)}.$$

In the next step we compute G_A, G_B and X. This is done by a bottom-up calculation performed on the tree decomposition. We begin by converting the given tree decomposition to a nice tree decomposition of the same width in linear time [Klo94]. Given a nice tree decomposition $(T, \mathcal{X} = \{X_t\}_{t \in V(T)})$ of G, recall that H_t refers to the subgraph induced on the union of vertices appearing in all bags present in the subtree rooted at t, except the vertices at the bag corresponding to node t itself. Notationally,

$$H_t := G_t[V(G_t) \setminus X_t].$$

We define the function $\beta : V(T) \to \mathbb{N}$ as follows:

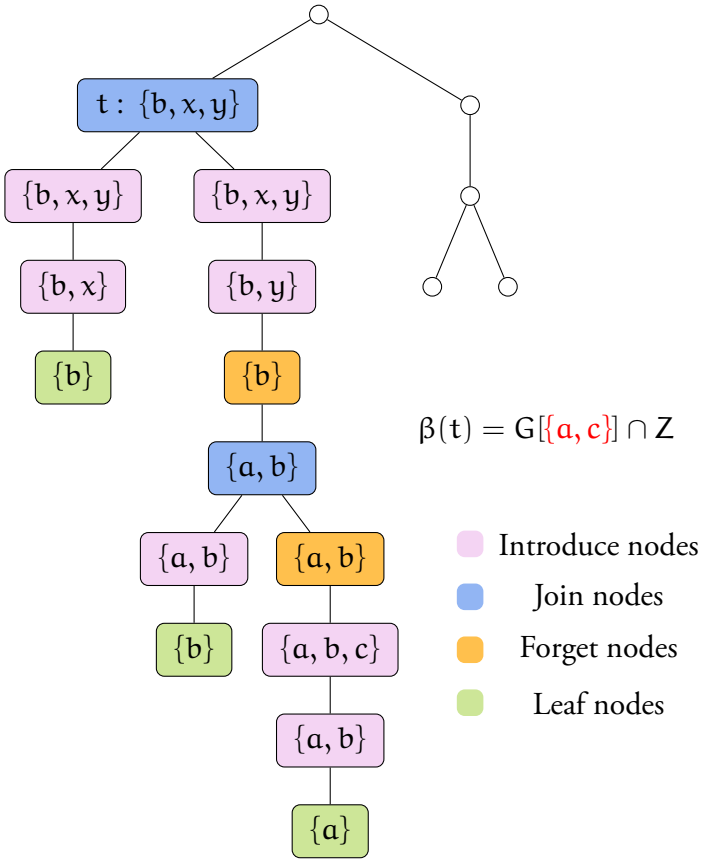
$$\beta(t) = |H_t \cap S|.$$

Notice that the value of β is zero at the leaves, and as we move our way up the tree starting from the leaves, β is non-decreasing. This follows from our discussion of the behavior of $\mu()$ preceding this algorithm, and is easily checked as well.

We compute β in a bottom up fashion starting from base nodes and moving towards the root. We abort this computation the first time that we find a node t such that $\beta(t) > (2/3)m$. We determine subgraphs G_A and G_B of G in the following manner:

- t **is a forget node with child node** s. In this case, let G_A be H_s and let G_B be $G \setminus (H_t \cup X_t)$.
- t **is a join node with child nodes** s **and** r
  - **If** $\beta(s) \leqslant m/2$: G_A is $H_s \cup G \setminus (H_t \cup X_t)$ and G_B is H_r.
  - **If** $\beta(r) \leqslant m/2$: G_A is $H_r \cup G \setminus (H_t \cup X_t)$ and G_B is H_s.

It is easy to check that these cases are exhaustive. We can now recursively solve the problem on the graphs induced on G_A and G_B, after picking X in the solution. In the instances that we recurse on, the hitting sets given by $G_A \cap Z$ and $G_B \cap Z$ respectively. Notice that the value of the parameter k is unchanged in the recursive instances - the depth of recursion is not determined by k, and k is simply used as a reference to



A cross-section of a nice tree-decomposition

Figure 8.3: An example of $\beta()$ at a join node

bound the width of the tree decompositions obtained in the recursive instances. The base cases are realized when the value of β at the root is zero or one. An outline of the algorithm can be found in Algorithm 2.

8.3.2 Analysis and Approximation Ratio

It is easy to check, and is exactly along the lines of the proof of Lemma 8.1, that Algorithm 2 returns a $\mathcal{F}$-hitting set, and that the algorithm runs in polynomial time. Both proofs can be carried out by a simple induction on the depth of recursion. We will provide a proof of the fact that the size of solution output by the algorithm is $O(k(\log k)^{3/2})$, and conclude this chapter with a brief note on how this algorithm can be used to derive an approximation of the promised ratio.

Lemma 8.3. *Let k be the size of the optimal $\mathcal{F}$-hitting set of the graph G. Then, the size of the hitting set output by Algorithm 2 is $O(k(\log k)^{3/2})$.*

Proof. Let

$$G_j(1), G_j(2), \ldots, G_j(2^j)$$

denote the graphs corresponding to instances obtained at recursion depth i. Notice that the recursion tree is a binary tree, and there are at most 2^i instances at recursion depth i (with the convention that the root of the tree that corresponds to the original instance has recursion depth 0). For a graph G, we will also use the notation G_A and G_B to denote the two instances that Algorithm 2 recurses on when given G as input.

At recursion depth j, let

$$\lambda_j(1), \lambda_j(2), \ldots, \lambda_j(2^j)$$

denote the sizes of the separators chosen for each of the 2^j instances (if there are fewer instances, then the remaining λ_j's are identified with zero). Let λ_j denote the sum of the sizes of all the separators chosen at recursion depth j:

$$\lambda_j = \lambda_j(1) + \lambda_j(2) + \ldots + \lambda_j(2^j).$$

Further, let

$$k_j(1), k_j(2), \ldots, k_j(2^j)$$

denote the sizes of the optimal $\mathcal{F}$-hitting set of the graphs $G_j(1), G_j(2), \ldots, G_j(2^j)$ (respectively). Recall that k is the size of the optimal $\mathcal{F}$-hitting set of the graph G. Note that the graphs corresponding to instances obtained at any given depth of recursion are subgraphs of G that are vertex-disjoint: this can be shown by induction on the depth of recursion. At the first level of recursion these graphs are obtained by the removal of a separator, and hence are vertex-disjoint. Assuming that the graphs at recursion depth j are vertex-disjoint, the graphs at recursion depth $(j+1)$ are vertex-disjoint because they are obtained from the graphs at recursion depth j after the removal of vertex separators. As a result, we have the following:

$$k_j(1) + k_j(2) + \ldots + k_j(2^j) \leqslant k.$$

Observe that the tree decomposition of the graph $G_j(i)$ for any $1 \leqslant i \leqslant 2^j$ has width at most $(k_j(i)+d)$ (c.f. Lemma 7.6) and therefore the approximate tree decomposition has width at most:

$$(k_j(i) + d)\sqrt{\log(k_j(i) + d)}.$$

Since $\lambda_j(i)$ is the size of the separator obtained by picking a bag in this tree decomposition, we have that:

$$\lambda_j(i) = (k_j(i) + d)\sqrt{\log(k_j(i) + d)},$$

and that λ_j is given by:

$$\sum_{1 \leqslant i \leqslant 2^j} (k_j(i) + d)\sqrt{\log(k_j(i) + d)}.$$

We claim that the above is at most $(k + d(2^j))\sqrt{\log(k + d)}$. This is easy to see:

$$\begin{aligned}
k_j(1) + k_j(2) + \ldots + k_j(2^j) &\leqslant k \\
(k_j(1) + d) + (k_j(2) + d) + \ldots + (k_j(2^j) + d) &\leqslant k + d(2^j) \\
\left((k_j(1) + d) + (k_j(2) + d) + \ldots + (k_j(2^j) + d)\right)\sqrt{\log(k+d)} &\leqslant (k + d(2^j))\sqrt{\log(k+d)} \\
\left(\sum_{1 \leqslant i \leqslant 2^j} (k_j(i) + d)\right)\sqrt{\log(k+d)} &\leqslant (k + d(2^j))\sqrt{\log(k+d)} \\
\sum_{1 \leqslant i \leqslant 2^j} \left((k_j(i) + d)\sqrt{\log(k+d)}\right) &\leqslant (k + d(2^j))\sqrt{\log(k+d)} \\
\sum_{1 \leqslant i \leqslant 2^j} \left((k_j(i) + d)\sqrt{\log(k_j(i)+d)}\right) &\leqslant (k + d(2^j))\sqrt{\log(k+d)},
\end{aligned}$$

where the last step follows from the fact that $k_j(i) \leqslant k$. Thus, we finally have that:

$$\lambda_j \leqslant (k + d(2^j))\sqrt{\log(k+d)}.$$

This establishes that the size of all the separators at any recursion depth j when put together is at most $(k + d(2^j))\sqrt{\log(k+d)}$. We are now ready to prove a bound on the size of the solution obtained by Algorithm 2.

Let $\mathfrak{S}(G, S)$ denote the output of the algorithm on input (G, S). We then have the following recursive relationship:

$$\begin{aligned}
\sum_{1 \leqslant i \leqslant 2^j} |\mathfrak{S}(G_j(i), S_j(i))| = \; & \sum_{1 \leqslant i \leqslant 2^j} \lambda_j + \\
& \sum_{1 \leqslant i \leqslant 2^j} |\mathfrak{S}((G_j(i))_A, S_j(i) \cap (G_j(i))_A)| + \\
& \sum_{1 \leqslant i \leqslant 2^j} |\mathfrak{S}((G_j(i))_B, S_j(i) \cap (G_j(i))_B)|
\end{aligned}$$

Notice that $(G_j(i))_A$ corresponds to the graph $G_{j+1}(2i-1)$ and the graph $(G_j(i))_B$ corresponds to the graph $G_{j+1}(2i)$, and as expected, the number of instances at recursion depth $(j+1)$ has potentially doubled from the number of instances at recursion depth j. The algorithm stops when $S \cap G_A$ is an empty set:

$$|\mathfrak{S}(G_j(i), \emptyset)| = 0$$

Note that the size of the hitting set output by the algorithm is given by:

$$|\mathfrak{S}(G_1(1), (S \cap G_1(1)))| + |\mathfrak{S}(G_1(2), (S \cap G_1(2)))| + \lambda_0,$$

or equivalently:

$$|\mathfrak{S}(G_A, (S \cap G_A))| + |\mathfrak{S}(G_B, (S \cap G_B))| + \lambda_0.$$

Let γ denote the depth of the recursion tree. Unfolding the recursion above, it is easy to see that we obtain the size of the hitting set output by the algorithm to be:

$$S(G) = \lambda_0 + \lambda_1 + \ldots + \lambda_\gamma.$$

Recall that:

$$\lambda_j \leqslant (k + d(2^j))\sqrt{\log(k+d)},$$

and therefore, the size of the hitting set amounts to:

$$(k\sqrt{\log(k+d)}) \cdot \gamma + \sum_{1 \leqslant j \leqslant \gamma} d(2^j)$$

Clearly, the crucial term that governs the size of the output is the depth of recursion. Notice that the choices of G_A and G_B were made to ensure:

$$\text{either } |S \cap G_A| \leqslant \frac{5}{6}|S|, \text{ or } |S \cap G_A| \leqslant \frac{1}{3}|S|$$

and

$$|S \cap G_B| \leqslant \frac{2}{3}|S|$$

So the depth of recursion is proportional to $O(\log |S|)$ or $O(\log(k^2\sqrt{k}))$, which amounts to $O(\log k)$. Substituting for γ, we see that the term:

$$\sum_{1 \leqslant j \leqslant \gamma} d(2^j)$$

is proportional to k, and

$$(k\sqrt{\log(k+d)}) \cdot \gamma$$

is simply $(k\sqrt{\log(k+d)}) \cdot (\log k)$. Therefore, we have that the size of the hitting set returned by the algorithm is $O(k(\log k)^{3/2})$, as desired.

□

To get an approximation algorithm out of Algorithm 2, we run it with different values of k, $1 \leqslant k \leqslant n$, starting from 1, and stop at the earliest point when we are successful. By our choice of k we know that G does not have $\mathcal{F}$-hitting set of size at most $k-1$ and hence $OPT \geqslant k$. This, together with Lemma 8.3,implies that the size of S returned by Algorithm 2 is $O(OPT(\log OPT)^{3/2})$. This gives us the promised approximation algorithm:

Theorem 8.1. *Let $\mathcal{F}$ be an obstruction set containing a planar graph, and let* OPT *be the size of the smallest F-hitting set. Given a graph G, in polynomial time we can find a subset $S \subseteq V(G)$ such that $G[V \setminus S]$ contains no element of $\mathcal{F}$ as a minor and $|S| = O(OPT \cdot (OPT)^{3/2})$.*

Algorithm 2 HIT-SET-II-(G, S)

1: **if** $S \cap G = \emptyset$ **then**
2: Return $\emptyset$
3: **end if**
4: Compute an approximate tree decomposition $(T, \mathcal{X} = \{X_t\}_{t \in V(T)})$ of width ℓ.
5: Convert $(T, \mathcal{X} = \{X_t\}_{t \in V(T)})$ to a nice tree decomposition of the same width.
6: Find a partitioning of vertex set $V(G)$ into G_A, G^B and X_t (a bag corresponding to a node in T) such that $|G_A \cap S| \leqslant (5/6)|S|$, and $|G_B| \leqslant (2/3)|S|$, as described in the proof.
7: Return HIT-SET-II-$(G_A, S \cap G_A) \cup$ HIT-SET-II-$(G_B, S \cap G_B) \cup X_t$

This concludes our description of the approximation algorithm for PLANAR $\mathcal{F}$-DELETION. To emphasize the consequences of this algorithm, we summarize a few implications of it, obtained by making different choices for the family $\mathcal{F}$. One of the consequences of this algorithm is an approximation for the TREEWIDTH η-DELETION SET problem, which is the following. Let η be a fixed constant. In the TREEWIDTH η-DELETION SET problem, we are given an input graph G and the objective is to delete the minimum number of vertices from a graph such that the resulting graph has treewidth at most η. For instance, the TREEWIDTH 1-DELETION SET is simply the FEEDBACK VERTEX SET problem. We obtain the following corollary of Theorem 8.1.

Corollary 8.2. Feedback Vertex Set, Diamond Hitting Set, Pathwidth One Deletion Set, Outerplanar Deletion Set *and* Treewidth η-Deletion Set *admit a factor* $O(\log^{3/2} n)$ *approximation algorithm on general undirected graphs.*

One recognizes one's course
by discovering the paths that stray from it.

Albert Camus

That's a nice trick.

The Matrix Reloaded

In this chapter we show that if the obstruction set $\mathcal{F}$ contains a planar graph then the Planar $\mathcal{F}$-Deletion problem has a linear vertex kernel on graphs excluding $K_{1,t}$ as an induced subgraph:

Theorem 9.1. *Let $\mathcal{F}$ be an obstruction set containing a planar graph. Then Planar $\mathcal{F}$-Deletion admits a linear vertex kernel on graphs excluding $K_{1,t}$ as an induced subgraph, where t is a fixed integer.*

Several well studied graph classes do not contain graphs with induced $K_{1,t}$. Of course, every graph with maximum vertex degree at most $(t-1)$ is $K_{1,t}$-free. The class of $K_{1,3}$-free graphs, also known as claw-free graphs, contains line graphs and de Bruijn graphs. Unit disc graphs are known to be $K_{1,7}$-free [CCJ90].

As a corollary we obtain that Feedback Vertex Set, Diamond Hitting Set, Pathwidth One Deletion Set, and Outerplanar Deletion Set admit a linear vertex kernel on graphs excluding $K_{1,t}$ as an induced subgraph. With the same methodology we also obtain a $O(k \log k)$ vertex kernel for Disjoint Cycle Packing on graphs excluding $K_{1,t}$ as an induced subgraph. We note that Disjoint Cycle Packing does not admit a polynomial kernel on general graphs [BTY09] unless coNP $\subseteq$ NP/poly.

We begin by recalling the fact that all YES-instances of this problem have treewidth bounded in the parameter k (see Lemma 7.6 in Chapter 7 for a proof):

Lemma. *Let $\mathcal{F}$ be an obstruction set containing a planar graph of size h. If G has an $\mathcal{F}$-hitting set S of size at most k, then $tw(G \setminus S) \leqslant d$ and $tw(G) \leqslant k + d$, where $d = 20^{2(14h-24)^5}$.*

Central to the reduction rules obtained in this chapter is the notion of a *protrusion*, and a method for reducing them. Recall that a protrusion is a part of the graph that has constant treewidth, and is "cut off" from the rest of the graph by a constant-sized separator. We state the result that lets us simplify protrusions whenever the problem in question has finite integer index.

The Protrusion Rule This rule suggests the replacement of any large r-protrusion for a fixed constant r that depends only on $\mathcal{F}$ (that is, only on the problem) with a *smaller and equivalent* r-protrusion. For this, we recall Lemma 6.3 (see Chapter 6):

Lemma ([BFL+09]). *Let Π be a problem that has finite integer index. Then there exists a computable function $\gamma : \mathbb{N} \to \mathbb{N}$ and an algorithm that given an instance (G, k) and an r-protrusion X of G of size at least $\gamma(r)$, runs in $O(|X|)$ time and outputs an instance (G^*, k^*) such that $|V(G^*)| < |V(G)|$, $k^* \leqslant k$, and $(G^*, k^*) \in \Pi$ if and only if $(G, k) \in \Pi$.*

Remark 9.1. *If G does not have $K_{1,t}$ as an induced subgraph then the proof of Lemma 6.3 also ensures that the graph G' does not contain $K_{1,t}$ as an induced subgraph. This ensures that the reduced instance belongs to the same graph class as the original. The remark is not only true about the class of graphs excluding $K_{1,t}$ as an induced subgraph, but also for any graph class $\mathcal{G}$ that can be characterized by either finite set of forbidden subgraphs or induced subgraphs or minors. That is, if G is in $\mathcal{G}$ then the graph G' returned by Lemma 6.3 is also in $\mathcal{G}$.*

Given this setup, we are only left with the task of efficiently finding large r-protrusions whenever the instance considered is large enough. Also, we would need to prove that Planar $\mathcal{F}$-Deletion has finite integer index. These tasks are accomplished in the next section, and in the subsequent section we perform an analysis to determine the size of the kernel that is obtained by repeated application of the protrusion rule until no longer possible.

9.1 Finding Protrusions

The first challenge is to argue the existence of a substantial-sized protrusion whenever the graph excludes $K_{1,t}$ as an induced subgraph, is a YES-instance of Planar $\mathcal{F}$-Deletion, and is suitably large. First, recall that by Lemma 7.3, we have the following:

♢ Let X be some subgraph of G that has treewidth τ for some constant τ. We know that given X, we can find a $(2\tau+2)$-protrusion within X. Further, the size of this protrusion depends inversely on the boundary of X — specifically, it turns out that the protrusion is of size:

$$\left(\frac{|X|}{4\cdot|\partial(X)|+1}\right).$$

Observe that we already know of a X that would fit the result above. Consider S, the approximate $\mathcal{F}$-hitting set from the previous chapter. When we remove S from G, recall that we are left with a subgraph of constant treewidth. In subsequent discussion, we use $G \setminus S$ as our choice of X.

Now, let us spend a moment considering how protrusions of the kind we find will be useful towards kernelization. From Lemma 6.3, we know that if we have an α-protrusion on more than $\gamma(\alpha)$ vertices, the graph can be subjected to a protrusion-based reduction. This also means that when the protrusion rule no longer applies, it is because the size of the protrusion falls short of $\gamma(\alpha)$. In particular, in our situation this means:

$$\overbrace{\left(\frac{|X|}{4\cdot|\partial(X)|+1}\right)}^{\text{The size of the protrusion}} \leqslant \gamma(2\tau+2)$$

Rewriting this, we obtain:

$$|X| \leqslant \gamma(2\tau+2) + 4\cdot|\partial(X)| + 1.$$

This means that we have a bound on X the moment we have some bound on $\partial(X)$, which, in our case, amounts to controlling the neighborhood of the approximate hitting set. Observe further that a bound on X implies a bound on the size of the entire graph: since X is $G \setminus S$, the total number of vertices is simply $(|X|+|S|)$ and $|S|$ is already polynomially bounded in k. Thus, our next focus is to obtain a bound on $\partial(X)$, or the neighborhood of S; and to this end we will show the following:

♢ If $\mathcal{F}$ contains a planar graph of size h, G excludes $K_{1,t}$ as an induced subgraph, and S is a $\mathcal{F}$-hitting set, then $\partial_G(G\setminus S) \equiv N(S) \leqslant g(h,t)\cdot|S|$ for some computable function g of h and t.

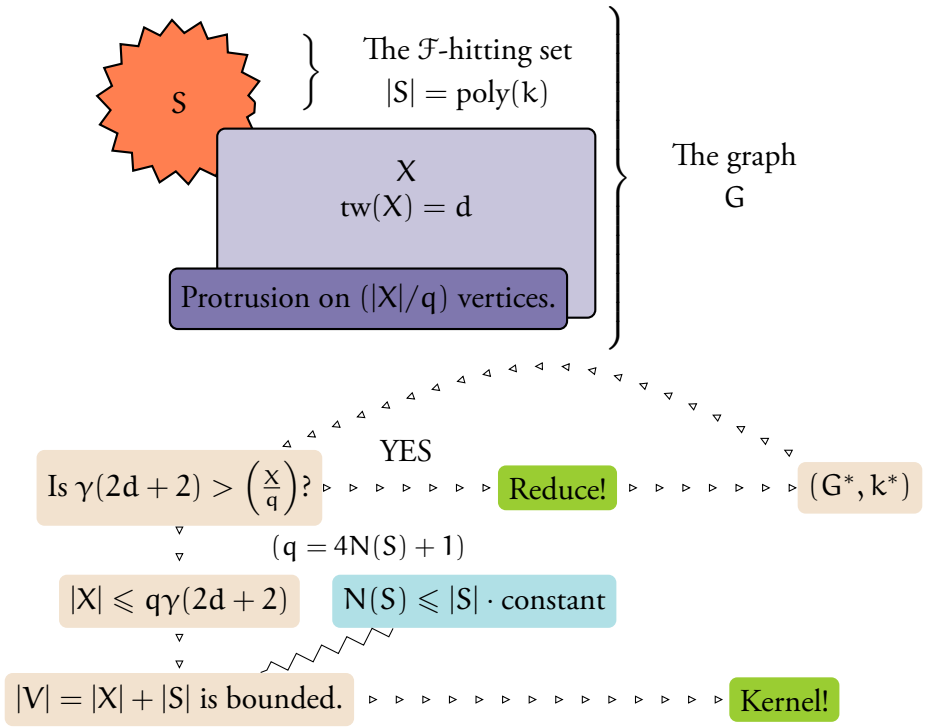


Figure 9.1: Simplifying Protrusions: the overall scheme.

We first make the arguments towards these two statements, after which we will show that Planar $\mathcal{F}$-Deletion has finite integer index. The overall strategy of implementing the protrusion replacement technique is shown in Figure 9.1.

9.2 Finding Protrusions

9.2.1 Bounding the boundary of X

We first show that the size of the neighborhood of any hitting set into the rest of the graph is linear in the size of the hitting set itself.

Lemma 9.1. *Let* G *be a graph excluding* $K_{1,t}$ *as an induced subgraph, and let* $\mathcal{F}$ *be an obstruction that contains a planar graph of size* h. *Let* S *be an* $\mathcal{F}$*-hitting set, and let* X *denote* $G \setminus S$. *We then have:*

$$|\partial(X)| \leqslant g(h,t) \cdot |S|,$$

for some computable function g *of* h *and* t.

Proof. The boundary of X is the set of vertices in X with neighbors in S, or equivalently, the neighborhood of S in X:

$$\partial(X) = \bigcup_{v \in S} N_X(v).$$

Thus, we have that

$$\begin{aligned} |\partial(X)| &\leqslant \textstyle\sum_{v \in S} |N_X(v)| \\ &\leqslant |S| \cdot \max_{v \in S} |N_X(v)|. \end{aligned}$$

Therefore, notice that it suffices to bound the neighborhood of an arbitrary but fixed vertex $v \in S$ by some function of h and t.

Since $X = G \setminus S$, by Lemma 7.6, $\mathrm{tw}(X) \leqslant d$ for $d = 20^{2(14h-24)^5}$. We will make use of the well known fact that a graph of treewidth d is $d+1$ colorable (see Proposition 2.3 for a short proof). We will now show that $|N_X(v)| \leqslant (t-1)(d+1)$. Consider the graph $G[N_X(v)]$. Since $\mathrm{tw}(G \setminus S) \leqslant d$ we have that $\mathrm{tw}(G[N_X(v)]) \leqslant d$ and hence $G[N_X(v)]$ is $(d+1)$ colorable.

Fix a coloring κ of $G[N_X(v)]$ with $d+1$ colors. Notice that no color class may admit t vertices or more: since the color classes induce an independent set, a color class of size t or more would imply the presence of $K_{1,t}$ as an induced subgraph in G, a contradiction. Since there are at most $(d+1)$ color classes with less than t vertices in each, we have:

$$|N_X(v)| \leqslant (t-1)(d+1).$$

Thus, for

$$g(h,t) = (t-1)(20^{2(14h-24)^5} + 1),$$

we have $|\partial(X) \leqslant |S| \cdot g(h,t)$, as desired. □

9.2.2 Finite Integer Index

Next, we show that Planar $\mathcal{F}$-Deletion has finite integer index. Recall Lemma 6.2, which gives us an easy sufficient condition for proving finite integer index:

Lemma ([BFL+09]). *Every strongly monotone* p-MIN-MSO *problem has finite integer index.*

We apply Lemma 6.2 above to show that Planar $\mathcal{F}$-Deletion has finite integer index. (We refer the reader to Definition 6.9 in Chapter 6 for the definition of strong monotonicity.)

Lemma 9.2. *Planar $\mathcal{F}$-Deletion has finite integer index.*

Proof. One can easily formulate Planar $\mathcal{F}$-Deletion in MSO, which shows that it is a p-min-MSO problem (see Section 7.3 in Chapter 7). To complete the proof that Planar $\mathcal{F}$-Deletion has finite integer index we show that $\Pi =$ Planar $\mathcal{F}$-Deletion is strongly monotone. Given a t-boundaried graph G, with $\partial(G)$ as its boundary, let $S'' \subseteq V(G)$ be a minimum set of vertices in G such that $G \setminus S''$ does not contain any graph in $\mathcal{F}$ as a minor. Let $S = S'' \cup \partial(G)$.

Now for any $(G', S') \in \mathcal{H}_t$ such that $\zeta_G^{\Pi}((G', S'))$ is finite, we have that $G \oplus G'[(V(G) \cup V(G')) \setminus (S \cup S')]$ does not contain any graph in $\mathcal{F}$ as a minor and $|S| \leqslant \zeta_G^{\Pi}((G', S')) + t$. This proves that Planar $\mathcal{F}$-Deletion is strongly monotone. By Lemma 6.2, Planar $\mathcal{F}$-Deletion has finite integer index. □

9.3 Analysis and Kernel Size – Proof of Theorem 9.1

Now we combine everything we have gathered, along with Lemma 7.3 so far into a proof of Theorem 9.1.

Proof of Theorem 9.1. Let (G, k) be an instance of Planar $\mathcal{F}$-Deletion and h be the size of a smallest planar graph in the obstruction set $\mathcal{F}$. We first apply Theorem 8.1, an approximation algorithm for Planar $\mathcal{F}$-Deletion with factor $O(\log^{3/2} OPT)$, and obtain a set S such that $G \setminus S$ contains no graph in $\mathcal{F}$ as a minor. If the size of the set S is more than $O(k \log^{3/2} k)$ then we return that (G, k) is a NO-instance to Planar $\mathcal{F}$-Deletion. This is justified by the approximation guarantee provided by the Theorem 8.1.

Now we obtain the kernel in two phases: we first apply the protrusion rule selectively (Lemma 6.3) and get a polynomial kernel. Then, we apply the protrusion rule exhaustively on the obtained kernel to get a smaller kernel. This is done in order to reduce the running time complexity of the kernelization algorithm. To obtain the kernel we follow the following steps.

Applying the Protrusion Rule. Let X denote $V(G) \setminus S$, and let d denote the treewidth of $G[X]$. By Lemma 7.6, $d \leqslant 20^{2(14h-24)^5}$. We apply Lemma 7.3 and obtain a $2(d+1)$-protrusion Y of G of size at least $\frac{|X|}{4|\partial(X)|+1}$. By Lemma 9.2, Planar $\mathcal{F}$-Deletion has finite integer index. Let $\gamma : \mathbb{N} \to \mathbb{N}$ be the function defined in Lemma 6.3. If $\frac{|X|}{4|\partial(X)|+1} \geqslant \gamma(2d+1)$, then using Lemma 6.3 we replace the $2(d+1)$-protrusion Y in G and obtain an instance (G^*, k^*) such that $|V(G^*)| < |V(G)|$, $k^* \leqslant k$, and (G^*, k^*) is a YES-instance of Planar $\mathcal{F}$-Deletion if and only if (G, k) is a YES-instance of Planar $\mathcal{F}$-Deletion . Recall that G^* also excludes $K_{1,t}$ as an induced subgraph.

Let (G^*, k^*) be a reduced instance with hitting set X. In other words, there is no $(2d+2)$-protrusion of size $\gamma(2d+2)$ in $G^* \setminus X$, and Protrusion Rule no longer applies. We claim that the number of vertices in this graph is bounded by $O(k \log^{3/2} k)$. Indeed, since we cannot apply the Protrusion Rule, we have that $\frac{|X|}{4|\partial(X)|+1} \leqslant \gamma(2d+2)$. Because $k^* \leqslant k$, we have that

$$|V(G^*)| \leqslant \gamma(2d+2)(4|\partial(X)|+1) + |S|.$$

By Lemma 9.1, $|\partial(X)| \leqslant g(h,t) \cdot |S|$ and thus

$$|V(G^*)| = O(\gamma(2d+2) \cdot k \log^{3/2} k) = O(k \log^{3/2} k).$$

This gives us a polynomial time algorithm that returns a vertex kernel of size $O(k \log^{3/2} k)$.

Now we give a kernel of smaller size. We would like to replace every large $(2d+2)$-protrusion in graph by a smaller one. We find a $(2d+2)$-protrusion Y of size at least $\gamma(2d+2)$ by guessing the boundary $\partial(Y)$ of size at most $2d+2$. This could be performed in time $k^{O(d)}$. So let (G^*, k^*) be the reduced instance on which we cannot apply the Protrusion Rule. If G is a YES-instance then there is a $\mathcal{F}$-hitting set X of size at most k such that $tw(G \setminus X) \leqslant d$. Now applying the analysis above with this X yields that $|V(G^*)| = O(k)$. Hence if the number of vertices in the reduced instance G^*, to which we can not apply the Protrusion Rule, is more than $O(k)$ then we return that G is a NO-instance. This concludes the proof of the theorem. □

Corollary 9.2. Feedback Vertex Set, Diamond Hitting Set, Pathwidth One Deletion Set, Outerplanar Deletion Set *admit linear vertex kernel on graphs excluding* $K_{1,t}$ *as an induced subgraph.*

The methodology used in proving Theorem 9.1 is not limited to Planar $\mathcal{F}$-Deletion. For example, it is possible to obtain an $O(k \log k)$ vertex kernel on $K_{1,t}$-free graphs for

Disjoint Cycle Packing, which is for a given graph G and positive integer k to determine if there are k vertex disjoint cycles in G. It is interesting to note that Disjoint Cycle Packing does not admit a polynomial kernel on general graphs [BTY09]. For our kernelization algorithm, we use the following Erdős-Pósa property [EP65b]: given a positive integer ℓ every graph G either has ℓ vertex disjoint cycles or there exists a set $S \subseteq V(G)$ of size at most $(4\ell \log \ell)$ such that $G \setminus S$ is a forest. So given a graph G and positive integer k, we first apply the factor 2 approximation algorithm given in [BBF99] and obtain a set S such that $G \setminus S$ is a forest. If the size of S is more than $(8k \log k)$ then we return that G has k vertex disjoint cycles. Else, we use the fact that Disjoint Cycle Packing [BFL+09] has finite integer index and apply the protrusion reduction rule in $G \setminus S$ to obtain an equivalent instance (G^*, k^*), as in Theorem 9.1. The analysis for kernel size used in the proof of Theorem 9.1 together with the observation that $\mathrm{tw}(G \setminus S) \leqslant 1$ shows that if (G, k) is a yes instance then the size of $V(G^*)$ is at most $O(k \log k)$.

Corollary 9.3. Disjoint Cycle Packing *has* $O(k \log k)$ *vertex kernel on graphs excluding* $K_{1,t}$ *as an induced graph.*

To every problem, there is a most simple solution.

Agatha Christie

But the simplest thing is difficult.

Karl Von Clausewitz

Let θ_c be a graph with two vertices and $c \geqslant 1$ parallel edges (see Figure 10.1). The result that will be the focus of this chapter is the following theorem.

Theorem 10.1. *Let $\mathcal{F}$ be an obstruction set containing θ_c. Then Planar $\mathcal{F}$-Deletion admits a kernel of size $O(k^2 \log^{3/2} k)$.*

Although the theorem addresses a restricted case of the Planar $\mathcal{F}$-Deletion problem, its applicability is not restricted to any graph class and it holds in general.

We use Θ_c-Deletion to refer to the problem of Planar $\mathcal{F}$-Deletion when $\theta_c \in \mathcal{F}$. Also, we use the term θ_c-hitting set to refer to a subset of vertices S such that $G \setminus S$ does not contain any minor models of θ_c.

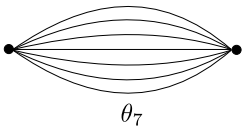


Figure 10.1: The θ_c graph, for $c = 7$.

A number of well-studied NP-hard combinatorial problems are special cases of Θ_c-Deletion. When $c = 1$, this is the classical Vertex Cover problem [NT74]. For $c = 2$, this is another well studied problem, the Feedback Vertex Set problem [BBF99, BYGNR98, CGHW98, Kar72]. When $c = 3$, this is the Diamond Hitting Set problem [FJP10].

We note that the size of the best known kernel for $c = 2$ is $O(k^2)$, which is very close to the size of the kernel in Theorem 10.1. Further, Dell and van Melkebeek showed that no NP-hard vertex deletion problem based on a graph property that is inherited

by subgraphs can have kernels of size $O(k^{2-\varepsilon})$ unless coNP $\subseteq$ NP/poly [DvM10]. Thus, the sizes of the kernels in Theorem 10.1 are tight up to a polylogarithmic factor.

In this chapter, we combine ideas from previous chapters involving kernelization for Independent Feedback Vertex Set (Chapter 5) and Planar $\mathcal{F}$-Deletion on claw-free graphs (Chapter 9). Our first goal is to reduce vertices of "large" degree so as to obtain an equivalent instance where the maximum degree is bounded. In particular, we will be reduce any given instance to a graph where the maximum degree is bounded by some polynomial in k. Notice that having done this, we have arrived at a subclass of $k^{O(1)}$-claw-free graphs. At this stage, we appeal to the procedures applied for Planar $\mathcal{F}$-Deletion on claw-free graphs to complete the reduction.

Our understanding of the minor models of θ_c from Section 7.3 of Chapter 7 will be useful in proving the reduction rule that bounds the maximum degree of the graph. This reduction rule will involve an application of the q-expansion lemma, in a manner that is similar to its application in the context of Independent Feedback Vertex Set (c.f. Chapter 5).

10.1 Finding hitting sets excluding a fixed vertex

Consider a vertex $v \in V(G)$. The kernelization algorithm requires a hitting set T_v of size $k^{O(1)}$ for all minor models of θ_c passing through v, assuming that v is not at the center of a θ_c flower of size more than k (see Chapter 7 for the definition of a θ_c-flower). It is required that $v \notin T_v$. In this section, we show that such a hitting set can be found in polynomial time provided $G \setminus \{v\}$ does not contain any minor model of θ_c, that is, all minor models of θ_c in G pass through v. Note that this is along the lines of Corollary 2.1 in [Tho10] (see Theorem 5.1), which achieved the same result except that the flowers were formed of cycles. Here, we have a more general structure, namely minor models of θ_c. In due course, we will apply this algorithm as a subroutine in finding hitting sets of minor models passing through a vertex v without any assumption on $G \setminus \{v\}$.

Lemma 10.1. *Let G be a n-vertex graph containing θ_c as a minor and v be a vertex such that $G' = G \setminus \{v\}$ does not contain θ_c as a minor and the maximum size of a flower containing v is at most k. Then there exists a set T_v of size $O(k)$ such that $v \notin T_v$ and $G \setminus T_v$ does not contain θ_c as a minor. Further, we can find the set T_v in polynomial time.*

Proof. We first recall Lemma 7.7, because of which we know that the treewidth of a graph that does not contain θ_c as a minor is bounded by $(2c-1)$. Thus, in particular, the treewidth of G' is at most $(2c-1)$.

The algorithm for finding T_v is exactly along the lines of the first approximation algorithm in Chapter 8. We only need a different choice of the β function to work with.

We first find an approximate tree decomposition of width ℓ using an algorithm of Feige et al. [FHL08] such that

$$\mathrm{tw}(G) \leqslant \ell \leqslant d'\mathrm{tw}(G)\sqrt{\log \mathrm{tw}(G)},$$

where d' is a fixed constant. Since $\mathrm{tw}(G') \leqslant (2c-1) = O(1)$, we know that the treewidth of the approximate tree decomposition is also a constant.

We convert the given tree decomposition to a nice tree decomposition of the same width in linear time [Klo94]. Given a nice tree decomposition $(T, \mathcal{X} = \{X_t\}_{t\in V(T)})$ of a graph G, recall that H_t refers to the subgraph induced on the union of vertices appearing in all bags present in the subtree rooted at t, except the vertices at the bag corresponding to node t itself. Notationally,

$$H_t := G_t[V(G_t) \setminus X_t].$$

Further, we let $\mathfrak{M}_{\theta_c}(G, v)$ to denote the maximum number of minor models of θ_c in the graph G, whose vertex sets intersect precisely at v. In other words, $\mathfrak{M}_{\theta_c}(G, v)$ is the size of the largest θ_c-flower passing through v.

We define the function $\beta : V(T) \to \mathbb{N}$ as follows:

$$\beta(t) = \mathfrak{M}_{\theta_c}(H_t \cup \{v\}, v),$$

that is, the value of β at a node t is size of the largest θ_c flower centered at v in the graph $H_t \cup \{v\}$ (recall that the tree decomposition computed is of the graph G', that does not contain v).

Notice that the value of β is zero at the leaves, and as we move our way up the tree starting from the leaves, β is non-decreasing — that is, it either increases or stays the same:

- If t is a leaf node, H_t is empty, and $\mathrm{tw}(H_t)$ is zero.

- If t is an introduce node, and s is the child of t, then H_t and H_s correspond to the same graph and the value of β does not change.
- If t is a forget node, and s is the child of t, then H_t has at most one vertex more than H_s, and the number of minor models of θ_c vertex disjoint except for v in $(H_s \cup \{v\})$ is at most one more than the number of them $(H_t \cup \{v\})$.
- If t is a join node with children r and s, it is evident that the number of vertex disjoint minor models of θ_c in H_t is at most

$$\mathfrak{M}_{\theta_c}(H_s \cup \{v\}, v) + \mathfrak{M}_{\theta_c}(H_r \cup \{v\}, v),$$

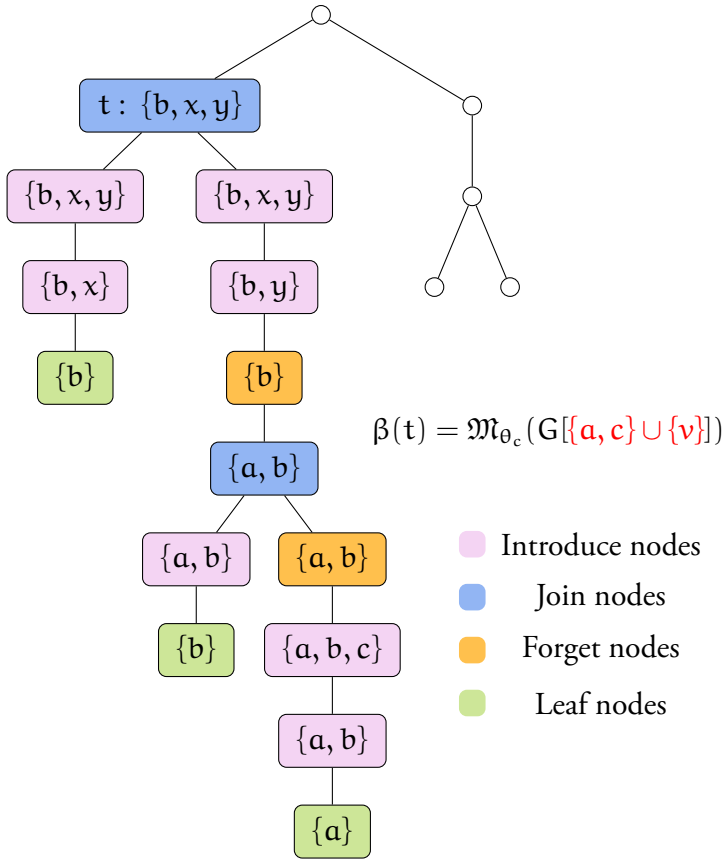
  and in this case as well, the value of $\beta()$ does not decrease from what it was at the child nodes.

We compute $\beta()$ in a bottom-up fashion, starting at the leaves. Notice that the value of $\beta()$ at any node t is in fact the same as the maximum size of a θ_c flower passing through v in the graph $H_t \cup \{v\}$, and thus the computation of $\beta()$ is polynomial time (c.f. Lemma 7.4). Starting from the leaves, let t be the earliest node at which the value of $\beta()$ becomes non-zero. Since $\beta()$ is a non-decreasing function from leaves to root, the node t can be identified unambiguously, except when $\beta(r) = 0$, and t does not exist. In this case, we return the empty set and abort.

Else, let G^* be the graph induced on the vertex set $V(G') \setminus H_t$ and let S be the subset of vertices in the node corresponding to the bag s, where s is the child node of t (in case t is a join node, we let s refer to either of the children). We now recursively solve the problem on G^*, after including S in our solution. Notice that the base case is achieved when the graph $G' \cup \{v\}$ has no minor models of θ_c.

Running Time: Observe that each recursive call provides evidence for at least minor model of θ_c passing through v, vertex-disjoint from those discovered by previous recursive calls. Since the size of the largest θ_c flower passing through v is at most k, the number of recursive calls is easily seen to be at most k. Since we spend only polynomial time in computing an approximate tree decomposition and the $\beta()$ function, the algorithm runs in polynomial time.

Correctness: Our proof is by induction on the depth of recursion. The base case is easily seen to be correct: if the size of the largest θ_c flower passing through v is zero, then there are no minor models to be hit, and the algorithm returns an empty set.



A cross-section of a nice tree-decomposition

Figure 10.2: An example of $\beta()$ at a join node

Let us continue to use the notation used in the description of the algorithm. The inductive hypothesis is that the algorithm correctly returns a hitting set for minor models of θ_c passing through v in the graph G^*. It remains to be shown that all the minor models of θ_c passing through v in the graph $H_t \cup \{v\}$ are hit by S, that is, $(H_t \cup \{v\}) \setminus S$ contains no minor models of θ_c.

This is easily seen by contradiction: indeed, if not, then $H_s \cup \{v\}$ contains minor models of θ_c and $\beta(s)$ is non-zero, contradicting the fact that t is the earliest node at which $\beta()$ assumes a non-zero value.

The Size of the Solution: We have at most k recursive calls. Let $s_1, \ldots, s_k$ denote the bags whose vertices were picked at recursion levels $1, \ldots, k$ respectively. Recall that the treewidth of G', the graph at the topmost level of recursion, was seen to be a constant. The graphs we work with at any deeper levels of recursion are subgraphs of G', and their treewidth is at most the treewidth of G', and therefore a constant. Therefore, we have that

$$|X_{s_i}| = O(1),$$

where we use X_s to denote the vertices in the bag at node s of a tree decomposition. We may hence conclude that the size of the solution obtained is $k \cdot \mathrm{tw}(G') = O(k)$.

□

10.2 Reducing the Maximum Degree of the Graph

In this section, we describe the reduction rules used by the kernelization algorithm. In contrast to the reduction rules employed by most known kernelization algorithms, these rules cannot always be applied on general graphs in polynomial time. Hence the algorithm does *not* proceed by applying these rules exhaustively, as is typical in kernelization programs. We describe how to arrive at situations where these rules can in fact be applied in polynomial time, and prove that even this selective application of rules results in a kernel of size polynomial in the parameter k.

We present a set of reduction rules which, given an input instance (G, k) of Θ_c-DELETION, obtains an equivalent instance (G', k') where $k' \leqslant k$ and the maximum degree of G' is at most a polynomial in k. This part of the kernelization program for Θ_c-DELETION is along the lines of the degree reduction procedures employed for the

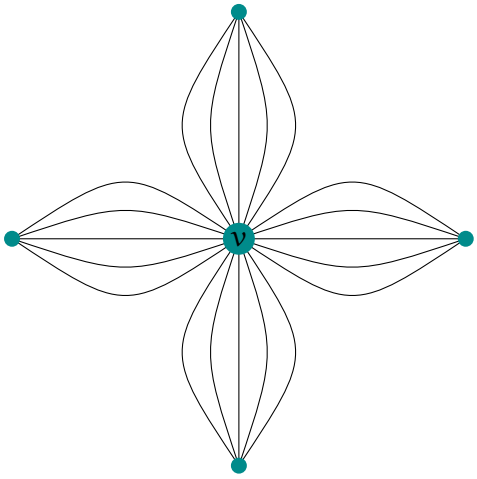


Figure 10.3: An example of a θ_5 flower of size 4. If $k = 3$, then the flower rule would involve deleting the vertex v and reducing k to 2.

Feedback Vertex Set (see [Tho10]) and the Independent Feedback Vertex Set (see Chapter 5) problems.

For this discussion, we say that a vertex v is *irrelevant* if it is not a part of any θ_c minor model, and is *relevant* otherwise. For each rule below, the input instance is (G, k).

Reduction Rule 10.1 (Irrelevant Vertex Rule). *Delete all irrelevant vertices in* G.

Recall that given a graph G and a vertex $v \in V(G)$, an θ_c*-flower of size* ℓ *passing through* v is a set of ℓ different θ_c minor-models in G, each containing v and no two sharing any vertex other than v (see Figure 10.3 for a simplified example).

Reduction Rule 10.2 (Flower Rule). *If a* θ_c*-flower of size greater than* k *passes through a vertex* v *of* G*, then include* v *in the solution and remove it from* G *to obtain the equivalent instance* $(G \setminus \{v\}, (k-1))$.

The argument for the soundness of these reduction rules is straight forward. One can test whether a particular vertex v is part of any minimal minor-model corresponding to θ_c using the rooted minor testing algorithm of Robertson and Seymour [RS95]. It is not clear, however, that we can check whether a vertex is a part of a θ_c flower of size greater than k in polynomial time. We defer the application of the flower rule and apply it only when the problem of finding a θ_c flower of size greater than k can be solved in polynomial time.

Given an instance (G, k) of Θ_c-DELETION, we first apply Lemma 8.1 on (G, k). The polynomial time algorithm described in Lemma 8.1, given a graph G and a positive integer k either reports that G has no θ_c-hitting set of size at most k, or finds a θ_c-hitting set of size at most $O(k \log^{3/2} k)$. If the algorithm reports that G has has no θ_c-hitting set of size at most k, then we return that (G, k) is a NO-instance of Θ_c-DELETION. Else, we assume that we have a hitting set S of size $O(k \log^{3/2} k)$. Now we proceed with a description of the selective application of the flower rule, and the process of obtaining hitting sets H_v for minor models of θ_c that pass through a vertex v. The latter is a non-trivial task since we require that $v \notin H_v$.

To apply the Flower Rule selectively we use S, the θ_c-hitting set. For a vertex $v \in S$ let $S_v := S \setminus \{v\}$ and let $G_v := G \setminus S_v$. By Lemma 7.7, we know that if the treewidth of a graph is more than $(2c-1)$, the the graph contains θ_c as a minor. Since deleting v from G_v makes it θ_c-minor-free, $\mathrm{tw}(G_v) \leqslant (2c-1)+1 = O(1)$. Now by Lemma 7.4, we find in linear time the size of the largest flower centered at v, in G_v. If for any vertex $v \in S$ the size of the flower in G_v is at least $k+1$, we apply the Flower Rule and get an equivalent instance $(G \leftarrow G \setminus \{v\}, k \leftarrow k-1)$. We apply the Flower Rule in such a manner until no longer possible. We abuse notation and continue to use (G, k) to refer to the instance that is reduced with respect to exhaustive application of the Selective Flower Rule. Thus, for every vertex $v \in S$ the size of any flower passing through v in G_v is at most k (see Figure 10.4).

Now we describe how to find, for a given $v \in V(G)$, a hitting set $H_v \subseteq V(G) \setminus \{v\}$ for all minor-models of θ_c that contain v. Since this hitting set is required to *exclude* v, H_v cannot be the trivial hitting set $\{v\}$. If $v \notin S$, then $H_v = S$. On the other hand, suppose $v \in S$. Since the maximum size of a flower containing v in the graph G_v is at most k by Lemma 10.1, we can find a set T_v of size $O(k)$ that does not contain v and hits all the θ_c minor-models passing through v in G_v. Hence in this case we set $H_v = S_v \cup T_v$ (See Figure 10.5.). We denote $|H_v|$ by h_v. Notice that H_v is defined algorithmically, that is, there could be many small hitting sets in $V(G) \setminus \{v\}$ hitting all minor-models containing v, and H_v is one of them.

We now turn to the last and most elaborate reduction rule of this section. This is the reduction rule that involves an application of the q-expansion lemma, with $q = c$.

Given an instance (G, k), S, and a family of sets H_v, we show that if there is a vertex v with degree more than $ch_v + c(c-1)h_v$, then we can reduce its degree to at most $ch_v + c(c-1)h_v$ by repeatedly applying the q-expansion lemma with $q = c$. Observe that

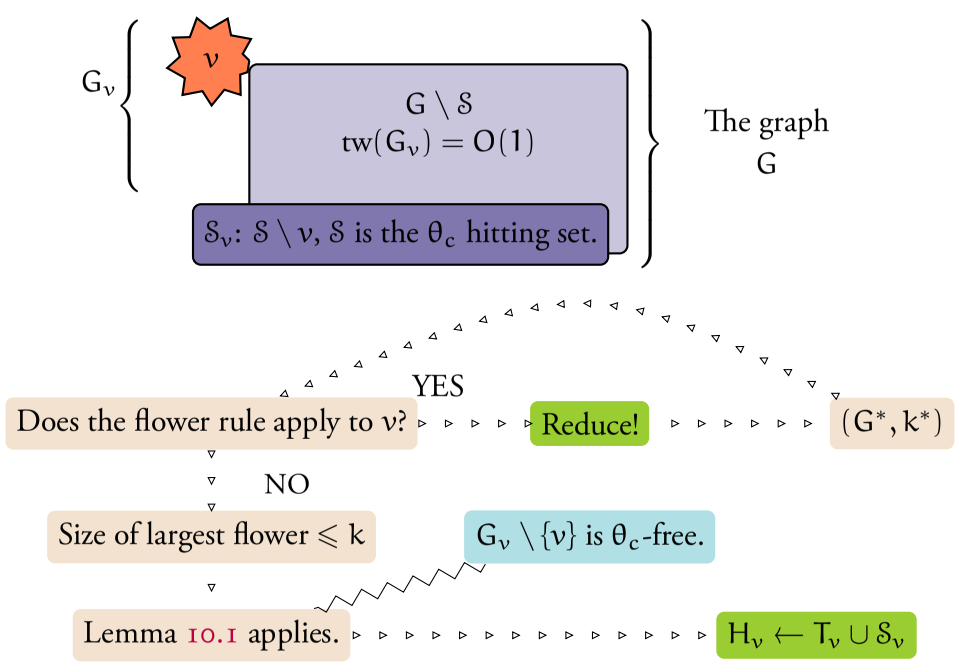


Figure 10.4: The Flower Rule either applies, or it is feasible to find a hitting set H_v of size $k^{O(1)}$ of all minor models passing through v, such that $v \notin H_v$. The hitting sets H_v will be useful for the subsequent reduction rule.

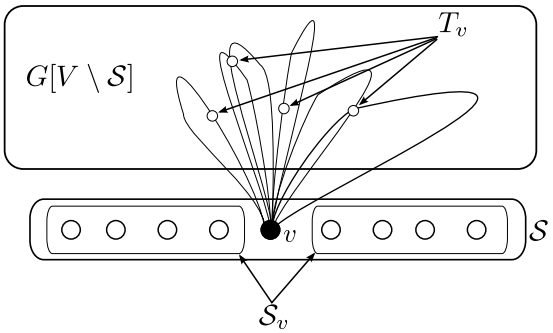


Figure 10.5: The hitting set in Selective Flower Rule

for every vertex v the set H_v is also a θ_c hitting set for G, that is, H_v hits *all* minor-models of θ_c in G.

Let v be a fixed vertex whose degree is more than $ch_v + c(c-1)h_v$. Consider the graph $G \setminus H_v$. Let the components of this graph that contain a neighbor of v be $C_1, C_2, \ldots, C_r$. Note that v cannot have more than $(c-1)$ neighbors into any component, else contracting the component will form a θ_c minor and will contradict the fact that H_v hits all the θ_c minors. Also note that none of the C_i's can contain a minor model of θ_c.

We say that a component C_i is adjacent to H_v if there exists a vertex $u \in C_i$ and $w \in H_v$ such that $(u, w) \in E(G)$. Next we show that vertices in components that are not adjacent to H_v are irrelevant in G. Recall a vertex is irrelevant if there is no minimal minor model of θ_c that contains it. Consider a vertex u in a component C that is not adjacent to H_v. Since $G[V(C) \cup \{v\}]$ does not contain any θ_c minor we have that if u is a part of a minimal minor model $M \subseteq G$, then $v \in M$ and also there exists a vertex $u' \in M$ such that $u' \notin C \cup \{v\}$. Then the removal of v disconnects u from u' in M, a contradiction to Observation 7.2 that for $c \geqslant 2$, any minimal θ_c minor model M of a graph G does not contain a cut vertex. Applying the Irrelevant Vertex Rule to the vertices in all such components leaves us with a new set of components $D_1, D_2, \ldots, D_s$, such that for every i, in D_i, there is at least one vertex that is adjacent to a vertex in H_v.

As before, we continue to use G to refer to the graph obtained after the Irrelevant Vertex Rule has been applied in the context described above. We also update the sets H_v for $v \in V(G)$ by deleting all the vertices w from these sets those have been removed using Irrelevant Vertex Rule.

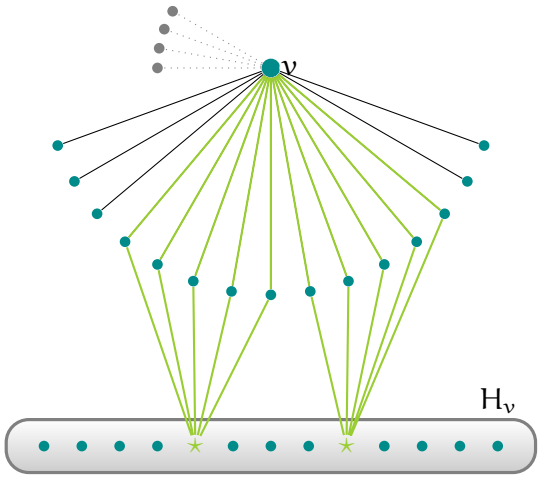


Figure 10.6: A picture of two c-stars, $c = 5$.

Now, consider a bipartite graph $\mathcal{G}$ with vertex bipartitions H_v and D. Here $D = \{d_1, \ldots, d_s\}$ contains a vertex d_i corresponding to each component D_i. We add an edge (v, d_i) if there is a vertex $w \in D_i$ such that $(v, w) \in E(G)$.

Even though we start with a simple graph (graphs without parallel edges) it is possible that after applying reduction rules parallel edges may appear. However, throughout the algorithm, we ensure that the number of parallel edges between any pair of vertices is at most c. Now, v has at most ch_v edges to vertices in H_v. Since v has at most $(c-1)$ edges to each D_i, it follows that if $d(v) > ch_v + c(c-1)h_v$, then the number of components $|D|$ is more than ch_v. Now by applying q-expansion lemma with $q = c$, $A = H_v$, and $B = D$, we find a subset $S \subseteq H_v$ and $T \subseteq D$ such that S has $|S|$ c-stars in T (see Figure 10.6) and $N(T) = S$ (see Figure 10.7).

The reduction rule involves deleting edges of the form (v, u) for all $u \in D_i$, such that $d_i \in T$, and adding c edges between v and w for all $w \in S$. We add these edges only if they were not present before so that the number of egdes between any pair of vertices remains at most c. For example, if v and w are non-adjacent, we add c edges, if v and w have c edges between them, then we add no extra edges.

Reduction Rule 10.3 (The Expansion Rule). *Let v be a vertex whose degree is more than*

$$ch_v + c(c-1)h_v,$$

where $h_v := |H_v|$, and H_v is a θ_c-hitting set for all θ_c-minor models passing through v. Let $\mathcal{G}$ be the bipartite graph with vertex bipartitions H_v and D, where $D = \{d_1, \ldots, d_s\}$

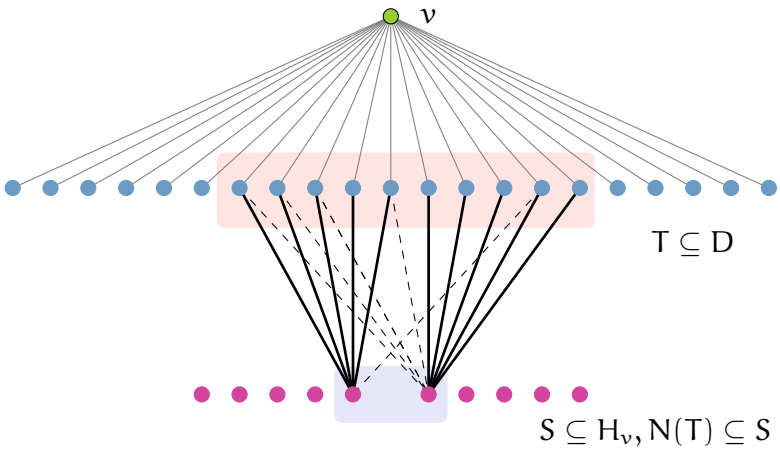


Figure 10.7: A picture of two c-stars, $c = 5$.

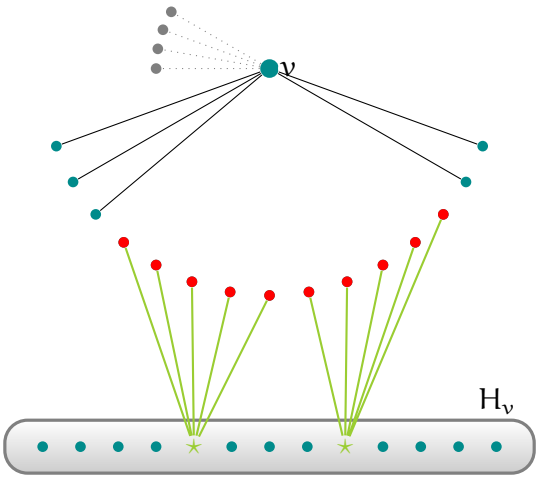


Figure 10.8: The first part of Reduction Rule 10.3. Involves deleting edges incident on v with their other endpoints in T. Vertices labeled $\star$ belong to S and vertices colored red are in T.

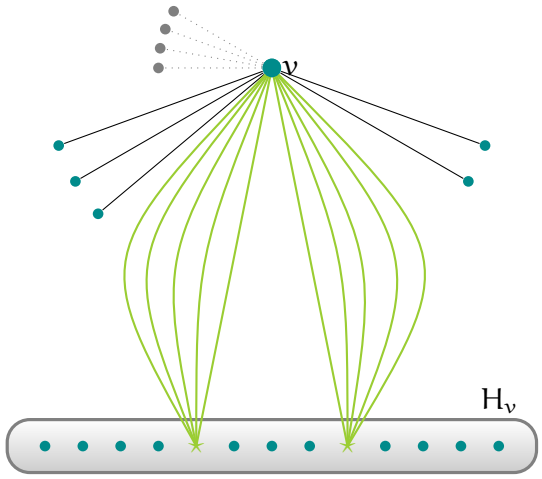


Figure 10.9: Part two of Reduction Rule 10.3. Vertices of T have been omitted for clarity.

contains a vertex d_i *corresponding to each component* D_i *of* $G \setminus H_v$*, and a vertex* $x \in H_v$ *and* $d_i \in D$ *are adjacent if there exists a vertex in the component* D_i *that is adjacent to* x*.*

Apply the q*-expansion lemma with* $q = c$*,* $A = H_v$*, and* $B = D$ *to find subsets* $S \subseteq H_v$ *and* $T \subseteq D$ *such that* S *has* $|S|$ c*-stars in* T *and* $N(T) = S$*.*

- ♢ *Delete all edges of the form* (v, u) *for all* $u \in D_i$*, such that* $d_i \in T$*. (See Figure 10.8.)*
- ♢ *Add edges between* v *and* w *for all* $w \in S$ *in such a way that the total number of edges between* v *and* w *is exactly* c*. That is, and if* v *and* w *have* $(c - r)$ *edges between them, we add* r *extra edges between them. (See Figure 10.9.)*

This completes the description of the q-expansion reduction rule with $q = c$. Let G_R be the graph obtained after applying the reduction rule. The following lemma shows the correctness of the rule.

Lemma 10.2. *Let* G*,* S *and* v *be as above and* G_R *be the graph obtained after applying the* c*-expansion rule. Then* (G, k) *is an yes instance of* Θ_c*-Deletion if and only if* (G_R, k) *is an yes instance of* Θ_c*-Deletion.*

Proof. We first show that if G_R has hitting set Z of size at most k, then the same hitting set Z hits all the minor-models of θ_c in G. Observe that either $v \in Z$ or $S \subseteq Z$. Suppose $v \in Z$, then observe that $G_R \setminus \{v\}$ is the same as $G \setminus \{v\}$. Therefore $Z \setminus \{v\}$,

a hitting set of $G_R \setminus \{v\}$ is also a hitting set of $G \setminus \{v\}$. This shows that Z is a hitting set of size at most k of G. The case when $S \subseteq Z$ is similar.

To prove that a hitting set of size at most k in G implies a hitting set of size at most k in G_R, it suffices to prove that whenever there is a hitting set of size at most k, there also exists a hitting set of size at most k that contains either v or all of S. Consider a hitting set W that does not contain v, and omits at least one vertex from S. Note the $|S|$ c-stars in $\mathcal{G}[\mathcal{S} \cup \mathcal{T}]$, along with v, correspond to minor-models of θ_c centered at v in G, vertex-disjoint except for v. Thus, such a hitting set must pick at least one vertex from each of the components. Let $\mathcal{D}$ be the collection of components D_i such that the (corresponding) vertex $d_i \in T$. Let X denote the set of all vertices of W that appeared in any $D_i \in \mathcal{D}$. Consider the hitting set W' obtained from W by removing X and adding S, that is, $W' := (W \setminus X) \cup S$.

We now argue that W' is also a hitting set of size at most k. Indeed, let S' be the set of vertices in S that do not already belong to W. Clearly, for *every* such vertex that W omitted, W must have had to pick distinct vertices from $\mathcal{D}$ to hit the θ_c minor-models formed by the corresponding c-stars. Formally, there exists a $X' \subseteq X$ such that there is a bijection between S' and X', implying that $|W'| \leqslant |W| \leqslant k$.

Finally, observe that W' must also hit all minor-models of θ_c in G. If not, there exists a minor-model M that contains some vertex $u \in X$. Hence, $u \in D_i$ for some i, and M contains some vertex in $H_v \setminus S$. However, v separates u from $H_v \setminus S$ in $G \setminus S$, contradicting Observation 7.2 that M does not contain a cut vertex. This concludes the proof. □

Observe that all edges that are added during the application of the q-expansion reduction rule have at least one end point in $\mathcal{S}$, and hence $\mathcal{S}$ remains a hitting set of G_R. We are now ready to summarize the algorithm that bounds the degree of the graph (see Algorithm 4).

Now we are ready to prove the lemma which bounds the maximum degree of the instance.

Lemma 10.3. *There exists a polynomial time algorithm that, given an instance* (G, k) *of* Θ_c*-Deletion returns an equivalent instance* (G', k') *such that* $k' \leqslant k$ *and that the maximum degree of* G' *is* $O(k \log^{3/2} k)$.

Proof. Let the instance output by Algorithm 4 be (H, l). By the correctness of reduction rules 10.1,10.2, and 10.3, it is clear that the instance (H, l) is equivalent to

Algorithm 3 BOUND-DEGREE-PRELUDE(G, k)

1: Apply reduction rule 10.1, the Irrelevant Vertex Rule, and let (G^*, k^*) denote the reduced instance.
2: **if** $(G^*, k^*) \equiv (G, k)$ **then**
3: Continue.
4: **else**
5: Return BOUND-DEGREE-PRELUDE(G^*, k^*).
6: **end if**
7: Apply Lemma 8.2. If the output of the algorithm is NO, then return NO and abort. Else, let S denote an approximate hitting set of size $O(k \log^{3/2} k)$.
8: For $v \in S$ determine if v is at the center of a θ_c-flower of size greater than k in the graph $(G \setminus S) \cup \{v\}$, using Lemma 7.4.
9: **if** v is at the center of a θ_c flower of size greater than k **then**
10: Apply reduction rule 10.2 and let (G^*, k^*) denote the reduced instance.
11: Return BOUND-DEGREE-PRELUDE(G^*, k^*).
12: **else**
13: Find T_v using Lemma 10.1.
14: Let $H_v := T_v \cup S_v$, where $S_v := S \setminus \{v\}$.
15: **end if**
16: For $v \notin S$, let $H_v := S$.
17: Return $[G, k, \{H_v \mid v \in V(G)]$.

the input instance (G, k). Further, by Lemma 10.2, the degree of every vertex in H is at most $ch_v + c(c-1)h_v \leqslant O(k \log^{3/2} k)$. It only remains to be shown that Algorithm 4 runs in polynomial time.

For $v \in V(G)$, let $\lambda(v)$ be the number of neighbors of v to which v has fewer than c parallel edges. Observe that the application of q-expansion reduction rule never increases $\lambda(v)$ for any vertex and decreases $\lambda(v)$ for at least one vertex. The other rules delete vertices, which can never increase $\lambda(v)$ for any vertex. Thus with each application of any reduction rule, the value of

$$\left(\sum_{v \in V(G)} \lambda(v) \right) + n$$

decreases by at least one, and since

$$\sum_{v \in V(G)} \lambda(v) \leqslant n^2,$$

Algorithm 4 BOUND-DEGREE(G, k)

1: $X :=$ BOUND-DEGREE-PRELUDE(G, k).
2: Let G, k and H_v for $v \in V(G)$ be as given by X.
3: **for** $v \in V(G)$ **do**
4: **if** $d(v) > ch_v + c(c-1)h_v$ **then**
5: Apply Reduction Rule 10.3, and let (G^*, k^*) denote the reduced instance.
6: Return BOUND-DEGREE(G^*, k^*).
7: **end if**
8: **end for**
9: Return (G, k).

it is clear that the algorithm runs in polynomial time.

This concludes the proof. □

10.3 Protrusion-Based Reductions

At this point, we have shown that any instance of Θ_c-DELETION can be reduced to one where the maximum degree is bounded by $O(k \log^{3/2} k)$.

If Lemma 10.3 returns that (G, k) is a NO-instance to Θ_c-DELETION then we return the same. Else we obtain an equivalent instance (G', k') such that $k' \leqslant k$ and the maximum degree of G' is bounded by $O(k \log^{3/2} k)$. We also have a θ_c-hitting set, S, of G' of size at most $O(k \log^{3/2} k)$ from Lemma 8.2. Let d denote the treewidth of the graph after the removal of S, that is, $d := \mathrm{tw}(G \setminus S)$.

Now, we obtain our kernel in two phases: we first apply the protrusion rule selectively (Lemma 6.3) and get a polynomial kernel. Then, we apply the protrusion rule exhaustively on the obtained kernel to get a smaller kernel. To obtain the kernel we follow the following steps.

Applying the Protrusion Rule. By Lemma 7.7, the treewidth of a graph that does not contain θ_c as a minor is bounded by $(2c-1)$. Hence $d \leqslant 2c-1$. Now we apply Lemma 7.3 and get a $2(d+1)$-protrusion Y of G' of size at least $\frac{|V(G')|-|S|}{4|N(S)|+1}$. By Lemma 9.2, Θ_c-DELETION has finite integer index. Let $\gamma : \mathbb{N} \to \mathbb{N}$ be the function defined in Lemma 6.3. Hence if $\frac{|V(G')|-|S|}{4|N(S)|+1} \geqslant \gamma(2d+1)$ then using Lemma 6.3 we replace the $2(d+1)$-protrusion Y of G' and obtain an instance G^* such that

$|V(G^*)| < |V(G')|$, $k^* \leqslant k'$, and (G^*, k^*) is a YES-instance of Θ_c-Deletion if and only if (G', k') is a YES-instance of Θ_c-Deletion .

Before applying the Protrusion Rule again, if necessary, we bound the maximum degree of the graph by reapplying Lemma 10.3. This is done because the application of the protrusion rule could potentially increase the maximum degree of the graph. We alternately apply the protrusion rule and Lemma 10.3 in this fashion, until either Lemma 10.3 returns that G is a NO instance, or the protrusion rule ceases to apply. Observe that this process will always terminate as the procedure that bounds the maximum degree never increases the number of vertices and the protrusion rule always reduces the number of vertices.

Let (G^*, k^*) be a reduced instance with hitting set S. In other words, there is no $(2d+2)$-protrusion of size $\gamma(2d+2)$ in $G^* \setminus S$, and the protrusion rule no longer applies. Now we show that the number of vertices and edges of this graph is bounded by $O(k^2 \log^3 k)$. We first bound the number of vertices. Since we cannot apply the Protrusion Rule, $\frac{|V(G^*)|-|S|}{4|N(S)|+1} \leqslant \gamma(2d+2)$. Since $k^* \leqslant k$ this implies that

$$\begin{aligned}
|V(G^*)| &\leqslant \gamma(2d+2)(4|N(S)|+1)+|S| \\
&\leqslant \gamma(2d+2)(4|S|\Delta(G^*)+1)+|S| \\
&\leqslant \gamma(2d+2)(O(k\log^{3/2} k) \times O(k\log^{3/2} k)+1)+O(k\log^{3/2} k) \\
&\leqslant O(k^2 \log^3 k).
\end{aligned}$$

To get the desired bound on the number of edges we first observe that since $tw(G^* \setminus S) \leqslant (2c-1) = d$, we have that the number of edges in $G^* \setminus S \leqslant d|V(G^*) \setminus S| = O(k^2 \log^3 k)$. Also the number of edges incident on the vertices in S is at most $|S| \cdot \Delta(G^*) \leqslant O(k^2(\log k)^3)$. This gives us a polynomial time algorithm that returns a kernel of size $O(k^2 \log^3 k)$.

Now we give a kernel of smaller size. To do so we apply combination of rules to bound the degree and the protrusion rule as before. The only difference is that we would like to replace any large $(2d+2)$-protrusion in graph by a smaller one. We find a $2d+2$-protrusion Y of size at least $\gamma(2d+2)$ by guessing the boundary $\partial(Y)$ of size at most $2d+2$. This could be performed in time $k^{O(d)}$. So let (G^*, k^*) be the reduced instance on which we can not apply the Protrusion Rule. Then we know that $\Delta(G^*) = O(k\log^{3/2} k)$. If G is a YES-instance then there exists a θ_c-hitting set S of size at most k such that $tw(G \setminus S) \leqslant (2c-1) = d$. Now applying the analysis above with this S yields that $|V(G^*)| = O(k^2 \log^{3/2} k)$ and $|E(G^*)| \leqslant O(k^2 \log^{3/2} k)$. Hence if the number of vertices or edges in the reduced instance G^*, to which we can

not apply the Protrusion Rule, is more than $O(k^2 \log^{3/2} k)$ then we return that G is a NO-instance. With this, we have a proof of Theorem 10.1.

Theorem 10.1 has following immediate corollary.

Corollary 10.2. *The* Vertex Cover , Feedback Vertex Set *and* Diamond Hitting Set *problems, when parameterized by solution size, admit kernels of size* $O(k^2 \log^{3/2} k)$.

We note, however, that Vertex Cover and Feedback Vertex Set are among the most deeply studied problems from the point of view of kernelization, and admit smaller kernels — specifically, Vertex Cover (see [Nie06]) is known to have a kernel on $2k$ vertices and Feedback Vertex Set is known to have a kernel on $O(k^2)$ vertices and edges (see [Tho10]).

The greatest challenge to any thinker
is stating the problem in a way that will allow a solution.

Bertrand Russel

Wait a minute, wait a minute.
You ain't heard nothin' yet!

The Jazz Singer

In this chapter we introduce the disjoint version of the Planar $\mathcal{F}$-Deletion problem, and show that it has a polynomial kernel on general graphs. The disjoint version of the problem, Disjoint Planar $\mathcal{F}$-deletion, is similar to Planar $\mathcal{F}$-Deletion, except for the following:

- a subset S of $(k+1)$ vertices in the input graph G are declared *forbidden*,
- $G \setminus S$ and $G[S]$ contain no minor models of any graph in $\mathcal{F}$, and
- we seek a $\mathcal{F}$-hitting set of size at most k that does not contain any of the forbidden vertices.

This style of reformulation is actually common in the literature because of a technique used to obtain faster FPT algorithms called *iterative compression*. The technique involves starting with a sub-optimal solution and improving it iteratively, and the "disjoint" version is typically what is encountered at every iteration of this technique. Iterative compression has been a vastly successful technique for achieving fast FPT algorithms. It was used for improving the running time of the Feedback Vertex Set problem, and the intermediate problem encountered therein was called the Forest Bipartition problem (see [CFL+08, CCL10]). The input to Forest Bipartition is a graph G and a partition of $V(G)$ into two parts A and B such that $G[A]$ and $G[B]$ are forests. The goal is to find $S \subseteq B$, of size at most k, such that $G \setminus S$ is a forest. The Disjoint Planar $\mathcal{F}$-deletion can be thought of as a considerably generalized version of this problem, and as we will see later in this chapter, we are able to use the kernelization algorithm for the disjoint version as a subroutine in an iterative compression algorithm to obtain a faster FPT algorithm for Planar $\mathcal{F}$-Deletion.

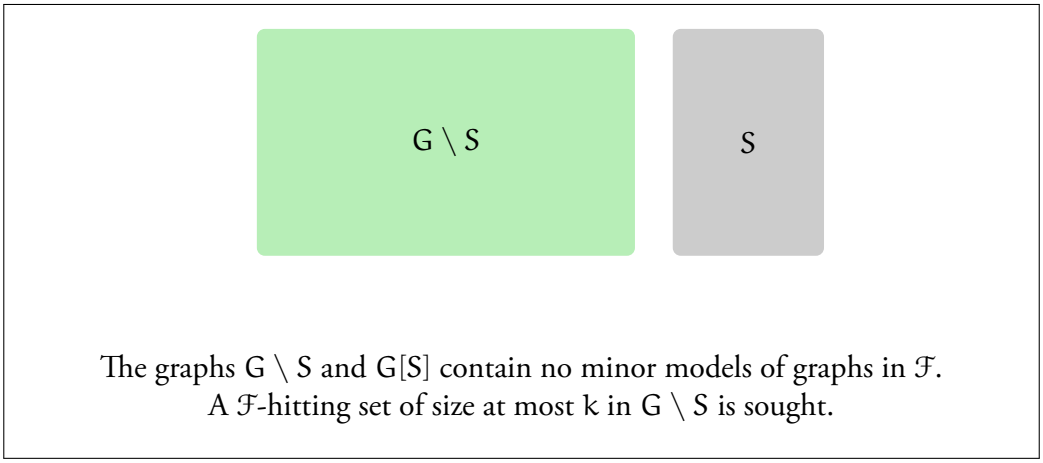


Figure 11.1: The disjoint version of Planar $\mathcal{F}$ Deletion.

Formally, the problem is the following:

Disjoint Planar $\mathcal{F}$-deletion

Input:	A graph G, a family of graphs $\mathcal{F}$, a positive integer k and a $\mathcal{F}$-hitting set S of size $(k+1)$.
Parameter:	k
Question:	Does there exist a $\mathcal{F}$-hitting set X such that $\lvert X\rvert \leqslant k$, and $S \cap S = \emptyset$?

In this chapter, we demonstrate a polynomial kernel for Disjoint Planar $\mathcal{F}$-deletion.

Theorem 11.1. *The* Disjoint Planar $\mathcal{F}$-deletion *problem has an* $O(k^3)$ *annotated kernel that does not increase the parameter.* Disjoint Planar $\mathcal{F}$-deletion *also has a polynomial kernel.*

Theorem 11.1 has an important corollary:

Corollary 11.2. *There exists an algorithm for Planar $\mathcal{F}$-Deletion that runs in time* $2^{O(k \log k)} n^2$.

The road map for demonstrating a polynomial kernel for Disjoint Planar $\mathcal{F}$-deletion is similar to our previous kernelization algorithms: we observe the protrusion reduction rule applies to the problem, and demonstrate that any instance of large enough size is either a NO instance or contains a protrusion that can be reduced.

So far, our strategy for demonstrating protrusions on sufficiently large YES instances had been to exploit an approximate hitting set and the structure of the rest of the graph. In this chapter, however, we use the decomposition lemma introduced in Chapter 7. This lemma is applied to an instance of Disjoint Planar $\mathcal{F}$-deletion to find protrusions in large instances, and the protrusions are reduced thereafter. The application involves exploiting the situation of the problem, namely that $G[B]$ and $G \setminus B$ do not contain some constant-sized planar graph as a minor. It turns out that we can think of an instance of Disjoint Planar $\mathcal{F}$-deletion as the incidence graph of a hypergraph, where the incidence graph excludes the complete graph on a constant number of vertices as a minor. The number of edges in such graphs is subject to known bounds, and these results are also stated and used in this chapter.

However, finding protrusions in the case of Disjoint Planar $\mathcal{F}$-deletion is not enough, because we are unable to demonstrate that the problem has finite integer index. However, we are able to show that an important equivalence relation, different from the canonical one, but devised to suit the Disjoint Planar $\mathcal{F}$-deletion problem, has finite index, and this leads to a more careful protrusion replacement technique that works to give us a polynomial kernel. The application of the decomposition lemma, and the workarounds to handle the fact that Disjoint Planar $\mathcal{F}$-deletion is not suitable for protrusion-based reductions right away, form the essence of this chapter.

11.1 Combinatorial Tools

We describe some results about hypergraphs that will be subsequently utilized in the analysis of kernelization.

11.1.1 Hypergraph Lemmata

A *hypergraph* $\mathcal{H}$ consists of a vertex set $V(\mathcal{H})$, and a hyperedge set $E(\mathcal{H})$ of subsets of $V(\mathcal{H})$. A hypergraph is *simple* if it has no multiple hyperedges and all its hyperedges have arity at least 2. The *incidence graph* of a hypergraph $\mathcal{H}$ is the bipartite graph $I(\mathcal{H})$ on the vertex set $V(\mathcal{H}) \cup E(\mathcal{H})$ such that $v \in V(\mathcal{H})$ is adjacent to $e \in E(\mathcal{H})$ in $I(\mathcal{H})$ if and only if $v \in e$. (See Figure 11.2 for an example.)

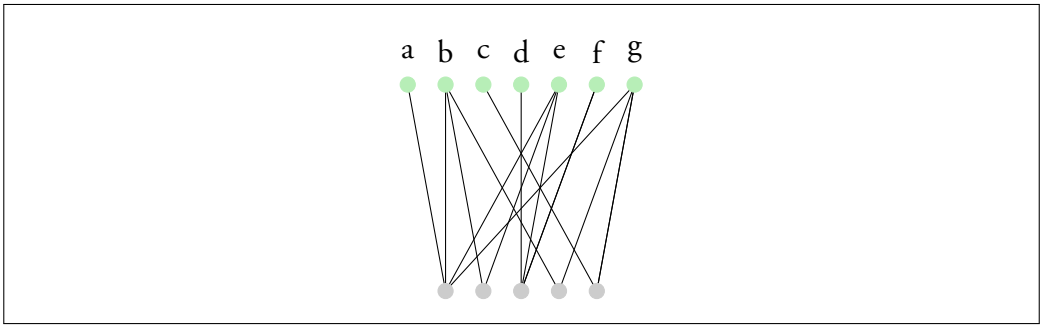


Figure 11.2: The incidence graph of a hypergraph on the vertex set $\{a, b, c, d, e, f, g\}$ with edges $\{a, b, e, g\}$,$\{b, e\}$,$\{d, e, f\}$,$\{b, g\}$ and $\{c, g\}$.

Kostochka [Kos82] show that there exists a constant s_h such that every graph with no K_h minor has average degree at most s_h. Later Thomason [Tho01] refined this result by proving that there is a constant s_h such that every n-vertex graph with no K_h - minor has at most $(s_h\sqrt{\log h})n$ edges. Moreover, this result is tight up to the value of $s_h = 0.319\ldots + o(1)$. The following lemma is due to Norine et al. [NSTW06]

Lemma 11.1 ([NSTW06]). *Let* G *be a graph that does not contain* K_h *as a minor. Then* G *has at most* $2^{s_h\sqrt{\log h}}n$ *cliques.*

The following lemma is proven in [FiOT10]. We provide its proof here for completeness.

Lemma 11.2 ([FmOT09]). *Let* $\mathcal{H}$ *be a simple* n*-vertex hypergraph such that its incidence graph* $I(\mathcal{H})$ *does not contain* K_h *as a minor. Then* $|E(\mathcal{H})| \leqslant 2^{s_h\sqrt{\log h}}n$.

Proof. Targeting towards a contradiction, let us assume that the lemma does not hold. Let $\mathcal{H}$ be the counterexample such that its incidence graph has minimum number of edges. Then for every hyperedge e of $\mathcal{H}$, every proper subset of e of size at least 2 also should be a hyperedge of $\mathcal{H}$. Indeed, if $e' \subset e$ is not a hyperedge of $\mathcal{H}$, then the incidence graph of the hypergraph obtained from $\mathcal{H}$ by replacing e by e' is proper subgraph of $I(\mathcal{H})$, which contradicts its minimality. Therefore, for every hyperedge e of $\mathcal{H}$ all its subsets of size 2 are also hyperedges of $\mathcal{H}$.

Let us construct a graph G. The vertex set of G is $V(\mathcal{H})$ and two vertices of G are adjacent if and only if there is a hyperedge in $\mathcal{H}$ of size two, containing these vertices. Then every clique of G corresponds to some hyperedge of $\mathcal{H}$, and G is a minor of $I(\mathcal{H})$.

By Lemma 11.1, the number of cliques in G, and thus the number of hyperedges in $I(\mathcal{H})$ is at most $2^{s_h\sqrt{\log h}}n$. □

Lemma 11.3. *Let $\mathcal{H}$ be an n-vertex hypergraph (not necessarily simple) such that its incidence graph $I(\mathcal{H})$ does not contain K_h as a minor. Then the number of hyperedges of $\mathcal{H}$ of size at least h is at most $2^{s_h\sqrt{\log h}}h(h-1)n/2$.*

Proof. Every hyperedge of size at least h cannot have multiplicity more than $h(h-1)/2$ because otherwise $I(\mathcal{H})$ would contain K_h as a minor. Then the proof follows from Lemma 11.2. □

In the next section we will use a (s,p)-dissolution, a notion introduced in Chapter 7. We recall the definition and the lemma associated with it here for convenience (see Chapter 7 for a detailed proof):

Definition. *Let G be a graph of treewidth b. An (s,p)-*dissolution *of G is defined to be a partition of the vertex set of G into $(p+1)$ parts $V_1,\ldots,V_p$ and U such that:*

- ♢ *Among the first p partitions, there are no edges with endpoints in different partitions, that is, for all $1 \leqslant r \neq s \leqslant p$, if $u \in V_r$ and $v \in V_s$, then $(u,v) \notin E(G)$.*
- ♢ *Each of the first p parts has size at least s, that is, $|V_i| \geqslant s$, for every i, $1 \leqslant i \leqslant p$.*
- ♢ *The neighborhood of V_i in U is at most $(2b+2)$:*

$$|N(V_i) \cap U| \leqslant 2b+2,$$

for every i, $1 \leqslant i \leqslant p$.

Lemma. *Let G be a graph of treewidth b. There is an integer constant d such that if G has at least $(d \cdot bsp)$ vertices (for some integers s and p), G admits a (s,p)-dissolution.*

11.2 Detecting Protrusions

In this section, we demonstrate how we infer the presence of a protrusion from the fact that we have "large" input instance. We will apply Lemma 7.1 for graphs of treewidth b (see Chapter 7 or the previous section). We have that there is an integer constant d such that if G has at least $(d \cdot bsp)$ vertices (for some integers s and p), G admits a (s,p)-dissolution. We will now exploit the structure of the (s,p)-dissolution to infer the existence of protrusions. To obtain a suitable protrusion, that is, one that has a

large enough number of vertices, we will also require s and p to be sufficiently large. The following lemma shows that there are we can achieve this with choices of s and p which are polynomial in k. This amounts to saying that as long as the size of the input instance is more than $k^{O(1)}$, we are able to detect a protrusion that will be appropriate for further reduction.

Lemma 11.4. *Let* (G, S, k) *be an instance of* Disjoint Planar $\mathcal{F}$-deletion. *For any* c*, there exist constants* α *and* β *(that depend only on* $\mathcal{F}$*) such that if the total number of vertices in* G *is more than* $(\alpha(k+1)+k)\cdot c\cdot(\beta(k+1)+k)$*, then, there exists a* r*-protrusion on at least* c *vertices, where* r *is bounded by a constant that depends only on the size of some planar graph in* $\mathcal{F}$*.*

Proof. We begin by applying Lemma 7.6 on $G \setminus S$ and concluding that $G \setminus S$ is a graph of constant treewidth, say b. Thus we may apply Lemma 7.1 to obtain a (s,p)-dissolution of $G \setminus S$, for values of s and p that will be determined in the course of this proof. For the rest of this discussion, we use h to denote the smallest planar graph in $\mathcal{F}$ (among those planar graphs of $\mathcal{F}$ that have the smallest size, H is chosen arbitrarily). Let $(V_1, \ldots, V_p, U)$ denote the (s,p)-dissolution of $G \setminus S$.

Given this setup, we first show that there exists a constant α such that if the number of parts p in the (s,p)-dissolution of $(G \setminus S)$ is more than $(\alpha\cdot(k+1)+k)$, then there exists a part V_i such that every connected component of V_i has at most h neighbors in S.

We let $C_1, \ldots, C_\gamma$ be the connected components of $G[V \setminus (U \cup S)]$. Consider the hypergraph $\mathcal{H}$ with the vertex set S: for every C_i, $1 \leqslant i \leqslant \gamma$, we define the hyperedge $e_i = N(C_i) \cap S$.

Observe that the incidence graph $I(\mathcal{H})$ is a minor of $G \setminus U$ — indeed, it is exactly the graph obtained by contracting the vertices of C_i into a single vertex in the graph $G \setminus U$. Now, if $G \setminus U$ were to be H-minor-free, then $I(\mathcal{H})$ would clearly be H-minor-free as well, and the hypotheses of Lemma 11.3 would hold. While we do not know in advance that $G \setminus U$ is H-minor-free, we do know that if the graph G is a YES instance of Disjoint Planar $\mathcal{F}$-deletion, then there exists $S^* \subseteq (V \setminus S)$, of size at most k, such that $G \setminus S^*$ does not contain H as a minor.

Therefore, we have that there is a subgraph $\mathcal{H}^*$ such that $I(\mathcal{H}^*)$ is indeed H-minor-free. We also know that $\mathcal{H}^*$ has at most k edges fewer than $\mathcal{H}$. Also, we may apply Lemma 11.3 to $I(\mathcal{H}^*)$. Recall that $\mathcal{H}^*$ is a hypergraph on $(k+1)$ vertices, therefore,

by Lemma11.3, the number of edges of size at least h, counting multiplicity, in $\mathcal{H}^*$ is at most

$$\frac{2^{s_h\sqrt{\log h}}h(h-1)}{2} \cdot (k+1).$$

We let α denote $2^{s_h\sqrt{\log h}}h(h-1)/2$.

Recall that we wish to arrive at the existence of a part V_i that does not contain any component with neighborhood larger than h in S. First, note that every component corresponds to an edge in $\mathcal{H}$. For components that have more than h neighbors in S, two cases arise: either the component corresponds to an edge of size at least h in $\mathcal{H}^*$, or it corresponds to an edge of $\mathcal{H}$ that does not exist in $\mathcal{H}^*$. What we have shown so far amounts to the following:

> There are at most $(\alpha \cdot (k+1))$ components of the first kind,
> and at most k components of the second.

Therefore, if p, the number of parts, is more than $(\alpha\cdot(k+1)+k)$, then there necessarily exists a part where no component has more than h neighbors in S.

Having found a part where all components have a constant-sized neighborhood in S, we are only one step away from deducing a protrusion: we need either one of these components to have enough vertices to qualify as a protrusion that can be subject to a reduction, or we need a collection of components whose size is large enough in the aggregate, and are such that all of them have the same neighborhood in S. We will now show that there exists a constant β such that if the size of each part s in the (s,p)-dissolution of $G \setminus S$ is at least $\beta \cdot k$, then there exists a set of components that have the desired property.

Recall that $\mathcal{H}^*$ is H-minor-free, and by Lemma 11.2, we have that the number of simple hyperedges (without counting multiplicity) in a $(k+1)$ vertex hypergraph whose incidence graph is H-minor-free is at most

$$2^{s_h\sqrt{\log h}} \cdot (k+1).$$

Recall also that $\mathcal{H}$ had at most k edges more than $\mathcal{H}^*$, and therefore, the number of edges in $\mathcal{H}$ is at most

$$2^{s_h\sqrt{\log h}} \cdot (k+1) + k.$$

We use β to denote $2^{s_h\sqrt{\log h}}$.

Now, let e be a fixed hyperedge of size at most h in $\mathcal{H}$. Let $c[e]$ denote the number of of components C such that $N(C) = e$, and let $w(c[e])$ denote the total number of vertices in all the components C such that $N(C) = e$. Notice that since the total number of distinct hyperedges is at most $\beta(k+1)+k$, and we have that:

$$\sum_{i=1}^{\beta(k+1)+k} w(c[e_i]) = s.$$

Therefore, if $s > (\beta(k+1)+k)c$, then there exists a collection of components in which the total number of vertices exceeds c, and whose neighborhood is exactly e (for some e). This completes the argument.

□

11.3 Replacing Protrusions

In this section we describe an algorithm that takes a protrusion of large enough size and replaces it with a smaller one. We first note that the results that are known about protrusion replacement cannot be used directly for Disjoint Planar $\mathcal{F}$-deletion. Here, we make a problem-specific adaptation of the results in [BFL+09] that is suitable in our context. We begin with some definitions. The first notion that we introduce is that of a *label-preserving isomorphism*.

Definition 11.1. *Let G_1 and G_2 be two graphs, and let t be a fixed positive integer. For $i \in \{1,2\}$, let f_{G_i} be a function that associates with every vertex of $V(G_i)$ some subset of $[t]$. The image of a vertex $v \in G_i$ under f_{G_i} is called the label of that vertex. We say that that G_1 is* label-wise isomorphic *to G_2, and denote it by $G_1 \cong_t G_2$, if there is an map $h : V(G_1) \to V(G_2)$ such that (a) h is one to one and onto; (b) $(u,v) \in E(G_1)$ if and only if $(h(u),h(v)) \in E(G_2)$ and (c) $f_{G_1}(v) = f_{G_2}(h(v))$. We call h a* label-preserving isomorphism.

Notice that the first two conditions of Definition 11.1 simply indicate that G_1 and G_2 are isomorphic. Now, let G be a t-boundaried graph, that is, G has t distinguished vertices, uniquely labeled from 1 to t. Given a t-boundaried graph G, we define a canonical labeling function $\mu_G : V(G) \to 2^{[t]}$. The function μ_G maps every distinguished vertex v with label $\ell \in [t]$ to the set $\{\ell\}$, that is $\mu_G(v) = \{\ell\}$, and for all vertices $v \in (V(G) \setminus \partial(G))$ we have that $\mu_G(v) = \emptyset$.

Next we define a notion of labeled edge contraction. Let H be a graph together with a function $f_H : V(H) \to 2^{[t]}$ and $(u, v) \in E(H)$. Furthermore, let H' be the graph obtained from H by identifying the vertices u and v into w_{uv}, removing all the parallel edges and removing all the loops. Then by *labeled edge contraction* of an edge (u, v) of a graph H, we mean obtaining a graph H' with the label function $f_{H'} : V(H') \to 2^{[t]}$. For $x \in V(H') \cap V(H)$ we have that $f_{H'}(x) = f_H(x)$ and for w_{uv} we define $f_{H'}(w_{uv}) = f_H(u) \cup f_H(v)$. Now we introduce a notion of labeled minors of a t-boundaried graph.

Definition 11.2. *Let H be a graph together with a function $f : V(H) \to 2^{[t]}$ and G be a t-boundaried graph with canonical labeling function μ_G. A graph H is called a labeled minor of G, if we can obtain a labeled isomorphic copy of H from G by performing edge deletion and labeled edge contraction.*

Remark 11.1. *We note that the notion of a label-preserving isomorphism for graphs depends only on the labeling function, and is oblivious to the boundary. In particular, if G and H are two labeled t-boundaried graphs that are label-wise isomorphic, a label preserving isomorphism is not required to necessarily map the boundary vertices of G to boundary vertices of H.*

Finally, we define the notion of h-*folios* and equivalence on t-boundaried graphs.

Definition 11.3. *A h-folio of a t-boundaried graph G is the set $\mathcal{M}_h(G)$ of all t-labeled minors of G on at most h vertices.*

Definition 11.4. *Let t and h be two fixed integers and G_1 and G_2 be two t-boundaried graphs. We say that $G_1 \equiv_t^h G_2$ if $\mathcal{M}_h(G_1) = \mathcal{M}_h(G_2)$.*

The number of t-labeled graphs on at most h vertices is a function of t and h, so the number of possible sets $\mathcal{M}_h(G)$ is as well. This yields the following observation

Observation 11.3. *For fixed integers h and t, $\equiv_t^h$ is an equivalence relation with finite index on t boundaried graphs.*

Proof. Clearly, the number of equivalence classes of $\equiv_t^h$ is exactly equal to the number of distinct h-folios that exist for the universe of t-boundaried graphs. A h-folio is a collection of labeled graphs on at most h vertices. The total number of labeled graphs on at most h vertices is $2^{\binom{h}{2}} \cdot (2^t)^h$. The number of possible h-folios is therefore $2^{2^{\binom{h}{2}+th}}$, and this establishes that $\equiv_t^h$ has finite index. □

Definition 11.5. *For a fixed family $\mathcal{F}$ of finite graphs and two t-boundaried graphs G_1 and G_2, we say that $G_1 \equiv_{\mathcal{F}} G_2$ if for all t-boundaried graphs G_3 and for all $Z \subseteq V(G_3)$,*

$$(G_1 \oplus G_3 \setminus Z) \text{ is } \mathcal{F} \text{ free if and only if } (G_2 \oplus G_3 \setminus Z) \text{ is } \mathcal{F} \text{ free}.$$

The main lemma of this section is the following.

Lemma 11.5. *Let $\mathcal{F}$ be a finite family of graphs, and let $q = \max_{H \in \mathcal{F}}\{|V(H)|\}$. Further, let t be a fixed positive integer, and let $l = (q + t)$. Then the relation $\equiv_t^l$ refines $\equiv_{\mathcal{F}}$. That is, if $G_1 \equiv_t^l G_2$ then $G_1 \equiv_{\mathcal{F}} G_2$. Also, for every fixed t, $\equiv_{\mathcal{F}}$ has finite index.*

Proof. Suppose $G_1 \equiv_t^l G_2$, and, for the sake of contradiction, $G_1 \not\equiv_{\mathcal{F}} G_2$. Then there exists a t-boundaried graph G_3 and some subset $Z \subseteq G_3$ is such that:

- either $(G_1 \oplus G_3 \setminus Z)$ is $\mathcal{F}$ free and $(G_2 \oplus G_3 \setminus Z)$ is not $\mathcal{F}$ free
- or $(G_1 \oplus G_3 \setminus Z)$ is not $\mathcal{F}$ free and $(G_2 \oplus G_3 \setminus Z)$ is $\mathcal{F}$ free

We consider the first situation. We assume that the graphs G_1, G_2 and G_3 have the canonical labeling associated with them. Since $(G_2 \oplus G_3 \setminus Z)$ is not $\mathcal{F}$ free , there exists a minor model of some graph $H \in \mathcal{F}$ in $(G_2 \oplus G_3 \setminus Z)$. We use $M_H^{(2,3)}$ to refer to a witness minor model. By Proposition 2.1 we have that there is a map $\phi : V(H) \to 2^{V(G')}$ such that for every vertex $v \in V(H)$, $G'[\phi(v)]$ is connected, for every pair of vertices $v, u \in V(H)$, $\phi(u) \cap \phi(v) = \emptyset$, and for every edge $uv \in E(H)$, there is an edge $u'v' \in E(G')$ such that $u' \in \phi(u)$ and $v' \in \phi(v)$. Let $P_1, P_2, \ldots, P_h$ be the these pairwise disjoint connected components of $M_H^{(2,3)}$, and let Q_i denote $P_i \cap V(G_2)$. Note that the Q_i corresponding to parts P_i that involve vertices from $\partial(G_2)$ may not induce connected subgraphs of G_2, but instead be composed of smaller components. Since there are at most t vertices on the boundary, the total number of components in the subgraph given by $\cup_{i=1}^h Q_i$ is at most $(h + t)$.

Now, we consider the labeled graph H^* obtained by contracting (using labeled edge contraction operations) the connected components of $\cup_{i=1}^h Q_i$ in G_2. Clearly, H^* belongs to the l-folio of G_2, and by our assumption on the equivalence of G_1 and G_2, H^* also belongs to the l-folio of G_1. This implies that G_1 contains a minor model of H^* that preserves labels. Specifically, there is a collection of connected components $R_1, \ldots, R_x$ that is, for every component of X of $\cup_{i=1}^h Q_i$ that contain any vertices

from $\partial(G_2)$, there exists a component R_j in G_1 that contains exactly the same set of boundary vertices.

Finally, we observe that the components $R_1, \ldots, R_x$ can be extended to a minor model $M_H^{(1,3)}$ of H in $G_1 \oplus G_3 \setminus Z$. Indeed, consider Y_i given by $P_i \cap V(G_3)$. To begin with, $M_H^{(1,3)}$ consists of all components Y_i and R_j that do not contain any vertices from $\partial(G_3)$. Now, consider $\{P_i \mid P_i \cap \partial(G_3) \neq \emptyset\}$, that is, the sub collection of those components of $M_H^{(2,3)}$ that have a non-trivial intersection with the boundary. For every such component X, let X_B denote the intersection of X with $\partial(G_3)$. Note that the elements of X_B also appear in some components of $R_1, \ldots, R_x$ (since the minor model was designed to preserve labels). Let $i_1^X, \ldots, i_m^X$ be such that

$$\cup_{j=1}^{m} R_{i_j^X} \cap \partial(G_3) = X_B.$$

We let M_X denote the subgraph induced on the vertices $\cup_{j=1}^{m} R_{i_j^X}$ with $X \cap V(G_3)$. Now, we show that M_X is a connected component of $G_1 \oplus G_3 \setminus Z$. We do this by showing that between any pair of vertices in X_B, there exists a path in M_X. Notice that this suffices, since every vertex in $M_X \setminus X_B$ has a path to some vertex in X_B.

To begin with, recall that X is a connected component in $G_2 \oplus G_3 \setminus Z$. Now, consider the subgraph $(X \cap V(G_2))$, and observe that it is a collection of connected components $T_1, \ldots, T_m$ that can be placed in one-one correspondence with the components $\{R_{i_j^X}\}_{i=1}^{m}$. It is also clear that each T_i contains at least one vertex from $\partial(G_2)$, and we let t_i be an arbitrarily chosen representative from $T_i \cap \partial(G_2)$. Since X is connected in $G_2 \oplus G_3 \setminus Z$, there is a path P_{ij} between t_i and t_j for all $1 \leqslant i \neq j \leqslant m$. The path P_{ij} comprises of subpaths that lie in vertices of $(X \cap V(G_3))$ and the components $T_1, \ldots, T_m$. Notice that since the T_x induces a connected component in G_2, there are no edges between vertices of T_i and T_j in G_2, and by the definition of gluing, there are no edges from vertices in $(V(G_3) \cap X) \setminus \partial(G_3)$ to any T_x. Thus, the path P_{ij} can only enter and exit a component at boundary vertices. For every subpath of P_{ij} that lies in entirely a component T_x let a and b denote the boundary vertices at which the subpath begins and ends (note that a and b need not be distinct). Now, let $R_{i_x^X}$ be the component corresponding to T_x in $(G_1 \oplus G_3 \setminus Z)$. Notice that $\{a, b\} \subseteq R_{i_x^X}$, and since $R_{i_x^X}$ is connected, there is a corresponding subpath connecting the vertices a and b in $(G_1 \oplus G_3 \setminus Z)$. Stitching these subpaths gives us a path Q_{ij} between vertices t_i and t_j in $G_1 \oplus G_3 \setminus Z$, and this shows that M_X indeed induces a connected component.

Now, for every X, we add M_X to $M_H^{(1,3)}$. It is easy to check that the $M_H^{(1,3)}$ that we have constructed is a minor model of H in $G_1 \oplus G_3 \setminus Z$. This contradicts the

assumption that $G_1 \oplus G_3 \setminus Z$ is $\mathcal{F}$-free. The argument for the second case is symmetric. Also, note that since $\equiv_{\mathcal{F}}$ is refined by $\equiv^h_t$, an equivalence relation with finite index (recall Observation 11.3), clearly, $\equiv_{\mathcal{F}}$ also has finite index for every fixed t. □

Finally, we introduce the protrusion replacement lemma.

Lemma 11.6. *Let $\mathcal{F}$ be a finite family of graphs. There exists a constant c and an algorithm that given a graph G, a r-protrusion X in G with $|X| > c$, outputs in $O(|X|)$ time an instance G^* such that $|V(G^*)| < |V(G)|$, and for every all $Z \subseteq (V(G) \setminus X) \cup \partial(X)$,*

$$(G \setminus Z) \text{ is } \mathcal{F} \text{ free } \text{ if and only if } (G^* \setminus Z) \text{ is } \mathcal{F} \text{ free }.$$

Proof. Let $\mathcal{S}$ be a set of representatives for the equivalence classes of $\equiv_{\mathcal{F}}$ and let $c = \max_{Y \in \mathcal{S}} |Y|$. Let ζ be a mapping from $2r$-boundaried graphs with at most $2c$ vertices to $\mathcal{S}$ such that for any $2r$-boundaried graph H on at most $2c$ vertices, $H \equiv_{\Pi} \zeta(H)$.

If $|X| > 2c$, then we find a $2r$-protrusion $X' \subseteq X$ such that $c < |X'| \leqslant 2c$ and work on X' instead of X. This can be done in time $O(|X|)$ since $G[X]$ has treewidth at most r. In particular, consider a nice tree-decomposition of $G[X]$ and pick a lowermost bag b such that the number of vertices appearing in bags below b is more than c'. Let X' be the set of vertices appearing in b or in bags below b. The choice of b ensures that $c < |X'| \leqslant 2c$. Notice that $\partial(X) \cap (X' \setminus \partial(X')) = \emptyset$. From now on, we assume that $|X| \leqslant 2c$.

Let $H = \zeta(G[X])$. We make G^* from G by replacing the $2r$-protrusion $G[X]$ with H. By the definition of $\equiv_{\mathcal{F}}$ we have that for every all $Z \subseteq (V(G) \setminus X) \cup \partial(X)$, $(G \setminus Z)$ is $\mathcal{F}$ free if and only if $(G^* \setminus Z)$ is $\mathcal{F}$ free . Since $|H| < |X|$ we have that $|V(G^*)| < |V(G)|$. The running time of the algorithm is $O(|X|)$. This concludes the proof. □

11.4 The Kernel and the Algorithm

We are now ready to describe the kernelization procedure for Disjoint Planar $\mathcal{F}$-deletion. We note that we will first obtain an annotated kernel of size $O(k^3)$. We then reduce this further to an instance of Disjoint Planar $\mathcal{F}$-deletion without annotations, whose size is bounded by $k^{O(1)}$.

Proof of Theorem 11.1. For this proof, we will require the fact that Disjoint Planar $\mathcal{F}$-deletion is an annotated p-min-MSO problem, which takes as input a triplet

(G, Y, k), and YES-instances are required to satisfy a CMSO-expressible property $P_\Pi(G, T)$, where T is a vertex subset of G of size at most k, and is such that $T \subseteq Y$. The input in the case of Disjoint Planar $\mathcal{F}$-deletion is easily seen to be $(G, G \setminus S, k)$, restricted to those graphs that satisfy the property that $G[S]$ and $G \setminus S$ are $\mathcal{F}$-free. Further, the property $P_\Pi(G, T)$ is given by: $(\wedge_{H \in \mathcal{F}} \neg \phi_H(G))$, where $\phi_H(G)$ denotes the fact that H is a minor of G, well-known to be expressible in MSO (see Chapter 7). From now onwards we will assume that we have an arbitrary YES-instance and we will bound its size by $O(k^3)$.

We treat (G, S, k), an instance of Disjoint Planar $\mathcal{F}$-deletion, as an instance of an annotated p-min-MSO problem with $Y = G \setminus S$. Now we outline a polynomial time procedure that takes as input (G, S, k) such that $G[S]$ and $G \setminus S$ does not contain any graph from $\mathcal{F}$ as a minor and an annotated set Y and outputs an instance (G', S, k) and an annotated set $Y' \subseteq G' \setminus S$ with the following properties. (a) G has a $\mathcal{F}$-hitting set $Z \subseteq Y$ of size at most k if and only if G' has $\mathcal{F}$-hitting set $Z' \subseteq Y'$ of size at most k; and (b) $G'[S]$ and $G' \setminus S$ does not contain any minor from $\mathcal{F}$. Of course, we can perform this procedure when the size of $V(G)$. We will run this procedure several times to reduce the instance size.

Let (G, S, k) be such that $G[S]$ and $G \setminus S$ does not contain any graph from $\mathcal{F}$ and Y be an annotated set. Recall Lemma 11.4, due to which there exist constants α and β (that depend only on $\mathcal{F}$) such that if the total number of vertices in G is more than $c \cdot (\alpha(k+1)+k) \cdot (\beta(k+1)+k)$, then, there exists a r-protrusion on at least c vertices, where r is bounded by a constant that depends only on the size of some planar graph in $\mathcal{F}$.

We consider the case when the size of the instance G is more than $c \cdot (\alpha(k+1) + k) \cdot (\beta(k+1) + k)$ for a suitable choice of c to be determined in the course of this proof. Let X be the protrusion in G given by Lemma 11.4. Since Disjoint Planar $\mathcal{F}$-deletion is an annotated p-min-MSO problem, by [BFL+09, Lemma 1], there is an integer q and an $O(|X|)$ time algorithm, that computes a set of vertices $Z \subseteq X \cap Y$ with $|Z| \leqslant qk$, such that if there exists $W \subseteq Y$ such that W is a solution for (G, S, k), then there exists a solution W' such that the part of W' that lies in the protrusion X is contained in Z, that is $W' \cap X \subseteq Z$. At this stage we update our instance (G, S, k) and Y to (G, S, k) and $(Y \setminus X) \cup Z$.

Furthermore, by [BFL+09, Lemma 2], given a subset Q of X that has $O(k)$ vertices, we can find a collection of $O(k)$ protrusions $X_1, \ldots, X_\ell$ that cover the protrusion X, that is, $\cup_{i=1}^{\ell} X_i = X$, with the additional property that all vertices of Q lie only on the

boundaries of these protrusions, that is, $Q \cap X_i \subseteq \partial(X_i)$. We set Q as $Z \cup (S \cap X)$. Now, choosing c to be greater than $qk + dl$ gives us a protrusion on enough vertices so that we are left with at least one protrusion among X_i's, say X_γ on at least d vertices such that Lemma 11.6 is applicable. Specifically, since $X_\gamma \cap Z$ lies in $\partial(X_\gamma)$, we know that there is always a solution that does not involve the non-boundary vertices of X_γ. Recall that Lemma 11.6 guarantees that if there exists a solution that lies outside $X_\gamma \setminus \partial(X_\gamma)$ in G, then there is a solution in the graph that is obtained after replacing X_γ with a smaller protrusion. Therefore, we apply Lemma 11.6 with $\mathcal{F}$, G and X_γ and obtain an equivalent instance G'. By the properties of G' returned by Lemma 11.6, we have that (G', S, k) and Y satisfies all the desired properties. In particular, note that due to the equivalence established in Lemma 11.6, $G' \setminus S$ is $\mathcal{F}$-free because $G \setminus S$ is $\mathcal{F}$-free. Also the replacement can be performed in $O(n)$ time, since it involves finding X_γ, which can be done in linear time, and then looking up a table of constant size, and performing the replacement, which only takes time proportional to the size of the protrusion.

Now we repeat the above procedure as long as the size of the reduced instance is large enough for an application of Lemma 11.4. Notice that if the input is a YES-instance, when the procedure terminates we have an equivalent instance (G', S, k) and an annotated set $Y' \subseteq G' \setminus S$ with $|V(G)| = O(k^3)$. Clearly, $G'[S]$ and $G' \setminus S$ are $\mathcal{F}$-free.

So overall, our kernelization algorithm does the following: it performs the above procedure on the given instance and if the size of reduced instance is more than $O(k^3)$, it returns NO, else it returns an annotated kernel of size $O(k^3)$, which will be crucial to our algorithm. Note that we can obtain the annotated kernel in $O(n^2)$ time.

We can remove the annotation by utilizing the fact that Disjoint Planar $\mathcal{F}$-deletion is NP-complete and thus there is a polynomial time reduction from our annotated p-min-MSO problem to Disjoint Planar $\mathcal{F}$-deletion. Thus we first obtain an $O(k^3)$ sized annotated kernel and then on this instance apply the polynomial time reduction to get a polynomial kernel. This completes the proof. □

11.4.1 A FPT Algorithm for $\mathcal{F}$-Deletion

We now turn to the FPT algorithm that is based on iterative compression, and also uses the kernel obtained in Theorem 11.1 in every iteration.

Proof of Corollary 11.2. Let n be the number of vertices of G and m be the number of edges of G. Given an instance of Planar $\mathcal{F}$-Deletion, we first apply Lemma 11 of [FLM+11], that given a graph G and a positive integer k, either reports that G has no $\mathcal{F}$-hitting set of size at most k or finds a $\mathcal{F}$-hitting set S of size at most $\ell = \eta k \log^{3/2} k$. In the first case we also return that G does not have a $\mathcal{F}$-hitting set of size at most k. Now we order the vertices of S as $v_1, \ldots, v_\ell$ and define $V_i = \{v_1, \ldots, v_i\} \cup (V(G) \setminus S)$ for every $1 \leqslant i \leqslant \ell$. Notice that if G has a $\mathcal{F}$-hitting set Y of size k then $Y \cap V_i$ is a $\mathcal{F}$-hitting set of $G[V_i]$ for every $i \leqslant \ell$. Furthermore, if Y is $\mathcal{F}$-hitting set of $G[V_i]$ then $Y \cup \{v_{i+1}\}$ is a $\mathcal{F}$-hitting set of $G[V_{i+1}]$. We iteratively go from 1 to ℓ and in the i^{th} step we will have $G[V_i]$, a solution S of size $k+1$ for $G[V_i]$, and the objective is to check whether there exists a solution of size at most k or not. We first "guess" the part of S that will participate in the final solution of size at most k — let us call this Z. If $G[S \setminus Z]$ does not contain any graph in $\mathcal{F}$ as minor then we get $(G, S \setminus Z, k - |Z|)$ – an instance of Disjoint Planar $\mathcal{F}$-deletion. We will give an algorithm for Disjoint Planar $\mathcal{F}$-deletion that runs in time $2^{O(k \log k)} n^2$. Note that we can perform one step of the iteration by solving 2^{k+1} instances of Disjoint Planar $\mathcal{F}$-deletion. Now, if $G[V_i]$ does not have solution of size k then we return that G does not have $\mathcal{F}$-hitting set of size at most k. Else, we have $\mathcal{F}$-hitting set Y of size at most k for $G[V_i]$. Using Y we get another instance, namely $(G[V_{i+1}], Y \cup v_{i+1}, k)$, that can be solved by running an algorithm for Disjoint Planar $\mathcal{F}$-deletion 2^{k+1} times. Thus in at most ℓ iterations we would have solved the problem. Now we give an algorithm to solve Disjoint Planar $\mathcal{F}$-deletion.

Given an instance (G, S, k) to Disjoint Planar $\mathcal{F}$-deletion, we set Y, the set of annotated vertices, to be $G \setminus S$ and run the kernelization algorithm described in Theorem 11.1. The annotated kernel is an instance (G', S, k') with a set Y' of annotated vertices, such that: G has a $\mathcal{F}$-hitting set of size at most k in $G \setminus S$ if and only if G' has a $\mathcal{F}$-hitting set of size at most $k' \leqslant k$ in $(G' \setminus S) \cap Y'$ and further, $|V(G')| \leqslant dk^3$. Applying a brute force algorithm that enumerates all subsets of size at most k of $(G' \setminus S) \cap Y'$ gives us a worst case running time of: $\binom{dk^3}{k}$, which amounts to $2^{O(k \log k)}$ time for solving the Planar $\mathcal{F}$-Deletion problem. Thus, the total time time to solve Planar $\mathcal{F}$-Deletion is $O((2^{k+1} 2^{k \log k}) 2^{k+1} \ell)$ plus the time taken to obtain S and to run the kernelization algorithm. While the kernelization algorithm runs in $O(n^2)$ time, the current implementation of Lemma 11 in [FLM+11] though runs in polynomial time, may take more than quadratic time. Here, we outline an algorithm that runs in time $O(2^{O(k)} n^2)$. This would imply an algorithm for Planar $\mathcal{F}$-Deletion that runs in time $2^{O(k \log k)} n^2$. The algorithm described in Lemma 11 in [FLM+11]

computes an approximate tree-decomposition and given a tree-decomposition runs in $O(n+m) = O(n^2)$ time. However, there is a known algorithm by Reed [Ree92] that runs in time $O(2^{O(k)}n \log n)$ and either reports that $tw(G) > k$ or returns a tree-decomposition of width $5k$. Using the algorithm of Reed [Ree92] in Lemma 11 of [FLM+11] instead of using polynomial time approximation for tree-decomposition we can obtain the desired S in $O(2^{O(k)}n^2)$. This completes the proof. □

It was one of those parties where you cough twice before you speak and then decide not to say it after all.

P G Wodehouse

You see, there is only one constant. One universal. It is the only real truth. Causality. Action, reaction. Cause and effect.

The Matrix Reloaded

Let G be a graph, and let $\mathcal{H}$ be a class of graphs. The packing and covering problems are two natural questions that arise in this setting:

- ♢ The $\mathcal{H}$-PACKING problem asks for a set of vertex-disjoint subgraphs of G, called an $\mathcal{H}$-*packing*, such that each of these subgraphs is isomorphic to some graph in $\mathcal{H}$.
- ♢ The $\mathcal{H}$-COVERING problem asks for a set $S \subseteq V(G)$ of vertices, an $\mathcal{H}$-*cover*, such that $G \setminus S$ contains no subgraph isomorphic to any graph in $\mathcal{H}$.

It often turns out that the two problems are closely related in the sense that the absence of a "large" packing implies the presence of a "small" cover. This notion is captured formally by the Erdős-Pósa property:

> The class $\mathcal{H}$ is said to have the Erdős-Pósa property if there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that, for every $k \geqslant 0$, every graph G either contains an $\mathcal{H}$-packing of size at least k, or has an $\mathcal{H}$-cover of size at most $f(k)$.

Let H be a fixed connected graph. The class $\mathcal{H} = \mathcal{M}H$ consists of graphs that contain H as a minor. Observe that for $H = \theta_1$, and $H = \theta_2$, $\mathcal{H} = \mathcal{M}H$ consists of all graphs that contain at least one edge and all graphs that contain at least one cycle, respectively. Note that the Erdős-Pósa property has already been established for these cases:

- ♢ A collection of vertex-disjoint subgraphs, each isomorphic to θ_1, is clearly a *matching*. It is well-known that if the maximum matching of a graph is of size at most k, then it has a vertex cover of size at most $2k$.

♢ By the classical result of Erdős-Pósa [EP65b], the class $\mathcal{H} = \mathcal{M}\theta_2$ has the Erdős-Pósa property with $f(k) = O(k \log k)$, that is, if the largest collection of vertex-disjoint cycles of a graph is less than k, then there exists a feedback vertex set of size $O(k \log k)$.

We will demonstrate that the class $\mathcal{H} = \mathcal{M}\theta_c$ has the Erdős-Pósa property for every fixed $c > 0$. To describe the Erdős-Pósa property formally, we need the notion of a $\mathcal{M}\theta_c$-cover; which we have already encountered: it is the $\mathcal{F}$-hitting set for $\mathcal{F} = \ \theta_c$.

Definition 12.1 (θ_c-hitting set). *Given a graph G and a vertex subset $S \subseteq V(G)$, we call S a θ_c-*hitting set *if $G \setminus S$ does not contain θ_c as a minor.*

In this chapter, as long as the context is unambiguous, we will use the term "hitting set" to refer to a θ_c-hitting set. The theorem that lies at the heart of this chapter is the following:

Theorem 12.1. [Erdős-Pósa property for θ_c] *Every graph G either contains k vertex-disjoint θ_c-minor models, or has a θ_c-hitting-set of size at most $f(k) = O(k^2 \log k)$.*

12.1 The Erdős-Pósa Property for θ_c

The Erdős-Pósa property for $\mathcal{M}\theta_c$ is shown by establishing the following two lemmas.

Lemma 12.1. *If the treewidth of a graph G is at least $2c^2k^2$, then G contains at least k vertex-disjoint θ_c-minor-models.*

Lemma 12.2. *If the treewidth of G is less than $2c^2k^2$ and G contains fewer than k vertex-disjoint θ_c-minor-models, then G contains a θ_c-hitting set of size at most $\eta k^2 = O(k^2)$, where the constant η depends only on c.*

The proof of Theorem 12.1 follows from the above two lemmas.

Proof of Theorem 12.1. Suppose graph G does not contain k vertex-disjoint θ_c-minor-models. Then by Lemma 12.1, G has treewidth at most $2c^2k^2$. Now by applying Lemma 12.2, we have that G contains a θ_c-hitting set of size at most $f(k)$. □

12.2 Unbounded treewidth implies a large packing

We require the notion of *brambles* for our proof of Lemma 12.1, so we introduce some elementary definitions first.

Brambles. We say that two subsets of $V(G)$ *touch* if they either have at least one vertex in common or if there is at least one edge with one endpoint in each subset.

Definition 12.2 (Brambles). *A* bramble $\mathcal{B}$ *of a graph* G *is a collection of mutually touching connected subgraphs, called the* elements *of the bramble.*

The canonical example of a bramble is the set of crosses (union of a row and a column) of an $(l \times l)$-grid.

Further, the following terms are also useful:

- A *hitting set* of a bramble is a subset of vertices S whose intersection with every element of the bramble is non-trivial.
- For a bramble $\mathcal{B}$, the *order* of $\mathcal{B}$ is the minimum cardinality of a hitting set of the bramble.
- For a graph G, *bramble number* of G is the maximum order of a bramble of G.

For instance, for the example of the set of crosses of the $(l \times l)$-grid above, the vertices of any row (or column), would intersect all elements of the bramble and would therefore constitute a hitting set.

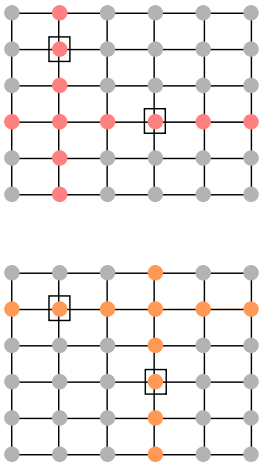


Figure 12.1: A pair of elements that share vertices, from the bramble that consists of the set of crosses of a grid.

Brambles and Treewidth. Let $\mathcal{B}$ be a bramble of a graph G, and let $(T, \{T_v : v \in V(T)\})$ be a tree-decomposition of G of the minimum width. Consider an element B of $\mathcal{B}$. Because B is connected, $T_B := \cup\{T_v : v \in V(B)\}$ is a subtree of T (see Lemma 2.2). Furthermore, because any two bramble elements touch, $\{T_B : B \in \mathcal{B}\}$ is a pairwise intersecting family of subtrees of T. Indeed, if a pair of brambles B_i and B_j have a common vertex v, then consider any vertex $x \in T$ such that $v \in T_x$ (such a bag exists because every vertex of G appears in at least one of the bags of its tree decomposition, by definition). Clearly, x is common to the subtrees T_{B_i} and T_{B_j}. Similarly, if there is an edge (u, v) with

$u \in B_i$ and $v \in B_j$, then the consider a vertex $y \in T$ such that $(u, v) \in T_y$ — again, such a bag exists because every edge of G appears in at least one of the bags of its tree decomposition, by definition. Note that y belongs to both T_{B_i} and T_{B_j}, and we have that $\{T_B : B \in \mathcal{B}\}$ is a pairwise intersecting family of subtrees of T.

By the Helly property of a family of pairwise intersecting subtrees (see Lemma 2.1), therefore, these subtrees have a vertex x in common. The corresponding subset T_x of V is thus a hitting set of $\mathcal{B}$. This shows that the order of $\mathcal{B}$ is at most $|T_x|$, which in turn is no greater than the tree-width of G. Because this is true for any bramble B, we conclude that the bramble number is bounded above by the size of the bags of the optimal tree decomposition, that is, $(1 + tw(G))$. In fact, the converse is also true: if the treewidth is large, then there exists a bramble of large order. This is summarized in the following theorem, first shown by Seymour and Thomas [ST93].

Proposition 12.2. [ST93] *The treewidth of a graph G is at least k if and only if it has a bramble of order $(k + 1)$.*

We now make another useful observation:

Lemma 12.3. [BBR07] *Let $\mathcal{B}$ be a bramble in a graph G. Then G contains a path that intersects every element of $\mathcal{B}$.*

Proof. Let P be a path in G that (1) intersects as many elements of $\mathcal{B}$ as possible, and (2) is as short as possible. Let v be an endpoint of P. There is a bramble element X that only intersects P at v, as otherwise we could delete v from P.

Suppose P does not intersect some bramble element Z. Since X and Z touch, there is a path Q starting at v through X to some vertex in Z, and $Q \cap P = \{v\}$. Thus $P \cup Q$ is a path that also hits Z. This contradicts that P hits the most elements of $\mathcal{B}$. □

Recall, at this point, that a minor model of θ_c consists of two connected components with c vertex-disjoint paths between them. We need to find, given only the assumption of "large" treewidth, k vertex-disjoint subgraphs, each containing a minor model of θ_c as a subgraph. We use the assumption of large treewidth to imply the existence of a bramble of large order. In this bramble, we utilize the path that hits every element to partition the elements of the brambles carefully into two smaller brambles, each with large enough order. Then we prove the existence of a large number of vertex disjoint paths "going across". This is shown by contradiction — the absence of a

large collection of vertex-disjoint paths will imply small cut between the two brambles, which turns out to be impossible because of the structure of the brambles and the fact that none of them admit small hitting set. We are now ready for a precise proof of Lemma 12.1.

Our proof of Lemma 12.1 uses some ideas from the proof of Lemma 3.2 in [WR08].

Proof of Lemma 12.1. We show that if the treewidth of a graph G is at least $2c^2k^2$, then G contains at least k vertex-disjoint θ_c-minor-models.

If the treewidth of G is at least $2c^2k^2$, then by Proposition 12.2, G contains a bramble (call it $\mathcal{B}$) of order at least $2c^2k^2 + 1$. By Lemma 12.3, there exists a path that visits every element of the bramble at least once. Let P be such a path, and let $v_1, \ldots, v_t$ be the vertices of P (stated in the order of their appearance in P). Note that $t \geqslant 2c^2k^2+1$, else P would be a hitting set of $\mathcal{B}$ with fewer vertices than the order of $\mathcal{B}$.

For $1 \leqslant i \leqslant t$, let B_i denote the set of all elements of $\mathcal{B}$ which contain the vertex v_i. Note that for $1 \leqslant i \leqslant t$, $\cup_{j=1}^{i} B_j$ is a bramble. Let O_i denote the order of this bramble. Let s be the smallest number such that $O_s \geqslant c^2k^2$. The existence of such s is guaranteed by the fact that $O_1 = 1$, $O_t > 2c^2k^2$, and for $1 \leqslant i \leqslant t-1$, $O_{i+1} \leqslant O_i + 1$.

Let $\mathcal{B}_1 = \cup_{i=1}^{s} B_i$, and let $\mathcal{B}_2 = \mathcal{B} \setminus \mathcal{B}_1$ (see Figure 12.2). Since the value of O_i only increases by one in a single step, we have that the order of $\mathcal{B}_2$ is at least c^2k^2, or else the union of the smallest hitting sets for $\mathcal{B}_1$ and for $\mathcal{B}_2$ would be a hitting set of $\mathcal{B}$ which has fewer vertices than the order of $\mathcal{B}$. Let P_1 be the sub-path of P starting at v_1 and ending at v_s, and P_2 the subpath starting at v_{s+1} and ending at v_t. By the above argument, P_1 and P_2 contain at least c^2k^2 vertices each.

Now, there must exist a collection, say $\mathcal{P}$, of at least c^2k^2 vertex-disjoint paths that begin in P_1 and end in P_2. If not, then by Menger's theorem, there exists a P_1–P_2 separator, say S, of size less than c^2k^2. Note that S cannot be a hitting set of the brambles $\mathcal{B}_1$ and $\mathcal{B}_2$, since the order of each of these is at least c^2k^2. So there exist elements $A \in \mathcal{B}_1, B \in \mathcal{B}_2$ such that $A \cap S = \emptyset = B \cap S$. But since A and B touch (being elements of $\mathcal{B}$), and $A \cap P_1 \neq \emptyset, B \cap P_2 \neq \emptyset$, S cannot be a P_1-P_2 separator.

We now show that $\mathcal{P} \cup P_1 \cup P_2$ contains k vertex-disjoint θ_c minor-models. Let E_p be the set of vertices that form the end points (on P_1 and P_2) of the paths in $\mathcal{P}$. For $i \in \{1, 2\}$, let $Q_i = P_i \cap E_p$. We label both Q_1 and Q_2 with a common index set $[M]$, where $M = |Q_1| = |Q_2|$. Let $f : [M] \to [M]$ be the following bijection: $f(i) = j$ if

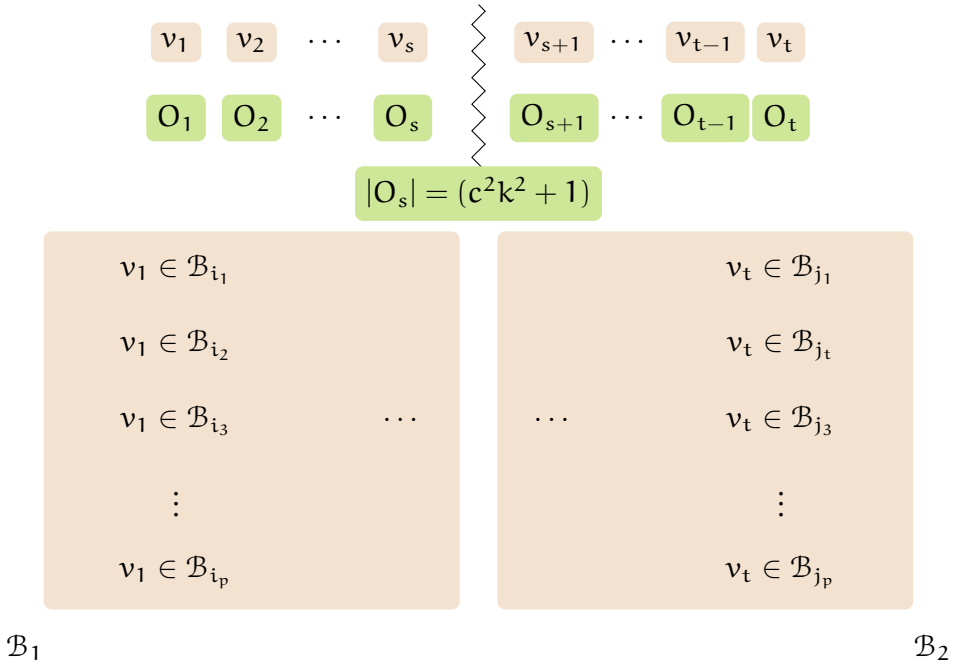


Figure 12.2: A schematic of the construction of brambles $\mathcal{B}_1$ and $\mathcal{B}_2$.

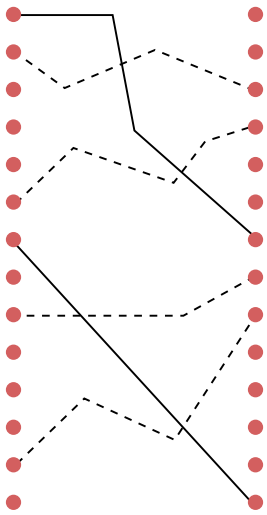


Figure 12.3: The dashed paths form a collection of cross-free paths.

and only if there is a path in $\mathcal{P}$ that begins in i and ends in j. We say that a subset of paths $C \subseteq \mathcal{P}$ is *cross-free* under this labeling if there does not exist $i, i' \in Q_1 \cap C; i < i'$ and $f(i) > f(i')$.

Note that since the paths in $\mathcal{P}$ are vertex-disjoint, the numbers $f(1), f(2), \ldots, f(M)$ form a permutation of M, and by Erdős-Szekeres Theorem [ES35], the sequence $\langle f(1), f(2), \ldots, f(M)\rangle$ contains a monotonically increasing or decreasing subsequence of length at least $\sqrt{|M|} = ck$. For $i \in \{1, 2\}$, let Q_i' denote the intersection of Q_i with this subsequence. It is easy to verify that the paths in $\mathcal{P}$ that have their end points in Q_1', Q_2' form a cross-free collection. These paths together with P_1, P_2 contain at least k vertex-disjoint θ_c minor-models. □

12.3 Bounded treewidth and a small packing number implies a small cover

In this section, we show that if G has treewidth at most $2c^2k^2$ and does not have more than $k' = k - 1$ disjoint minor-models of θ_c, then there exists a set $S \subseteq V(G)$, $|S| = O(k^2)$, such that $G \setminus S$ does not contain θ_c as a minor.

The hitting set is obtained using methods that are similar to the second approximation algorithm (Algorithm 2) in Chapter 8. There are, however, some notable points of difference. In this algorithm, our choice of measure effectively bounds the treewidth of the graph. This plays into the analysis of the size of the solution that we obtain. Also, this algorithm is not a polynomial time procedure, since it relies on the computation of an optimal tree decomposition. Finally, given the premise that the input graph G has treewidth at most $2c^2k^2$ and does not have more than $k' = k - 1$ disjoint minor-models of θ_c, this algorithm always returns a hitting set of size $O(k^2)$, and hence constitutes a proof of Lemma 12.2.

Recall, from Chapter 8, that Algorithm 2 required a function μ on graphs that has the following properties:

- the value of μ on the graph corresponding to the input instance is polynomial in k,
- we can find a small separator X that separates the input graph into two parts G_A and G_B such that the value of the function on each of G_A and G_B is at most a constant fraction of what it was on the entire graph,
- and the problem is easily solved when the value of the function is a small constant.

The algorithm that determines a hitting set, as before, recurses on the parts G_A and G_B and picks X in the solution. Intuitively, X serves to cover all the minor models that have vertices in G_A *and* G_B, and the remaining minor models are covered recursively. Our description of the algorithm proceeds in three parts. We first propose a choice of μ and demonstrate that the three properties are satisfied. We then provide a formal description of the algorithm. Finally, we demonstrate correctness by showing that the algorithm always returns a hitting set of size $O(k^2)$.

12.3.1 The choice of μ

The maximum number of vertex-disjoint minor models of θ_c in G turns out to be a suitable choice for μ. For the rest of this discussion, we will use the term *packing number* to refer to the maximum number of vertex-disjoint minor models of θ_c present in a graph. Thus, we are suggesting that the packing number of G is an appropriate choice for $\mu(G)$. We now verify each of the three properties stated previously.

The value of μ on the entire graph is $k^{O(1)}$. The first criteria is easily established, as we have the premise that G does not have more than $(k-1)$ disjoint minor-models of θ_c.

Existence of a small separator that reduces the packing number by a constant fraction in each part. Our next goal is to find a small separator that separates the graph into parts where the packing number is a constant fraction of the packing number of the entire graph. Again, we consider the tree decomposition of G with the intention of using the bags as separators. For any bag t, let us consider the behavior of μ on the graphs H_t, which is the subgraph induced on the union of vertices appearing in all bags present in the subtree rooted at t, except the vertices at the bag corresponding to node t itself. We use m to denote $\mu(G)$.

For the given tree decomposition, let us analyze the behavior of μ on the graphs H_t. It is easy to verify the following:

- the value of $\mu(H_t)$ at an introduce node t is the same as the value of $\mu(H_s)$ if s is the child of t.
- the value of $\mu(H_t)$ at a forget node t is at most one more than value of $\mu(H_s)$ if s is the child of t (it is possible that the forgotten vertex v was marked, and this vertex appears in the graph H_t but not in H_s.

- the value of $\mu(H_t)$ at a join node t is $\mu(H_r)+\mu(H_s)$, where r and s are the children of t.

Consider the node with the largest distance from the root at which $\mu(H_t)$ is more than $(m/3)$: that is, a node t such that for some constant α, $\mu(H_t) > (m/3)$ and $\mu(H_s) \leqslant (m/3)$, where s is the child of t (if t has two children, then the aforesaid is required to be true of *each* of the children). Notice that any node t which realizes this property is either a join node or an introduce node, because the value of $\mu()$ does not "jump" at an introduce node.

Notice that if t is a forget node with child s, the subgraphs H_s and $G \setminus (H_t \cup X_t)$ are exactly the pieces we need, separated by X_s. Notice that $\mu(G \setminus H_t)$ is at most $m(1-\frac{1}{3})$, since the packing number of the graph is at most m and the packing number of H_t is at least $(m/3)$ by choice of t. Thus, $\mu(G \setminus (H_t \cup X_t))$ is also at most $(m(1-\frac{1}{3}))$, since $G \setminus (H_t \cup X_t)$ is a subgraph of $G \setminus H_t$.

Now we turn to the case when t is a join node. Let r and s be the child nodes of t. Notice that $X_t = X_s = X_r$, and the natural candidates for G_A and G_B (the subgraphs to recurse on) would be some combinations of the pieces obtained after the deletion of X_t. Let's consolidate what we know:

$$\mu(H_s) \leqslant m/3 \text{ and } \mu(H_r) \leqslant m/3,$$

$$\mu(H_s) + \mu(H_r) \leqslant m$$

$$\mu(H_t) > m/3,$$

and

$$\mu(G \setminus (H_t \cup X_t)) \leqslant \left(1 - \frac{1}{3}\right) m.$$

Thus, note that $\mu(H_s \cup H_r) \leqslant (2/3)m$ and $\mu(G \setminus (H_t \cup X_t)) \leqslant (2/3)m$. Thus we achieve a constant factor drop in both instances $G_A := (H_s \cup H_r)$ and $G_B := G \setminus (H_t \cup X_t)$. Our observations are summarized in the following claim:

Claim 12.1. *In a graph G, there exists a separator S such that $G \setminus S$ admits a partition into two parts G_A and G_B with the following properties:*

$$\mu(G_A) \leqslant (2/3)\mu(G) \text{ and } \mu(G_B) \leqslant (2/3)\mu(G).$$

Thus, we will be able to obtain separators whose sizes are proportional to the treewidth. Again, because of our premise, the treewidth is bounded by $2c^2k^2$, and so is the size of the separators that are obtained as bags of the tree decomposition.

Remark 12.1. *A crucial point is that at every stage of recursion, the packing number drops by a fraction, and the treewidth decreases appropriately — since we already know, by Lemma 12.1 that if the treewidth is large, then the packing number is large as well. So, for example, if the packing number is* $(k/2)$*, then the treewidth is at most* $2c^2(k/2)^2$*. At every level of recursion, the algorithm computes an optimal tree decomposition of the current instance before proceeding. This will be apparent in the description of the algorithm in the next section.*

Finding a hitting set of bounded size when the value of μ is a constant. Towards establishing the last property, we show that if the maximum number of vertex disjoint minor models of θ_c is bounded by a constant, say d, then the treewidth is also bounded by a constant. Indeed, recall that any graph with treewidth greater than 20^{2t^5} contains a $t \times t$ grid as a minor. Further, notice that a $(d+1)c \times d$ grid contains more than d vertex disjoint minor models of θ_c. Therefore, if the treewidth exceeds 20^{2t^5} for $t = dc$, then we contradict our assumption that the maximum number of vertex disjoint minor models of θ_c is bounded by d. Hence, if the maximum number of vertex disjoint minor models of θ_c is bounded, by say d, then the treewidth is bounded by $20^{(2dc)^5}$, a constant.

We describe the procedure for obtaining a hitting set when $\mu(G) = 1$, as this is the case that is relevant to the algorithm. Now, we compute the tree decomposition which, as discussed, will have bags of constant size. Starting from the leaves and working our way upwards, we identify the first node t at which the value of the measure flips from 0 to 1. We claim that X_t, which is the set of vertices in the bag corresponding to the node t, constitutes a θ_c-hitting set of constant size. Indeed, let the components of $G \setminus X$ in H_t be $H_1, \ldots, H_p$, and let $H_{p+1}, \ldots, H_q$ denote remaining the components of $G \setminus X$. If there is a minor model of θ_c in any H_i, $1 \leqslant i \leqslant p$, then we contradict the choice of t as being the first node at which the value of $\mu()$ changed from 0 to 1. On the other hand, if there is a minor model of θ_c in any H_i, $p < i \leqslant q$, then we contradict the assumption that $\mu(G) = 1$. The fact that X_t is constant size follows from our previous inference that G has constant treewidth when the packing number is a constant (in this case, 1).

Thus we have shown the following claim:

Claim 12.2. *If* $\mu(G) = 1$, *then* G *admits a* θ_c*-hitting set of constant size.*

12.3.2 The Algorithm for finding a θ_c-hitting set

We are now ready for a formal description of the algorithm that finds a θ_c-hitting set, which is provided below, in Algorithm 5.

Algorithm 5 HIT-SET$(G, \mu(G))$

1: Compute an optimal tree decomposition $(T, \mathcal{X} = \{X_t\}_{t \in V(T)})$ of G.
2: Convert $(T, \mathcal{X} = \{X_t\}_{t \in V(T)})$ to a nice tree decomposition of the same width.
3: **if** $\mu(G) = 1$ **then**
4: Return a solution of constant size (see Claim 12.2).
5: **else**
6: Find a partitioning of vertex set $V(G)$ into G_A, G_B and X_t (a bag corresponding to a node in T) such that $\mu(G_A) \leqslant (2/3)\mu(G)$, and $\mu(G_B) \leqslant (2/3)\mu(G)$ (See Claim 12.1).
7: Return HIT-SET$(G_A, (2/3)\mu(G)) \cup$ HIT-SET$(G_B, (2/3)\mu(G)) \cup X_t$
8: **end if**

12.3.3 Analysis and Approximation Ratio

It is easy to see that Algorithm 5 runs in polynomial time. This follows by a simple induction on the depth of recursion. We will provide a proof of correctness and the fact that that the size of solution output by the algorithm is $O(k^2)$.

The fact that the subset of vertices returned by the hit all minor models of θ_c in the graph follows from induction on the depth of recursion. The correctness of the base case follows from Claim 12.2. For the induction step, recall that we begin by identifying a set node t in the tree decomposition of G, and the vertices in the corresponding bag X_t are included in our solution. With the induction hypothesis, we may assume that we have hitting sets of minor models of θ_c in the graphs G_A and G_B (obtained in accordance with Claim 12.1). Let us denote these hitting sets by S_1 and S_2, and now consider $G \setminus \{S_1 \cup S_2\}$. Note that any remaining minor models have vertices in G_A *and* G_B, and therefore in X_t, because minor models are connected subgraphs and X_t separates G_A from G_B. Therefore, $S_1 \cup S_2 \cup X_t$ indeed hits all minor models of θ_c in G.

The size of the solution returned by Algorithm 5 is governed by the following recurrence (see Claim 12.1):

$$S(G, \mu(G)) \leqslant S(G_A, (2/3)\mu(G)) + S(G_B, (2/3)\mu(G)) + tw(G).$$

Note that by Lemma 12.1, we have that $tw(G) \leqslant c^2\mu(G)^2$. We know that $\mu(G)$ is bounded by k, and we abuse notation and use k to denote $\mu(G)$. Thus, note that $S(G, \mu(G))$ is upper bounded by $T(G, k)$, which is given by the following recurrence:

$$T(G, k) = T(G_A, 2/3k) + T(G_B, 2/3k) + c^2k^2.$$

The base case is that when $k = 1$, we have, from Claim 12.2:

$$T(G, 1) = O(1).$$

Expanding the recurrence above, together with the fact that $T(G, 1) = O(1)$ we obtain:

$$T(G, k) = c^2k^2 + \left(\sum_{i=1}^{d} \left(\frac{8}{9}\right)^i \cdot c^2k^2\right) + O(1)$$

The following sum is a geometric series and tends to a constant in the limit $i \to \infty$, and is therefore bounded by a constant when taken only up to a finite number of terms:

$$\sum_{i=1}^{d} \left(\frac{8}{9}\right)^i.$$

The above proves that the recurrence solves to $O(k^2)$, as desired.

12.4 Edge-Disjoint $\{\theta_c\}$-packing: Some Observations

We demonstrate a polynomial time procedure for bounding (by a polynomial function of k) the maximum degree of instances of the problem of packing at least k edge-disjoint copies of θ_c minor models in G. Formally, the problem is the following:

EDGE-DISJOINT θ_c PACKING

Input:	A graph G, a non-negative integer k
Parameter:	k
Question:	Does there exist $S_1, S_2, \ldots, S_r \subseteq V$, $r \geqslant k$, such that $G[S_i]$ induces a θ_c minor model, and $E(S_i) \cap E(S_j) = \emptyset$ for all $1 \leqslant i \leqslant r$?

Our recipe for bounding the degree is as follows: We first argue, using the Erdős-Pósa property, that either the input graph admits a "small" hitting set for all minor models passing through a vertex, or that there must be a "large" number of vertex-disjoint minor models of θ_c. In the latter case, we are already done, as we may return a trivial YES-instance as a kernel. Otherwise, we find an approximate hitting set, and use it to find hitting sets for every vertex v: these sets H_v hit all minor models that contain v. Thus, for every v, we have a small set of vertices that hits all minor models that pass through v. We are now able to use expansion properties to bound the degree of every vertex.

We begin by recalling the Erdős-Pósa property for packing disjoint minor models of θ_c:

Every graph G either contains k vertex-disjoint θ_c-minor models, or has a θ_c-hitting-set of size at most $f(k) = O(k^2)$.

Thus, notice that we may apply an approximation algorithm to find a θ_c-hitting set, with one of the following outcomes: we are either able to conclude that the graph has no θ_c-hitting set of size $O(k^2)$, or we have a θ_c-hitting set of size $O(k^2 \log k)$. Notice that in the first case, due to the Erdős-Pósa property, we may conclude that the input instance is a YES instance of EDGE-DISJOINT θ_c PACKING, and we are done. Thus we are left with the situation when we have a θ_c-hitting set of size $O(k^2 \log k)$. Let us call this hitting set S. Before we describe the procedure for bounding the degree using S, let us observe the following reduction rules:

Reduction Rule 12.1 (Irrelevant Vertex Rule). *If a vertex v is not contained in any minimal minor model of θ_c in G, then delete v.*

Definition 12.3 (r-flower). *A r-flower passing through v is a collection of r subsets of $V(G)$:*

$$\{S_1, S_2, \ldots, S_r\}$$

that interesect pairwise exactly at v*:*

$$S_i \cap S_j = \{v\}, \quad \forall 1 \leqslant i, j \leqslant r$$

and are such that $G[S_i]$ *contains a minor model of* θ_c *for all* $1 \leqslant i \leqslant r$.

Reduction Rule 12.2 (Flower Rule). *If a* k*-flower passes through a vertex* v *of* G*, then return a trivial YES instance.*

The soundness of the irrelevant vertex rule is evident. From the definition of a r-flower, it follows that the structure of a r-flower corresponds to r edge-disjoint minor models of θ_c. Therefore, if a graph admits a k-flower, then we have witnesses for k edge-disjoint minor models of θ_c and therefore we may declare the input a YES instance.

We note that one can test whether a particular vertex v is part of any minimal minor-model corresponding to θ_c using the rooted minor testing algorithm of Robertson and Seymour [RS95]. It is not clear, however, that one might check whether a vertex is a part of a θ_c flower of size at least k in polynomial time. We defer the application of the flower rule and apply it only when the problem of finding a θ_c flower of size at least k can be solved in polynomial time.

For a vertex $v \in \mathcal{S}$ let $\mathcal{S}_v := \mathcal{S} \setminus \{v\}$ and let $G_v := G \setminus \mathcal{S}_v$. By Lemma 7.7, we know that any graph of treewidth greater than $(2c-1)$ contains a θ_c, as a minor. Since deleting v from G_v makes it θ_c-minor-free, we have that for a fixed c, $\mathrm{tw}(G_v) \leqslant (2c-1)+1 = O(1)$. Now by Lemma 7.4, we find in linear time the size of the largest flower containing v in G_v. If for any vertex $v \in \mathcal{S}$ the size of the flower in G_v is at least $k+1$, we apply Flower Rule and stop. So from now onwards we assume that for every vertex $v \in \mathcal{S}$ the maximum size of a flower passing through v in G_v is at most k.

Now we describe how to find, for a given $v \in V(G)$, a hitting set $H_v \subseteq V(G) \setminus \{v\}$ for all minor-models of θ_c that contain v. Since this hitting set is required to *exclude* v, H_v cannot be the trivial hitting set $\{v\}$. If $v \notin \mathcal{S}$, then $H_v = \mathcal{S}$. On the other hand, suppose $v \in \mathcal{S}$. Since the maximum size of a flower containing v in the graph G_v is at most k by Lemma 10.1, we can find a set T_v of size $O(k)$ that does not contain v and hits all the θ_c minor-models passing through v in G_v. Hence in this case we set $H_v = \mathcal{S}_v \cup T_v$. We denote $|H_v|$ by h_v. Note that H_v is defined algorithmically,

that is, there could be many small hitting sets in $V(G) \setminus \{v\}$ hitting all minor-models containing v, and H_v is one of them.

Observe that for every vertex v the set H_v is also a θ_c hitting set for G, that is, H_v hits *all* minor-models of θ_c in G. Consider the graph $G \setminus H_v$. Let the components of this graph that contain a neighbor of v be $C_1, C_2, \ldots, C_r$. Note that v cannot have more than $(c-1)$ neighbors into any component, else contracting the component will form a θ_c minor and will contradict the fact that H_v hits all the θ_c minors. Also note that none of the C_i's can contain a minor model of θ_c.

We say that a component C_i is adjacent to H_v if there exists a vertex $u \in C_i$ and $w \in H_v$ such that $(u, w) \in E(G)$. Next we show that vertices in components that are not adjacent to H_v are irrelevant in G. Recall a vertex is irrelevant if there is no minimal minor model of θ_c that contains it. Consider a vertex u in a component C that is not adjacent to H_v. Since $G[V(C) \cup \{v\}]$ does not contain any θ_c minor we have that if u is a part of a minimal minor model $M \subseteq G$, then $v \in M$ and also there exists a vertex $u' \in M$ such that $u' \notin C \cup \{v\}$. Then the removal of v disconnects u from u' in M, a contradiction to Observation 7.2 that for $c \geqslant 2$, any minimal θ_c minor model M of a graph G does not contain a cut vertex. Applying the Irrelevant Vertex Rule to the vertices in all such components leaves us with a new set of components $D_1, D_2, \ldots, D_s$, such that for every i, in D_i, there is at least one vertex that is adjacent to a vertex in H_v.

As before, we continue to use G to refer to the graph obtained after the Irrelevant Vertex Rule has been applied in the context described above. We also update the sets H_v for $v \in V(G)$ by deleting all the vertices w from these sets those have been removed using Irrelevant Vertex Rule.

Now, consider a bipartite graph $\mathcal{G}$ with vertex bipartitions H_v and D. Here $D = \{d_1, \ldots, d_s\}$ contains a vertex d_i corresponding to each component D_i. We add an edge (v, d_i) if there is a vertex $w \in D_i$ such that $(v, w) \in E(G)$.

Now, we describe a greedy algorithm that demonstrates the presence of at least k edge-disjoint minor models of θ_c, all containing the vertex v, when the degree of v is suitably large. Consider the vertices in H_v, and let

$$\{u_1, \ldots, u_{h_v}\}$$

be a list of vertices in H_v in order of decreasing degree in $\mathcal{G}$: that is, the vertices that "see" the largest number of components appear the earliest in this ordering. Let the

modified degree of u_i, denoted x_i, be defined as the number of 'new' component vertices seen by u_i, that is, the number of components that are adjacent to u_i but not to u_j for any $j < i$. Formally:

$$x_i := |N(d_i) \setminus \cup_{1 \leqslant j < i} N(d_j) \cap N(d_i)|$$

Further, let $y(i)$ denote the number of "new" (in the same sense as before) edge-disjoint minor models that are possible between d_i and v:

$$y_i := \lfloor x_i / c \rfloor$$

Notice that if there exists j, $1 \leqslant j \leqslant s$, such that:

$$\sum_{i=1}^{j} y_i \geqslant k,$$

then we obtain k edge disjoint minor models of θ_c, all intersecting exactly at v. In this case, we return YES and stop. Else, notice that the number of components is precisely the sum of all x_i, which is in turn bounded:

$$\sum_{i=1}^{h_v} x_i \leqslant \sum_{i=1}^{h_v} (cy_i + c) \leqslant c \left\{ \sum_{i=1}^{h_v} y_i + \sum_{i=1}^{h_v} 1 \right\} \leqslant c(k + h_v).$$

Thus we arrive at the following reduction rule:

Reduction Rule 12.3 (High Degree). *If a vertex v has degree more than $c(k+h_v)$, then return a trivial YES instance.*

Notice that after the application of the high degree reduction rule, we are either done, or left with a graph where the degree of every vertex is bounded by $c(k + h_v) = O(k^2 \log k)$.

With this, we arrive at the following theorem:

Theorem 12.3. *We can reduce, in polynomial time, a general instance of* EDGE-DISJOINT θ_c PACKING *to an equivalent instance where the maximum degree is bounded by* $O(k^2 \log k)$.

Using the arguments in Section 10.3, we can infer the existence of a suitable protrusion given an instance where the maximum degree is bounded by $k^{O(1)}$. However, since we are unable to demonstrate that the problem is CMSO-expressible, or that it has finite integer index, the protrusion-based reduction rules do not apply. From this point, the argument can be completed to one for obtaining a polynomial kernel if we are able to prove that Edge-disjoint θ_c packing is CMSO-expressible, or that it has finite integer index. We leave this as an interesting open problem.

It isn't that they can't see the solution.
It is that they can't see the problem.

Chesterton, G. K.

The irrationality of a thing is no argument against its existence,
rather a condition of it.

Friedrich Wilhelm Nietzsche

So far, most of our work involved the design of kernelization algorithms leading up to polynomial-sized kernels. While every problem that is fixed-parameter tractable admits a kernel[1], there are fixed-parameter tractable problems for which polynomial-sized kernels are not known. An explanation was provided in the work of Bodlaender et al. [BDFH09] where it was shown that unless NP $\subseteq$ CoNP/poly, there are fixed-parameter tractable problems that do not admit a polynomial sized kernel. This triggered further work on showing lower bounds of kernels, and in this chapter, we briefly summarize developments in the area starting from the framework developed in the paper by Bodlaender et al. [BDFH09]. We will apply this framework to concrete examples in the next two chapters.

13.1 Composition Algorithms

In the classical setting, a distillation algorithm is defined as follows.

Definition 13.1. *Let* $Q, Q' \subseteq \{0,1\}^*$ *be classical problems. A distillation from* Q *in* Q' *is a polynomial time algorithm* $\mathcal{D}$ *that receives as inputs finite sequences* $\overline{x} = (x_1, \ldots, x_t)$ *with* $x_i \in \{0,1\}^*$ *for* $i \in [t]$ *and outputs a string* $\mathcal{D}(\overline{x}) \in \{0,1\}^*$ *such that*

1. $|\mathcal{D}(\overline{x})| = (\max_{i \in [t]} |x_i|)^{O(1)}$

[1] The size of the kernel that any FPT problem is guaranteed to have is proportional to f, if the FPT algorithm runs in time $f(k)p(n)$. Note that the size of such a kernel cannot be a polynomial function of k, unless P=NP.

2. $\mathcal{D}(\overline{x}) \in Q'$ *if and only if for some* $i \in [t] : x_i \in Q$.

If $Q' = Q$ we speak of a self-distillation. We say that Q has a distillation if there is a distillation from Q in Q' for some Q'. The following theorem is crucial to the theory of lower bounds in kernelization.

Theorem 13.1 ([FS11]). *If any* NP*-complete problem has a distillation algorithm then NP* $\subseteq$ *CoNP/poly.*

In this section we introduce composition algorithms, distillation-type algorithms for parameterized problems. The existence of a composition algorithm for a parameterized problem along with a polynomial kernel implies a *distillation* algorithm for the corresponding classical problem (c.f [BDFH09]).

The formal definition of a composition algorithm is the following:

Definition 13.2 ([BDFH09]). *Let* Π *be a parameterized problem. A composition of* Π *is a polynomial time algorithm* $\mathbb{A}$ *that receives as inputs finite sequences* $\overline{x} = ((x_1, k), (x_2, k), \ldots, (x_t, k))$ *with* $x_i \in \{0,1\}^*$ *for* $i \in [t]$. *The algorithm is required to output an instance* $\mathbb{A}(\overline{x}) := (y, l) \in \{0,1\}^* \times \mathbb{N}$ *such that*

1. $l = k^{O(1)}$
2. $(y, l) \in \Pi$ *if and only if for some* $i \in [t] : (x_i, k) \in \Pi$.

It turns out that if the parameterized version of a NP-complete problem admits both a composition and a polynomial kernel, then it also has a distillation. This will imply that if a parameterized problem has a composition algorithm, then it has no polynomial kernel unless NP $\subseteq$ CoNP/poly, due to Theorem 13.1.

Theorem 13.2 ([BDFH09]). *Let* Π *be a compositional parameterized problem, and such that* P*, the underlying classical problem, is* NP*-complete. If* Π *has a polynomial kernel, then* P *also has a distillation algorithm.*

Algorithms that compose multiple instances of a problem into one (respecting a number of properties) have been developed for various problems ([BDFH09, CFM11, DLS09]). As an illustration, we provide a rather simple example of composition algorithm.

k-PATH
Input: A graph G and a non-negative integer k.
Parameter: k
Question: Does G have a path of length k?

This problem is shown to be in FPT with running time $2^{O(k)}n^{O(1)}$ via the technique of color coding. The algorithm randomly "colors" the vertex set with k colors, that is, it picks uniformly at random a function $c : V \to [k]$ and hopes that the vertices of the k-path we are after are colored distinctly. That is, the function restricted to at least one witness k-length path is injective. It then uses dynamic programming to actually identify the colorful path from the colored graph. The algorithm is derandomized using a family of hash functions, and the reader is referred to [AYZ95] for more details.

We present now a composition for k-PATH. Suppose the input instances are:

$$(G_1, k), \ldots, (G_t, k)$$

A composition algorithm is trivial, it simply provides the disjoint union G' of all the graphs as the output. Notice that this consumes linear time, and leaves the parameter unchanged. Clearly, G' has a path of length at most k if, and only if, there exists i, $1 \leqslant i \leqslant t$, such that G_i has a path of length at most k.

Informally speaking, the disjoint union strategy works whenever we look for connected structures that are required to satisfy properties that are functions of the vertices participating in the structure (and are independent of rest of the graph). Typically, the fact that the property is "local" helps the forward direction and the connectivity aids the reverse direction of the argument. Examples include k-CYCLE, k-TREE or k-OUT TREE in a directed graph. Thus these problems, though FPT, do not admit polynomial kernels unless NP $\subseteq$ CoNP/poly.

13.2 Polynomial Parameter Transformations

In this section, we introduce the notion of transformations, which will allow us to prove results for problems that do not obviously have compositions.

We begin by describing what we mean by a polynomial parameter transformation [BTY09, DLS09].

Definition 13.3 (Polynomial parameter transformation). *Let* P *and* Q *be parameterized problems. We say that* P *is polynomial parameter reducible to* Q*, written* $P \preccurlyeq_{ppt} Q$*, if there exists a polynomial time computable function* $f : \{0,1\}^* \longrightarrow \{0,1\}^*$*, and a polynomial* $p : \mathbb{N} \to \mathbb{N}$*, and for all* $x \in \{0,1\}^*$ *and* $k \in \mathbb{N}$ *if* $f(x,k) = (y,l)$*, then the following hold:*

1. $(x,k) \in P$*, if and only if* $(y,l) \in Q$*, and*
2. $l \leqslant p(k)$

We call f *a polynomial parameter transformation from* P *to* Q*.*

Notice the differences with the notion of fixed-parameter reductions introduced by Downey and Fellows (see [DF95c, DF95a, DF95b]). In general fixed-parameter reductions, one is allowed $f(k)p(|x|)$ time for an input instance (x,k) where f is an arbitrary function and p is a polynomial function, and the resulting parameter is allowed to be an arbitrary function of the original parameter. Here the running time allowed is only a polynomial in $|x|$ and k and the resulting parameter value is only a polynomial function of the original parameter.

Polynomial parameter transformations are used to show non-existence of polynomial sized kernels using the following theorem.

Theorem 13.3 ([BTY09]). *Let* (A,k) *and* (B,l) *be parameterized problems such that* A *is* NP*-complete, and* $B \in NP$*. Suppose that there is a polynomial parameter transformation from* A *to* B*. Then, if* B *has a polynomial kernel, then* A *has a polynomial kernel.*

As an easy corollary of Theorem 13.3, note that whenever (A,k) and (B,l) are parameterized problems (such that A is NP-complete, and $B \in NP$) and $A \preccurlyeq_{ppt} B$, if A is compositional, then B does not have a polynomial kernel unless $NP \subseteq CoNP/poly$. A natural strategy to prove that a problem A is unlikely to admit a polynomial kernel, is to reduce some NP-complete problem B, for which we have a composition, to A using a polynomial parameter transformation.

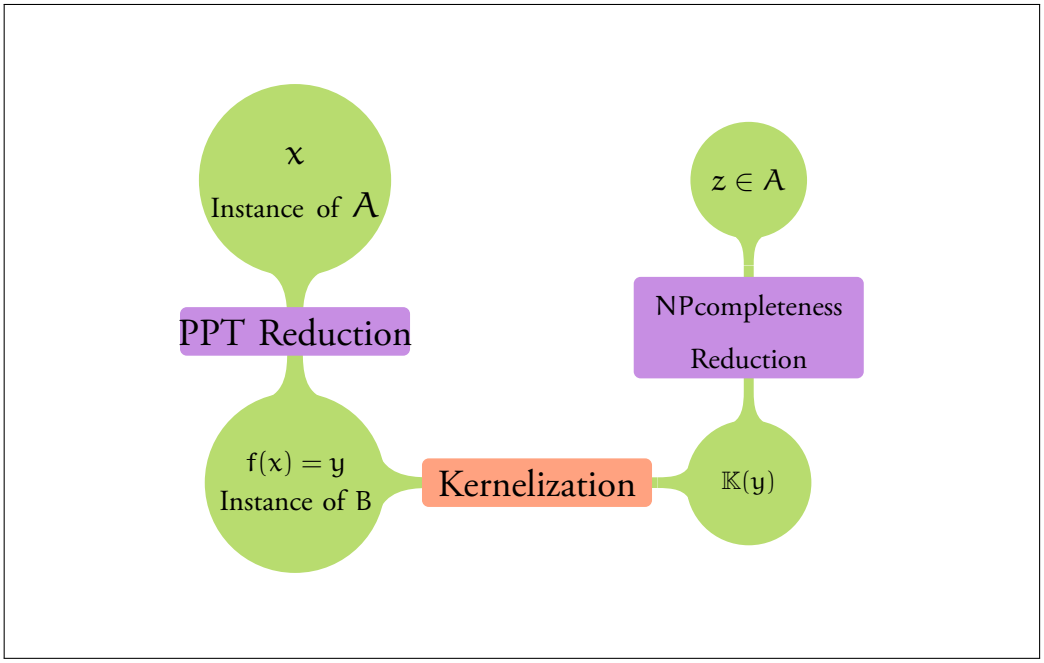


Figure 13.1: Illustrating the utility of PPT Reductions in Kernelization: A PPT reduction from A to B allows us to obtain a kernel for A whenever B admits a kernelization algorithm.

13.2.1 (Vertex) Disjoint Cycles

Consider the following two parameterized problems.

Vertex Disjoint Cycles
Input: Undirected graph $G = (V, E)$ and a non-negative integer k.
Parameter: k
Question: Does G contain at least k vertex-disjoint cycles?

Edge Disjoint Cycles
Input: Undirected graph $G = (V, E)$ and a non-negative integer k.
Parameter: k
Question: Does G contain at least k edge-disjoint cycles?

The problem of Vertex Disjoint Cycles is strongly related to the Feedback Vertex Set (FVS) problem, wherein the question is whether there exist k vertices whose deletion makes the graph acyclic (usually studied with k as the parameter). Clearly, if a graph has more than k vertex disjoint cycles, then it cannot have a FVS of size k or less, as any FVS has to pick at least one vertex from every cycle. If there are at most k vertex disjoint cycles, the implications are less immediate, but an upper bound of $O(k \log k)$ on the size of the optimal FVS is known, due to a result by Erdős and Pósa [EP65a]. For the FVS problem, there is a kernel of size $O(k^2)$ ([Tho10]) by Thomassé, who improved upon a kernel of size $O(k^3)$ ([Bod07]). The Edge Disjoint Cycles problem has a polynomial kernel (see [BTY09]).

In contrast, it is shown in [BTY09] that the Vertex Disjoint Cycles problem does not admit a polynomial kernel through a polynomial parameter transformation from an intermediate problem called Disjoint Factors. For the rest of this discussion, when we say disjoint cycles we mean *vertex* disjoint cycles.

We begin by describing the Disjoint Factors problem. For a positive integer k, we use L_k to denote the alphabet $\{1, 2, \ldots, k\}$, and L_k^* to denote the set of all words over L_k. For $x \in L_k$, an x *factor* of a word $w_1 \cdots w_r \in L_k^*$ is a substring $w_i \cdots w_j; 1 \leqslant i < j \leqslant r$ such that $w_i = w_j = x$. A word $w \in L_k^*$ is said to have the *disjoint factor property* if there exist disjoint (non-overlapping) factors $f_1, f_2, \ldots, f_k$ in w, where for $1 \leqslant i \leqslant k$, f_i is an i factor of w.

Disjoint Factors is the following problem:

Disjoint Factors
Input: A word $w \in L_k^*$
Parameter: $k \geqslant 1$
Question: Does w have the Disjoint Factors property?

In [BTY09], a composition algorithm is described for the Disjoint Factors problem, implying that polynomial kernelization is infeasible:

Theorem 13.4 ([BTY09]). *Disjoint Factors parameterized by* k *does not admit a polynomial kernel unless NP* $\subseteq$ *CoNP/poly.*

We are now ready to describe a polynomial parameter transformation from Disjoint Factors to Vertex Disjoint Cycles.

Given an input (W, k) of Disjoint Factors, with $W = w_1 \cdots w_n$, a word in $\{0,1\}^*$, we build a graph $G = (V, E)$ as follows. First, we take n vertices $v_1, \ldots, v_n$, and edges $\{v_i, v_{i+1}\}$ for $1 \leqslant i < n$, i.e., these vertices form a path of length n. Let P denote this subgraph of G. Then, for each $i \in L_k$, we add a vertex x_i, and make x_i incident to each vertex v_j with $w_j = i$, i.e., to each vertex representing the letter i. See Figure 13.2 for an illustration.

We next claim that G has k disjoint cycles if and only if (W, k) has the requested k disjoint factors. Suppose G has k disjoint cycles $c_1, \ldots, c_k$. As P is a path, each of these cycles must contain at least one vertex not on P, i.e., of the form x_j, and hence each of these cycles contains exactly one vertex x_j (as the cycles are vertex disjoint, and there are k vertices available outside P). For $1 \leqslant j \leqslant k$, the cycle c_j thus consists of x_j and a subpath of P. This subpath must start and end with a vertex incident to x_j. These both represent letters in W equal to j. Let F_j be the factor of W corresponding to the vertices on P in c_j. Now, $F_1, \ldots, F_k$ are disjoint factors, each of length at least two (as the cycles have length at least three), and F_j starts and ends with j, for all j, $1 \leqslant j \leqslant k$.

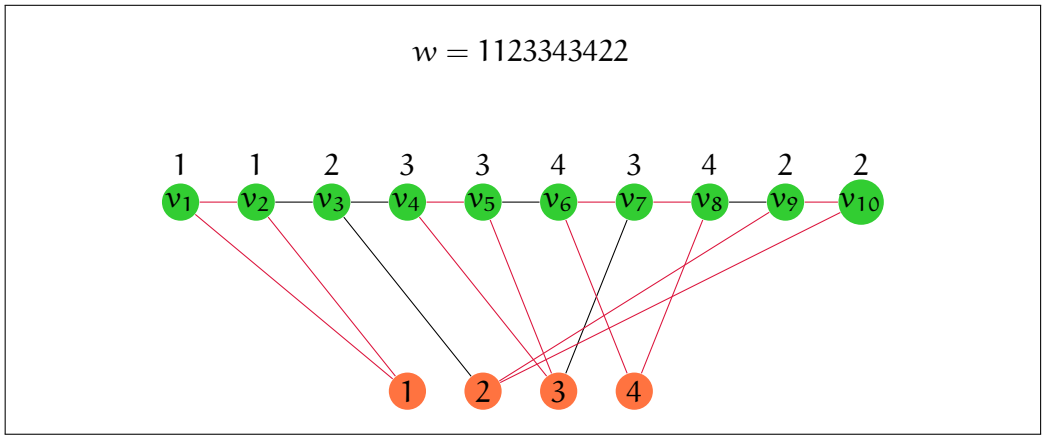


Figure 13.2: Disjoint Factors $\preccurlyeq_{ppt}$ Disjoint Cycles

Conversely, if we have disjoint factors $F_1, \ldots, F_k$ with the properties as in the Disjoint Factors problem, we build k vertex disjoint cycles as follows: for each j, $1 \leqslant j \leqslant k$, take the cycle consisting of x_j and the vertices corresponding to factor F_j. Thus we have shown:

Theorem 13.5 ([BTY09]). *Disjoint Factors does not admit a polynomial kernel unless NP $\subseteq$ CoNP/poly.*

13.2.2 Turing Kernelization

Although a parameterized problem may not necessarily admit a polynomial kernel, it may evidently admit many of them, with the property that the instance is in the language if and only if at least one of the kernels corresponds to an instance that is in the language. It is established in [FFL+09] that k-Leaf Out-Branching admits n independent kernels of size $O(k^3)$. It was not a kernel in the usual "many to one" sense, though it was kernel in the "one to many" sense. This brings us to the notion of Turing kernelization. In order to define this we first define the notion of t-*oracle*.

Definition 13.4. *A* t*-oracle for a parameterized problem* Π *is an oracle that takes as input* (I, k) *with* $|I| \leqslant t$, $k \leqslant t$ *and decides whether* $(I, k) \in \Pi$ *in constant time.*

Definition 13.5. *A parameterized problem* Π *is said to have* $g(k)$-sized turning kernel *if there is an algorithm which given an input* (I, k) *together with a* $g(k)$*-oracle for* Π *decides whether* $(I, k) \in \Pi$ *in time polynomial in* $|I|$ *and* k. $|x'|, k' <= g(k)$.

Observe that the well known notion of kernel or many to one kernel is a special case of turing kernelization. In particular, many to one kernels are equivalent to turing kernels where the kernelization algorithm is only allowed to make one oracle call and must return the same answer as the oracle.

I don't believe it.
Prove it to me and I still won't believe it.

Douglas Adams

Relax Luther, it's much worse than you think.

Mission Impossible

In this chapter, we demonstrate kernelization lower bounds for two packing problems: Vertex Disjoint θ_c-Packing and Vertex Disjoint Odd Cycle Packing. In particular, these results imply that these problems do not have polynomial kernels unless NP $\subseteq$ CoNP/poly.

First, we study the Vertex Disjoint θ_c-Packing problem, which is a generalization of the Disjoint Cycles problem. In the parameterized Disjoint Cycles problem the input consists of an undirected graph G and a positive integer parameter k, and the question is whether G contains at least k vertex-disjoint cycles. This problem is NP-complete [GJ79], and it has long been known to have a kernel of size *exponential* in k, from an early result due to Bodlaender [Bod94] and a folklore theorem of parameterized complexity [DF99]. It was open for a long time whether this problem has a *polynomial* kernel. Quite recently, Bodlaender et al. [BTY09] showed that Disjoint Cycles does *not* have a polynomial kernel unless NP $\subseteq$ CoNP/poly, which is widely believed to be unlikely.

Recall that for a positive integer c, we define θ_c to be the multigraph which consists of two vertices and c parallel edges between them. For each positive integer c, we define the Vertex Disjoint θ_c-Packing problem as follows:

Vertex Disjoint θ_c-Packing
Input: An undirected graph G and a positive integer k.
Parameter: k
Question: Does G contain at least k vertex-disjoint subgraphs $H_1, H_2, \ldots, H_k$ such that for each $1 \leqslant i \leqslant k$, (1) H_i contains a θ_c as a *minor*, and, (2) no proper subgraph of H_i contains θ_c as a minor?

Note that for $c = 1$, the Vertex Disjoint θ_c-Packing problem is equivalent to asking whether there exists a *matching* of size (number of edges) at least k in G, and can be solved in polynomial time by finding a maximum matching in G [MV80]. For $c = 2$, this problem is equivalent to the Disjoint Cycles problem, and hence is NP-complete [GJ79] and has no polynomial kernel unless NP $\subseteq$ CoNP/poly [BTY09].

In this chapter, we show that for each fixed $c \geqslant 3$ the problem remains NP-complete and has no polynomial kernel unless NP $\subseteq$ CoNP/poly.

The second problem that we explore is the Vertex Disjoint Odd Cycle Packing problem. Cycles of odd length are interesting for various reasons. One of them is that they provide a natural analog of the Feedback Vertex Set problem. Recall that the Feedback Vertex Set question asks, for a graph G, a smallest subset of vertices S such that the graph $G \setminus S$ has no cycles. Finding a feedback vertex set is interesting because graphs that do not have cycles (forests) are an interesting (and from the algorithmic point of view, a "tractable") graph class. Similarly, the class of bipartite graphs (graphs whose vertex set can be partitioned into two independent parts) is another interesting graph class, and relevant to us because they are exactly the graphs that do not have cycles of odd length.

It is natural to ask both the covering and packing question for odd cycles. The *covering* question corresponds to Odd Cycle Traversal, and has been shown FPT [RSV04, LSS09], while the question of whether it admits a polynomial kernel is regarded an interesting open problem in the kernelization literature. The complementary packing problem has two versions again:

♢ Are there at least k vertex disjoint cycles of odd length?
♢ Are there at least k edge disjoint cycles of odd length?

The vertex disjoint version has been studied for the special case of packing triangles, for which a cubic kernel has been obtained [FHR+05]. In this chapter, we show that the general question (for both versions), is not likely to admit a polynomial kernel.

14.1 Vertex-Disjoint $\{\theta_c\}$-packing: No Polynomial Kernels

In this section, we show the lower bound for Vertex Disjoint θ_c-Packing:

Theorem 14.1. *For any fixed integer* $c \geqslant 3$*, the Vertex Disjoint* θ_c*-Packing problem is* NP-complete*, and has no kernel of size bounded by* k^d*, for any fixed constant* d*, unless NP* $\subseteq$ *CoNP/poly.*

Proof. This proof is by a reduction from Disjoint Factors. We begin by noting that the derived classical problem corresponding to Disjoint Factors is NP-complete [BTY09]. Also, the derived classical problem corresponding to Vertex Disjoint θ_c-Packing is easily seen to be in NP. Further, Disjoint Factors is known to admit a composition algorithm [BTY09], and hence does not have polynomial kernels unless NP $\subseteq$ CoNP/poly. We now proceed to describe the reduction.

We begin by recalling the definition of Disjoint Factors, the problem we will reduce from towards the proof the theorem. For a positive integer k, we use L_k to denote the alphabet $\{1, 2, \ldots, k\}$, and L_k^* to denote the set of all words over L_k. For $x \in L_k$, an x *factor* of a word $w_1 \cdots w_r \in L_k^*$ is a substring $w_i \cdots w_j; 1 \leqslant i < j \leqslant r$ such that $w_i = w_j = x$. A word $w \in L_k^*$ is said to have the *disjoint factor property* if there exist disjoint (non-overlapping) factors $f_1, f_2, \ldots, f_k$ in w, where for $1 \leqslant i \leqslant k$, f_i is an i factor of w.

Disjoint Factors is the following problem:

Disjoint Factors
Input: A word $w \in L_k^*$
Parameter: $k \geqslant 1$
Question: Does w have the Disjoint Factors property?

Note that to prove the theorem, it suffices to demonstrate a polynomial parameter transformation from Disjoint Factors to Vertex Disjoint θ_c-Packing (this follows from Theorems 13.3 and 13.4).

Given an instance of Disjoint Factors consisting of a positive integer k and a word $w = x_1 x_2 \ldots x_n \in L_k^*$, we construct an instance (G, k) of Vertex Disjoint θ_c-Packing as follows (See Figure 14.1.). We create vertices $v_1, v_2, \ldots, v_n$ and add the edges $(v_i, v_{i+1}); 1 \leqslant i < n$. That is, we create a path (called the *spine* of G) that mimics w. For $1 \leqslant i \leqslant n$, let $L_k(x_i)$ be the letter that corresponds to vertex v_i on this spine. We then introduce a "dominating" vertex for each letter in the alphabet. That is, for $1 \leqslant i \leqslant k$, we create a vertex D_i and make D_i adjacent to all the vertices $v_j; L_k(v_j) = i$. We use $D(v_j)$ to denote the dominating vertex corresponding to

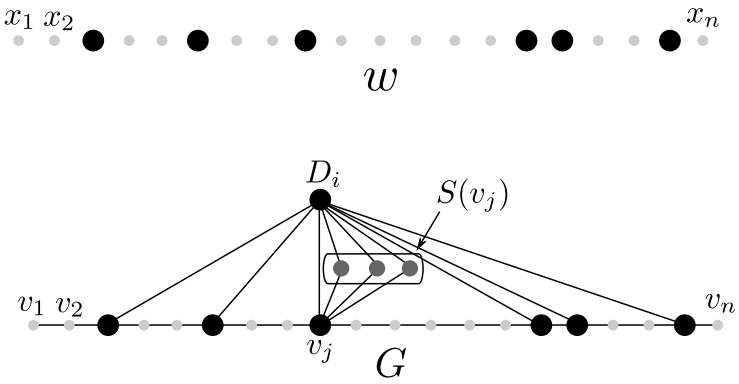


Figure 14.1: Reduction from Disjoint Factors to Vertex Disjoint θ_c-Packing for $c = 5$. The big black vertices all correspond to the letter i.

vertex v_j. Finally, for each $v_j; 1 \leqslant j \leqslant n$, we introduce a set $S(v_j)$ of $(c-2)$ new "special" vertices, and make each of these new vertices adjacent to v_j and to $D(v_j)$.

In the reduced instance (G, k), G is the graph constructed as above, and k has the same value as for the input Disjoint Factors instance. It is easy to see that this construction can be performed in polynomial time. Now, Vertex Disjoint θ_c-Packing is evidently in NP, and Disjoint Factors is NP-complete [BTY09, Theorem 5]. From Lemma 14.1, the reduction described above is both a Karp reduction and a polynomial parameter transofrmation, and the theorem follows. □

Lemma 14.1. *Let* $(w = x_1x_2\ldots x_n, k)$ *be an instance of Disjoint Factors, and let* (G, k) *be an instance of Vertex Disjoint* θ_c*-Packing constructed from* (w, k) *as described in the proof of Theorem 14.1. Then* (w, k) *is a YES instance of Disjoint Factors if and only if* (G, k) *is a YES instance of Vertex Disjoint* θ_c*-Packing.*

Proof. Let $\ell = x_i, \cdots x_j = \ell$ be an ℓ-factor of w for some $1 \leqslant \ell \leqslant k$. The subgraph H of G induced by the vertex set $\{D_\ell, v_i, \ldots, v_j\} \cup S(v_i)$ forms a minimal θ_c minor model (See Figure 14.2.). Indeed, one can pick exactly c vertex-disjoint paths between D_ℓ and v_i in H, consisting of: the edge (D_ℓ, v_i), the $(c-2)$ paths $\langle D_\ell, y, v_i\rangle; y \in S(v_i)$ each of length two, and the path that consists of the edge (D_ℓ, v_j) followed by the path from v_j to v_i along the spine of G. Note that this θ_c minor model intersects the spine only at vertices corresponding to the ℓ-factor $x_i \ldots x_j$ of w, and that the only other vertices it contains are the dominating vertex corresponding to ℓ and special vertices corresponding to v_i. It follows that if w

has the disjoint factor property, then G contains k vertex-disjoint minor-models of θ_c, each constructed from one of the disjoint factors in the manner specified above.

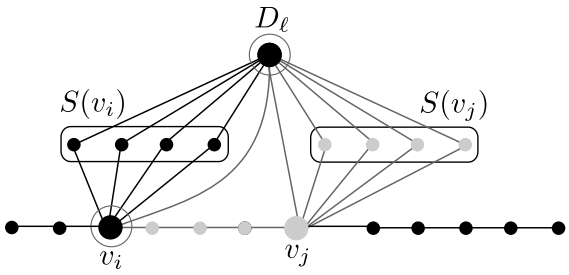


Figure 14.2: Hardness reduction: The forward direction.

Now suppose G contains k vertex-disjoint minimal minor-models of θ_c. If we remove all the dominating vertices $D_i; 1 \leqslant i \leqslant k$ from G, then the remaining graph does not contain any cycle. Therefore any minor model of θ_c in G must contain at least one of the dominating vertices. Since there are k vertex-disjoint minimal minor-models in G, each of these must contain exactly one of the dominating vertices. For letter $\ell; 1 \leqslant \ell \leqslant k$, let M_ℓ denote the minimal θ_c minor model of G that contains D_ℓ. Now, if $P_\ell = v_i \cdots v_j$ is a subpath of the spine such that $L_k(v_i) = L_k(v_j) = \ell$, then $x_i \cdots x_j$ is an ℓ-factor of w. If each $M_\ell; 1 \leqslant \ell \leqslant k$ contains such a P_ℓ, then since the M_ℓs are vertex-disjoint, w has the disjoint factors property, and the claim is proved. We now argue that each $M_\ell; 1 \leqslant \ell \leqslant k$ must contain such a P_ℓ. So let $1 \leqslant \ell \leqslant k$.

First we show that M_ℓ must contain at least two vertices v_i, v_j from the spine such that $L_k(v_i) = \ell = L_k(v_j)$. If M_ℓ contains no such vertex, then in M_ℓ, either D_ℓ is isolated, or all of D_ℓ's neighbors (which are all special vertices) are pendant vertices. By Observation 7.2, neither of this can happen, and so M_ℓ contains at least one such vertex. Suppose M_ℓ contains exactly one such vertex, say v_i. Then, since any vertex in M_ℓ has degree at least 2 (Observation 7.2), and since D_ℓ is the only "dominating" vertex (see the construction) in M_ℓ, the only special vertices that are present in M_ℓ are those that are adjacent to v_i. If M_ℓ contains no neighbor of v_i that lies on the spine, then M_ℓ is the subgraph of G induced on D_ℓ, v_i, and some subset of $S(v_i)$. It is clear that such a subgraph cannot contain a θ_c as a minor (Figure 14.3), and so M_ℓ contains at least one neighbor of v_i that lies on the spine of G.

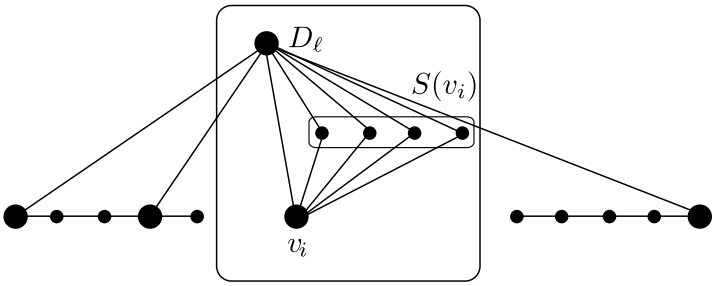


Figure 14.3: Hardness reduction: One case that is ruled out in the reverse direction.

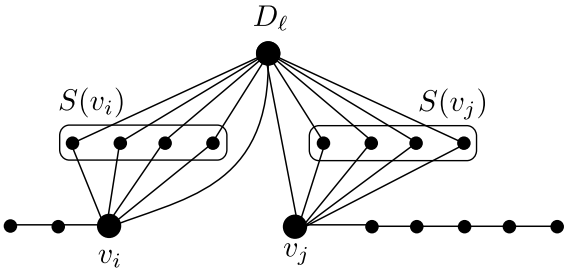


Figure 14.4: Hardness reduction: The second case that is ruled out in the reverse direction.

Let w be such a neighbor. From the assumption, $L_k(w) \neq \ell$. Since w has degree at least 2 in M_i, and none of the vertices in $S(w)$ can be in M_i (since they would have degree one in M_i), M_i contains a neighbor of w other than v that lies on the spine in G. Repeating this argument, we reach a vertex on the spine that can have only one neighbor in M_i, a contradiction. Thus M_ℓ contains at least two vertices v_i, v_j such that $L_k(v_i) = L_k(v_j) = \ell$. If M_ℓ does not contain the subpath of the spine from v_i to v_j, then M_ℓ is the subgraph of G induced on D_ℓ, v_i, v_j, and some subset of $S(v_i) \cup S(v_j)$ (Figure 14.4). In particular, by Observation 7.2 and a similar argument as above, no other vertex on the spine is in M_ℓ. D_ℓ is then a cut vertex for this subgraph, which contradicts Observation 7.2. It follows that M_ℓ must contain a subpath of the spine from v_i to v_j. □

14.2 Odd Cycle Packing: No Polynomial Kernels

In this section, we show kernel lower bounds for Vertex Disjoint Odd Cycle Packing and Edge Disjoint Odd Cycle Packing.

Theorem 14.2. *The Vertex Disjoint Odd Cycle Packing and Edge Disjoint Odd Cycle Packing problems are* NP-complete*, and have no polynomial kernels, unless NP* $\subseteq$ *CoNP/poly.*

Proof. We begin again by noting that the derived classical problem corresponding to Disjoint Factors is NP-complete [BTY09, Theorem 5]. It is clear that the derived classical problem corresponding to Vertex Disjoint Odd Cycle Packing is in NP.

We refer the reader to the beginning of the proof of Theorem 14.1 in the previous section or Section 13.2.1 in Chapter 13 for the definition of disjoint factors.

We will demonstrate polynomial parameter transformations from Disjoint Factors to Vertex Disjoint Odd Cycle Packing and Edge Disjoint Odd Cycle Packing. This fact that this suffices to prove the theorem follows from Theorems 13.3 and 13.4.

The same construction works for both the Vertex Disjoint Odd Cycle Packing and Edge Disjoint Odd Cycle Packing problems. The proofs of equivalence are also nearly identical, as we will see shortly.

Given an input (W, k) of Disjoint Factors, with $W = w_1 \cdots w_n$, a word in $\{0,1\}^*$, we build a graph $G = (V, E)$ as follows. First, we have n vertices $v_1, \ldots, v_n$. Let P denote this subgraph of G. Then, for each $i \in L_k$, we add a vertex x_i, and make x_i incident to each vertex v_j with $w_j = i$, i.e., to each vertex representing the letter i.

We repeat this construction again, essentially duplicating the graph we have built so far, as follows. We have n more vertices $u_1, \ldots, u_n$. Then, for each $i \in L_k$, we add a vertex y_i, and make y_i incident to each vertex u_j with $w_j = i$. This time, however, we add the edges $\{u_i, u_{i+1}\}$ for $1 \leqslant i < n$ — so that these vertices form a path of length n. Let Q denote this subgraph of G.

We now subdivide all the edges on the path P and let r_i denote the vertex between v_i and v_{i+1}. Similarly, we subdivide all the edges on the path Q and let s_i denote the vertex between u_i and u_{i+1}. For $1 \leqslant i \leqslant (n-1)$, we add the edges (v_i, s_i), and finally add edges (x_j, y_j) for $1 \leqslant j \leqslant k$.

This completes the construction, see Figure 14.5 for an illustration.

$w = 1122312 \leftarrow$ The factor indicated below.

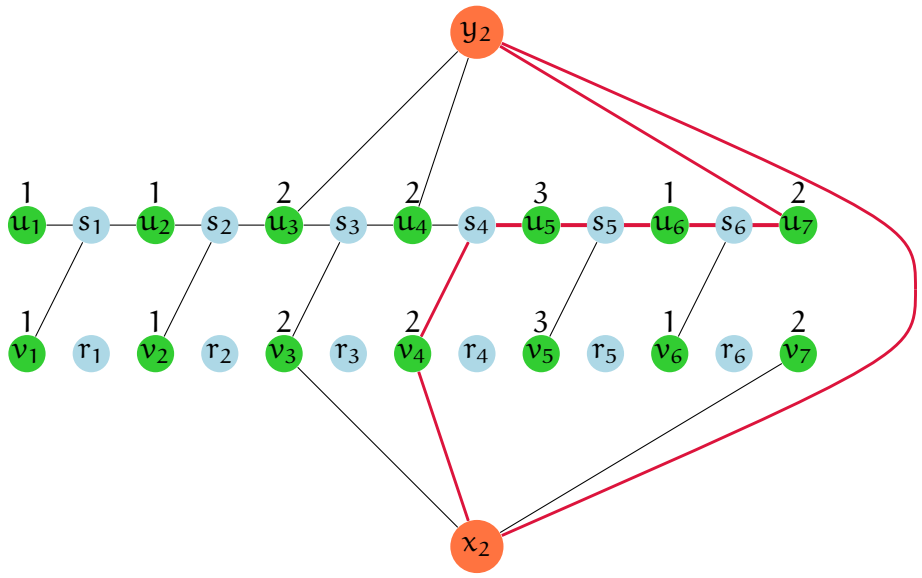


Figure 14.5: Disjoint Factors $\preccurlyeq_{ppt}$ Odd Vertex Disjoint Cycle Packing

We now argue the correctness of the reduction. In the forward direction, we show that if w contains k disjoint factors, one for each letter in L, then G contains k mutually vertex disjoint cycles, each of odd length. Notice that since any collection of vertex disjoint cycles is also edge disjoint, the claim demonstrates the correctness of the forward direction for both Vertex Disjoint Odd Cycle Packing and Edge Disjoint Odd Cycle Packing.

Claim 14.1. *If w contains k disjoint factors, one for each letter in L, then G contains k mutually vertex disjoint cycles, each of odd length.*

For a fixed letter $i \in L$, we first show that an i-factor corresponds to a cycle of odd length in G. Let the i factor be $w_p \cdots w_q$. Indeed, consider the following cycle:

$$x_i — v_p — s_p — u_{p+1} — \cdots — s_q — u_q — y_i — x_i.$$

This cycle comprises of a sub path on Q and four other edges: (x_i, v_p), (v_p, s_p), (u_q, y_i) and (y_i, x_i). Thus, if we show that the sub path on Q is of odd length, it follows that the entire cycle has odd length. Notice that any sub path of Q that corresponds to a factor (that is, the end points of the path are vertices i and j such that $w_i = w_j$) has even length, since the path is subdivided. The

sub path of Q that belongs to the cycle is easily seen to be a sub path corresponding to a factor but short of one edge (the edge (u_p, s_p)). Therefore, this sub path is of odd length.

We now argue that two cycles corresponding to different factors do not have any vertices in common. Given the way the cycles are chosen above, it is clear that the vertices x_i, y_i belong to exactly one of the chosen cycles. The vertex v_p is chosen on the basis of the fact that w_p is the starting point of the i-factor, thus if v_p belongs to two cycles, that means that the corresponding factors in w are not disjoint. Similarly, the sub paths on the path Q correspond to the sub word $w_{p+1} \dots w_q$. Thus, it is clear that if the sub paths corresponding to two different cycles intersect, then the corresponding factors are not disjoint in w. It is easily checked that the vertices s_p belongs to exactly one of the chosen cycles: By construction, a neighbor of s_p from P is always picked in the cycle that s_p belongs to, and s_p has an unique neighbor (namely, v_p) on P. Since v_p belongs to exactly one of the chosen cycles, so does s_p. This concludes the proof of Claim 14.1.

In the reverse direction, we show that if G contains k mutually *edge* disjoint cycles, each of odd length, then w contains k disjoint factors, one for each letter in L. Notice that no assumptions are made about whether the cycles share vertices, so in particular the proof holds when all the cycles are vertex disjoint. Thus the following claim establishes the reverse direction of the reduction for both Vertex Disjoint Odd Cycle Packing and Edge Disjoint Odd Cycle Packing.

Claim 14.2. *If* G *contains* k *mutually edge disjoint cycles, each of odd length, then* w *contains* k *disjoint factors, one for each letter in* L.

Let $C_1, \dots, C_k$ denote the k mutually edge disjoint cycles, each of odd length. We first demonstrate that each cycle C_j must involve x_p and y_p for some $1 \leqslant p \leqslant k$.

Notice that the graph G without the edges $\{(x_p, y_p) \mid 1 \leqslant p \leqslant k\}$ is bipartite. This is demonstrated by the following two-coloring of $V(G)$ (see Figure 14.6): vertices x_p and y_p are colored red, their neighbors on the paths P and Q are colored blue, and the remaining "subdivison vertices" are colored red. It is easy to to verify that this is indeed a two-coloring of G in the absence of the edges

$$E_C := \{(x_p, y_p) \mid 1 \leqslant p \leqslant k\}.$$

Therefore, it is clear that any odd cycle C_j must involve one of the edges in E_C. As there are only k edges in E_C and the cycles $C_1, \dots, C_k$ are edge-disjoint, it is clear

that each C_i contains exactly one edge of the form (x_p, y_p). This shows that each cycle C_j must involve x_p and y_p for some $1 \leqslant p \leqslant k$.

The intersection of the cycle that contains $\{x_p, y_p\}$ with the path Q determines a p-factor by construction, and the fact that the cycles are disjoint imply that the factors thus obtained are also disjoint. This concludes the proof of Claim 14.2.

The proof of the theorem follows from Claims 14.1 and 14.2 above. □

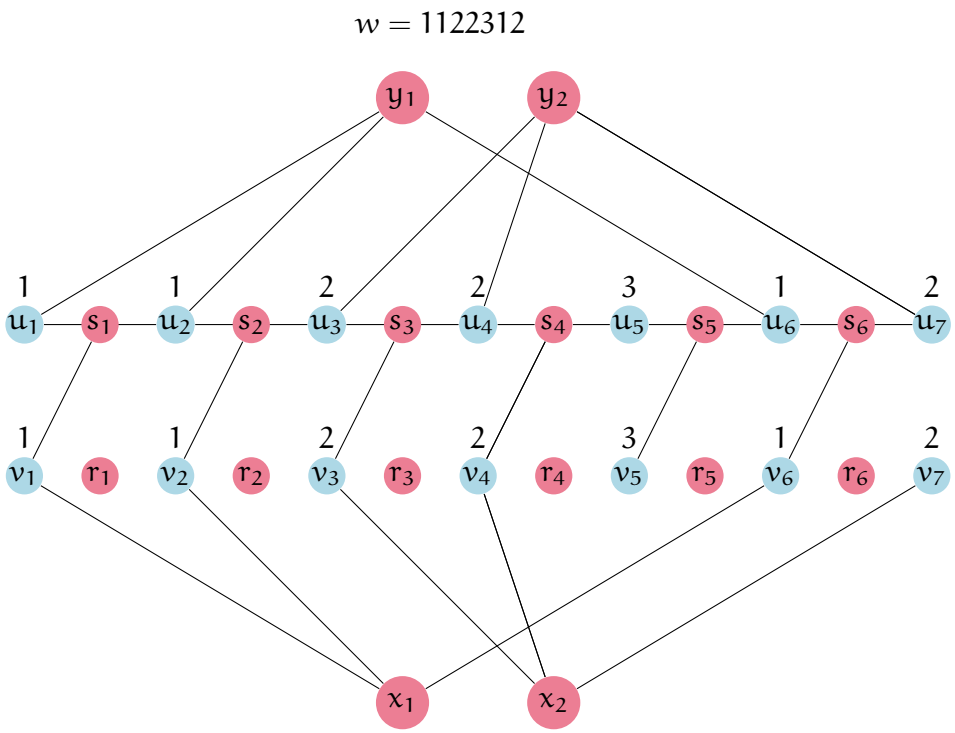


Figure 14.6: A two-coloring of the reduced instance without the edges (x_p, y_p), $1 \leqslant p \leqslant k$.

Just when you think it can't get any worse, it can.
And just when you think it can't get any better, it can.

Nicholas Sparks

"Would you tell me, please, which way I ought to go from here?"
"That depends a good deal on where you want to get to."
"I don't much care where –"
"Then it doesn't matter which way you go."

Lewis Carroll

The Graph Motif problem concerns a vertex-colored undirected graph G and a *multiset* M of colors. We are asked whether there is a set S of vertices of G such that the subgraph induced on S is connected and there is a color-preserving bijective mapping from S to M. That is, the problem is to find if there is a connected subgraph H of G such that the multiset of colors of H is identical to M.

The Graph Motif problem has immense utility in bioinformatics, especially in the context of metabolic network analysis (eg. motif search in metabolic reaction graphs with vertices representing reactions and edges connecting successive reactions) [BHK+09, LFS06] . The problem is NP-complete even in very restricted cases, such as when G is a tree with maximum degree 3, or when G is a bipartite graph with maximum degree 4 and M is a multiset over just two colors. When parameterized by |M|, the problem is FPT, and it is W[2]-hard when parameterized by the *number of colors* in M, even when G is a tree [FFHV07].

The Colorful Motif problem is a simpler version of the Graph Motif problem, where M is a set (and not a multiset). Even this problem is NP-hard on simple classes of graphs, such as when G is a tree with maximum degree 3 [FFHV07]. The problem is FPT on general graphs when parameterized by |M|, and the current fastest FPT algorithm, by Guillemot and Sikora, runs in $\mathcal{O}^*(2^{|M|})$ time and polynomial space [GS10].

We now turn to an example of a seemingly simple graph class on which the problem continues to be intractable. A graph is called a *comb graph* if (i) it is a tree, (ii) all vertices are of degree at most 3, (iii) all the vertices of degree 3 lie on a single simple

path. Cygan et al. [CPPW10] recently showed that the problem is NP-hard even on comb graphs. Further, they show that the parameterized version of the problem is unlikely to admit a polynomial kernel on forests (a disjoint union of trees) unless NP $\subseteq$ CoNP/poly, which would in turn imply an unlikely collapse of the Polynomial Hierarchy [CCHO05]. Hierarchy collapses to the third level.

We begin by pushing the borders of classical tractability. We show that while the problem is polynomial time on *caterpillars* (trees where the removal of all leaf vertices results in a *path*, called the *spine* of the caterpillar), it is NP-hard on *lobsters* (trees where the removal of all leaf vertices results in a *caterpillar*). In fact, we show that even more is true: the problem is NP-hard even on rooted trees of height two, or equivalently, on trees of diameter at most four. See Figure 15.1 for an overview of some of the graph classes mentioned.

Next, we extend the known results on the hardness of kernelization for this problem [CPPW10]. Specifically, we show that the lower bound can be tightened to hold for comb graphs as well. This is established by demonstrating a simple but unusual composition algorithm for the problem restricted to comb graphs. The composition is unusual because it is not the trivial composition (via disjoint union), and yet, it does not employ gadgets to encode the identity of the instances. To the best of our knowledge, this is an uncommon style of composition. On the positive side, we show a straightforward argument that yields polynomially many polynomial kernels for the problem on comb graphs, *a la* the many polynomial kernels obtained for k-Leaf Out Branching [FFL+09]. Again, to the best of our knowledge, this is one of the very few examples of many polynomial kernels for a parameterized problem for which polynomial kernelization is infeasible.

However, in our attempts to obtain many polynomial kernels for the more general case of trees, we learn that some natural approaches fail. Specifically, we show that two natural variants of the problem — Rooted Colorful Motif[1], Subset Colorful Motif[2] — do not admit polynomial kernels unless NP $\subseteq$ CoNP/poly. This shows, for instance, that the "guess" for obtaining many polynomial kernels has to be more than, or different from, a subset of vertices.

While we show that Colorful Motif is NP-hard on trees of diameter at most four, the kernelization complexity of the problem on this class of graphs is still open. However, we show that Colorful Motif is NP-hard on general graphs of diameter *three*,

[1] Does there exist a colorful subtree that contains a specific vertex?
[2] Does there exist a colorful subtree that contains a specific subset of vertices?

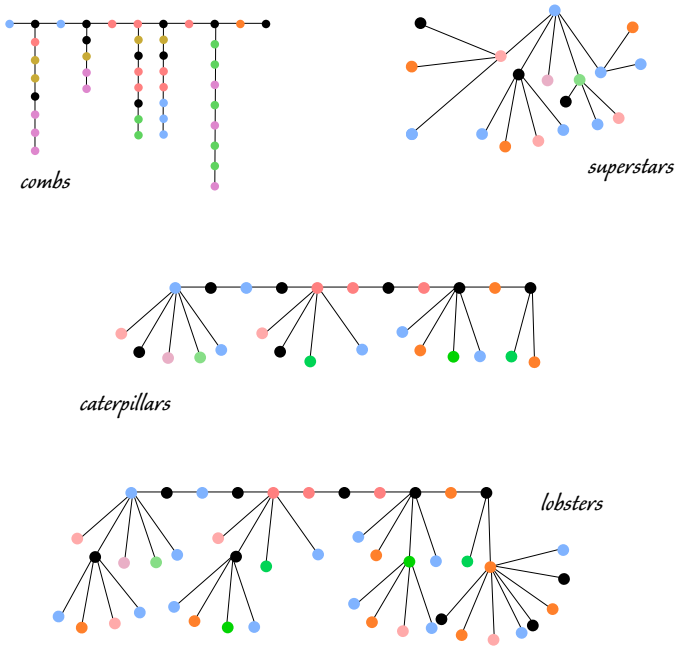


Figure 15.1: Graph classes: combs, superstars, caterpillars and lobsters.

and the same reduction also shows that polynomial kernels are unlikely for graphs of diameter three. We employ a reduction from Colorful Motif on general graphs. Using similar techniques, we show that the problem is NP-hard on general graphs of diameter *two*. This turns out to be useful to show the NP-hardness of Connected Dominating Set on the same class of graphs.

The results we obtain in this chapter contribute to the rapidly growing collection of problems for which polynomial kernels do not exist under reasonable complexity-theoretic assumptions. Given that many of our results pertain to very special graph classes, we hope these hardness results — which make these special problems available as starting points for further reductions — will be useful in settling the kernelization complexity of many other problems. In fact, we demonstrate the utility of the NP-completeness of Colorful Motif on graphs of diameter two, by showing that Connected Dominating Set on graphs of diameter two is NP-complete. The classical complexity of Connected Dominating Set on graphs of diameter two was hitherto unknown, although it was known to be NP-complete on graphs of diameter three, and trivial on graphs of diameter one. We summarize these results in Table 15.1.

Graph Class	Classical Complexity	Kernelization Complexity
Colorful Motifs		
Comb Graphs	NP-Complete [CPPW10]	No Poly Kernel [Lemma 15.4], Many Poly Kernels [Lemma 15.5]
Superstar Graphs	NP-Complete [Theorem 15.2]	-
Lobsters	NP-Complete [Corollary 15.6]	No Poly Kernel [Corollary 15.6]
Diameter Two Graphs	NP-Complete [Corollary 15.4]	-
Diameter Three Graphs	NP-Complete [Corollary 15.5]	No Poly Kernel [Corollary 15.5]
Rooted Colorful Motifs		
Trees	NP-Complete [Proposition 15.7]	No Poly Kernel [Proposition 15.8]
Subset Colorful Motifs		
Trees	NP-Complete [Proposition 15.9]	No Poly Kernel [Proposition 15.9]

Table 15.1: Summary of Results. Apart from the results tabulated, we also prove that Connected Dominating Set on Graphs of Diameter Two is NP-Complete.

15.1 Hardness On Superstar Graphs

We begin by observing that the Colorful Motif problem is NP-complete even on simple classes of graphs. It is already known that the problem is NP-complete on comb graphs [CPPW10]. In this section, we show that the problem is NP-complete on superstars — or equivalently, on rooted trees of height at most two. To begin with, consider Colorful Motif on paths. A solution corresponds to a colorful subpath, which, if it exists, we can find in polynomial time by guessing its end points. It is easy to see that this approach can be extended to a polynomial time algorithm for Colorful Motif on caterpillars, in which case we are looking for a colorful "subcaterpillar": We may guess the end points of the spine of the subcaterpillar, and for any given guess, if the subpath on the spine does not span the entire set of colors, we check if they can be found on the leaves.

Recall that a lobster is a tree where the removal of all leaf vertices results in a caterpillar. Lobsters are a natural generalization of caterpillars, and we show that the Colorful Motif problem is NP-hard on lobsters. In fact, we show that the problem is NP-hard on lobsters whose spine has just *one* vertex. Observe that every such graph is a superstar graph; thus we show that the problem is NP-hard on superstars. To show these hardness results, we reduce from the following variant of the well-known Set Cover problem:

Colorful Set Cover

Input: A finite universe U, a finite family $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$ of subsets of U such that there is no i, j for which $F_i \cup F_j = U$, and a function of representatives $C : \mathcal{F} \to U$ such that $C(F_i) \in F_i$.

Question: Does there exist $\mathcal{R} \subseteq \mathcal{F}$ such that $\bigcup_{S \in \mathcal{R}} S = U$ and C is injective on $\mathcal{R}$?

We will need the fact that Set Cover is NP-complete even when no two sets in the family span the universe. Formally, if the input to Set Cover is restricted to families that have the property that no two subsets in the family are such that their union is the universe, it remains NP-complete:

At-Least-Three Set Cover

Input: A finite universe U, a finite family $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$ of subsets of U, such that there is no i, j for which $F_i \cup F_j = U$.

Question: Does there exist $\mathcal{R} \subseteq \mathcal{F}$ such $\bigcup_{S \in \mathcal{R}} S = U$?

Proposition 15.1. At-Least-Three Set Cover *is* NP-complete.

Proof. The statement follows by an easy reduction from the well-known Set Cover problem which is among Karp's original list of 21 NP-complete problems [Kar72]:

Set Cover
Input: A finite universe U, a finite family $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$ of subsets of U, and a positive integer k.
Question: Does there exist $\mathcal{R} \subseteq F$, with $|\mathcal{R}| \leqslant k$, such that $\bigcup_{S \in \mathcal{R}} S = U$?

If $k = 1$, check if there exists i such that $F_i = U$. If this is the case, return a trivial YES-instance of At-Least-Three Set Cover. If $k > 1$, then in time $\binom{n}{2}$, examine if that there exists i, j for which $F_i \cup F_j = U$: if there is, return again a trivial YES-instance of At-Least-Three Set Cover, and if not, return the original instance. The correctness of this reduction is immediate.

□

Lemma 15.1. Colorful Set Cover *is NP-hard.*

Proof. We establish this by a reduction from At-Least-Three Set Cover. Let $(U, \mathcal{F} = \{F_1, \ldots, F_n\}, k)$ be an instance of At-Least-Three Set Cover. We construct an instance $(U', \mathcal{F}', C)$ of Colorful Set Cover as follows (see Figure 15.2 for an example). We construct $\mathcal{F}'$ essentially by making k "copies" of every family in $\mathcal{F}$. More precisely, let $X = \{x_1, x_2, \ldots, x_k\}$ be a set disjoint from U. For $1 \leqslant i \leqslant n, 1 \leqslant j \leqslant k$, let $F_{ij} = F_i \cup \{x_j\}$. For $1 \leqslant i \leqslant n, 1 \leqslant j \leqslant k$, we define $C(F_{ij}) = x_j$. We set $U' = U \cup X$ and $\mathcal{F}' = \{F_{ij} \mid 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant k\}$. Note that, as required, $U' = U \cup X$, and there are no indices i, j, p, q for which $F_{ij} \cup F_{pq} = U'$.

Now, suppose the given At-Least-Three Set Cover instance is a YES instance, and $R = \{F_{i_1}, F_{i_2}, \ldots, F_{i_t}\}$ be a set cover of size at most k in the At-Least-Three Set Cover instance. In the reduced instance, consider the set $R' = \{F_{i_1} \cup \{x_1\}, F_{i_2} \cup \{x_2\}, \ldots, F_{i_t} \cup \{x_t\}\}$. For $1 \leqslant j \leqslant k$, let $F'_j = \{F_{ij} \mid 1 \leqslant i \leqslant n\}$. If $t < k$, then arbitrarily pick an element from each $F'_j, t < j \leqslant k$ and add it to R'. Call

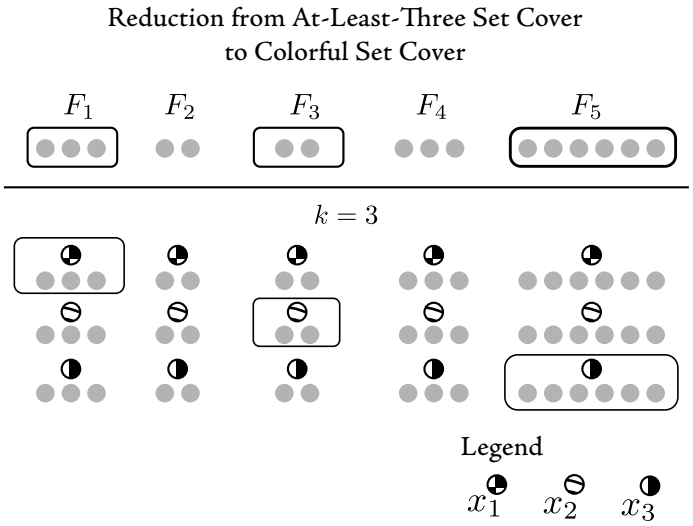


Figure 15.2: An Illustration of the Reduction

this new set R''. We claim that R'' is a solution for the instance Colorful Set Cover of $(U', \mathcal{F}', C)$. Consider $d \in X$. Since we have picked at least one element from each F'_j into our solution, d is covered by R''. Suppose $d \in U$. Let $d \in F_{i_j} \in R$ (note that such an F_{i_j} exists since R is a solution for the Set Cover instance). Then, $F_{i_j} \cup \{x_j\} \in R' \subseteq R''$, implying that d is covered by R'' and hence, the reduced instance is a YES instance of Colorful Set Cover.

Conversely, suppose the reduced instance is a YES instance of Colorful Set Cover, and let R'' be a colorful set cover. It is easy to see that R'' contains exactly one element from each F'_j. Let $R'' = \{F''_1, F''_2, \ldots, F''_k\}$. Define $R = \{F''_1 \backslash X, F''_2 \backslash X, \ldots, F''_k \backslash X\}$. Note that any $d \in U$ covered by some $F''_i \in R''$ will be covered by $F''_i \backslash X \in R$. Since $|R| \leqslant k$, it is indeed a solution for the At-Least-Three Set Cover instance.

□

Theorem 15.2. Colorful Motif *on superstar graphs is NP-hard.*

Proof. The proof is by reduction from Colorful Set Cover. Let $(U, \mathcal{F} = \{F_1, \ldots, F_n\}, C)$ be an instance of Colorful Set Cover. Without loss of generality, we assume that $\bigcup_{S \in \mathcal{F}} S = U$. We construct a graph T as follows. For each set $F_i \in \mathcal{F}$ and each element $x \in F_i$, add a vertex $x[i]$ to $V(T)$. Note that the same element in U may appear any number of times as a vertex, once for each time it appears in the sets F_i, $1 \leqslant i \leqslant n$. For each set F_i, add a new "set" vertex u_i and make it adjacent to all the vertices $x[i]$

that correspond to elements of F_i. Finally, add a "root" vertex r and make r adjacent to all the vertices u_i. This completes the description of T. More formally,

$$V(T) = \{x[i] \mid x \in F_i, 1 \leqslant i \leqslant n\} \cup \{u_1, \ldots, u_n\} \cup \{r\}$$
$$E(T) = \{(x[i], u_i) \mid x \in F_i, 1 \leqslant i \leqslant n\} \cup \{(u_i, r) \mid 1 \leqslant i \leqslant n\}$$

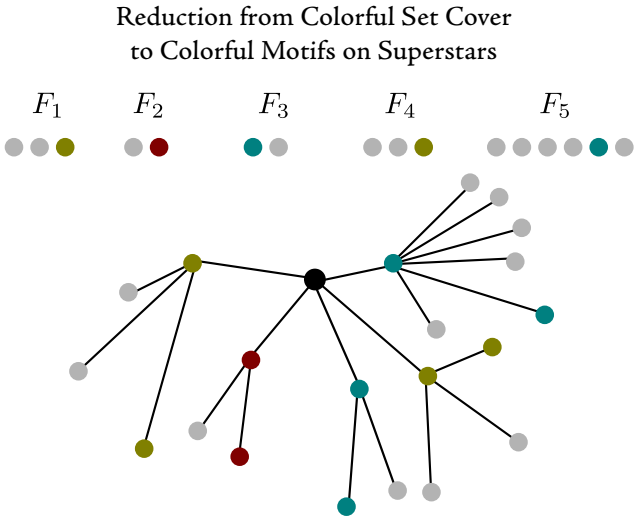


Figure 15.3: Reduction from Colorful Set Cover to Colorful Motif on superstar graphs. Note that the highlighted color in each F_i corresponds to $C(F_i)$

For every $x \in U$, we let c_x be a color in the set of colors of the Colorful Motif instance. Also, for simplicity, let f_i denote $C(F_i)$. The coloring function c labels each vertex $x[i], x \in F_i$ with the color c_x. The "set" vertex u_i gets the color c_{f_i} where F_i is the set that u_i represents. The root vertex r gets a new color, named c_r. Formally,

$$c(x[i]) = \{c_x \mid x \in F_i, 1 \leqslant i \leqslant n\}$$
$$c(u_i) = \{c_{f_i} \mid 1 \leqslant i \leqslant n, \textbf{ and } c(r) = c_r\}.$$

Note that the color set used contains one color for each element in U, and the new color c_r. T is clearly a superstar; the reduced instance is $(T, |U| + 1, c)$.

Now, suppose that the given Colorful Set Cover instance is a YES instance, and let $R = \{F_{i_1}, F_{i_2}, \ldots, F_{i_t}\} \subseteq F$ be a solution for this instance. Consider the subtree T' of T consisting of (1) the subtrees rooted at $u_{i_1}, \ldots, u_{i_t}$, (2) the vertex r, and (3) the edges $(r, u_{i_1}), \ldots, (r, u_{i_t})$. From the subtree rooted at each u_j in T', remove

those leaves that have the same color as u_j. From the remaining leaves, arbitrarily delete all but one leaf of each color. Call the resulting tree T''. It is clear from the construction of T'' that it does not contain two vertices of same color. Now, consider the color c_x for any $x \in U \cup \{r\}$. We claim that T'' contains a vertex of color c_x. If $c_x = c_r$, then we are done since T'' contains the vertex r. Suppose $c_x \neq c_r$. Let F_p be an element of R which covers the element $x \in U$ in the solution to the Colorful Set Cover instance. Then, either $C(F_p) = x$, in which case we can immediately see that T''contains u_p which is colored with color c_x, or $C(F_p) \neq x$, in which case our construction of T'' ensures that the subtree of T'' rooted at u_p has a leaf with color c_x. Thus T'' is a colorful subtree of T with $|U| + 1$ colors.

Conversely, suppose that the reduced instance is a YES instance of Colorful Motif and let T'' be a colorful subtree of T with $|U| + 1$ colors. Let $\{u_{i_1}, u_{i_2}, \ldots, u_{i_t}\} = V(T'') \cap \{u_1, \ldots, u_n\}$. Let $R = \{F_{i_1}, F_{i_2}, \ldots, F_{i_t}\}$. In the original instance of Colorful Set Cover, the elements of R get the colors $\{c(u_{i_1}), c(u_{i_2}), \ldots, c(u_{i_t})\}$, and these are all distinct. Consider $d \in U$. Since T'' is colorful with $|U| + 1$ colors, some vertex of T'' is colored c_d, and $c_d \neq c_r$. Hence, the vertex of T'' colored c_d is either some u_i or a leaf adjacent to some u_i. Since we are picking the corresponding F_i in R, we ensure that R covers d and hence the given instance is a YES instance of Colorful Set Cover. □

Proposition 15.3. *Let* (T, C) *be an instance of* Colorful Motif*, where* T *is a superstar graph. Let* $u_1, \ldots u_r$ *be the children of the root of* T*. Let* V_i *denote the set of leaves adjacent to* u_i*, and let* U_i *denote* $V_i \cup \{u_i\}$*. For* $X \subseteq V(T)$*, let* $c(X)$ *denote the set of colors used on* X*, that is:*

$$c(X) = \{d \mid \exists x \in X, c(x) = d\}.$$

The Colorful Motif *problem is* NP-hard *on superstar graphs even on instances where no two subtrees are colored with the entire set of colors: that is, for any* $i \neq j$*,*

$$(c(U_i) \cup c(U_j)) \setminus C \neq \phi.$$

Proof. Note that the claim follows from the reduction stated in the proof of theorem 15.2, because of the fact that an instance of Colorful Set Cover is such that there is no i, j for which $F_i \cup F_j = U$. This rules out the possibility of the reduced instance obtained in the proof above having two subtrees that are colored with the entire set of colors.

□

15.2 Colorful Motifs on Graphs of Diameter Two and Three

In this section, we consider the Colorful Motif problem restricted to graphs of diameter two and three. We show that the Colorful Motif problem on superstars reduces to Colorful Motif on graphs of diameter two, thereby establishing that the problem is NP-complete. Also, we show that the Colorful Motif problem on general graphs reduces to Colorful Motif on graphs of diameter three, thereby establishing that the problem is NP-complete, and that polynomial kernels are infeasible. These reductions are quite similar, with only subtle differences.

Lemma 15.2. *The* Colorful Motif *problem on superstars with parameter* k *reduces to* Colorful Motif *with parameter* k *on graphs of diameter two.*

Proof. Note that superstars are graphs of diameter four. Let (T, k, c) be an instance of Colorful Motif on superstars. A superstar can be, by definition, rooted at a vertex r such that the graph is a rooted tree of height two. Let r denote the root of T, and assume that the neighbors of r are ordered in some arbitrary but fixed fashion. Let $T(i)$ denote the graph induced on the i^{th} neighbor of r, and the leaves of T that are adjacent to it, that is,

$$T(i) := T[v_i \cup \big(N[v_i] \setminus r\big)].$$

We refer to the graph induced on $T(i)$ as the i^{th} subtree. Note that, by Proposition 15.3, we may assume that any colorful subtree of T intersects non-trivially with more than two subtrees. We now describe an instance (Q, k, c_q) that is equivalent to (T, k, c), and is such that Q has diameter two. The graph Q is obtained from T in the following steps: First, we add $\binom{k}{2}$ new vertices:

$$V(Q) = V(T) \cup \{v[i, j] \mid i, j \in [k] \text{ and } i \neq j\}.$$

We use X to denote the set of newly introduced vertices, that is, $\{v[i, j] \mid i, j \in [k] \text{ and } i \neq j\}$. The new vertices get the same color as r, that is, $c_q(u) = c(r)$ for all vertices $u \in V(Q) \setminus V(T)$. For all "original" vertices u in $V(T)$, $c_q(u) = c(u)$. The

edge set of Q retains all the original edges in T, Further, for every pair of distinct subtrees $T(i)$ and $T(j)$, we make the vertex $v[i,j]$ global[3] to all the vertices in $T(i) \cup T(j)$. Finally, make every $v[i,j]$ global to the set of all vertices in the closed neighborhood of r, and induce a clique on all vertices $v[i,j]$ (see Figure 15.4). Formally:

(a) $\forall\{u,v\}, u \in T(i), v \in T(j)$, for $i,j \in [k]$ and $i \neq j$, $(u,v) \in E(Q)$ if, and only if, $(u,v) \in E(T)$

(b) $\forall i,j \in [k], i \neq k$, and $u \in T(i) \cup T(j)$, $(v[i,j],u) \in E$

(c) $\forall u \in X$ and $\forall v \in N[r]$, $(u,v) \in E$

(d) $\forall u,v \in X$, $(u,v) \in E$.

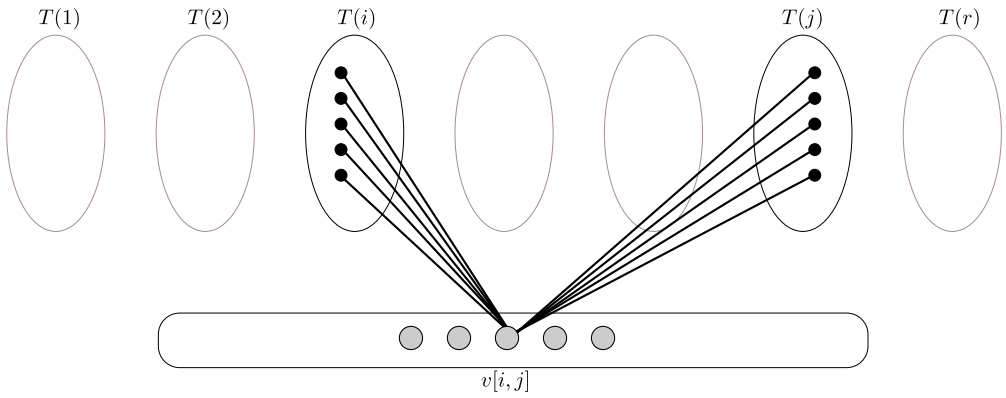


Figure 15.4: A Slice of the graph Q

Notice that Q is a graph of diameter two. Indeed, any pair of vertices within the same subtree have distance at most two (via their parent), and any pair of vertices in distinct subtrees $T(i)$ and $T(j)$ are reachable by a path of length at most two: if $i \neq k+1$ and $j \neq k+1$, then $v[i,j]$ is a common neighbor of any pair of vertices (u,v) such that $u \in T(i)$ and $v \in T(j)$, thus making them distance two apart. Any $v[i,j]$ is distance one from any vertex in $T(i)$ or $T(j)$ and is distance two from a vertex in any other subtree $T(l)$, because $v[l,u]$ (for any choice of u) is a common neighbor of a vertex in $T(l)$ and $v[i,j]$ (recall that the graph induced on vertices $v[i,j]$ induces a clique, and $v[l,u]$ is adjacent to every vertex in $T(l)$).

[3] The operation of making a vertex v global to a set S involves making v adjacent to every vertex in S.

We now claim that (T, k, c) is a YES-instance of Colorful Motif if, and only if, (Q, c_q, k) is a YES-instance of Colorful Motif. Notice that any colorful subtree T' in T is a colorful subtree of Q. In the converse, let R be a colorful subtree of Q. Observe that none of the newly introduced vertices (or edges) are used in R: such vertices connect precisely two subtrees, and therefore R intersects at most two subtrees of Q. Notice that $R \setminus v[i, j]$ is a colorful subtree of T, which contradicts our assumption that any colorful subtree of T intersects non-trivially with more than two subtrees. Thus, R functions as a colorful subtree T' of T. It is easy to see that this is also a polynomial parameter transformation. This completes the proof the lemma.

□

Corollary 15.4. *The* Colorful Motif *problem is* NP-hard *on graphs of diameter two.*

Proof. By Theorem 15.2, Colorful Motif is NP-hard on superstars, and due the polynomial reduction established in Lemma 15.2, we have that the Colorful Motif problem is NP-hard on graphs of diameter two. □

Lemma 15.3. *The* Colorful Motif *problem with parameter* k *reduces to* Colorful Motif *with parameter* $(k+1)$ *on graphs of diameter three.*

Proof. Let (T, c, k) be an instance of Colorful Motif with k colors. Let $T(i)$ denote the subset of vertices in T that have color i, that is:

$$T(i) = \{v \in T \mid c(v) = i\}.$$

We refer to the set of vertices in $T(i)$ as the *color class* i. We now describe an instance $(R, c_r, k+1)$ that is equivalent to T, and is such that R has diameter three. To this end, we describe an intermediate instance $(Q, c_q, k+1)$ based on (T, c, k). The graph Q is obtained from T in the following steps: First, we add $\binom{k}{2}$ new vertices:

$$V(Q) = V(T) \cup \{v[i, j] \mid i, j \in [k] \text{ and } i \neq j\}.$$

The new vertices form the color class $(k+1)$, that is, $c_q(u) = k+1$ for all vertices $u \in V(Q) \setminus V(T)$. We abuse notation and use $T(i)$ to refer to color class i in Q, for $i \in [k+1]$. For all "original" vertices u in $V(T)$, $c_q(u) = c(u)$. The edge set of Q retains all the original edges in T, and further, we add edges so that every color class induces a clique. Finally, for every pair of distinct color classes $T(i)$ and $T(j)$, we make the vertex $v[i, j]$ global to all the vertices in $T(i) \cup T(j)$ (see Figure 15.5). Formally:

(a) For every $i \in [k+1]$, $\forall\{u,v\} \in T(i)$, $(u,v) \in E(Q)$

(b) $\forall\{u,v\}, u \in T(i), v \in T(j)$, for $i,j \in [k]$ and $i \neq j$, $(u,v) \in E(Q)$ if, and only if, $(u,v) \in E(T)$

(c) $\forall i,j \in [k], i \neq k$, and $u \in T(i) \cup T(j)$, $(v[i,j],u) \in E$

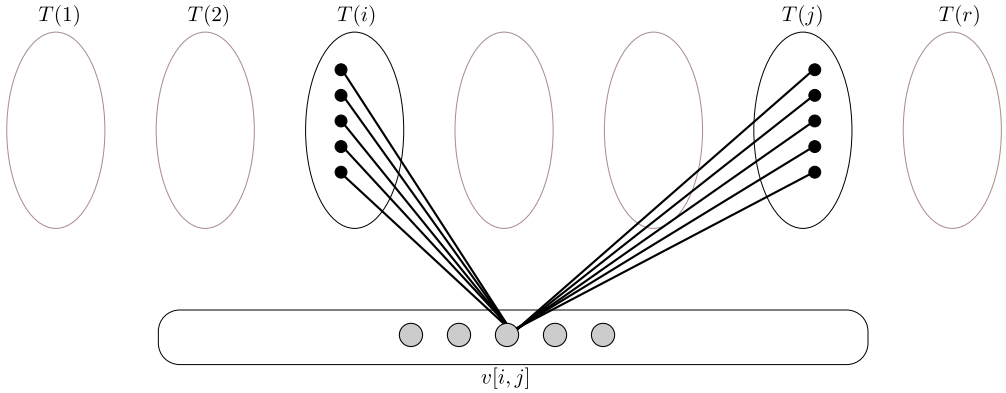


Figure 15.5: A Slice of the graph Q

Notice that Q is a graph of diameter two. Indeed, any pair of vertices within the same color class $T(i)$ have distance one, and any pair of vertices in distinct color classes $T(i)$ and $T(j)$ are reachable by a path of length at most two: if $i \neq k+1$ and $j \neq k+1$, then $v[i,j]$ is a common neighbor of any pair of vertices (u,v) such that $u \in T(i)$ and $v \in T(j)$, thus making them distance two apart. Any $v[i,j]$ is distance one from any vertex in $T(i)$ or $T(j)$ and is distance two from a vertex in any other color class $T(l)$, because $v[l,l']$, for any $l' \neq l$, is a common neighbor of a vertex in $T(l)$ and $v[i,j]$ (recall that $T(k+1)$ induces a clique, and $v[l,l']$ is adjacent to every vertex in $T(l)$).

We are now ready to provide a description of $(R, c_r, k+1)$. The graph R is obtained from Q by replacing every $v[i,j] \in T(k+1)$ with the following vertex set:

$$V[i,j] = \{v[i,j]^{(1)}, v[i,j]^{(2)}, \ldots, v[i,j]^{(d_{ij})}\},$$

where d_{ij} is the number of neighbors of $v[i,j]$ in $V(R) \setminus T(k+1)$. We assume that the vertices in $N(v[i,j])$ are ordered in some arbitrary but fixed fashion. We then add the edges $(v[i,j]^{(l)}, u_l)$, where u_l is the l^{th} neighbor of $v[i,j]$. We then add edges so that $T(k+1)$ induces a clique. Again, for all original vertices u in $V(Q)$, $c_r(u) = c_q(u)$. For all vertices $u \in T(k+1)$, we let $c_r(u) = k+1$.

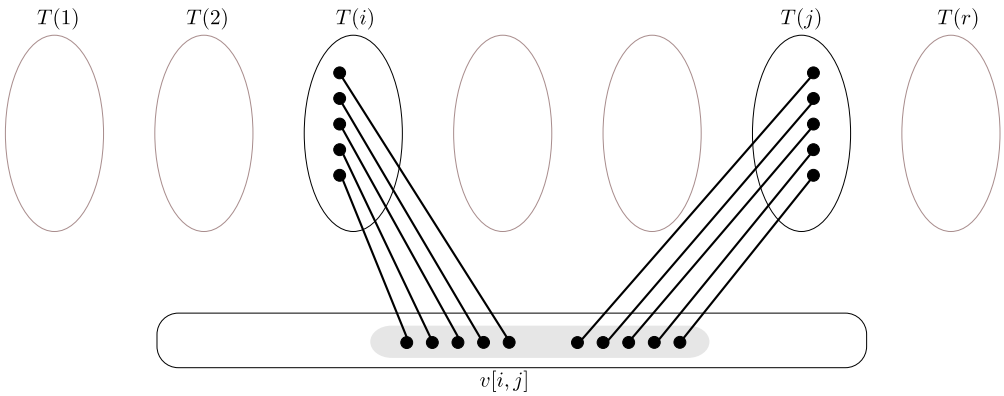


Figure 15.6: A Slice of the Graph R

Observe that R is a graph of diameter three. Again, every pair of vertices within the same color class $T(i)$ have distance one. Any pair of vertices in distinct color classes $T(i)$ and $T(j)$, if $i \neq k+1$ and $j \neq k+1$ are reachable by a path of length at most three. Let $u \in T(i)$ be the l_u^{th} neighbor of $v[i,j]$ in Q, and let $v \in T(j)$ be the l_v^{th} neighbor of $v[i,j]$. Then $(u, v[i,j]^{l_u}) \in E$, $(v[i,j]^{l_u}, v[i,j]^{l_v}) \in E$, and $(v[i,j]^{l_v}, v) \in E$, establishing a path of length three between u and v. Further, it is easy to see that any $v[i,j]^l$ is distance two from any vertex in $T(i)$ or $T(j)$ or any other color class $T(l)$, for similar reasons.

We now claim that (T, c, k) is a YES-instance of Colorful Motif if, and only if, $(R, c_r, k+1)$ is a YES-instance of Colorful Motif. Notice that any colorful subtree T' in T can be trivially extended to a colorful subtree R' in R - indeed, we may let R' to be $T' \cup \{(u,v)\}$, where $u \in T'$ and $v \in T(k+1)$ such that $(u,v) \in E$ (note that such a v always exists). Conversely, let R' be a colorful subtree of R. Let u be the vertex in R' from $T(k+1)$. Notice that u is necessarily a leaf of R', since no vertex $u \in T(k+1)$ has degree more than one outside $T(k+1)$. Notice that the subtree $R' \setminus \{u\}$ gives us a colorful subtree T' of T, as required. It is easy to see that this is also a polynomial parameter transformation. This completes the proof the lemma.

□

Corollary 15.5. *The* Colorful Motif *problem is* NP-hard *on graphs of diameter three, and does not admit a polynomial kernel unless NP ⊆ CoNP/poly.*

Proof. The Colorful Motif is NP-hard and does not admit a polynomial kernel unless NP ⊆ CoNP/poly (this follows from the results in [CPPW10]). Note that the

polynomial reduction established in Lemma 15.3, is a polynomial parameter transformation, therefore the corollary follows. □

15.3 Many Polynomial Kernels on Combs

In [CPPW10], Cygan et al. show that Colorful Motif is NP-complete on *comb graphs*, defined as follows:

Definition 15.1. *A graph* $G = (V, E)$ *is called a* comb graph *if (i) it is a tree, (ii) all vertices are of degree at most 3, (iii) all the vertices of degree 3 lie on a single simple path. The maximal path, which starts and ends in degree 3 vertices is called the* spine *of a comb graph. One of the two endpoint vertices of the spine is arbitrarily chosen as the first vertex, and the other as the last. A path from a degree 3 vertex to a leaf which contains exactly one degree 3 vertex, is called a* tooth.

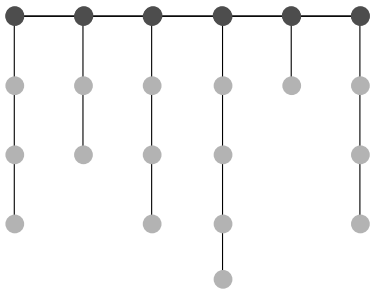


Figure 15.7: A Comb Graph

In this section, we present a composition algorithm for Colorful Motif on comb graphs. Note that in [CPPW10], it is observed that Colorful Motif is unlikely to admit polynomial kernels on forests. This simple composition obtained using disjoint union does not work "as is" when we restrict our attention to comb graphs, since the graph resulting from the disjoint union of comb graphs is not a comb graph, as it is not connected.

15.3.1 A Composition Algorithm

We begin by introducing some notation that will be useful presently. Let (T, k, c) be an instance of Colorful Motif restricted to comb graphs, that is, let T denote a comb graph, and let $c : V(T) \to [k]$ be a coloring function.

Let T_p and T_q be two comb graphs, and let $l_p \in V(T_p)$ be the last vertex on the spine of T_p, and let $f_q \in V(T_q)$ be the first vertex on the spine of T_q. We define $T_p \odot T_q$ as follows:

(i) $V(T_p \odot T_q) = V(T_p) \uplus V(T_q) \cup \{v_p, v_q\}$, where $\{v_p, v_q\}$ are "new" vertices, and

(ii) $E(T_p \odot T_q) = E(T_p) \uplus E(T_q) \cup \{(l_p, v_p), (v_p, v_q), (v_q, f_q)\}$

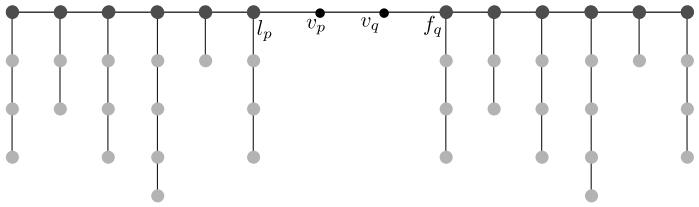


Figure 15.8: An illustration of the $T_p \odot T_q$ operation

We are now ready to describe the composition algorithm:

Lemma 15.4. *The* Colorful Motif *problem does not admit a polynomial kernel on comb graphs unless NP* $\subseteq$ *CoNP/poly.*

Proof. Let $(T_1, c_1, k), (T_2, c_2, k), \ldots (T_t, c_t, k)$ be the instances that are input to the composition algorithm. Let $\mathcal{T}$ denote the graph:

$$\mathcal{T} = T_1 \odot T_2 \odot \cdots \odot T_t.$$

Let N denote the set of all new vertices introduced by the $\odot$ operations. Notice that any vertex of $\mathcal{T}$ that does not belong to N is a vertex from one of the instances T_i. We will refer to such vertices in $\mathcal{T}$ as being from $\mathcal{T}(T_i)$.

We refer to the pair of vertices in N adjacent to the endpoints of the spine of a T_i as the *guards* of T_i (notice that any T_i has at most two guard vertices). We define the coloring function c on $\mathcal{T}$ as follows. For every vertex $u \in T_i$, $c(u) = c_i(u)$. For every vertex u that is a guard of T_i, $c(u) = c(v)$, where v is the vertex of T_i adjacent to u. We now claim that $(\mathcal{T}, k)$ is the composed instance.

We first show that if $\exists i, i \in [t]$, such that T_i is a YES-instance of Colorful Motif, then $\mathcal{T}$ is a YES instance of Colorful Motif. Notice that the colorful connected subtree of T_i also exists in $\mathcal{T}$, and hence we are done.

Conversely, let $\mathcal{I}$ be a YES-instance of Colorful Motif. Let R be a colorful connected subtree of $\mathcal{I}$. Notice that by construction, if $V(R)$ intersects with $\mathcal{I}(T_i)$ and $\mathcal{I}(T_j)$ for $i \neq j$, then it contains two vertices of the same color — indeed, this follows from the observation that $V(R)$ would have to contain the guard vertices of T_i and T_j to be connected, and this would lead to multiple occurrences of the colors that occur at the endpoints of the spines of T_i and T_j.

Thus, $V(R)$ intersects with $\mathcal{I}(T_i)$ for exactly one value of i, $1 \leqslant i \leqslant t$. Clearly, R is a colorful connected subtree of T_i. Thus, by Theorem 13.2, Colorful Motif admits no polynomial kernel on comb graphs unless NP $\subseteq$ CoNP/poly. □

Corollary 15.6. *The* Colorful Motif *problem on lobsters does not admit a polynomial kernel unless NP $\subseteq$ CoNP/poly.*

Proof. Recall that a lobster is a tree where the removal of all leaf vertices results in a caterpillar. Clearly, the problem is NP-complete due to Theorem 15.2. Further, observe that the composition described in the proof of 15.4 can be imitated with minor changes to obtain a similar result on lobsters. Indeed, we only need to observe that it suffices to subdivide only edges along the spine of the lobster (which is the spine of the caterpillar obtained by removing all the leaves of the lobster). If a solution does not intersect the spine, it is contained in one of the "dangling superstars", and this is easily detected before applying the composition. □

15.3.2 Many Polynomial Kernels

Although a parameterized problem may not necessarily admit a polynomial kernel, it may admit many of them, with the property that the instance is in the language if and only if at least one of the kernels corresponds to an instance that is in the language. We now show that the Colorful Motif problem admits n kernels of size $\mathcal{O}(k^2)$ each on comb graphs. This is established by showing that a closely related variant, the Rooted Colorful Motif problem, admits a polynomial kernel. The Rooted Colorful Motif problem is the following:

Rooted Colorful Motif

Input:	A graph $G = (V, E)$, $k \in \mathbb{N}$, a coloring function $c : V \to [k]$, and $r \in V$.
Parameter:	k
Question:	Does G contain a subtree T on k vertices, containing r, such that c restricted to T is bijective?

Lemma 15.5. *The* Colorful Motif *problem admits many polynomial kernels on comb graphs.*

Proof. Let $T = (V, E)$ be a comb graph, and let (T, k, c, u) be an instance of Rooted Colorful Motif. We first show that T can be reduced to an equivalent instance T_u on at most $\mathcal{O}(k^2)$ vertices. Notice that we may obtain an equivalent instance T_u from T by deleting all vertices in T that lie outside the k-neighborhood of u. This is because any colorful subtree of T rooted at u will not involve vertices outside the k-neighborhood of u.

We now observe that T_u has at most $\mathcal{O}(k^2)$ vertices. Note that in the k-neighborhood of u in T, there are at most $2k$ vertices that belong to the spine and from each of these vertices, there is a tooth of length at most k. Also, if u lies on a tooth, there are at most $2k$ more vertices on the same tooth, and if u lies on the spine then there are at most k more vertices on the tooth rooted at u, if at all one exists. Hence, in T_u, there are at most $2k$ vertices of degree 3 each having a tooth of length at most k and at most one other tooth, which is of length at most $2k$. So, the total number of vertices in T_u is at most $2k + 2k \cdot k + 2k = \mathcal{O}(k^2)$. Notice that there are n choices for u from T, and repeating the procedure above by "guessing the root" gives us n polynomial kernels for the problem, as desired.

□

15.4 Hardness of Kernelization for Restricted Variants

In this section, we demonstrate the infeasibility of some strategies for showing many polynomial kernels for Colorful Motif restricted to trees. Observe that the Colorful Motif problem is unlikely to admit a polynomial kernel on trees, since a polynomial kernel on trees would imply a polynomial kernelization procedure for comb graphs, which is infeasible (see Lemma 15.4).

15.4.1 Hardness with a Fixed Root

In the case of comb graphs, we were able to establish that the problem of finding a colorful subtree with a fixed root admits a $\mathcal{O}(k^2)$ kernel. Unfortunately, this ap-

proach does not extend to trees, as we establish that Rooted Colorful Motif is compositional on trees.

Proposition 15.7. *The* Rooted Colorful Motif *problem restricted to trees is* NP-hard.

Proof. By reduction from Colorful Motif restricted to trees, which is NP-hard on trees (in fact, on the even more restricted class of comb graphs) as shown by Cygan et al [CPPW10].

Let (T, k, c) be an instance of Colorful Motif where T is a tree. Let $\{v_1, v_2, \ldots, v_\ell\}$ be the vertices in T that have color 1 in T. Make ℓ copies $T_1, T_2, \ldots, T_\ell$ of T. Add a new vertex v, and for $1 \leqslant i \leqslant \ell$, make v adjacent to the copy of v_i in T_i; let this graph be T'. Let c' be the coloring function that gives all copies of a vertex in T the same color as it has in T, and the color $k+1$ to the new vertex v. $(T', k+1, c', v)$ is the reduced instance of Rooted Colorful Motif. T' is clearly a tree, and it is easy to see that (T, k, c) is a yes instance of Colorful Motif if and only if $(T', k+1, c', v)$ is a yes instance of Rooted Colorful Motif. □

Proposition 15.8. *The* Rooted Colorful Motif *problem when restricted to trees does not admit a polynomial kernel unless NP* $\subseteq$ *CoNP/poly.*

Proof. From Proposition 15.7, the Rooted Colorful Motif problem is NP-complete when restricted to trees. We now describe a composition algorithm for Rooted Colorful Motif on trees.

Let $(T_1, v_1, c_1, k), (T_2, v_2, c_2, k), \ldots (T_r, v_r, c_r, k)$ be the input instances, where $T_i = (V_i, E_i)$. Consider the tree $\mathcal{T} = (V, E)$ described as follows:

1. $V = \{u\} \cup \{u_1, u_2, \ldots, u_r\} \cup_{i \in [r]} V_i$
2. $(u, u_i) \in E$, for all $i \in [r]$, and $(u_i, v_i) \in E$, for all $i \in [r]$
3. $E_i \subset E$ for all $i \in [t]$.

Consider the coloring function $c : V \to [k+2]$ defined as follows: for all $v \in V_i$, $c(v) = c_i(v)$. Further, $c(u_i) = k+1$ for all u_i, and $c(u) = k+2$.

We now claim that $(\mathcal{T}, k+2, c, v)$ is the composed instance. Indeed, if T_i has a colorful subtree rooted at v_i then we have a colorful subtree (on $k+2$ colors) rooted

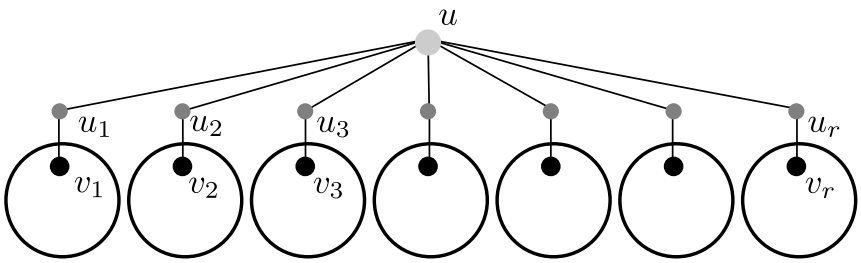


Figure 15.9: An illustration of the composition

at v, which extends the tree rooted at v_i using the edges (v_i, u_i) and (u_i, u). In the reverse direction, suppose $\mathcal{T}$ has a colorful subtree T' rooted at u. Then, observe that T' contains vertices from *exactly* one of the T_is. This is because if T' contains vertices from T_i and T_j then T' contains vertices u_i and u_j both of which have the same color , a contradiction to the assumption that T' was colorful. This completes the proof.

□

15.4.2 Hardness with a Fixed Subset of Vertices

Now, we have seen that "fixing" one vertex does not help the cause of kernelization for trees in general. In fact, more is true: fixing any constant number of vertices does not help. The problem we study in this section is the following:

Subset Colorful Motif

Input:	A graph $G = (V, E)$, a coloring function $c : V \to [k]$, and a set of vertices $U \subseteq V$, $\lvert U\rvert = s = \mathcal{O}(1)$.
Parameter:	k
Question:	Does G contain a subtree T on k vertices, such that $U \subseteq V(T)$, and c restricted to T is bijective?

Proposition 15.9. *The* Subset Colorful Motif *problem restricted to trees does not admit a polynomial kernel unless NP ⊆ CoNP/poly.*

Proof. To prove this proposition, it is sufficient to show that there exists a polynomial parameter transformation from the Rooted Colorful Motif problem restricted to trees to the Subset Colorful Motif problem restricted to trees (Theorem 13.3 and Propositions 15.7 and 15.8). We now proceed to give such a transformation.

Let (T, v, c, k), where $T = (V, E)$, be an input instance to Rooted Colorful Motif. Let $s' = s - 1$. Construct the tree $T' = (V', E')$ as follows: (i) $T' = (V', E')$ where $V' = V \cup \{u_1, u_2, \ldots, u'_s\}$ (ii) $E' = E \cup \{(v, u_1), \ldots, (v, u'_s)\}$

Now, we define a coloring function $c' : V' \to [k + s']$ as follows:

(i) $c'(v) = c(v), \forall v \in V$

(ii) $c'(u_j) = k + j, \forall j \in [s']$

$(T', c', \{v, u_1, \ldots, u'_s\}, k + s')$ is the reduced instance of Subset Colorful Motif. It is easy to see that the reduction is a polynomial parameter transformation, and the proposition follows. □

15.5 Connected Dominating Set

In this section, we show that Connected Dominating Set on graphs of diameter two is NP-complete. The classical complexity of Connected Dominating Set on graphs of diameter two was hitherto unknown, although it was known to be NP-complete on graphs of diameter three, and trivial on graphs of diameter one. We establish this by a non-trivial reduction from Colorful Motif on graphs of diameter two, which is NP-complete by Lemma 15.2.

Theorem 15.10. *The* Connected Dominating Set *problem, when restricted to graphs of diameter two, is* NP-complete.

Proof. We establish this by demonstrating a reduction from Colorful Motif on graphs of diameter two. Let (G, k, c) be an instance of Colorful Motif on graphs of diameter two. Let $C(i)$ denote the subset of vertices in C that have color i. That is, $C(i) = \{v \in G \mid c(v) = i\}$. We refer to set of vertices in $C(i)$ as the *color class* i. We now describe an equivalent instance (H, k) of Connected Dominating Set on graphs of diameter two.

We define the following simple operations before we begin:

1. *inducing a clique on* S, where $S \subseteq V$: Add the edges (u, v) for all $u, v \in S$, $u \neq v$.

2. *inducing a complete bipartite graph on* (S, T), where $S, T \subseteq V$, $S \cap T = \emptyset$: Add the edges (u, v) for all $u \in S$, $v \in T$.

3. *inducing a matching on* (S, T), where $S, T \subseteq V$, $S \cap T = \emptyset$, $|S| = |T|$ and there exists a natural ordering of the vertices in S and T (that is, $S = \{s_1, s_2, \ldots, s_r\}$ and $T = \{t_1, t_2, \ldots, t_r\}$): Add the edges (s_i, t_i) for all i, $1 \leqslant i \leqslant r$.

4. *making* u *global to* S, given a vertex u and a set $S \subseteq V$: Add all edges (u, v) where $v \in S$.

The graph H is obtained from G, in a series of several steps. We describe the graph H by the sequence of modifications made to G to arrive at H.

(1) To begin with, let $H = G$.

(2) Induce a clique on $C(i)$, for all $i \in [k]$.

(3) For every color class $C(i)$, add $(k + 1)$ new vertices $v_i(1), v_i(2), \ldots, v_i(k + 1)$. We refer to this set as $H(i)$. Induce a complete bipartite graph on $(C(i), H(i))$.

(4) For every pair (i, j) such that $i \neq j$, induce a matching on $(H(i), H(j))$, where the ordering on the vertices in H_i and H_j is the natural one established by the labels. (That is, the j^{th} vertex of H_i is $v_i(j)$.) Further, for every pair of vertices $(v_i(p), v_j(q))$, where $p \neq q$, add an edge and subdivide it. We refer to the set of all such subdivided vertices as $D(i, j)$. We let $D = \cup_P D(i, j)$, where $P = \binom{[k]}{2}$.

(5) Induce a clique on D, the set of subdivided vertices.

(6) For every $u \in D(i, j)$, we make u global to $C(l)$, for every l, where $l \neq i$ and $l \neq j$.

Claim 15.1. *The graph* H *has diameter two.*

Proof. We show this by demonstrating a path of length at most two for every pair of vertices u and w. This is done in a series of cases, summarized in table 15.2.

The detailed case analysis proceeds as follows:

(i) $u \in C(i)$ and $w \in C(i)$. In this case the distance between u and w is one, because $C(i)$ is a clique.

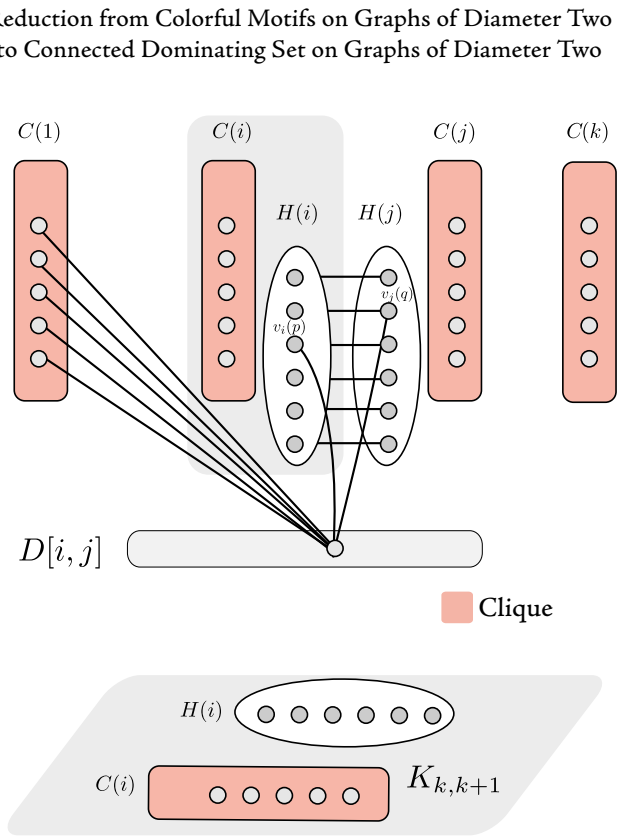


Figure 15.10: An Illustration of the Reduction in the proof of Theorem 15.10

(ii) $u \in C(i)$ and $w \in C(j)$, where $j \neq i$. In this case, there exists a path of length at most two because the original graph has diameter two.

(iii) $u \in H(i)$ and $w \in H(i)$. In this case there exists a path of length two because for every vertex $x \in C(i)$, by construction in step 3, (u, x) and (w, x) are both edges in the graph.

(iv) $u \in H(i)$ and $v \in H(j)$, $i \neq j$. Let $u = v_i(p)$ and let $w = v_j(q)$. If $p = q$, the edge introduced by the matching make u and v adjacent, otherwise, there exists a subdivided vertex that is a common neighbor of both u and w, which implies a path of length two.

(v) $u \in H(i)$ and $w \in C(i)$. By step 3 in the construction, (u, w) is an edge in the graph.

(vi) $u \in H(i)$ and $w \in C(j)$, where $j \neq i$. Let $u = v_i(p)$. Then $(u, v_j(p))$ is a

	C	H	D
C	G has diameter two	Complete bipartition and matching edges	D is a clique, and a neighbor into D always exists
H	[A Symmetric Case]	Matching edges, or a common neighbor in D	D is a clique, and a neighbor into D always exists
D	[A Symmetric Case]	[A Symmetric Case]	Clique

Table 15.2: The Diameter Two Argument Summary

matching edge and $(v_j(p), w)$ is an edge (by step 3 in the construction). This establishes a path of length two.

(vii) $u \in D$ and $w \in D$. In this case the distance between u and w is one, because D is a clique by step 5 of construction.

(viii) $u \in D(i,j)$ and $w \in C(l)$, where $l \neq i$ and $l \neq j$. In this case the distance between u and w is one, because u is global to $C(l)$ by step 6 of construction.

(ix) $u \in D(i,j)$ and $w \in C(i)$ (respectively, $w \in C(j)$). u is adjacent to a vertex in $H(i)$ (respectively, in $H(j)$), which is in turn adjacent to w. This establishes a path of length two.

(x) $u \in D$ and $w \in H(j)$ for some j. Either u is adjacent to w, or is adjacent to a vertex that is adjacent to w (recall that w is adjacent to at least one vertex in D and D induces a clique).

This case analysis establishes that H is indeed a graph of diameter two.

□

Now we prove that any colorful subtree of G is a connected dominating set of H. It is clearly connected. Notice that all vertices in $H(i)$ are dominated for all $i \in [k]$: any vertex in $H(i)$ is adjacent to all vertices in $C(i)$ and a colorful tree contains one vertex from $C(i)$. A similar argument shows that all vertices in D are dominated.

Conversely, we argue the graph induced on any connected dominating set of size at most k induces a colorful subtree in G. Notice that it suffices to prove that any

connected dominating set intersects non-trivially with every $C(i)$, that is, if S is a connected dominating set of size at most k, then:

$$|S \cap C(i)| \geqslant 1.$$

Because $|S| \leqslant k$, this implies that $|S \cap C(i)| = 1$. Because S is connected, the vertices of S induce a colorful subtree. Therefore, we now only need to establish that $|S \cap C(i)| \geqslant 1$. For the sake of contradiction, suppose not. In particular, let

$$|S \cap C(i)| = 0$$

Notice that no vertex $u \notin C(i)$ dominates more than one vertex in $H(i)$. Indeed:

(a) If $u \in H(i)$, then because $H(i)$ induces an independent set, u dominates no vertex other than itself.

(b) If $u \in D$, u has at most one neighbor in $H(i)$, and therefore dominates at most one vertex of $H(i)$.

(c) If $u \in H(j)$, $j \neq i$, u has at most one neighbor in $H(i)$ (indeed, the matching partner is the only one), and therefore dominates at most one vertex of $H(i)$.

(d) It is easy to see that all other vertices have no neighbors in $H(i)$.

Therefore, any dominating set that does not intersect with $C(i)$ is forced to pick more than k vertices to dominate all vertices in $H(i)$, contradicting the assumption that S was a dominating set of size at most k. This completes the proof.

□

You're searching, Joe, for things that don't exist; I mean beginnings.
Ends and beginnings – there are no such things. There are only middles.

Robert Frost

...he seemed to approach the grave as an hyperbolic curve approaches a straight line —
less directly as he got nearer, till it was doubtful if he would ever reach it at all.

Thomas Hardy

In this final chapter, we summarize the results presented in this thesis and establish possible directions for further research.

16.1 The $\mathcal{F}$-deletion Problem

In this thesis we gave the first kernelization algorithms for a subset of $\mathcal{F}$-Deletion problems and a generic approximation algorithm for the $\mathcal{F}$-Deletion problem when the set of excluded minors $\mathcal{F}$ contains at least one planar graph. Our approach generalizes and unifies known kernelization algorithms for p-Vertex Cover and p-Feedback Vertex Set. By the celebrated result of Robertson and Seymour, every $\mathcal{F}$-Deletion problem is FPT and our work naturally leads to the following questions:

♢ Does every $\mathcal{F}$-Deletion problem have a polynomial kernel?

♢ Can it be that for some finite sets of minor obstructions $\mathcal{F} = \{O_1, \ldots, O_p\}$ the answer to this question is NO?

♢ The case when $\mathcal{F} = \{K_5, K_{3,3}\}$, which amounts to vertex deletion to planar graphs, is an interesting challenge.

Our early work has led us to the following conjecture:

The $\mathcal{F}$-Deletion problem admits a polynomial kernel if, and only if, $\mathcal{F}$ contains a planar graph.

Towards the "forward" direction of this conjecture, in this thesis we showed the following:

★ $\mathcal{F}$-Deletion admits an approximation algorithm with ratio $O(\log^{3/2} OPT)$, when the class of excluded minors for $\mathcal{F}$ contains at least one planar graph.

★ Then we restrict ourselves to a subclass of graphs, called t-claw-free graphs (those that exclude $K_{1,t}$ as an induced subgraph), and show that $\mathcal{F}$-Deletion admits a polynomial kernel on this graph class, again provided $\mathcal{F}$ contains at least one planar graph.

★ In an independent endeavor, we show that when $\mathcal{F}$ contains the θ_c graph (a pair of vertices with c edges between them), then $\mathcal{F}$-Deletion admits a polynomial kernel on general graphs.

★ The disjoint version of the $\mathcal{F}$-Deletion admits a polynomial kernel on general graphs if $\mathcal{F}$ contains at least one planar graph. Recall that the input to the disjoint version is a graph with a partition of the vertex set into two parts such that the graph induced on either partition does not contain $\mathcal{F}$ as a minor. One of the parts is declared forbidden, and we are required to find an optimal $\mathcal{F}$-hitting set from the other part.

★ An important consequence of the polynomial kernel for this problem is an uniform FPT algorithm for the $\mathcal{F}$-Deletion problem that runs in time $2^{O(k \log k)} n^{O(1)}$. No algorithm running in time $(2^{k^{O(1)}} n^{O(1)})$ was known previously.

The remaining components of this conjecture are left open.

The approximation algorithm obtained in this thesis is rather general. While several generic approximation algorithms are known for problems of minimum vertex deletion to obtain subgraphs with property P, like when P is a hereditary property with a finite number of minimal forbidden subgraphs [LY93], or can be expressed as a universal first order sentence over subsets of edges of the graph [KT95], we are not aware of any generic approximation algorithm for the case when a property P is characterized by a finite set of forbidden minors.

In fact, our approximation algorithm combined with the fact that a graph G that excludes a fixed graph H as a minor has treewidth at most $O(\sqrt{|V(G)|})$ gives us an approximation algorithm for $\mathcal{F}$-Deletion (note that we now do not require $\mathcal{F}$ to contain a planar graph) with factor $O(\sqrt{n})$. In the context of approximation, we conclude with the following open problems:

$\diamondsuit$ Does $\mathcal{F}$-Deletion admit a constant-factor approximation algorithm when the class of excluded minors for $\mathcal{F}$ contains at least one planar graph?

$\diamondsuit$ Can we show that $\mathcal{F}$-Deletion (without any restrictions on the family $\mathcal{F}$) does not admit an approximation algorithm within a $n^{\left(\frac{1}{2}-\varepsilon\right)}$ factor, for any fixed $\varepsilon > 0$?

Another line of thought is to consider $\mathcal{F}$-Deletion wherein we require the solution to possess a fixed, desirable property, such as connectivity, or independence. In this thesis, we demonstrated a polynomial kernel for . It will be interesting to see which other subclasses of the $\mathcal{F}$-Deletion problem admit polynomial kernels when the solution is required to be independent. However, the question of connectivity seems to be settled in the negative (as far as kernelization is concerned).

16.2 Packing Variants of $\mathcal{F}$-deletion

In this thesis, we also consider the complementary "packing" question: we wish to maximize the number of vertex (or edge) disjoint minor models of graphs in $\mathcal{F}$. We provide an Erdős-Pósa style result for the case when $\mathcal{F} = \{ \theta_c \}$. This borrows techniques from the approximation algorithm for the $\mathcal{F}$-Deletion problem and also makes use of known connections between *brambles* and treewidth.

In this context, we apply the "Erdős-Pósa property" of θ_c minor models to make partial progress towards finding polynomial kernels for packing edge disjoint minor models of θ_c. In particular, in polynomial time, we are able to reduce any instance of the problem to one where the maximum degree is bounded by a poylnomial in k. Independently, but on a related note, we show two lower bounds, demonstrating that the problem of finding if there are at least k *vertex-disjoint* minor models of θ_c is unlikely to admit a polynomial kernel parameterized by k, and this is also true for the problem of checking if there are at least k *vertex-disjoint* cycles of odd length (again parameterized by k).

The two natural questions in this context are the following:

$\diamondsuit$ For what choices of H does the problem of Edge-Disjoint $M(H)$-packing problem admit a polynomial kernel?

$\diamondsuit$ For what choices of H does the problem of Vertex-Disjoint $M(H)$-packing problem admit a polynomial kernel?

A dichotomy result for either of the problems above would be the most desirable result in this direction.

16.3 Colorful Motifs

We studied the problem of Colorful Motif on various graph classes. We proved that the problem of Colorful Motif restricted to superstars is NP-Complete. We also showed NP-completeness on graphs of diameter two. We applied this result towards settling the classical complexity of Connected Dominating Set on graphs of diameter two — specifically, we show that it is NP-Complete. Further, we showed that on graphs of diameter two, the problem is NP-Complete *and* is unlikely to admit a polynomial kernel.

Next, we showed that obtaining polynomial kernels for Colorful Motif on comb graphs is infeasible, but we show the existence of n polynomial kernels. Further, we study the problem of Colorful Motif on trees, where we observe that the natural strategies for many polynomial kernels are not successful. For instance, we show that "guessing" a root vertex, which helped in the case of comb graphs, fails as a strategy because the Rooted Colorful Motif problem has no polynomial kernels on trees. We summarize our results about Colorful Motif in special graph classes below:

★ On the class of comb graphs, Colorful Motif is NP-Complete and Rooted Colorful Motif has an $\mathcal{O}(k^2)$ kernel. Equivalently, Colorful Motif has n kernels of size $\mathcal{O}(k^2)$ each.

★ Subset Colorful Motif does not admit a polynomial kernel on trees unless NP $\subseteq$ CoNP/poly.

Finally, we leave open the questions of whether the Colorful Motif problem admits polynomial kernels on superstars, and many polynomial kernels when restricted to trees.

16.4 Concluding Remarks

The notion of kernelization is popular in practice — in many cases, it can be thought of as a precise way of stating all the heuristic-based preprocessing steps that have been

popular and effective for a long time. In theory, the notion is important for more than one reason — there is an increasingly popular feeling that kernelization is *the* way of understanding fixed-parameter tractability. The theorem that establishes the equivalence of these notions is more than a syntactic equality — it encodes an entire philosophy, and immediately puts on offer a possible "right way" of viewing FPT. Kernelization complexity is fast evolving as one the most important benchmarks for a deeper understanding of problem complexity. In this thesis, we hope that we have demonstrated this to a large extent with the help of questions that are generic, fundamental and practical.

References

[ABH+10] Abhimanyu M. Ambalath, Radheshyam Balasundaram, Chintan Rao H, Venkata Koppula, Neeldhara Misra, Geevarghese Philip, and M S Ramanujan, *On the kernelization complexity of colorful motifs*, International Symposium on Parameterized and Exact Computation (IPEC), 2010, pp. 14–25. 11

[AFLS07] Faisal N. Abu-Khzam, Michael R. Fellows, Michael A. Langston, and W. Henry Suters, *Crown structures for vertex cover kernelization*, Theory Comput. Syst. **41** (2007), no. 3, 411–430. 29

[ALS91] S Arnborg, J Lagergren, and D Seese, *Easy problems for tree-decomposable graphs*, J. Algorithms **12** (1991), no. 2, 308–340. 20, 83

[AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick, *Color-coding*, Journal of the ACM **42** (1995), no. 4, 844–856. 177

[BBF99] Vineet Bafna, Piotr Berman, and Toshihiro Fujito, *A 2-approximation algorithm for the undirected feedback vertex set problem*, SIAM J. Discr. Math. **12** (1999), no. 3, 289–297. 122, 123

[BBR07] Etienne Birmelé, John Adrian Bondy, and Bruce A. Reed, *Brambles, prisms, and grids*, Graph theory in Paris, Trends Math., Birkhäuser Verlag, 2007, pp. 37–44. 160

[BDFH09] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin, *On Problems Without Polynomial Kernels*, J. Comput. Syst. Sci. **75** (2009), no. 8, 423–434. 175, 176

[BFL+09] H. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos, *(Meta) Kernelization*, Proceedings of the 50th Annual Symposium on Foundations of Computer Science (FOCS), IEEE, 2009, pp. 629–638. 19, 59, 60, 64, 70, 116, 119, 122, 148, 153

[BHK+09] Sharon Bruckner, Falk Hüffner, Richard M. Karp, Ron Shamir, and Roded Sharan, *Topology-free querying of protein interaction networks*, The 13th Annual International Conference on Research in Computational Molecular Biology (RECOMB), LNCS, vol. 5541, Springer, 2009, pp. 74–89. 9, 193

[Bod94] Hans L. Bodlaender, *On disjoint cycles*, International Journal of Foundations of Computer Science **5** (1994), 59–68. 183

[Bod96] Hans L. Bodlaender, *A linear-time algorithm for finding tree-decompositions of small treewidth*, SIAM J. Comput. **25** (1996), no. 6, 1305–1317. 20, 79, 83

[Bod07] ———, *A Cubic Kernel for Feedback Vertex Set*, STACS'07: Proceedings of the 24th Annual Conference on Theoretical Aspects of Computer Science (Berlin, Heidelberg), Springer-Verlag, 2007, pp. 320–331. 180

[BPT92] Richard B. Borie, Gary R. Parker, and Craig A. Tovey, *Automatic Generation of Linear-Time Algorithms from Predicate Calculus Descriptions of Problems on Recursively Constructed Graph Families*, Algorithmica 7 (1992), 555–581. 20, 83

[BTTvL95] Hans L. Bodlaender, Richard B. Tann, Dimitris M. Thilikos, and Jan van Leeuwen, *On interval routing schemes and treewidth*, Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science, vol. 1017, 1995, pp. 181–196. 87

[BTY09] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo, *Kernel Bounds for Disjoint Cycles and Disjoint Paths*, Proceedings of the 17th Annual European Symposium (ESA 2009), Lecture Notes in Comput. Sci., vol. 5757, Springer, 2009, pp. 635–646. 115, 122, 178, 180, 181, 183, 184, 185, 186, 189

[BYGNR98] Reuven Bar-Yehuda, Dan Geiger, Joseph Naor, and Ron M. Roth, *Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference*, SIAM J. Computing **27** (1998), no. 4, 942–959. 123

[CCHO05] Jin-yi Cai, Venkatesan T. Chakaravarthy, Lane A. Hemaspaandra, and Mitsunori Ogihara, *Competing provers yield improved Karp-Lipton collapse results*, Information and Computation **198** (2005), no. 1, 1–23. 194

[CCJ90] Brent N. Clark, Charles J. Colbourn, and David S. Johnson, *Unit disk graphs*, Discrete Math. **86** (1990), no. 1-3, 165–177. 115

[CCL10] Yixin Cao, Jianer Chen, and Yang Liu, *On feedback vertex set new measure and new structures*, 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT), Lecture Notes in Computer Science, vol. 6139, 2010, pp. 93–104. 141

[CFJ05] Benny Chor, Michael R. Fellows, and David W. Juedes, *Linear kernels in linear time, or how to save* k *colors in* $O(n^2)$ *steps*, Proceedings of the 30th International Workshop on Graph-Theoretic Concepts in Computer Science (WG), LNCS, vol. 3353, Springer, 2005, pp. 257–269. 25, 35

[CFKX07] 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** (2007), no. 4, 1077–1106. 22

[CFL+08] Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger, *Improved algorithms for feedback vertex set problems*, J. Comput. Syst. Sci **74** (2008), no. 7, 1188–1198. 141

[CFM11] Yijia Chen, Jörg Flum, and Moritz Müller, *Lower bounds for kernelizations and other preprocessing procedures*, Theory Comput. Syst **48** (2011), no. 4, 803–839. 176

[CGHW98] Fabián A. Chudak, Michel X. Goemans, Dorit S. Hochbaum, and David P. Williamson, *A primal-dual interpretation of two 2-approximation algorithms for the feedback vertex set problem in undirected graphs*, Operations Research Letters **22** (1998), no. 4-5, 111–118. 123

[CKJ01] Jianer Chen, Iyad A. Kanj, and Weijia Jia, *Vertex Cover: Further observations and further improvements*, J. Algorithms **41** (2001), no. 2, 280–301. 22

[CM93] Bruno Courcelle and M Mosbah, *Monadic second-order evaluations on tree-decomposable graphs*, Theor. Comp. Sci. **109** (1993), no. 1–2, 49–82. 20, 83

[Cou90] Bruno Courcelle, *The monadic second-order logic of graphs. i. recognizable sets of finite graphs*, Information and Computation **85** (1990), no. 1, 12–75. 20, 83

[Cou97] Bruno Courcelle, *The expression of graph properties and graph transformations in monadic second-order logic*, Handbook of Graph Grammars

and Computing by Graph Transformations, Volume 1: Foundations, World Scientific, 1997, pp. 313–400. 20

[CPPW10] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk, *Improved FPT algorithm and quadratic kernel for pathwidth one vertex deletion*, Proceedings of the 5th International Symposium on Parameterized and Exact Computation (IPEC 2010), Lecture Notes in Comput. Sci., Springer, 2010, pp. 95–106. 194, 196, 197, 206, 207, 211

[DF95a] Rod G. Downey and Michael R. Fellows, *Fixed-parameter Tractability and Completeness I: Basic Results*, SIAM J. Comput. **24** (1995), no. 4, 873–921. 178

[DF95b] ———, *Fixed-parameter Tractability and Completeness II: On Completeness for* W[1], Theor. Comput. Sci. **141** (1995), no. 1-2, 109–131. 178

[DF95c] ———, *Parameterized Computational Feasibility*, Feasible Mathematics II, Boston: Birkhäuser, 1995, pp. 219–244. 178

[dF97] Babette de Fluiter, *Algorithms for graphs of small treewidth*, Ph.D. thesis, Utrecht University, 1997. 59, 64

[DF99] R. G. Downey and M. R. Fellows, *Parameterized Complexity*, Springer, 1999. 4, 183

[Die05] Reinhard Diestel, *Graph theory*, third ed., Graduate Texts in Mathematics, vol. 173, Springer-Verlag, Berlin, 2005. 14

[DLS09] Michael Dom, Daniel Lokshtanov, and Saket Saurabh, *Incompressibility Through Colors and IDs*, ICALP '09: Proceedings of the 36th International Colloquium on Automata, Languages and Programming (Berlin, Heidelberg), Springer-Verlag, 2009, pp. 378–389. 176, 178

[DST02] Josep Diaz, Maria Serna, and Dimitrios M. Thilikos, *Counting* H*-colorings of partial* k*-trees*, Theoretical Computer Science **281** (2002), 291–309. 75

[DvM10] Holger Dell and Dieter van Melkebeek, *Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses*, Proceedings of 42th ACM Symposium on Theory of Computing (STOC 2010), ACM, 2010, pp. 251–260. 124

[EP65a] P. Erdős and P. Pósa, *On Independent Circuits Contained in a Graph*, Canadian Journal of Mathematics **17** (1965), 347–352. 180

[EP65b] Paul Erdős and Louis Pósa, *On independent circuits contained in a graph*, Canadian J. Math. **17** (1965), 347–352. 122, 158

[ES35] P. Erdös and G. Szekeres, *A combinatorial problem in geometry*, Compositio Math. **2** (1935), 463–470. 163

[Fel06] Michael R. Fellows, *The lost continent of polynomial time: Preprocessing and kernelization*, Proceedings of 2nd International Workshop on Parameterized and Exact Computation (IWPEC), Lecture Notes in Comput. Sci., vol. 4169, Springer, 2006, pp. 276–277. 22

[FFHV07] Michael R. Fellows, Guillaume Fertin, Danny Hermelin, and Stéphane Vialette, *Sharp tractability borderlines for finding connected motifs in vertex-colored graphs*, Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP), LNCS, vol. 4596, 2007, pp. 340–351. 9, 193

[FFL+09] Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Daniel Raible, Saket Saurabh, and Yngve Villanger, *Kernel(s) for Problems with No Kernel: On Out-Trees with Many Leaves*, Proceedings of the 26th Annual Symposium on Theoretical Aspects of Computer Science (STACS), 2009, pp. 421–432. 182, 194

[FG06] Jörg Flum and Martin Grohe, *Parameterized complexity theory*, Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, Berlin, 2006. 4, 23, 26

[FHL08] Uriel Feige, Mohammadtaghi Hajiaghayi, and James R. Lee, *Improved approximation algorithms for minimum weight vertex separators*, SIAM J. Comput. **38** (2008), no. 2, 629–657. MR MR2411037 (2009g:68267) 93, 100, 105, 125

[FHR+05] Mike Fellows, Pinar Heggernes, Frances Rosamond, Christian Sloper, and Jan Arne Telle, *Finding k disjoint triangles in an arbitrary graph*, Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science, vol. 3353, 2005, pp. 235–244. 184

[FiOT10] Fedor V. Fomin, Sang il Oum, and Dimitrios M. Thilikos, *Rank-width and tree-width of H-minor-free graphs*, Eur. J. Comb. **31** (2010), no. 7, 1617–1628. 144

[FJP10] Samuel Fiorini, Gwenaël Joret, and Ugo Pietropaoli, *Hitting diamonds and growing cacti*, Proceedings of the 14th Conference on Integer Programming and Combinatorial Optimization (IPCO 2010), Lecture Notes in Comput. Sci., vol. 6080, Springer, 2010, pp. 191–204. 123

[FLM+10a] Fedor V Fomin, Daniel Lokshtanov, Neeldhara Misra, Geevarghese Philip, and Saket Saurabh, *An Erdös-Pósa result for generalized cycles*, Unpublished Manuscript, 2010. 10

[FLM+10b] ———, *Lower bounds for some packing problems*, Unpublished Manuscript, 2010. 11

[FLM+11] ———, *Hitting forbidden minors: Approximation and kernelization*, Proceedings of Symposium on Theoretical Aspects of Computer Science, 2011. 10, 155, 156

[FmOT09] Fedor V. Fomin, S.-il Oum, and Dimitrios M. Thilikos, *Rank-width and tree-width of* H*-minor-free graphs*, Manuscript (2009). 144

[FS11] Lance Fortnow and Rahul Santhanam, *Infeasibility of instance compression and succinct PCPs for NP*, Journal of Computer and System Sciences 77 (2011), no. 1, 91 – 106. 176

[GGHN06] Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier, *Data reduction, exact, and heuristic algorithms for clique cover*, Proceedings of the 8th Workshop on Algorithm Engineering and Experiments (ALENEX '06), SIAM, 2006, pp. 86–94. 26, 27

[GJ79] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory ofNP–Completeness*, Freeman, 1979. 183, 184

[GN07] Jiong Guo and Rolf Niedermeier, *Invitation to data reduction and problem kernelization*, SIGACT News **38** (2007), no. 1, 31–45. 4

[GS10] Sylvain Guillemot and Florian Sikora, *Finding and counting vertex-colored subtrees*, Mathematical Foundations of Computer Science, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2010, pp. 405–416. 10, 193

[HN94] Dorit S. Hochbaum and Joseph Naor, *Simple and fast algorithms for linear and integer programs with two variables per inequality*, SIAM J. Comput. **23** (1994), no. 6, 1179–1192. 33

[Kar72] Richard M. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Communications, 1972, pp. 85–103. 123, 198

[Klo94] Ton Kloks, *Treewidth – computations and approximations*, Lecture Notes in Comput. Sci., vol. 842, Springer, 1994. 18, 93, 106, 125

[Kos82] A. V. Kostochka, *The minimum Hadwiger number for graphs with a given mean degree of vertices*, Metody Diskret. Analiz. (1982), no. 38, 37–58. 144

[KT95] P. G. Kolaitis and M. N. Thakur, *Approximation properties of NP minimization classes*, J. Comput. System Sci. **50** (1995), 391–411. 220

[LFS06] Vincent Lacroix, Cristina G. Fernandes, and Marie-France Sagot, *Motif search in graphs: Application to metabolic networks*, IEEE/ACM Transactions on Computational Biology and Bioinformatics **3** (2006), no. 4, 360–368. 9, 193

[LSS09] Daniel Lokshtanov, Saket Saurabh, and Somnath Sikdar, *Simpler parameterized algorithm for OCT*, Combinatorial Algorithms, Lecture Notes in Computer Science, vol. 5874, 2009, pp. 380–384. 184

[LY93] C. Lund and M Yannakakis, *The approximation of maximum subgraph problems*, Proceedings of the 20th International Colloquium Automata, Languages and Programming (ICALP 1993), Lecture Notes in Comput. Sci., vol. 700, Springer, 1993, pp. 40–51. 220

[MPRS10] Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, and Saket Saurabh, *Independent feedback vertex set*, 2010, Accepted at COCOON 2011. 10

[MRS11] Neeldhara Misra, Venkatesh Raman, and Saket Saurabh, *Lower bounds on kernelization*, Discrete Optim. **8** (2011), no. 1, 110–128. 4

[MV80] Silvio Micali and Vijay V. Vazirani, *An* $O(\sqrt{|V|}|E|)$ *Algorithm for Finding Maximum Matching in General Graphs*, Proceedings of the 21st Annual Symposium on Foundations of Computer Science (FOCS), 1980, pp. 17–27. 184

[Nie06] Rolf Niedermeier, *Invitation to Fixed Parameter Algorithms (Oxford Lecture Series in Mathematics and Its Applications)*, Oxford University Press, USA, March 2006. 4, 26, 140

[NSTW06] Serguei Norine, Paul Seymour, Robin Thomas, and Paul Wollan, *Proper minor-closed families are small*, J. Combin. Theory Ser. B **96** (2006), no. 5, 754–757. 144

[NT74] G. L. Nemhauser and L. E. Trotter, Jr., *Properties of vertex packing and independence system polyhedra*, Math. Programming **6** (1974), 48–61. 123

[Ree92] Bruce A. Reed, *Finding approximate separators and computing tree width quickly*, Proceedings of the twenty-fourth annual ACM Symposium on Theory of Computing (STOC), ACM Press, 1992, pp. 221–228. 156

[RS86] Neil Robertson and Paul D. Seymour, *Graph minors. II. Algorithmic aspects of tree-width*, Journal of Algorithms 7 (1986), no. 3, 309–322. 17

[RS95] ———, *Graph minors. XIII. The disjoint paths problem*, J. Comb. Theory Ser. B **63** (1995), 65–110. 129, 170

[RST94] Neil Robertson, Paul D. Seymour, and Robin Thomas, *Quickly excluding a planar graph*, J. Comb. Theory Ser. B **62** (1994), 323–348. 86, 87, 92

[RSV04] Bruce Reed, Kaleigh Smith, and Adrian Vetta, *Finding odd cycle transversals*, Operations Research Letters **32** (2004), no. 4, 299 – 301. 184

[ST93] Paul D. Seymour and Robin Thomas, *Graph searching and a min-max theorem for tree-width*, J. Combin. Theory Ser. B **58** (1993), no. 1, 22–33. 8, 160

[Tho01] Andrew Thomason, *The extremal function for complete minors*, J. Combin. Theory Ser. B **81** (2001), no. 2, 318–338. 144

[Tho10] Stéphan Thomassé, *A quadratic kernel for feedback vertex set*, ACM Transactions on Algorithms **6** (2010), no. 2. 8, 22, 29, 47, 51, 124, 129, 140, 180

[WR08] David R. Wood and Bruce A. Reed, *Polynomial treewidth forces a large grid-like-minor*, Tech. Report arXiv:0809.0724v3, arxiv.org, 2008. 161