Parameterized Algorithms for Finding Large Sparse Subgraphs

Parameterized Algorithms for Finding Large Sparse Subgraphs

Kernelization and Beyond

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de rector magnificus prof.dr.ir. F.P.T. Baaijens, voor een commissie aangewezen door het College voor Promoties, in het openbaar te verdedigen op woensdag 14 september 2022 om 16:00 uur

door

Hubertus Thilman Donkers

geboren te Vlissingen

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

Voorzitter:	prof.dr. J.J. Lukkien
Promotor:	prof.dr. M.T. de Berg
Copromotor:	dr. B.M.P. Jansen
Leden:	prof.dr. H.L. Bodlaender (Universiteit Utrecht)
	dr.hab. M. Pilipczuk (University of Warsaw)
	prof.dr. F.C.R. Spieksma

Het onderzoek of ontwerp dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

Art attribution

All artwork is created by Huib Donkers. Some are derived from existing artwork.

The image on page iv is based on the photograph *Morning Storm II* by Malcolm Bawn. Lighthouses can guide sailors to safety and warn them of danger. They can be the first sign of coming home. The lighthouse in the image symbolizes the help, guidance, and sense of home that I received from the people around me.

The image on page 0 depicts the start of a short game of chess. It relates to the example of assigning chess sets to chess players that I use in the introduction to explain the Vertex Cover problem.

The image on page 10 is based on a still frame from the video series *Rebuilding Tally Ho* by Leo Sampson Goolden. The workshop with tools and a partially built hull relates to the contents of the Chapter *Preliminaries* in which I describe the basic framework and a number of tools on which this thesis relies.

The image on page 22 shows a coastal town with a docked boat. An outerplanar graph can be seen as a road network (without over- or underpasses) where each node is reachable by boat, i.e., they are all adjacent to a single body of water which represents the outer face. The town in the image represents a vertex of such an outerplanar graph.

The image on page 78 displays a Greek temple housing an oracle. A Turing kernelization algorithm transforms a single large problem instance into multiple smaller ones which can be solved by an external algorithm. In the analysis we assume the small instances are solved in constant time by a hypothetical algorithm referred to as an *oracle*.

The image on page 110 and on the cover is based on a photograph from *freepik.com*. The concept of an antler decomposition as introduced in Chapter 5 of this thesis is named after the large acyclic structures attached to a stag's head.

The image on page 150 is based on a photograph from *pixabay.com*. It displays a secluded tree in the literal sense.

The image on page 176 is a drawing of the sailing yacht *De Verrassing* as it sets course to unknown destinations at dusk, the end of the day and the end of this thesis.

Printed by Ipskamp Printing, Enschede.

A catalogue record is available from the Eindhoven University of Technology Library
ISBN: 978-90-386-5550-5

Acknowledgements

This thesis is the result of five years of work. But time hasn't been the only ingredient. This thesis wouldn't have been as it is today without the people that helped and supported me along the way.

First and foremost my supervisor Bart. We worked together on a research project before I started my PhD and I knew I would be in good hands when I asked for you to be my PhD supervisor. Thank you for the help and guidance, our weekly meetings, and for making sure I actually finish things from time to time, rather than endlessly improving them.

Secondly my promotor Mark, thank you for being available for any questions I had and for your suggestions for my thesis. Many thanks also to the other members of my committee, Hans, Marcin, and Frits for reviewing my thesis. In general I want to thank the algorithms community for their (anonymous) constructive feedback on my papers and for sharing ideas and pleasant conversations during workshops, conferences, and summer schools. In particular I would like to thank my co-authors Jari and Michał, I enjoyed working with you.

Luckily the past five years didn't consist only of research. Being among my colleagues made for a pleasant atmosphere that allowed for serious research as well as fun. I want to thank my office mates from before the pandemic, Max, Max, Thom and later Dani, Bram, and Aleksandr, for the laughs and jokes and for watering the plants[1]. Thank you Jari and Shivesh, my new office mates after the pandemic, for brushing up my geography and other trivia. Also thanks to my remote office mates, Michał and Ben, who completed our small FPT research group. Thank you for the interesting, fun, and eventually explosive weekly meetings.

Of course all my other colleagues from the ALGA group played a major role in making my time at the university enjoyable as well, with our daily lunches, occasional boulder or board game nights, and other activities. Thank you Agnes, Aleksandar, Andrés, Arpan, Arthur, Astrid, Bettina, Hans, Henk, Ignaz, Irene, Irina, Jules, Kevin, Kevin, Leonidas, Leonie, Leyla, Marcel, Martijn, Max, Mehran, Meivan, Morteza, Nathan, Pantea, Quirijn, Sándor, Sudeshna, Tim, Tom, Willem, and Wouter.

[1] once a year, roughly

Beyond the academic world, I have been fortunate enough to be surrounded by amazing people that have both supported me in my incomprehensible job as well as distracted me from it. I want to thank my housemates, old and new, with whom I enjoyed Christmas dinners, barbecues, boulder sessions, rollercoasters, slacklining, and so much more. Then, the sailors that make up my sailing team(s) and also the competition, thank you for keeping me sharp in the weekends and sore on Mondays.

Thank you Astrid, for putting up with all my sailing activities and for occasionally taking me away from them to places I hadn't been before, thank you for dealing with my annoying jokes[2] and for making me laugh. You're amazing.

Finally my family, my parents Barbara and Winfried, my brother Jan Maarten, thank you for your support in my work, hobbies, and life in general.

[2] 

Contents

1

Introduction

Even when a problem seems well-suited for a computer to solve, it may be very difficult or impossible to do so efficiently. Consider for example the following problem:

> Imagine you run a chess club. The club has 17 members, 9 chess sets, and a clubhouse. To stop the spread of a pandemic disease it is decided that the clubhouse should no longer be used. Instead you wonder if you can hand out the 9 chess sets to 9 chess players such that any time two players want to play each other, at least one of them has a chess set.

Of course, when we want all pairs of players to be able to play each other, there can only be one player without a chess set, so we would require 16 sets. However if we don't need every two players to be able to play each other, but only a certain list of pairs then it becomes less clear whether 9 chess sets are sufficient. Consider as an example the situation described in Figure 1.1 and try to determine if 9 chess sets is enough.

What you see in Figure 1.1 is a network or *graph*, with the stick figures as *vertices* and the lines between them as *edges*. The problem of determining whether we have enough chess sets is called Vertex Cover. We want to select a set of 9 vertices such that for every edge in the graph, at least one of its endpoints is selected. You may have noticed that this is not an easy task. In fact, this is one of the classical NP-hard problems, which means we expect the computing time and

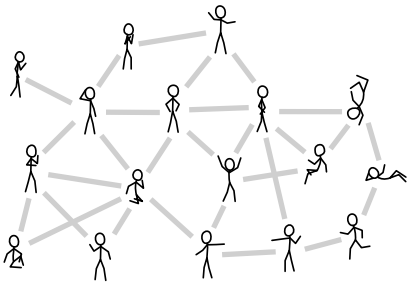


Figure 1.1 Every stick figure represents a member of our chess club. There is a line between two members if they want to play each other, meaning at least one of them needs to have a chess set.

power required to find a solution to increase superpolynomially with the size of the input. In our example this means we cannot expect even the most powerful computers to determine if a large chess club has enough chess sets.

Unfortunately many of the problems we want computers to solve are NP-hard. Accepting that we will not be able to solve large instances of these problems in general, the question becomes "what *can* we achieve to come closer to a solution?" One option is to allow solutions that are suboptimal. *Approximation algorithms* can efficiently find a solution that is slightly suboptimal for varying interpretations of the word slightly. However, if you do not want to compromise on the quality of the solution then this is not an option. An alternative is to simplify the problem input as much as possible, in the hope that it may be simplified to such an extent that even an inefficient algorithm is able to solve it within reasonable time. This approach of simplifying the problem before attempting to solve it is called *preprocessing*.

A preprocessing algorithm aims to modify the input in such a way that the solution remains the same but the new input can be solved faster. With the view that the size of the input is the main factor in the running time, preprocessing is a matter of "compressing" the input, decreasing its size but keeping the essential information that is required to find a solution. The notion of *kernelization* formalizes this approach. A kernelization algorithm is an efficient algorithm (it runs in polynomial time) that compresses the input down to a certain size. A central theme in kernelization research is to determine to what size the input can be compressed. The more we can reduce the size of the problem, the less time is required to solve it.

A kernelization algorithm often works using a number of *reduction rules*, which describe one small simplification of the problem. For the problem of distributing chess sets among the members of our chess club, one such reduction rule could be the following: if one chess player only has one chess partner and never plays anyone else, then we can safely give their partner a chess set (see Figure 1.2).

Any distribution that would assign a chess set to a player with only one partner also works if this partner was assigned the chess set instead. Once we assign this chess set to the partner, we can forget about both these chess players and figure out how to distribute the chess sets among the remaining players: we have made the problem a little bit smaller! We obtain a kernelization algorithm if we can efficiently apply a number of reduction rules like this such that afterwards we can give a guarantee on the size of the remaining problem instance.

It may not be directly clear what sort of guarantee a kernelization algorithm can give on the size of the compressed problem instance. Even something as simple as "the compressed instance will always be smaller than the original instance" does not result in a useful concept of a preprocessing algorithm because such algorithms likely do not exist for NP-hard problems. This is because if there did exist an efficient preprocessing algorithm that can always reduce the size of the input, then we could run this algorithm again on the compressed instance to make it even smaller. If we keep doing this we will reach a point where the compressed input is so small that finding a solution is trivial. So solving the problem could then be done efficiently by repeated preprocessing. Since we are looking specifically at NP-hard problems which we believe cannot be solved efficiently, we immediately also rule out any hope for a preprocessing algorithm that guarantees a reduction in input size.

So what type of guarantee can kernelization give us? The solution comes from *Parameterized Complexity*. Rather than analyze the running time of an algorithm as a function of only the input size n, we can consider other properties of the input that may affect the difficulty of solving the problem. A common property to look at is the size of an optimal solution of the problem. We call such a value the parameter and usually denote it by k. The running time of an algorithm can then be expressed as a function of both n and k. This idea of looking at other properties than only the size of the input allows us to formulate a usable guarantee on how much a kernelization algorithm can compress a problem instance. Namely, we can guarantee that the size of the compressed instance is bounded by a function of this parameter k. So then, regardless of how large the input size is, as long as the parameter k is small, the kernelization algorithm can be used to efficiently compress large problems into small ones.

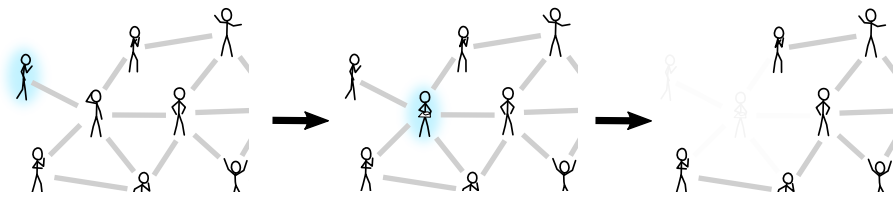


Figure 1.2 A simple reduction rule: find a chess player with only one partner. Give their partner a chess set and continue with a smaller graph.

Parameterized complexity

Fast exact algorithms for NP-hard problems are unlikely to exist. This means that as the size of the problem instance gets larger we can no longer guarantee that the algorithm terminates within reasonable time. This does not mean however that such an algorithm will always fail to deliver an answer to a large problem instance. In parameterized complexity we analyze under what conditions algorithms can solve large instances quickly. To do this, problem instances are associated with a parameter which measures in a way the "complexity" of the instance. With this parameter we can give a more precise guarantee on how long an algorithm takes to solve problem instances of size n with parameter k. This has led to algorithms that can very efficiently solve large NP-hard problem instances under the condition that the parameter is small. If a problem can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$ for some function f, then this problem is said to be *fixed-parameter tractable* (FPT). Such an algorithm runs in polynomial time as long as the parameter is bounded by a constant, and the degree of this polynomial does not depend on this constant.

For many problems the size of a solution gives a natural parameterization. For example if we consider the Vertex Cover example of distributing chess sets parameterized by the number of available chess sets then this problem is fixed-parameter tractable. This means that, even though we concluded earlier that we cannot expect even the most powerful computers to determine whether a large chess club has enough chess sets, we can now be more specific and claim that it is the number of available chess sets rather than the number of members that makes the problem hard.

While considering the size of a solution as parameter often gives positive results, we are completely free to choose another parameterization. Ultimately we want a parameter that overestimates the actual complexity of the problem as little as possible. The natural parameter "solution size" for Vertex Cover for example, hugely overestimates the actual complexity of the problem. Instead parameters that tell something about the structure of the graph have turned out to be better indicators of the complexity of the problem. Examples are the feedback vertex number or treewidth of the graph. These can be arbitrarily smaller than the solution size, meaning that even problem instances with a large solution size can still have a relatively simple structure that allows them to be solved efficiently. Such parameters are often called *structural parameters*.

Finding large sparse subgraphs

In the example problem of distributing chess sets among the members of a chess club, we are looking for a small group of members to hand out chess sets to. From another perspective we are looking for a large group of members that don't need a chess set, because they will never play against another member in this large group. If we consider a graph as in Figure 1.1 but then restricted to only the people that do not receive a chess set, this should be a graph without edges, because such an

edge would then represent two people who want to play each other but neither of which has a chess set. So the problem of distributing chess sets is the same as finding a large, so called, *induced subgraph* that has no edges. The problems studied in this thesis revolve around finding large induced subgraphs that belong to a certain class of graphs.

Edgeless graphs form a very simple class of graphs. A more interesting graph class may be the class of acyclic graphs, graphs that do not contain a cycle. Where edgeless graphs do not have any edges, acyclic graphs can have several edges but they always have fewer edges than they have vertices. In general, if a graph class consists only of graphs that have fewer edges than a constant multiple of the number of vertices, then this is called a *sparse* graph class.

Similar to how edgeless graphs and acyclic graphs are not allowed to contain edges and cycles respectively, we can define other graph classes by giving one or multiple forbidden patterns. For the graph classes studied in this thesis, these forbidden patterns can be defined by a finite set $\mathcal{F}$ containing graphs. The class of graphs that do not contain (as a minor, a variation of subgraph) a forbidden pattern described by $\mathcal{F}$ are called $\mathcal{F}$-minor-free graphs. It turns out that regardless of the choice of $\mathcal{F}$, this graph class is a sparse graph class, i.e., $\mathcal{F}$-minor-free graphs have few edges compared to vertices.

The problem of finding a large induced subgraph that is $\mathcal{F}$-minor free, or equivalently, finding a small set of vertices whose removal makes the graph $\mathcal{F}$-minor free is called $\mathcal{F}$-Minor-Free Deletion. The problems Vertex Cover and Feedback Vertex Set (finding a large acyclic induced subgraph) are two examples of $\mathcal{F}$-Minor-Free Deletion problems. Many other problems can be described using a set of forbidden patterns. Observing this as a common theme among these problems often allows us to generalize results and techniques developed from one problem to be applied to other $\mathcal{F}$-Minor-Free Deletion problems.

Beyond kernelization

In this thesis we study several $\mathcal{F}$-Minor-Free Deletion problems using different algorithmic paradigms. The notion of kernelization described before gives us the first rigorous preprocessing framework. A kernelization algorithm is useful as a preprocessing step to run before solving the problem because it reduces the size of the problem instance. If the reduced instance is small enough, this means that it can be solved in reasonable time afterwards. The notion of *Turing kernelization* extends this idea of reducing the input size as a preprocessing step. Instead of requiring that the result of solving the reduced instance tells us the answer to the original problem, a Turing kernelization may take this result and give us another small instance to solve instead. It may continue to do this polynomially many times, after which it should produce the final solution. In this thesis we actually describe a slightly simpler version of a Turing kernelization which does not have to wait for the result of the small problem instances, but instead it produces a list of small problem instances whose results are easily combined into a solution to the

original problem.

Both traditional kernelization and Turing kernelization focus on reducing the *size* of the instances to be solved. This works well together with the idea that problem instances of small enough size can be solved in reasonable time. However, as we have seen in parameterized complexity, the size of the instances may not always play the largest role in the running time. Instead, the value of the parameter largely determines whether an instance can be solved in reasonable time. Although research in kernelization and its variations have resulted in fast practical algorithms and effective reduction rules, this should mainly be attributed to reducing, in a sense, the complexity of the problem rather than just its size. We see that many of the successful reduction rules often succeed in reducing not only the problem size, but also the parameter. The reduction rule we gave earlier for distributing chess sets reduces the number of chess sets to be distributed for example.

Adhering to the strict framework of kernelization algorithms with their goal of reducing the size of a problem instance may not always lead us towards such reduction rules that prove so effective in practice, resulting in a significant speed up for solving the preprocessed problem instance. Taking inspiration from reduction rules such as the one described earlier, we shift our focus from reducing the size of a problem to reducing the parameter.

In this thesis we apply several algorithmic techniques to find large sparse subgraphs. The next sections describe the techniques and problems discussed in each chapter of this thesis.

Kernelization for outerplanar vertex deletion

In Chapter 3 we investigate kernelization for Outerplanar Vertex Deletion. An outerplanar graph is a graph that can be embedded in the plane such that all vertices are incident to the outer face, see Figure 1.3 for an example. In the Outerplanar Vertex Deletion problem the objective is to find a small set of

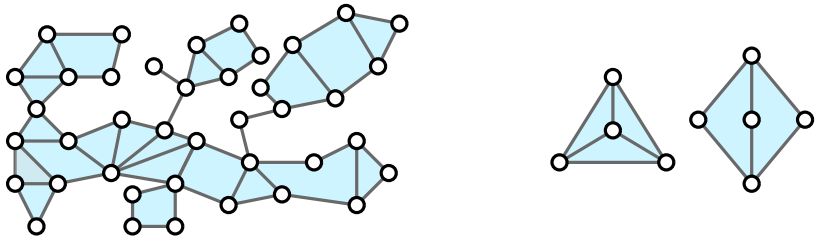


Figure 1.3 On the left is an example of an outerplanar graph, drawn on the plane without crossing edges. This divides the plane into bounded faces (marked blue) and one unbounded region (marked white) called the outer face. All vertices are incident to the outer face. On the right are (the only) two forbidden patterns in outerplanar graphs.

vertices whose deletion makes the graph outerplanar. Since a graph is outerplanar if and only if it does not contain one of two forbidden patterns, this problem can be described as an $\mathcal{F}$-Minor-Free Deletion problem.

It was already known [61] that a large collection of $\mathcal{F}$-Minor-Free Deletion problems, including Outerplanar Vertex Deletion, admit a polynomial kernel when parameterized by the solution size. However it was unknown what this polynomial bound on the size of the kernel is. We study specifically the Outerplanar Vertex Deletion problem and using several algorithmic techniques we present a new kernelization algorithm that reduces the size of the problem instance to $\mathcal{O}(k^4)$. This includes several new reduction rules for Outerplanar Vertex Deletion and how they can be applied efficiently. In proving the correctness of these reduction rules we uncover a number of structural properties of outerplanar graphs.

A Turing kernelization dichotomy for $\mathcal{F}$-Minor-Free Deletion

In Chapter 4 of this thesis we study the applicability of Turing kernelization to the class of all $\mathcal{F}$-Minor-Free Deletion problems. We prove, under some complexity theoretic assumptions, that for a certain structural parameterization of $\mathcal{F}$-Minor-Free Deletion there does not exist a polynomial Turing kernel for all choices of $\mathcal{F}$, apart from a special case that we identify. This means for example that we do not expect a polynomial Turing kernelization for $\{P_3\}$-Minor-Free Deletion parameterized by the feedback vertex number. The remaining choices of $\mathcal{F}$, for which this hardness result does not apply, are relatively limited in their complexity. This allows us to give a polynomial Turing kernelization for these remaining choices of $\mathcal{F}$, completing the dichotomy.

Finding antler structures to solve Feedback Vertex Set

As discussed, kernelization and its variations aim to reduce the problem in size. However, the size of the input may not actually play the largest role in the time it takes to find a solution. For fixed-parameter tractable problems such as $\mathcal{F}$-Minor-Free Deletion, the value of the parameter typically has the largest influence on the running time of the algorithm rather than the size of the input. In Chapter 5 we investigate the possibility of an efficient preprocessing algorithm that reduces the parameter instead of the input size. It is not clear how to formalize such a preprocessing step. We cannot guarantee to always strictly reduce the parameter in polynomial time just like we cannot guarantee to always strictly reduce the size of the problem in polynomial time: such a preprocessing step could be applied iteratively resulting in a trivial instance, meaning the problem can be solved in polynomial time, something we believe is impossible for NP-hard problems. We ask ourselves, under what conditions can we guarantee to reduce the parameter in polynomial time?

For the problem of Feedback Vertex Set parameterized by the solution size, we describe a certain graph structure which we call *antler* and show that the presence of sufficiently simple antlers in the graph allows us to efficiently reduce the parameter (the size of a solution). Finding an antler structure in the graph allows us to identify a set of vertices that is part of an optimal solution. This reduces the problem to finding the remaining part of a solution which is strictly smaller. This way the solution size can be reduced by finding antler structures.

Finding secluded sparse graphs

Chapters 3 to 5 investigate preprocessing methods $\mathcal{F}$-Minor-Free Deletion problems. In Chapter 6 we consider a related problem, finding a large sparse *secluded* subgraph. A subgraph is k-secluded when it has at most k neighbors. For example the reduction rule described in Figure 1.2 starts with finding a 1-secluded vertex. In fact, the reduction rules discussed in Chapters 3 and 5 and many other reduction rules for $\mathcal{F}$-Minor-Free Deletion problems work in secluded subgraphs that belong to a certain graphs class. In Chapter 6 we consider the problem of finding a maximum size secluded subgraph that belongs to one of the most elementary classes of sparse graphs: trees.

Finding maximum size k-secluded trees is known to be FPT when parameterized by k [67], but the known algorithm has a running time with a doubly exponential dependency on the parameter [66]. We improve this result and present an algorithm with running time $2^{\mathcal{O}(k \log k)} n^4$, which not only finds *a* maximum size k-secluded tree, but it can be used to identify *all* maximum size k-secluded trees in a graph.

Secluded trees are closely related to antler structures as discussed in Chapter 5, since an antler can be shown to consist of a small number of secluded trees. Research into secluded graph structures may open pathways to parameter reducing reduction rules using local certificates of optimality like presented in Chapter 5.

List of publications

Chapter 3 is based on the following publication.

> Huib Donkers, Bart M. P. Jansen, and Michał Włodarczyk. Preprocessing for outerplanar vertex deletion: An elementary kernel of quartic size. In Petr A. Golovach and Meirav Zehavi, editors, *IPEC 2021, September 8-10, 2021, Lisbon, Portugal*, volume 214 of *LIPIcs*, pages 14:1–14:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.IPEC.2021.14.

Chapter 4 is based on the following publication.

> Huib Donkers and Bart M. P. Jansen. A Turing kernelization dichotomy for structural parameterizations of $\mathcal{F}$-minor-free deletion. *Jour-*

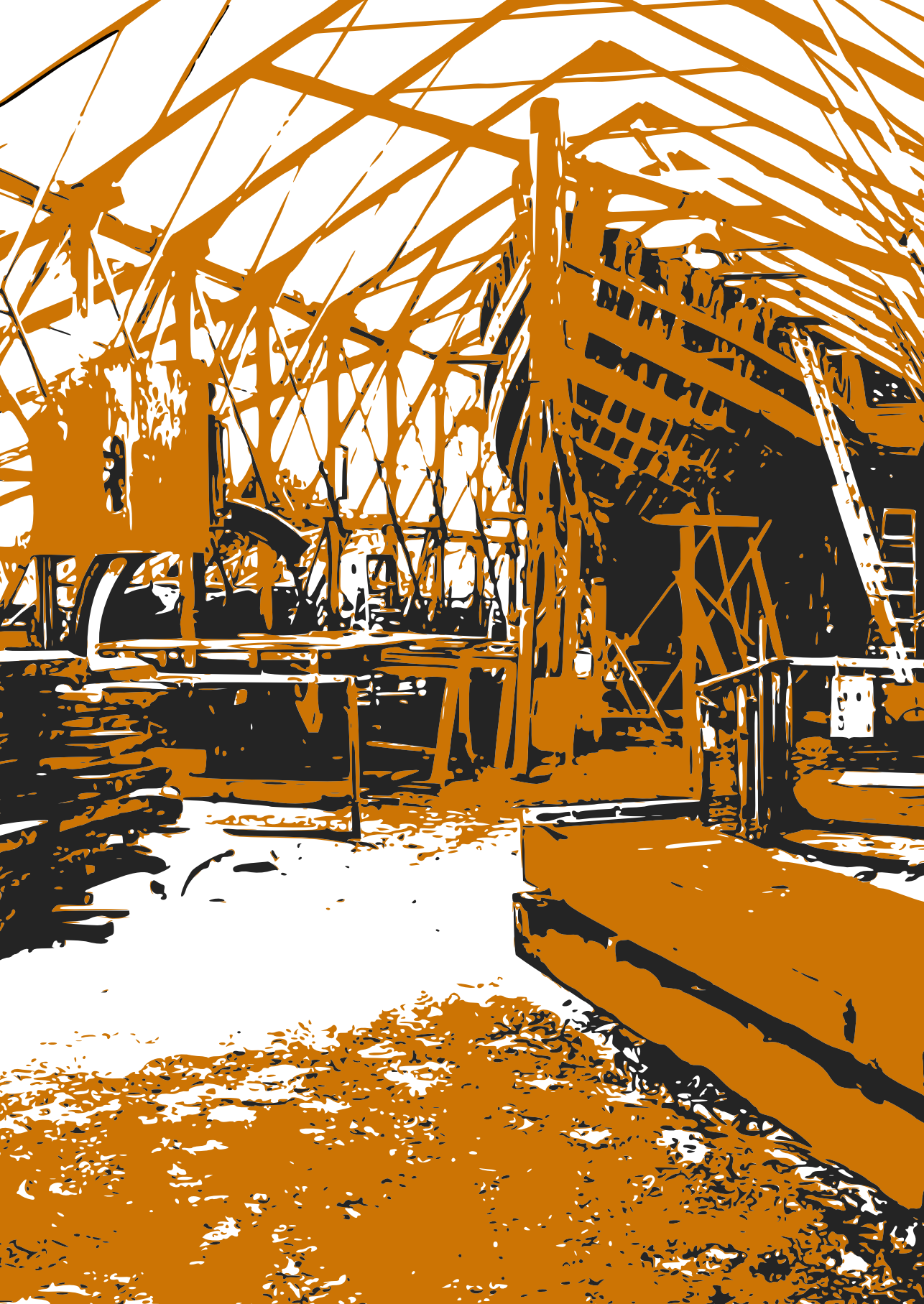
nal of Computer and System Sciences, 119:164–182, 2021. doi:10.1016/j.jcss.2021.02.005.

Chapter 5 is based on the following publication, which was awarded the Best Paper and Best Student Paper at WG 2021.

Huib Donkers and Bart M. P. Jansen. Preprocessing to reduce the search space: Antler structures for feedback vertex set. In Lukasz Kowalik, Michał Pilipczuk, and Paweł Rzążewski, editors, *WG 2021, Warsaw, Poland, June 23-25, 2021, Revised Selected Papers*, volume 12911 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2021. doi:10.1007/978-3-030-86838-3_1.

Chapter 6 is based on the following publication.

Huib Donkers, Bart M.P. Jansen, and Jari J.H. de Kroon. Finding k-secluded trees faster. In *WG 2022*. To appear.



2

Preliminaries

2.1 General notation

For a positive integer ℓ we denote the set $\{1, \ldots, \ell\}$ by $[\ell]$, with $[0] = \emptyset$. Let $X \times Y$ denote the Cartesian product of the sets X and Y defined as $\{(x, y) \mid x \in X, y \in Y\}$. For a set X and an integer $\ell > 0$ let X^ℓ denote the ℓ-ary Cartesian power of X defined as $\{(x_1, \ldots, x_\ell) \mid x_i \in X \text{ for all } i \in [\ell]\}$. Let 2^X denote the powerset of a set X consisting of all subsets of X. For a set X and integer ℓ let $\binom{X}{\ell}$ denote the family of all subsets $X' \subseteq X$ with $|X'| = \ell$. For any family of sets $X_1, \ldots, X_\ell$ indexed by $[\ell]$ we define for all $i \in [\ell]$ the following:

$$X_{<i} = \bigcup_{1 \leq j < i} X_j \ , \qquad X_{>i} = \bigcup_{i < j \leq \ell} X_j \ ,$$
$$X_{\leq i} = \bigcup_{1 \leq j \leq i} X_j \ , \text{ and} \qquad X_{\geq i} = \bigcup_{i \leq j \leq \ell} X_j \ .$$

For a function $f\colon A \to B$, let $f^{-1}\colon B \to 2^A$ denote the *preimage function* of f, that is $f^{-1}(a) = \{b \in B \mid f(b) = a\}$. For a set $A' \subseteq A$, we use $f(A')$ as a shorthand for $\bigcup_{a \in A'} f(a)$, and similarly $f^{-1}(A') = \bigcup_{a \in A'} f^{-1}(a)$.

2.2 Graph theory

All graphs considered in this thesis are finite and undirected. With the exception of Chapter 5 all graphs used in this thesis are simple graphs. In Chapter 5 we use a generalization of simple graphs called multigraphs, which we formally define in Section 5.2.

A *simple graph* G is a pair $(V(G), E(G))$ of its vertex set $V(G)$ and edge set $E(G)$. When the graph is clear from context we use n and m to denote $|V(G)|$ and $|E(G)|$ respectively. An *edge* in a graph is a set of two vertices that form its endpoints, i.e. $E(G) \subseteq \binom{V(G)}{2}$. This means a simple graph has no self-loops (an edge of which both endpoints are the same vertex) and no multi-edges (multiple edges between two vertices). A multigraph differs from a simple graph precisely on these two points. For brevity we sometimes write uv to denote the edge $\{u, v\}$ of a simple graph. For an edge $e = uv \in E(G)$ let $V(e)$ denote the set of vertices that form its endpoints, i.e., $V(e) = \{u, v\}$.

We call the graph on the empty vertex set the *null graph*. For a non-negative integer n we use $n \cdot G$ to denote the graph consisting of n disjoint copies of G.

Subgraphs For a vertex set $S \subseteq V(G)$ let $G[S]$ be the subgraph of G induced by S, that is, $G[S] = (S, E(G) \cap \binom{S}{2})$. We use $G - S$ denote the subgraph of G induced by $V(G) \setminus S$. For a vertex $v \in V(G)$ we use $G - v$ as shorthand for $G - \{v\}$. For an edge set $A \subseteq E(G)$ we denote by $G \setminus A$ the graph with vertex set $V(G)$ and edge set $E(G) \setminus A$. For $e \in E(G)$ we write $G \setminus e$ as a shorthand for $G \setminus \{e\}$.

Neighborhood and degree Let $N_G(v)$ denote the open neighborhood in G of a vertex v defined as $N_G(v) = \{u \in V(G) \mid uv \in E(G)\}$. For $S \subseteq V(G)$ we define the open neighborhood of S as $N_G(S) = \bigcup_{v \in S} N_G(v) \setminus S$. The closed neighborhood of a single vertex v is $N_G[v] = N_G(v) \cup \{v\}$, and the closed neighborhood of a vertex set S is $N_G[S] = N_G(S) \cup S$.

The *degree* $\deg_G(v)$ of a vertex v in a graph G is the number of edge-endpoints incident to v, i.e., $\deg_G(v) = |\{e \in E(G) \mid v \in e\}| = |N_G(v)|$.

For (not necessarily disjoint) vertex sets $A, B \subseteq V(G)$, we define $E_G(A, B)$ as the set of edges with one endpoint in A and the other in B, i.e., $E_G(A, B) = \{uv \in E(G) \mid u \in A, v \in B\}$.

Common abbreviations In all notation using the graph as subscript, we may omit the subscript if the graph is clear from the context. To simplify the presentation, in expressions taking one or more vertex sets as parameter such as $N_G(..)$ and $E_G(.., ..)$, we sometimes use a subgraph H of G as argument as a shorthand for the vertex set $V(H)$ that is formally needed. In general, for functions taking as argument a vertex set, we may sometimes give a graph as argument instead of its vertex set. One common example of this occurs when using a weight function $w \colon V(G) \to \mathbb{N}^+$ assigning weights to vertices in G. If H is

a subgraph of G, then using the shorthand defined earlier we may denote $\bigcup_{v \in V(H)}$ as $w(V(H))$ which we may abbreviate further to $w(H)$.

2.2.1 Structural graph properties

For $\ell \geq 1$ let P_ℓ denote the graph $(\{v_1, \ldots, v_\ell\}, \{\{v_i, v_{i+1}\} \mid i \in [\ell - 1]\})$. A *path* on ℓ vertices is any graph isomorphic to P_ℓ. To describe a path we may give a sequence of vertices or a sequence of edges instead of describing the entire graph. A path between two vertices u and v is called a uv-path. A graph G is connected if for any two vertices $a, b \in V(G)$ there is an ab-path in G. A *connected component* (sometimes referred to simply as *component*) of a graph is an inclusion-wise maximal subgraph that is connected. For a graph G and vertex set $A \subseteq V(G)$, we say A is connected in G if $G[A]$ is connected. When G is clear from context we simply say A is connected.

A graph that is acyclic is called a *forest*, and if the graph is also connected we call it a *tree*. A *rooted tree* is a tree with a specified root vertex. For vertices u, v in a tree T with root r, we say that u is an *ancestor* of v (or equivalently, v is a *descendant* of u) if u lies on the (unique) path from r to v.

A vertex set $X \subseteq V(G)$ is an *independent set* in G if $E_G(X, X) = \emptyset$. A *vertex cover* in G is a set of vertices $Y \subseteq V(G)$ such that $V(G) \setminus Y$ is an independent set, i.e., every edge in $E(G)$ has at least one endpoint in Y. The *vertex cover number* $\mathsf{vc}(G)$ of a graph G is the smallest size of a vertex cover in G. A graph is *bipartite* if there is a partition of $V(G)$ into two independent sets A and B. A *feedback vertex set* (FVS) in a graph G is a vertex set $X \subseteq V(G)$ such that $G - X$ is acyclic. The *feedback vertex number* of a graph G, denoted by $\mathsf{fvs}(G)$, is the minimum size of a FVS in G.

A vertex $v \in V(G)$ is a *cut vertex* in G if $G - v$ has more connected components than G. A graph G is *biconnected* when it is connected and has no cut vertices. A graph G is 2-*connected* when it is biconnected and has at least three vertices, or equivalently, when for every pair of vertices $u, v \in V(G)$ there exists a cycle in G going through both u and v. A *biconnected component* of a graph is an inclusion-wise maximal subgraph which is biconnected.

If X is the set of cut vertices in a graph G and Y is the set of biconnected components of G, then consider the graph T with a vertex v_B for each biconnected component B in G, a vertex v_c for each cut vertex c in G, and an edge between a vertex v_B and v_c if and only if $c \in V(B)$. It can easily be seen that T is bipartite, but one can also show the stronger claim that T is acyclic [42, Lemma 3.1.4]. For this reason we call T the *block-cut forest*, or when G is connected, a *block-cut tree*.

A graph G is a *clique*, or a *complete graph*, if $E(G) = \binom{V(G)}{2}$. For an integer q, the graph K_q is the complete graph on q vertices. For integers p, q, the graph $K_{p,q}$ is the bipartite graph $(A \cup B, E)$, where $|A| = p$, $|B| = q$, and $uv \in E$ whenever $u \in A$ and $v \in B$.

A *leaf* of a (possibly cyclic) graph G is a vertex v for which $\deg_G(v) \leq 1$. A

vertex v in a graph G is said to be a *universal vertex* in G if for all $u \in V(G) \setminus \{v\}$ we have $uv \in E(G)$.

Minors A contraction of $uv \in E(G)$ introduces a new vertex adjacent to all of $N_G(\{u, v\})$, after which u and v are deleted. For $A \subseteq V(G)$ such that $G[A]$ is connected, contracting A into a new vertex v replaces all vertices of A by the single vertex v with $N_G(v) = N_G(A)$, this is equivalent to exhaustively contracting the edges in $G[A]$.

A graph H is a *minor* of a graph G if H can be obtained from G by a (possibly empty) series of edge contractions, edge deletions, and vertex deletions. If this series is non-empty then H is called a *proper minor* of G. A *minor model* of H (sometimes abbreviated to H-model) in G is a function $\varphi\colon V(H) \to 2^{V(G)}$ such that

1. for every vertex $v \in V(H)$, the graph $G[\varphi(v)]$ is connected,
2. for distinct $u, v \in V(H)$ we have $\varphi(v) \cap \varphi(u) = \emptyset$, and
3. for every edge $uv \in E(H)$ we have $|E_G(\varphi(u), \varphi(v))| \geq 1$.

The sets $\varphi(v)$ are called *branch sets*. Clearly, H is a minor of G if and only if there is an H-model in G.

Recall that since an H-model $\varphi\colon V(H) \to 2^{V(G)}$ is a function, we may use the shorthand notation $\varphi(S) = \bigcup_{v \in S} \varphi(v)$ for a vertex set $S \subseteq V(H)$ and $\varphi(H') = \varphi(V(H'))$ for a subgraph H' of H. An H-model φ is called *minimal* if there does not exist an H-model φ' with $\varphi'(H) \subsetneq \varphi(H)$.

Treewidth A *tree decomposition* of graph G is a pair (T, χ) consisting of a rooted tree T and a function $\chi\colon V(T) \to 2^{V(G)}$, such that:

1. for each $v \in V(G)$ the nodes $\{t \mid v \in \chi(t)\}$ induce a non-empty connected subtree of T, and
2. for each edge $uv \in E(G)$ there is a node $t \in V(T)$ with $\{u, v\} \subseteq \chi(t)$.

The sets $\chi(t)$ for $t \in V(T)$ are called *bags*. The width of a tree decomposition is the size of the largest bag minus one, i.e., $\max_{t \in V(T)} |\chi(t)| - 1$. The *treewidth* of a graph G, denoted by $\mathsf{tw}(G)$, is the minimum width of a tree decomposition of G. If w is a constant, then there is a linear-time algorithm that given a graph G either outputs a tree decomposition of width at most w or correctly concludes that treewidth of G is larger than w [18].

Graph classes In Chapter 1 we have loosely introduced the idea of sparse graph classes. A graph class is a collection of graphs. Interesting graph classes often contain infinitely many graphs that satisfy a certain property, such as being acyclic or biconnected. A graph class $\mathcal{G}$ is *sparse* if there exists a function $f(n) \in \mathcal{O}(n)$

such that for all graphs $G \in \mathcal{G}$ we have that $|E(G)| \leq f(|V(G)|)$. Since an acyclic graph has fewer edges than vertices, the class of acyclic graphs is a sparse graph class. On the other hand, for any integer $n \geq 0$ there is a biconnected graph with n vertices and $\binom{n}{2} = \Theta(n^2)$ edges (namely K_n), so the class of all biconnected graphs is not a sparse graph class.

Planar graphs are graphs that can be embedded in the plane, i.e., they can be drawn on the plane without crossing edges. In such an embedding, each $v \in V(G)$ is mapped to a point p_v on the plane and each edge $uv \in E(G)$ to a curve with endpoints p_u and p_v whose interior does not intersect any other point or curve of the drawing. This divides the plane into one or more connected regions, one of which is infinite. These regions are called *faces* and the region that is infinite is called the *outer face*. Planar graphs on $n \geq 3$ vertices have at most $3n - 6$ edges, hence the class of planar graphs is a sparse graph class. *Outerplanar graphs* (discussed in detail in Chapter 3) are planar graphs that have an embedding in the plane where all vertices are incident to the outer face. Outerplanar graphs on n vertices have at most $2n$ edges [19, Lemma 78, Lemma 91].

Many of the sparse graph classes considered in this thesis can be described by a set of forbidden minors. If $\mathcal{F}$ is a set of graphs then a graph that does not contain any graph of $\mathcal{F}$ as a minor is called *$\mathcal{F}$-minor free*. When $\mathcal{F} = \{H\}$ is a singleton set, we often write *H-minor free*. The class of all K_3-minor free graphs is precisely the class of all acyclic graphs meaning it is a sparse graph class. In fact for any $\mathcal{F}$, the class of all $\mathcal{F}$-minor free graphs is a sparse graph class [99].

Similar to $\mathcal{F}$-minor free graphs, a graph that does not contain any graph of $\mathcal{F}$ as a subgraph is called *$\mathcal{F}$-subgraph free*. However the class of all $\mathcal{F}$-subgraph free graphs is not necessarily a sparse graph class. For example, for all integers $n > 0$ there is a K_3-subgraph free graph on $2n$ vertices with $\frac{1}{2}n^2$ edges (namely $K_{n,n}$), so the class of all K_3-subgraph free graphs is not a sparse graph class.

Of particular interest in this thesis are graph classes that are minor-closed. A graph class is *minor-closed* if for any graph G in the graph class, any minor of G is also contained in the graph class. The class of planar graphs is minor-closed since, intuitively, if a graph can be drawn on the plane without edge crossings, then after contracting edges or removing vertices or edges, it can still be drawn on the plane without edge crossings. Similarly, the class of outerplanar graphs is minor-closed as well.

An important result in graph theory is the *graph minor theorem* by Robertson and Seymour [113], which implies that a minor-closed graph class can be characterized by a finite set of forbidden minors. That is, for any minor-closed graph class $\mathcal{G}$, there is a finite set of graphs $\mathcal{F}$ such that G is $\mathcal{F}$-minor free if and only if $G \in \mathcal{G}$. For the class of planar graphs these forbidden minors are well known to be K_5 and $K_{3,3}$ [125]. These forbidden minors are sometimes called *obstructions* to planarity. For outerplanar graphs the forbidden minors, or obstructions to outerplanarity, are K_4 and $K_{2,3}$ [42].

Since the treewidth of a graph cannot increase when taking minors, the graph

class consisting of all graphs G with $\mathsf{tw}(G) \leq \eta$ for some constant η is a minor-closed graph class, and can therefore be described as an $\mathcal{F}$-minor free graph class using a specific collection of forbidden minors $\mathcal{F}$ depending on η. Determining the set $\mathcal{F}$ of forbidden minors for a given treewidth η is difficult and it is known that the number of graphs in $\mathcal{F}$ increases at least exponentially with η [109], but possibly much faster. However we do know that at least one of the graphs in $\mathcal{F}$ is planar. This follows from the fact that the η by η grid graph[1] is planar and has treewidth exactly η. Conversely, the *grid minor theorem* by Robertson and Seymour [111] shows that there is a function f such that any graph G with $\mathsf{tw}(G) \geq f(n)$ contains $\boxplus_n$ as a minor, for any n. Since any planar graph is a minor of a large enough grid, this also implies that there is a bound on the treewidth of graphs that do not contain a fixed planar graph as minor. Combining these results, it follows that $\mathcal{F}$-minor-free graphs have bounded treewidth if and only if $\mathcal{F}$ contains a planar graph.

2.3 Hitting forbidden minors

A vertex set X in a graph G for which $G - X$ belongs to a certain graphs class is called a *modulator* to this graph class, e.g., a modulator to planarity is a set of vertices to remove from the graph such that the remaining graph is planar, or $\{K_5, K_{3,3}\}$-minor free. Usually the graph class in consideration is clear from context and we may use the single word "modulator" to describe the set.

Many of the algorithmic problems central to this thesis are about determining whether a graph has a small modulator to a class of $\mathcal{F}$-minor free graphs. Such a modulator "hits", so to speak, all forbidden $\mathcal{F}$-minors in the given graph G, such that $G - X$ is $\mathcal{F}$-minor free. These problems of hitting forbidden minors are $\mathcal{F}$-Minor-Free Deletion problems, defined for any fixed finite family of graphs $\mathcal{F}$ as follows.

$\mathcal{F}$-Minor-Free Deletion **Input:** A graph G and an integer k **Question:** Does there exist a vertex set $X \subseteq V(G)$ of at most k vertices such that $G - X$ is $\mathcal{F}$-minor free?

In the context of an $\mathcal{F}$-Minor-Free Deletion problem, we may call a modulator X of size at most k a *solution*.

Many classic problems in the field of algorithms can be expressed as an $\mathcal{F}$-Minor-Free Deletion problem. For example Vertex Cover and Feedback Vertex Set are equivalent to $\{K_2\}$-Minor-Free Deletion and $\{K_3\}$-Minor-Free Deletion respectively. The $\{K_5, K_{3,3}\}$-Minor-Free Deletion and $\{K_4, K_{2,3}\}$-Minor-Free Deletion problems are commonly referred to as Planar Deletion and Outerplanar Vertex Deletion. The $\mathcal{F}$-Minor-

[1] The *grid graph* $\boxplus_\eta$ has vertex set $\{v_{a,b} \mid a, b \in [\eta]\}$ and edge set $\{\{v_{a,b}, v_{a',b'}\} \mid a, a', b, b' \in [\eta], |a - a'| + |b - b'| = 1\}$).

Free Deletion problem in general has been studied extensively in recent years [12, 60, 61, 65, 81, 86].

2.4 Algorithms and complexity

A *decision problem* is commonly defined in complexity theory as a language over a finite alphabet Σ. Such a language is a set $\Pi \subseteq \Sigma^*$ of finite strings consisting of elements from Σ. An instance $x \in \Sigma^*$ of a problem Π is a yes-instance if $x \in \Pi$ and a no-instance otherwise. Decision problems can be categorized in complexity classes such as P, NP and coNP. These classes are at the bottom of a hierarchy of infinitely many classes closed under polynomial-time many-one reductions. This is called the polynomial hierarchy (cf. [9, Chapter 5]). It is conjectured that all these classes are distinct. We assume the reader is familiar with polynomial-time reductions and the classes P, NP, and coNP.

The problems discussed in this thesis are NP-hard, in particular, if each graph in $\mathcal{F}$ contains at least one edge, it follows from the general results of Lewis and Yannakakis [93] that $\mathcal{F}$-Minor-Free Deletion is NP-hard.

Approximation We define the notion of an approximation algorithm in the context of $\mathcal{F}$-Minor-Free Deletion problems. For any $\mathcal{F}$ and any graph G, let $\mathsf{opt}_{\mathcal{F}}(G)$ denote the minimum number of vertex deletions to make the graph G $\mathcal{F}$-minor free, i.e., an instance (G, k) of the $\mathcal{F}$-Minor-Free Deletion problem is a yes-instance if and only if $k \geq \mathsf{opt}_{\mathcal{F}}(G)$.

For a constant $\alpha > 1$, an *α-approximation algorithm* for $\mathcal{F}$-Minor-Free Deletion is an algorithm that takes as input a graph G and returns a vertex set X of size at most $\alpha \cdot \mathsf{opt}_{\mathcal{F}}(G)$ such that $G - X$ is $\mathcal{F}$-minor free. Approximation algorithms are useful when they have a substantially better running time than the best exact algorithms. In this thesis we only consider polynomial-time approximation algorithms for NP-hard problems. Vertex Cover and Feedback Vertex Set both admit a polynomial-time 2-approximation algorithm, and in general, all $\mathcal{F}$-Minor-Free Deletion problems admit a polynomial-time α-approximation algorithm for some constant α [61].

Parameterized complexity A *parameterized decision problem* is a language $\Pi \subseteq \Sigma^* \times \mathbb{N}$, where each instance $(x, k) \in \Sigma^* \times \mathbb{N}$ has a parameter k that captures its complexity in some well-defined way. For example, an instance (x, k) of Vertex Cover (or any $\mathcal{F}$-Minor-Free Deletion problem) parameterized by solution size is a yes-instance if x correctly encodes a graph G and G has a vertex cover (or a modulator to an $\mathcal{F}$-minor free graph) of size at most k. For structural parameterizations we consider as part of the input a witness for the parameter. For example, an instance (x, k) of $\mathcal{F}$-Minor-Free Deletion parameterized by the feedback vertex number is a yes-instance if x encodes a graph G, an integer ℓ,

and a feedback vertex set X of G with $|X| = k$, and if G has a modulator S to an $\mathcal{F}$-minor-free graph with $|S| \leq \ell$ (cf. [53, § 2.2]).

The size $|(x, k)|$ of an instance (x, k) of a parameterized problem is conventionally defined as $|x| + k$, which corresponds to encoding k in unary. This is how parameterized decision problems correspond to (classical) decision problems. For a parameterized decision problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ the corresponding (classical) decision problem is a language $\Pi' \subseteq (\Sigma \cup \{\#\})^*$ over the alphabet Σ extended with a separator symbol $\#$, where Π' contains, for each $(x, k) \in \Pi$, the string x followed by $\#$ followed by k times some fixed symbol $1 \in \Sigma$. When we say a parameterized decision problem belongs to some class of (classical) decision problems we formally mean that the decision problem corresponding to the parameterized decision problem belongs to that class. The problems and parameterizations considered in this thesis are always such that if (x, k) is a YES-instance then $k \leq |x|$, so we can immediately remove any instance (x, k) with $k > |x|$ from consideration. Hence the value and encoding of k never affects the asymptotic bounds on the size of a problem instance and we may use $|x|$ in asymptotic bounds rather than $|(x, k)| = |x| + k$.

We say that a parameterized problem Π is *fixed-parameter tractable* (FPT) if there is an algorithm that given a parameterized instance $(x, k) \in \Sigma^* \times \mathbb{N}$, decides whether $(x, k) \in \Pi$ in time $f(k) \cdot |x|^{\mathcal{O}(1)}$ for some computable function f. For problems that are FPT, such algorithms allow NP-hard problems to be solved efficiently on instances whose parameter is small. The class of FPT problems is closed under parameterized reductions, defined as follows.

Definition 2.4.1 ([35, Definition 13.1])**.** Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized decision problems. A *parameterized reduction* from A to B is an algorithm that, given an instance (x, k) of A, outputs an instance (x', k') of B such that

- $(x, k) \in A$ if and only if $(x', k') \in B$,
- $k' \leq g(k)$ for some computable function g, and
- the running time is $f(k) \cdot |x|^{\mathcal{O}(1)}$ for some computable function f.

It can be seen that if there is a parameterized reduction from a parameterized decision problem A to a parameterized decision problem B, then existence of an FPT algorithm for B immediately yields an FPT algorithm for A. A hierarchy of complexity classes $\mathsf{W}[0] \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \ldots$ can be defined using parameterized reductions, where each class is closed under parameterized reductions. Important in this thesis are the classes $\mathsf{W}[0]$ and $\mathsf{W}[1]$. The class $\mathsf{W}[0]$ contains precisely those parameterixed decision problems that are FPT, and this class is therefore commonly referred to as FPT. The class $\mathsf{W}[1]$ contains all decision problems for which a parameterized reduction to k-Clique exists; k-Clique is $\mathsf{W}[1]$-complete. A parameterized decision problem Π is $\mathsf{W}[1]$-hard if there is a parameterized reduction from k-Clique to Π. This implies that if any $\mathsf{W}[1]$-hard problem can be shown to be FPT, then all problems in $\mathsf{W}[1]$ are FPT. It is widely believed

that W[1]-hard problems do not admit an FPT algorithm. For a more elaborate introduction to the field we refer to the books Parameterized Complexity [47] and Parameterized Algorithms [35].

All of the $\mathcal{F}$-Minor-Free Deletion problems are fixed-parameter tractable when parameterized by k [115] and also when parameterized by treewidth [34].

Kernelization A kernelization algorithm is a polynomial-time preprocessing algorithm that takes as input a parameterized problem instance and returns an equivalent parameterized problem instance that has a bounded size. Formally, for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ and a function $f\colon \mathbb{N} \to \mathbb{N}$, a *kernelization algorithm* for Π of size f is an algorithm that, on input $(x, k) \in \Sigma^* \times \mathbb{N}$, takes time polynomial in $|x| + k$ and outputs $(x', k') \in \Sigma^* \times \mathbb{N}$ such that

1. $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$, and
2. both $|x'| \leq f(k)$ and $k' \leq f(k)$.

The term "kernelization algorithm" is sometimes abbreviated to "kernelization", or simply "kernel".

It can easily be seen that if there exists a kernelization algorithm for a decidable parameterized problem then it can be combined with a brute force algorithm to obtain an FPT algorithm. So a decidable problem is FPT if there exists a kernelization algorithm for it. The converse is also true, if a problem is fixed-parameter tractable, i.e., it can be solved in $f(k) \cdot n^c$ time for some computable function f and constant c, then a kernel can be obtained by running this algorithm for n^{c+1} steps. If this produces an answer then a trivial instance of constant size can be returned reflecting this answer. If this does not produce an answer then it is guaranteed that $n < f(k)$ hence the input instance already satisfies the conditions for the output instance and no modification is required. It follows that a decidable parameterized problem is fixed-parameter tractable if and only if it admits a kernelization algorithm [35, Lemma 2.2]. While all fixed-parameter tractable problems admit a kernelization, they do not all admit a small kernelization. Of particular interest are kernels of polynomial size. Determining which parameterized problems admit kernelization algorithms of polynomial size has become a rich area of algorithmic research [20, 62, 96].

Hardness results for kernelization can be obtained through various methods. In this thesis we use polynomial-parameter transformations to obtain hardness results for kernelization.

Definition 2.4.2. A *polynomial-parameter transformation* from parameterized problem A to parameterized problem B is a polynomial-time algorithm that, given an instance (x, k) of A, outputs an instance (x', k') of B such that

1. $(x, k) \in A$ if and only if $(x', k') \in B$, and
2. $k' \leq f(k)$ for some polynomial function f.

If for two NP-complete parameterized decision problems A and B there is a polynomial-parameter transformation from A to B and B admits a polynomial kernel, then a polynomial kernel for A can be constructed as follows. First apply the polynomial-parameter transformation to the input instance (x_1, k_1) to obtain (in polynomial time) an equivalent instance (x_2, k_2) of problem B. Then run the kernelization algorithm on (x_2, k_2) to obtain an equivalent instance (x_3, k_3) of problem B with $|x_3| \leq k_2^{\mathcal{O}(1)} \leq k_1^{\mathcal{O}(1)}$ and similarly $k_3 \leq k_1^{\mathcal{O}(1)}$. Finally, since A and B are NP-complete, there is a polynomial-time algorithm that transforms the instance (x_3, k_3) of B into an equivalent instance (x_4, k_4) of A and since this algorithm spends only polynomial time in the size of the input $|x_3| + k_3 \leq k_1^{\mathcal{O}(1)}$ we have that the size of the output $|x_4| + k_4 \leq (|x_3| + k_3)^{\mathcal{O}(1)} \leq k_1^{\mathcal{O}(1)}$ since it has to write down the parameter value k_4 resulting from the reduction in unary.

Using the observation that polynomial kernels are preserved under polynomial-parameter transformations, we can transfer known hardness results from one problem to another. In particular it is known that the CNF-SAT problem with clauses of unbounded size and parameterized by the number of variables does not admit a polynomial kernel unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ (cf. [72, Lemma 9]), so if there is a polynomial-parameter transformation from CNF-SAT to a parameterized decision problem Π, then if Π admits a polynomial kernel then so does CNF-SAT, i.e., Π does not admit a polynomial kernel unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. It is widely believed that $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$, hence this gives strong evidence that such a problem Π does not admit a polynomial kernel.

There are two important reasons why it is widely believed that $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$. First is its relation to classical complexity. When the conjecture fails and $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, then this implies that the polynomial hierarchy collapses to the third level [127]. This hierarchy is based on satisfiability problems on quantified Boolean formulas with an increasing number of alternating quantifiers. A collapse of the polynomial hierarchy to a certain level implies that all Boolean formulas with arbitrarily many alternating quantifiers can be converted in polynomial time into equisatisfiable formulas with some constant number of alternating quantifiers, which seems impossible.

Second, it seems fundamentally easier to verify that a Boolean formula is satisfiable (there exists a satisfying assignment) than it is to verify that a Boolean formula is not satisfiable (there does not exist a satisfying assignment). This suggests that NP and coNP are likely incomparable. We know $\mathsf{coNP} \subseteq \mathsf{coNP/poly}$, so $\mathsf{NP} \not\subseteq \mathsf{coNP}$ does not directly contradict $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. However, the scenario that $\mathsf{NP} \not\subseteq \mathsf{coNP}$ but $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ seems impossible, since access to polynomial-size advice that may only depend on the length of the input does not seem of much use when, for example, verifying that a Boolean formula is not satisfiable. This is because there are exponentially many formulas of a given length, so it seems impossible that a polynomial-size advice can be useful for many formulas.

It is unknown whether all $\mathcal{F}$-Minor-Free Deletion problems admit a polynomial kernel when parameterized by k. When $\mathcal{F}$ contains a planar graph we

speak of Planar-$\mathcal{F}$ Deletion, and these problems (parameterized by k) admit a polynomial kernelization [61]. Since under this condition we know that $\mathcal{F}$-minor free graphs have a bounded treewidth, we have that any solution X is also a modulator to a graph of bounded treewidth. This insight was crucial to the method used by Fomin et al. [61] to obtain these polynomial kernels. Kernelizability of the remaining $\mathcal{F}$-Minor-Free Deletion problem remains an open problem.

Turing kernelization Turing kernelization [54] is a relaxation of the traditional form of kernelization. Some problems that do *not* admit polynomial kernelizations, do admit polynomial Turing kernelizations [17, 78, 83, 95, 126]. Note that a parameterized problem that has a kernel of size $\mathcal{O}(k^c)$ can be solved by a polynomial-time algorithm that first spends polynomial time to prepare a query of size $\mathcal{O}(k^c)$, and then queries an oracle for its answer. Turing kernelization investigates if and how polynomial-time algorithms can solve NP-hard parameterized problems by querying an oracle for the answers to instances of size $k^{\mathcal{O}(1)}$, potentially multiple times. Formally, for a parameterized problem $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ and a function $f\colon \mathbb{N} \to \mathbb{N}$, a *Turing kernelization* of size f is an algorithm taking input $(x, k) \in \Sigma^* \times \mathbb{N}$. This algorithm can query an oracle to obtain the answer to any instance of problem $\mathcal{Q}$ of size and parameter bounded by $f(k)$ in a single step, and using this power solves any instance (x, k) in time polynomial in $|x| + k$.

Similar to how polynomial-parameter transformations (Definition 2.4.2) preserve existence of polynomial kernels for NP-complete problems, they also preserve existence of polynomial Turing kernels. Hermelin et al. [72] describe a hierarchy of complexity classes $\mathsf{MK}[1] \subseteq \mathsf{WK}[1] \subseteq \mathsf{MK}[2] \subseteq \mathsf{WK}[2] \subseteq \ldots$ based on polynomial-parameter transformations. Relevant in this thesis are WK[1] and MK[2]. A problem complete for WK[1] is Hitting Set parameterized by the number of sets. Two problems that are complete for MK[2] are Hitting Set parameterized by the number of elements in the universe and cnf-sat parameterized by the number of variables. It is known [72] (cf. [48, Theorem 30.12.3]) that problems that are WK[1]-hard (and therefore also problems that are MK[2]-hard) do not admit a polynomial kernel unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. Additionally it is conjectured (cf. [48, Conjecture 30.12.1]) that they also do not admit polynomial Turing kernels. As such, a polynomial-parameter transformation from cnf-sat to a parameterized decision problem Π (which proves MK[2]-hardness of Π) gives strong evidence that Π does not admit a polynomial Turing kernel.

3

A Kernel for Outerplanar Vertex Deletion

3.1 Introduction

In this chapter we investigate kernelization for Outerplanar Deletion. This problem belongs to the class of Planar-$\mathcal{F}$ Deletion problems, which consists of all $\mathcal{F}$-Minor-Free Deletion problems for which the family $\mathcal{F}$ contains a planar graph. Since the family of $\mathcal{F}$-minor-free graphs has bounded treewidth if and only if $\mathcal{F}$ includes a planar graph [111], this restriction ensures that removing a solution to the problem yields a graph of constant treewidth. Hence any solution is a treewidth-η modulator for some $\eta \in \mathbb{N}$ depending on $\mathcal{F}$. For this more restricted class Fomin et al. [61] have shown that polynomial kernels exist for each choice of $\mathcal{F}$. However, the running time of this kernelization algorithm is described by the authors as "horrendous" and regarding the size the authors state the following in the arXiv version of their work:

> The size of the kernel, however, is not explicit. Several of the constants that go into the proof of Lemma 29 depend on the size of the largest graph in certain antichains in a well-quasi-order and thus we don't know what the (constant) exponent bounding the size of the kernel is. We leave it to future work to make also the size of the kernel explicit.

For some specific Planar-$\mathcal{F}$ Deletion problems kernels with explicit size are known. Most famous are Vertex Cover and Feedback Vertex Set which admit kernels with respectively a linear and quadratic number of vertices [30, 76, 121]. Additionally, if θ_c denotes the graph with two vertices and $c \geq 1$ parallel edges, then $\{\theta_c\}$-Minor-Free Deletion admits a kernel with $\mathcal{O}(k^2 \log^{3/2} k)$ vertices and edges [60, Theorem 1.2]; note that the cases $c = 1$ and $c = 2$ correspond to Vertex Cover and Feedback Vertex Set. Another problem for which an explicit kernel size bound is known is Pathwidth-one Deletion, where the goal is to obtain a graph of pathwidth one, i.e., each connected component is a caterpillar. First a kernel of quartic size was obtained [105] which was later improved to a quadratic kernel [37]. If we want to remove at most k vertices to obtain a graph of treedepth at most η, we obtain the Treedepth-η Deletion problem. Since this property can be characterized by forbidden minors and bounded treedepth implies bounded treewidth, this problem is also a special case of Planar-$\mathcal{F}$ Deletion. Giannopoulou et al. [65] have shown that for every η, there is a kernel with $2^{\mathcal{O}(\eta^2)} \cdot k^6$ vertices for Treedepth-η Deletion. They have also proven that, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, there is no universal constant c for which all $\mathcal{F}$-Minor-Free Deletion problems admit a kernel of size $\mathcal{O}(k^c)$ because the degree of the polynomial which bounds the kernel size must increase as a function of $\mathcal{F}$.

In this chapter we investigate Outerplanar Deletion, which asks for a graph G and parameter k whether a set $S \subseteq V(G)$ of size k exists such that $G - S$ is outerplanar. A graph is outerplanar if it admits a planar embedding for which all vertices lie on the outer face, or equivalently, if it contains neither K_4 nor $K_{2,3}$ as a minor [42]. An outerplanar graph on n vertices has at most $2n$ edges [19], so outerplanar graphs form a sparse graph class. The class of outerplanar graphs is a rich superclass of forests and are frequently studied in graph theory [28, 33, 43, 55, 120], graph drawing [16, 64, 100], and optimization [41, 94, 101, 107].

Since outerplanarity can be characterized as being $\{K_4, K_{2,3}\}$-minor free [28], the problem belongs to the class of Planar-$\mathcal{F}$ Deletion problems. It is arguably the easiest problem in the class for which no explicit polynomial kernel is known. This makes Outerplanar Deletion a well-suited starting point to deepen our understanding of Planar-$\mathcal{F}$ Deletion problems in the search for explicit kernelization bounds.

Results Let $\mathsf{opd}(G)$ denote the minimum size of a vertex set $S \subseteq V(G)$ such that $G - S$ is outerplanar. Our main result is the following theorem:

Theorem 3.1.1. *The* Outerplanar Deletion *problem admits a polynomial-time kernelization algorithm that, given an instance (G, k), outputs an equivalent instance (G', k'), such that $k' \leq k$, graph G' is a minor of G, and G' has $\mathcal{O}(k^4)$ vertices and edges. Furthermore, if $\mathsf{opd}(G) \leq k$, then $\mathsf{opd}(G') = \mathsf{opd}(G) - (k - k')$.*

The algorithm behind Theorem 3.1.1 is elementary, consisting of a subroutine to build a decomposition of the input graph G using marking procedures in a tree

decomposition, together with a series of explicit reduction rules. In particular, we avoid the use of protrusion replacement (summarized below). Concrete bounds on the hidden constant in the $\mathcal{O}$-notation follow from our arguments. The size bound depends on the approximation ratio of an approximation algorithm that bootstraps the decomposition phase, for which the current state-of-the-art is 40. We will therefore present a formula to obtain a concrete bound on the kernel size, rather than its value using the current-best approximation (which would exceed 10^5).

Theorem 3.1.1 presents the first concrete upper bound on the degree of the polynomial that bounds the size of kernels for Outerplanar Deletion. We hope that it will pave the way towards obtaining explicit size bounds for all Planar-$\mathcal{F}$ Deletion problems and give an impetus for research on the kernelization complexity of the Planar Deletion problem, which is one of the major open problems in kernelization today [117, 4:28],[62, Appendix A].

Via known connections [61] between kernelizations that reduce to a minor of the input graph and bounds on the sizes of obstruction sets, we obtain the following corollary.

Corollary 3.1.2. *If G is a graph such that $\mathsf{opd}(G) > k$ but each proper minor G' of G satisfies $\mathsf{opd}(G') \leq k$, then G has $\mathcal{O}(k^4)$ vertices and edges.*

The existence of a polynomial bound with unknown degree follows from the work of Fomin et al. [61]; Corollary 3.1.2 gives the first explicit size bounds and contributes to a large body of research on minor-order obstructions (e.g. [27, 44, 45, 46, 91, 114, 116]).

Techniques The known kernelization algorithms [60, 61] for Planar-$\mathcal{F}$ Deletion make use of (near-)protrusions. A protrusion is a vertex set that induces a subgraph of constant treewidth and boundary size. Protrusion replacement is a technique where sufficiently large protrusions are replaced by smaller ones without changing the answer. Protrusion techniques were first used to obtain kernels for problems on planar and other topologically-defined graph classes [21]. Later Fomin et al. [60] described how to use protrusion techniques for problems on general graphs. They proved [60, Lemma 3.3] that any graph G, which contains a modulator X to constant treewidth such that $|X|$ and the size of its neighborhood can be bounded by a polynomial in k, contains a protrusion of size $|V(G)|/k^{\mathcal{O}(1)}$ that can be found efficiently. For any fixed $\mathcal{F}$ containing a planar graph, they present a method to obtain a small modulator to an $\mathcal{F}$-minor-free graph, which has constant treewidth. This leads to a polynomial kernel for Planar-$\mathcal{F}$ Deletion on graphs with bounded degree since the size of the neighborhood of the modulator can be bounded so protrusion replacement can be used to obtain a polynomial kernel. Specifically for $\{\theta_c\}$-Minor-Free Deletion they give reduction rules to reduce the maximum degree in a general graph, which leads to a polynomial kernel on general graphs.

The kernel for Planar-$\mathcal{F}$ Deletion given by Fomin et al. [61] does not rely on bounding the size of the neighborhood of the modulator followed by protrusion replacement. Instead they present the notion of a near-protrusion: a vertex set that will become a protrusion after removing any size-k solution from the graph. With an argument based on well-quasi-ordering they determine that if such near-protrusions are large enough one can, in polynomial time, reduce to a proper minor of the graph without changing the answer.

In this chapter we present a method for Outerplanar Deletion to decrease the size of the neighborhood of a modulator to outerplanarity. This relies on a process that was called "tidying the modulator" in earlier work [124] and also used in the kernelization for Chordal Vertex Deletion [82]. The result is a larger modulator $X \subseteq V(G)$ but with the additional feature that it retains its modulator properties when omitting any single vertex, that is, $G - (X \setminus \{x\})$ is outerplanar for each $x \in X$. We proceed by decomposing the graph into near-protrusions, following along similar lines as the decomposition by Fomin et al. [60] but exploiting the structure of outerplanar graphs at several steps to obtain such a decomposition with respect to our larger tidied modulator, without leading to worse bounds. With the additional properties of the modulator X obtained from tidying we no longer need to rely on well-quasi-ordering, but instead are able to reduce the size of the neighborhood of the modulator in two steps. The first reduces the number of connected components of $G - X$ which are adjacent to any particular modulator vertex $x \in X$. In the case of $\{\theta_c\}$-minor-free graphs, if $G - (X \setminus \{x\})$ is $\{\theta_c\}$-minor free then bounding the number of components of $G - X$ adjacent to each $x \in X$ is sufficient to bound $|N_G(X)|$, since any $x \in X$ has less than c neighbors in any component of $G - (X \setminus \{x\})$. One of the major difficulties we face when working with $\{K_{2,3}\}$-minor-free graphs is that in such a graph there can be arbitrarily many edges between a vertex x and a connected component of $G - (X \setminus \{x\})$. Therefore we present an additional reduction rule that reduces, in a second step, the number of edges between a vertex and a connected component. After these two steps we obtain a bound on the size of the neighborhood of the modulator. At this point, standard protrusion replacement could be applied to prove the *existence* of a kernel for Outerplanar Deletion with $\mathcal{O}(k^4)$ vertices. In order to give an explicit kernelization algorithm we present a number of additional reduction rules to avoid the generic protrusion replacement technique. This eventually leads to a kernel with at most $c \cdot k^4$ vertices and edges for Outerplanar Deletion. It is conceptually simple (yet tedious) to extract the explicit value of c from the algorithm description.

Organization In the next section we give basic definitions and notation we use throughout the rest of the chapter, together with structural observations for outerplanar graphs. Section 3.3 describes how we obtain small modulators to outerplanarity with progressively stronger properties, and finally we obtain a modulator of size $\mathcal{O}(k^4)$ such that each remaining component has only 4 neighbors in the mod-

ulator, effectively forming a decomposition into protrusions. The second stage of the kernelization reduces the size of the connected components outside the modulator. These reduction rules are described in Section 3.4. In Section 3.5 we finally tie everything together to obtain a kernel with $\mathcal{O}(k^4)$ vertices and edges.

3.2 Preliminaries

Graph theory The *boundary* of a vertex set $S \subseteq V(G)$ is the set $\partial_G(S) = N_G(V(G) \setminus S)$.

Definition 3.2.1. For a vertex set $S \subseteq V(G)$, we refer to the graph induced by S and its neighbors as $G\langle S\rangle = G[N_G[S]]$.

When H is an induced subgraph of G we write briefly $G\langle H\rangle = G\langle V(H)\rangle$ and $\partial_G(H) = \partial_G(V(H))$.

For two disjoint sets $X, Y \subseteq V(G)$, we say that $S \subseteq V(G) \setminus (X \cup Y)$ is an (X,Y)-*separator* if the graph $G - S$ does not contain any path from any $u \in X$ to any $v \in Y$. By Menger's theorem, if $x, y \in V(G)$ are non-adjacent in G then the size of a minimum (x,y)-separator is equal to the maximum number of internally vertex-disjoint paths from x to y.

Definition 3.2.2. For a vertex set $X \subseteq V(G)$ the *component graph* $\mathcal{C}(G,X)$ is the bipartite graph $(X \cup Y, E)$ with bipartition X, Y, where Y is the set of connected components of $G - X$, and $(v, C) \in E$ if there is at least one edge between $v \in X$ and the component $C \in Y$.

Planar and outerplanar graphs A plane embedding of graph G is given by a mapping from $V(G)$ to $\mathbb{R}^2$ and a mapping that associates with each edge $uv \in E(G)$ a simple curve on the plane connecting the images of u and v, such that the curves given by two distinct edges can intersect only at the image of a vertex that is a common endpoint of both edges. A face in a plane embedding of a graph G is a subset of the plane enclosed by images of some subset of the edges. We say that a vertex v lies on a face f if the image of v belongs to the closure of f. In every plane embedding there is exactly one face of infinite area, referred to as the outer face. Let F denote the set of faces in a plane embedding of G. Then Euler's formula states that $|V(G)| - |E(G)| + |F| = 2$. Given a plane embedding of G we define the dual graph $\widehat{G}$ with $V(\widehat{G}) = F$ and edges given by pairs of distinct faces that are incident to an image of a common edge from $E(G)$. A *weak dual* graph is obtained from the dual graph by removing the vertex created in place of the outer face.

A graph is called planar if it admits a plane embedding. By Wagner's theorem, a graph G is planar if and only if G contains neither K_5 nor $K_{3,3}$ as a minor. A graph is called outerplanar if it admits a plane embedding with all vertices lying on the outer face. A graph G is outerplanar if and only if G contains neither K_4

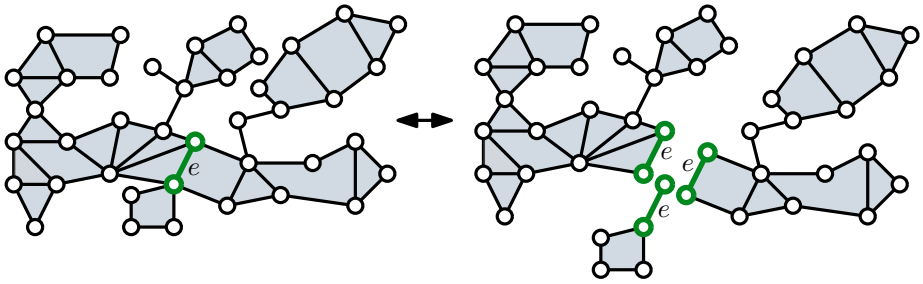


Figure 3.1 The figure shows the induced subgraphs on which Condition 1 of Lemma 3.2.4 has to be evaluated.

nor $K_{2,3}$ as a minor [28]. If a graph G is planar (resp. outerplanar) and H is a minor of G, then H is also planar (resp. outerplanar). The weak dual of an embedded biconnected outerplanar graph G is either the null graph, if G is a single edge or vertex, or a tree otherwise [55]. A graph G is planar (resp. outerplanar) if and only if every biconnected component in G induces a planar (resp. outerplanar) graph.

Observation 3.2.3. *Let $v \in V(G)$. The graph G is outerplanar if and only if for each connected component C of $G - v$ the graph $G\langle C\rangle$ is outerplanar.*

For a graph G we call $S \subseteq V(G)$ an *outerplanar deletion set* if $G - S$ is outerplanar. The *outerplanar deletion number* of G, denoted $\mathsf{opd}(G)$, is the size of a smallest outerplanar deletion set in G.

Structural properties of outerplanar graphs We present a number of structural observations of outerplanar graphs which will be useful in our later argumentation. The first is a characterization of outerplanar graphs similar to Observation 3.2.3. Rather than looking at the components of a graph with one vertex removed, it considers the components of a graph with both endpoints of an edge removed. This allows us for example to easily argue about outerplanarity of graphs obtained from "gluing" two outerplanar graphs on two adjacent vertices, see also Figure 3.1. Recall that $G\langle C\rangle = G[N_G[V(C)]]$.

Lemma 3.2.4. *Let G be a graph and $e \in E(G)$. Then G is outerplanar if and only if both of the following conditions hold:*

1. *for each connected component C of $G - V(e)$ the graph $G\langle C\rangle$ is outerplanar, and*

2. *the graph $G \setminus e$, obtained from G by removing the edge e, does not have three induced internally vertex-disjoint paths connecting the endpoints of e.*

Proof. ($\Rightarrow$) Suppose G is outerplanar. Then every subgraph of G is outerplanar, showing the first condition holds. If $G \setminus e$ has three induced internally vertex-disjoint paths connecting the endpoints of $e = xy$, then each path has at least one interior vertex which shows that G has a $K_{2,3}$-minor, contradicting outerplanarity of G.

($\Leftarrow$) Suppose the two conditions hold, and suppose for a contradiction that G is not outerplanar. Then G contains K_4 or $K_{2,3}$ as a minor. We consider the two cases separately.

G has a $K_{2,3}$-minor. Suppose that G contains $K_{2,3}$ as a minor. It is easy to see that there exist two vertices $u, v \in V(G)$ and three disjoint connected vertex sets A_1, A_2, A_3 such that A_i contains a vertex of both $N_G(u)$ and $N_G(v)$ for all $i \in [3]$. Let G_i be the graph obtained from $G[\{u, v\} \cup A_i]$ by removing the edge uv, if it exists. There exists an (u, v)-path in G_i, so by taking a shortest path there exists an *induced* (u, v)-path P_i in G_i. Since the edge uv does not belong to G_i, path P_i has at least one interior vertex. The three (u, v)-paths P_1, P_2, P_3 in G obtained in this way are internally vertex-disjoint, have at least one interior vertex, and are induced after removing the edge uv if it exists. We use this to derive a contradiction.

If the edge uv exists and is equal to e, then the existence of P_1, P_2, P_3 shows that the second condition is violated and leads to a contradiction. So in the remainder, we may assume that $e \neq uv$. Hence at least one vertex of $\{u, v\}$ lies in a connected component C of $G - V(e)$. Assume without loss of generality that $u \notin V(e)$ and u lies in $V(C)$. We show that $v \in V(G\langle C\rangle)$ in this case. Suppose that v does not belong to $G\langle C\rangle$. Then in particular $v \notin V(e)$ and the vertices $V(e)$ separate u from v; but since P_1, P_2, P_3 are three internally vertex-disjoint paths, vertices u and v cannot be separated by the set $V(e)$ of two vertices. It follows that $u, v \in V(G\langle C\rangle)$.

We claim that each path P_i is a subgraph of $G\langle C\rangle$. To see this, note that the path starts and ends in $G\langle C\rangle$. The two vertices $V(e)$ are the only vertices of $G\langle C\rangle$ which have neighbors in G outside $G\langle C\rangle$. So a path starting and ending in $G\langle C\rangle$ has to leave $G\langle C\rangle$ at one vertex of $V(e)$ and enter $G\langle C\rangle$ at the other; but then e is a chord of this path other than uv. Since the paths P_i do not have such chords, it follows that each path P_i is a subgraph of $G\langle C\rangle$.

By the above we have that the graph $G\langle C\rangle$ contains three internally vertex-disjoint paths P_1, P_2, P_3 with at least one interior vertex each. But then $G\langle C\rangle$ contains $K_{2,3}$ as a minor and is not outerplanar; a contradiction to the first condition.

G has a K_4-minor. In the remainder, we may assume that G contains K_4 as a minor but does not contain $K_{2,3}$ as a minor, as otherwise the previous case applies. Observe that this means that G contains K_4 as a subgraph: any subdivision of K_4 leads to a $K_{2,3}$ minor.

So let H be a K_4 subgraph in G. Observe that there cannot be two connected components of $G - V(e)$ that both contain a vertex of H: any two vertices of the clique H are connected by an edge, which merges the connected components. So there is one connected component C of $G - V(e)$ that contains all vertices of $V(H) \setminus V(e)$. But then H is a subgraph of $G\langle C\rangle$, proving that $G\langle C\rangle$ is not outerplanar and contradicting the first condition. □

In order to more easily apply Lemma 3.2.4, we show that no two induced paths as referred to in Lemma 3.2.4(2) can lie in the same connected component C as referred to in Lemma 3.2.4(1).

Lemma 3.2.5. *Suppose G is outerplanar with an edge $uv \in E(G)$. If P_1, P_2 are internally vertex-disjoint (u, v)-paths in $G \setminus uv$, then the interiors of P_1 and P_2 lie in different connected components of $G - \{u, v\}$.*

Proof. Suppose for contradiction that the interiors of P_1 and P_2 are in the same connected component of $G - \{u, v\}$, and let P be a path from $V(P_1)$ to $V(P_2)$ in $G - \{u, v\}$. Let G' be the graph obtained from G by contracting the interiors of P_1 and P_2 into a single vertex p_1 and p_2 respectively and contracting P to realize the edge p_1p_2. Clearly G' is a minor of G so then G' doesn't contain a K_4-minor. Observe however that $\{u, v, p_1, p_2\}$ induce a K_4 subgraph in G'. Contradiction. □

We now give a condition under which an edge can be added to an outerplanar graph without violating outerplanarity. Intuitively, this corresponds to adding an edge between two vertices that lie on the same interior face.

Lemma 3.2.6. *Suppose G is outerplanar and vertices x, y lie on an induced cycle D with $xy \notin E(G)$. Then adding the edge xy to G preserves outerplanarity.*

Proof. Let D_1, D_2 be the two parts of the cycle $D - \{x, y\}$. We claim that D_1 and D_2 belong to different connected components of $G - \{x, y\}$. Suppose not, and let P be a path from $V(D_1)$ to $V(D_2)$ in $G - \{x, y\}$ that intersects $V(D_1)$ and $V(D_2)$ in exactly one vertex v_1 and v_2, respectively. The path P has at least one interior vertex since the cycle D is induced. But then P together with the two induced (v_1, v_2)-paths along D give a $K_{2,3}$-minor; a contradiction to the assumption that G is outerplanar.

Hence D_1 and D_2 belong to different connected components of $G - \{x, y\}$. Let G' be the graph obtained from G by adding the edge xy. We show that for each connected component C of $G' - \{x, y\}$ the graph $G'\langle C\rangle$ is a minor of G and therefore outerplanar. This follows from the fact that, by the argument above, C contains at most one segment of the cycle D and therefore we can contract the remaining segment to realize the edge xy.

Using the above, we prove that G' is outerplanar by applying Lemma 3.2.4 to edge xy. The preceding argument shows that the first condition is satisfied. To see that the second condition is satisfied as well, note that G is outerplanar and therefore $G' \setminus xy = G$ does not contain three internally vertex-disjoint paths connecting the endpoints of e. □

Finally, we observe that if an outerplanar graph G has a cycle C, then any component of $G - V(C)$ is adjacent to at most two vertices of the cycle (else there would be a K_4 minor), and these must be consecutive on the cycle (else there would be a $K_{2,3}$ minor).

Lemma 3.2.7. *If C is a cycle in an outerplanar graph G, then each connected component of $G - V(C)$ has at most two neighbors in C, and they must be consecutive along the cycle.*

Proof. Suppose for a contradiction that some component D of $G - V(C)$ has two neighbors x, y which are not consecutive along C. Then the cycle provides two vertex-disjoint (x, y)-paths with at least one interior vertex each, and component D provides a third (x, y)-path with an interior vertex. This yields a $K_{2,3}$-minor where $\{x\}$ and $\{y\}$ are the branch sets of the degree-3 vertices, contradicting outerplanarity.

Now suppose that some component D of $G - V(C)$ has three or more neighbors on C. Let P_1, P_2, P_3 be three vertex-disjoint paths that cover the entire cycle C such that each path contains a neighbor of D and observe that $V(P_1), V(P_2)$, $V(P_3), V(D)$ form the branch sets of a K_4-minor in G, contradicting outerplanarity. □

LCA closure If a graph is outerplanar, then its treewidth is at most 2 [19, Lemma 78]. Since n-vertex graphs of treewidth w can have at most $w \cdot n$ edges [19, Lemma 91] we obtain the following.

Observation 3.2.8. *If G is an outerplanar graph, then $|E(G)| \leq 2 \cdot |V(G)|$.*

Let T be a rooted tree and $S \subseteq V(T)$ be a set of vertices in T. We define the *lowest common ancestor* (LCA) of u and v (not necessarily distinct), denoted as $\mathsf{LCA}(u, v)$, to be the deepest node x which is an ancestor of both u and v. The LCA closure of S is the set

$$\overline{\mathsf{LCA}}(S) = \{\mathsf{LCA}(u, v) : u, v \in S\}.$$

Lemma 3.2.9. *[62, Lemmas 9.26, 9.27, 9.28] Let T be a rooted tree, $S \subseteq V(T)$, and $M = \overline{\mathsf{LCA}}(S)$. All of the following hold.*

1. *Each connected component C of $T - M$ satisfies $|N_T(C)| \leq 2$.*
2. *$|M| \leq 2 \cdot |S| - 1$.*
3. *$\overline{\mathsf{LCA}}(M) = M$.*

Lemma 3.2.10. *If (T, χ) is a tree decomposition of width at most c of a graph G, and $B \subseteq V(T)$ is a set of nodes of T closed under taking lowest common ancestors (i.e., $\overline{\mathsf{LCA}}(B) = B$), then for $M = \bigcup_{t \in B} \chi(t)$ and any connected component C of $G - M$ we have $|N_G(C) \cap M| \leq 2c$.*

Proof. Let T_C denote the subgraph of T induced by the nodes whose bag contains a vertex of C. Since C is a connected component of $G-M$, we have $V(T_C)\cap B = \emptyset$ and T_C is a connected tree rather than a forest. Hence there exists a tree T' in the forest $T - B$ such that T_C is a subtree of T'. Since B is closed under taking lowest common ancestors, it follows from Lemma 3.2.9 that for $Z := N_T(V(T'))$ we have $|Z| \leq 2$. For each $z \in Z$, let $f(z)$ denote the first node outside $V(T_C)$ on the unique shortest path in T from $V(T_C)$ to z. Note that we may have $z = f(z)$. Let $g(z)$ denote the unique neighbor in T of node $f(z)$ among $V(T_C)$. Observe that both $f(z)$ and $g(z)$ lie on each path in T connecting a node of T_C to z.

By definition of T_C we have that each bag of T_C intersects $V(C)$ while $\chi(f(z))$ does not. Hence $\chi(f(z)) \neq \chi(g(z))$. As each bag has size at most $c + 1$, it follows that $|\chi(f(z)) \cap \chi(g(z))| \leq c$ for each $z \in Z$. To prove the desired claim that $|N_G(C) \cap M| \leq 2c$, it therefore suffices to argue that $N_G(C) \cap M \subseteq \bigcup_{z\in Z}(\chi(f(z)) \cap \chi(g(z)))$.

Consider a vertex $v \in N_G(C) \cap M$. We argue that $v \in \chi(f(z)) \cap \chi(g(z))$ for some $z \in Z$, as follows. Since $v \in M$, there exists a node $b^* \in B$ such that $v \in \chi(b^*)$. Since $v \in N_G(C)$ there exists $u \in V(C)$ such that $\{u, v\} \in E(G)$. Hence there is a bag in the tree decomposition containing both u and v, and as vertices of $V(C)$ only occur in bags of the subtree T_C, we find that v occurs in at least one bag of T_C. Since the occurrences of v form a connected subtree of T, and v appears in at least one bag of T_C and at least one bag of B, while the only neighbors in B of the supertree T' of T_C are the nodes in Z, it follows that v occurs in at least one bag $\chi(z)$ for some $z \in Z$. But since all paths from T_C to z pass through $f(z)$ and $g(z)$ as observed above, this implies $z \in \chi(f(z)) \cap \chi(g(z))$; this concludes the proof. □

3.3 Splitting the graph into pieces

In this section we show how to reduce any input of Outerplanar Deletion to an equivalent instance which admits a decomposition into a modulator of bounded size along with a bounded number of outerplanar components containing at most four neighbors of the modulator.

3.3.1 The augmented modulator

The starting point for both our kernelization algorithm and the one from Fomin et al. [61] is to employ a constant-factor approximation algorithm. We however begin with a different approximation algorithm, which has two advantages. First, the algorithm is constructive: it relies only on separating properties of bounded-treewidth graphs and rounding a fractional solution from a linear programming relaxation. Second, the approximation factor can be pinned down to a concrete value.

Theorem 3.3.1. *[71] There is a polynomial-time deterministic 40-approximation algorithm for* Outerplanar Deletion.

Proof. The article [71] only states that the approximation factor is constant. However, it also provides a recipe to retrieve its value. From [71, Theorem 1.1] we get that the approximation factor for Outerplanar Deletion is $2 \cdot \alpha(3)$, for a function α satisfying the following: the problem k-Subset Vertex Separator admits a polynomial-time $(\alpha(k), \mathcal{O}(1))$-bicriteria approximation algorithm. Without going into details, one can check that such an algorithm has been given by Lee [92]: by examining the proof of Lemma 2 therein for $\varepsilon = \frac{1}{4}$ we see that one can construct a polynomial-time $(8 \cdot H_{2k}, 2)$-bicriteria approximation algorithm, where H_k is the k-th harmonic number. We check that $2 \cdot 8 \cdot H_6 < 40$. Both algorithms in question are deterministic. □

In our setting, for a given graph G and integer k, we want to determine whether G admits an outerplanar deletion set of size at most k. Thanks to the theorem above, we can assume that we are given an outerplanar deletion set X (also called a modulator to outerplanarity) of size at most $40 \cdot k$. As a next step, we would like to augment this set to satisfy a stronger property. This step is inspired by the technique of tidying the modulator from van Bevern, Moser, and Niedermeier [124]. For each vertex $v \in X$ we would like to be able to "put it back" into $G - X$ while maintaining outerplanarity. In order to do so, we look for a set of vertices from $V(G) \setminus X$ that needs to be removed if v is put back. Since $G - X$ is outerplanar and hence has treewidth at most two, we can construct such a set of moderate size by a greedy approach. We scan a tree decomposition in a bottom-up manner and look for maximal subgraphs that are outerplanar when considered together with v. When such a subgraph cannot be further extended we mark one bag of a decomposition, which gives 3 vertices to be removed. We show that this idea leads to a 3-approximation algorithm. While this approach based on covering/packing duality is well-known, we present the proof for completeness.

Lemma 3.3.2. *There is a polynomial-time algorithm that, given a graph G, an integer k, and a vertex v such that $G - v$ is outerplanar, either finds an outerplanar deletion set $S \subseteq V(G) \setminus \{v\}$ in G of size of most $3k$ or correctly concludes that there is no outerplanar deletion set $S \subseteq V(G) \setminus \{v\}$ in G of size of most k.*

Proof. Since $G - v$ is outerplanar, its treewidth is at most two. A tree decomposition (T, χ) of $G - v$ of this width can be computed in linear time [18].

Consider a process in which we scan the tree decomposition in a bottom-up manner and mark some nodes of T. In the i-th step we will mark a node $t_i \in V(T)$ and maintain a family $Y_1, \ldots, Y_i$ of disjoint subsets of $V(G) \setminus \{v\}$, so that for each $j \in [i]$ the graph $G[Y_i \cup \{v\}]$ is not outerplanar. We begin with no marked vertices and an empty family of vertex sets. Let $U(t)$ be the set of vertices appearing in a bag in the subtree of T rooted at $t \in V(T)$. In the i-th step we choose a lowest node $t_i \in V(T)$ (breaking ties arbitrarily), so that $U(t_i) \cup \{v\} \setminus$

$\bigcup_{j=1}^{i-1} Y_j$ induces a non-outerplanar subgraph of G. If there is no such node, we terminate the process. Otherwise we set $Y_i = U(t_i) \setminus \bigcup_{j=1}^{i-1} Y_j$ and continue the process.

By the definition, the sets $Y_1, \ldots, Y_i$ are disjoint and each of them, when considered together with v, induces a subgraph which is not outerplanar. Suppose that the procedure has executed for at least $k+1$ steps. Then for any set $S \subseteq V(G) \setminus \{v\}$ of size of most k, there is some $i \in [k+1]$ such that $Y_i \cap S = \emptyset$. Since $G[Y_i \cup \{v\}]$ is not outerplanar, we can conclude that S is not an outerplanar deletion set. Hence we can conclude that no set as desired exists and terminate.

Suppose now that the procedure has terminated at the k'-th step, where $k' \leq k$. Since t_i is chosen as a lowest node among those satisfying the given condition, we get that $U(t_i) \cup \{v\} \setminus (\chi(t_i) \cup \bigcup_{j=1}^{i-1} Y_j)$ induces an outerplanar subgraph of G. Observe that $S = \bigcup_{j=1}^{k'} \chi(t_j)$ separates Y_i from Y_j in $G - v$ for each pair $1 \leq i < j \leq k'$, because in particular $\chi(t_i) \subseteq S$. Let $Y_0 = V(G) \setminus \bigcup_{j=1}^{k'} Y_j$. Then also $G[Y_0 \cup \{v\}]$ is outerplanar and S separates Y_0 from any Y_i in $G - v$. We apply Observation 3.2.3 to $G - S$ with cut vertex v and check that any connected component C of $G - S - v$ is contained in some set Y_i, so $G\langle C \rangle$ is outerplanar, and thus $G - S$ is outerplanar. The size of each bag in (T, χ) is at most 3, hence $|S| \leq 3k' \leq 3k$. The claim follows. □

Observe that if it is impossible to remove k vertices avoiding v from $G - (X \setminus \{v\})$ to make it outerplanar, then any outerplanar deletion set in G of size at most k must contain v. In this situation it suffices to solve the problem on $G - v$. Otherwise, we identify a set $R(v)$ of at most $3k$ vertices whose removal allows v to be put back in $G - X$ without spoiling outerplanarity. After inserting $R(v)$ into the set X, we could put v back "for free". Let us formalize this idea of augmenting the modulator.

Definition 3.3.3. A (k, c)-*augmented modulator* in graph G is a pair of disjoint sets $X_0, X_1 \subseteq V(G)$ such that:

1. $G - X_0$ is outerplanar,
2. for each $v \in X_0$, there is a set $R(v) \subseteq X_1$, such that $|R(v)| \leq 3k$ and $G - ((X_0 \setminus \{v\}) \cup R(v))$ is outerplanar, and
3. $|X_0| \leq c \cdot k$, $X_1 = \bigcup_{v \in X_0} R(v)$, which implies $|X_1| \leq 3c \cdot k^2$.

We classify the pairs of vertices within $X_0 \cup X_1$. A pair (u, v): $u, v \in X_0 \cup X_1$ is of type:

A: if $u, v \in X_0$ or $(u \in X_0, v \in R(u))$ or $(v \in X_0, u \in R(v))$,

B: if (u, v) is not of type A and $\{u, v\} \cap X_0 \neq \emptyset$,

C: if $u, v \in X_1$.

We note that the number of type-A pairs is at most $c(3+c)\cdot k^2$, the number of type-B pairs is at most $3c^2\cdot k^3$, and the number of type-C pairs is at most $9c^2\cdot k^4$.

The downside of the augmented modulator is that its size can be as large as $\mathcal{O}(k^2)$. However, in return we obtain an even stronger property than previously sketched. For most of the pairs of vertices u, v from the augmented modulator (X_0, X_1), putting them back into $G-(X_0\cup X_1)$ at the same time still does not break outerplanarity. This property will come in useful for bounding the size of the kernel.

Observation 3.3.4. *Let (X_0, X_1) be a (k,c)-augmented modulator in a graph G. Then for each $v\in X_0\cup X_1$, the graph $G-(X_0\cup X_1\setminus\{v\})$ is outerplanar. Furthermore, if $u,v\in X_0\cup X_1$ and the pair (u,v) is of type B or C, then the graph $G-(X_0\cup X_1\setminus\{u,v\})$ is outerplanar.*

Let us summarize what we can compute so far. We say that instances (G,k) and (G',k') are equivalent if $\mathsf{opd}(G)\leq k \Leftrightarrow \mathsf{opd}(G')\leq k'$.

Lemma 3.3.5. *There is a polynomial-time algorithm that, when given an instance (G,k), either correctly concludes that $\mathsf{opd}(G)>k$ or outputs an equivalent instance (G',k'), where $k'\leq k$ and G' is a subgraph of G, along with a $(k',40)$-augmented modulator in G'. If $\mathsf{opd}(G)\leq k$ then it holds that $\mathsf{opd}(G')=\mathsf{opd}(G)-(k-k')$. Moreover, if for every vertex $v\in V(G)$ there is an outerplanar deletion set $S\subseteq V(G)\setminus\{v\}$ in G of size at most k, then $k'=k$.*

Proof. We run the 40-approximation algorithm from Theorem 3.3.1 to obtain an outerplanar deletion set X_0. If $|X_0|>40\cdot k$, we conclude that $\mathsf{opd}(G)>k$. Otherwise, we iterate over $v\in X_0$ and execute the subroutine from Lemma 3.3.2 with respect to the graph $G_v=G-(X_0\setminus\{v\})$. If for any vertex v we have concluded that G_v does not admit any outerplanar deletion set $S\subseteq V(G_v)\setminus\{v\}$ of size at most k, then the same holds for G. This implies that any outerplanar deletion set in G of size at most k (if there is any) must include the vertex v and the instance $(G-v,k-1)$ is equivalent to (G,k). Furthermore, in this case $\mathsf{opd}(G')=\mathsf{opd}(G)-1$ as long as $\mathsf{opd}(G)\leq k$. We can thus remove the vertex v from G, decrease the value of parameter k by 1, and start the process from scratch. If during this process we reach an instance $(G',0)$, then (G,k) is satisfiable if and only if G' is outerplanar. Observe that if for every vertex $v\in V(G)$ there is an outerplanar deletion set $S\subseteq V(G)\setminus\{v\}$ in G of size at most k, then this holds also for the graph G_v and thus we will not apply the reduction rule decreasing the value of k.

Suppose now that for each $v\in X_0$ we have obtained a set $S_v\subseteq V(G_v)\setminus\{v\}$ of size at most $3k$ such that $G-((X_0\setminus\{v\})\cup S_v)$ is outerplanar. Then setting $R(v)=S_v$ and $X_1=\bigcup_{v\in X_0}R(v)$ satisfies the requirements of Definition 3.3.3. □

The reduction step above is the only one in our algorithm that may decrease the value of k. Moreover, no further reduction will modify the outerplanar deletion

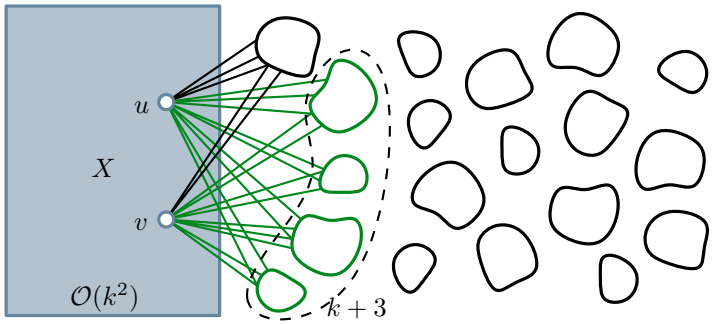


Figure 3.2 Illustration of Reduction Rule 3.1. For each pair $u, v \in X = X_0 \cup X_1$ we choose up to $k+3$ components of $G - X$ with edges to both u and v and mark the corresponding edges in the component graph $\mathcal{C}(G, X)$. If a pair (v, C) is not marked in the end, all the edges between v and C are removed.

number as long as $\mathsf{opd}(G) \leq k$. This observation will come in useful for bounding the size of minimal minor obstructions to having an outerplanar deletion set of size k.

As the next step, we would like to bound the number of connected components in $G - (X_0 \cup X_1)$ and the number of connections between the components and the modulator vertices. We show that if vertices $u, v \in X_0 \cup X_1$ are adjacent to sufficiently many components, then at least one of u, v must be removed in any solution of size at most k. Together with the "putting back" property of the augmented modulator, this allows us to forget some of the edges without modifying the space of solutions of size at most k. We formalize this idea with the following marking scheme, an illustration of which can be found in Figure 3.2.

Reduction Rule 3.1. Let G be a graph, $k \in \mathbb{N}$, and (X_0, X_1) be a (k, c)-augmented modulator in G. Consider the component graph $\mathcal{C}(G, X_0 \cup X_1)$. For each pair $u, v \in X_0 \cup X_1$ choose up to $k+3$ components C_i with edges to both u and v, and mark the edges $(u, C_i), (v, C_i)$ in $\mathcal{C}(G, X_0 \cup X_1)$. If an edge (v, C) is unmarked in the end, remove all the edges between v and C in G. If some component C of $G - (X_0 \cup X_1)$ or a vertex $v \in X_0 \cup X_1$ becomes isolated, remove it from G.

Lemma 3.3.6 (Safeness)**.** *Let G be a graph, $k \in \mathbb{N}$, and (X_0, X_1) be a (k, c)-augmented modulator in G. Let G' be obtained from G by applying Reduction Rule 3.1 with respect to (X_0, X_1, k). If $\mathsf{opd}(G) > k$ then $\mathsf{opd}(G') > k$ and if $\mathsf{opd}(G) \leq k$ then $\mathsf{opd}(G') = \mathsf{opd}(G)$.*

Proof. It suffices to show that any solution in G' of size at most k is also valid in G. Removing an outerplanar connected component is always safe so it suffices to argue for the correctness of the edge removal rule. Consider a single step of the reduction

in a graph G, in which we have removed the edges between vertex $v \in (X_0 \cup X_1)$ and a connected component C of $G - (X_0 \cup X_1)$. Let G' be the graph after this modification and S be an outerplanar deletion set of size at most k in G'. If $v \in S$, then $G' - S = G - S$ so let us assume that $v \notin S$.

Suppose there is another $u \in X_0 \cup X_1$ with an edge to C in G. Since the pair (v, C) was not marked, there are $k + 3$ components C_i, different from C, of $G - (X_0 \cup X_1)$ with edges to both u and v. These pairs were marked, so they cannot be removed in any previous reduction step. By a counting argument, at least 3 of these components have empty intersections with S. If $u \notin S$, then these components together with $\{u, v\}$ form a minor model of $K_{2,3}$ in $G' - S$, which is not possible. Therefore, $u \in S$.

It follows that v is the only neighbor of C in $G - S$. By Observation 3.3.4 we can "put back" v into $G - (X_0 \cup X_1)$ without spoiling the outerplanarity and so the graph $(G - S)\langle C\rangle$ being the subgraph of $G[C \cup \{v\}]$ is outerplanar. The graph $G - S - C$ is a subgraph of $G' - S$, so it is also outerplanar. The intersection of their vertex sets is exactly $\{v\}$ so from Observation 3.2.3 we obtain that $G - S$ is outerplanar. □

Now we show that after application of Reduction Rule 3.1 the component graph $\mathcal{C}(G, X_0 \cup X_1)$ cannot be too large. This will come in useful for proving further upper bounds. We could trivially bound the number of its edges by $|X_0 \cup X_1|^2 \cdot (k + 3) = \mathcal{O}(k^5)$ but, thanks to the properties of the augmented modulator, we can be more economical. First, we need a simple observation about bipartite outerplanar graphs.

Proposition 3.3.7. *Consider an outerplanar bipartite graph $(X \cup Y, E)$ such that all the vertices in Y have degree at least two. Then $|Y| \leq 4 \cdot |X|$ and $|E| \leq 10 \cdot |X|$.*

Proof. Remove part of the edges so that each vertex in Y has degree exactly two. Now contract each vertex from Y to one of its neighbors. The constructed graph is a minor of $(X \cup Y, E)$ with a vertex set X, so it is outerplanar and the number of edges is at most $2 \cdot |X|$ by Observation 3.2.8. Each edge could have been obtained by at most 2 different contractions, as otherwise $(X \cup Y, E)$ would contain $K_{2,3}$ as a minor. Therefore $|Y| \leq 4 \cdot |X|$. Again by Observation 3.2.8, the number of edges in $(X \cup Y, E)$ is at most $2 \cdot (|X| + |Y|) \leq 10 \cdot |X|$. □

Recall the types of pairs from Definition 3.3.3 and their properties from Observation 3.3.4. We know that the number of type-A pairs is at most $c(3 + c) \cdot k^2$ and the number of type-B pairs is at most $3c^2 \cdot k^3$. Moreover, pairs of type B can be inserted back into $G - (X_0 \cup X_1)$ without affecting its outerplanarity.

Lemma 3.3.8. *After the application of Reduction Rule 3.1 with respect to a (k, c)-augmented modulator (X_0, X_1), the component graph $\mathcal{C}(G, X_0 \cup X_1)$ contains at most $f_1(c) \cdot (k + 3)^3$ vertices and edges, where $f_1(c) = 14c^2 + 60c$.*

Proof. For pairs of type A we have marked at most $(k+3) \cdot c(3+c) \cdot k^2$ edges. If (u, v) is of type B, then by Observation 3.3.4 the graph $G - (X_0 \cup X_1 \setminus \{u, v\})$ is outerplanar and there can be at most 2 components adjacent to both u, v as otherwise we would obtain a $K_{2,3}$-minor. Hence, for pairs of type B we have marked at most $2 \cdot 3c^2 \cdot k^3$ edges.

Next, we argue that the total number of edges marked due to pairs of type C is $30c \cdot k^2$. Let $E_C \subseteq E(\mathcal{C}(G, X_0 \cup X_1))$ denote the set of these edges. Let $Y_C \subseteq V(\mathcal{C}(G, X_0 \cup X_1))$ be the set of these connected components of $G - (X_0 \cup X_1)$ which are incident to at least one edge from E_C in $\mathcal{C}(G, X_0 \cup X_1)$. By the definition of the marking scheme, if $C \in Y_C$ then C is in fact incident to at least 2 edges from E_C, and their other endpoints belong to X_1. Consider the subgraph $(X_1 \cup Y_C, E_C)$ of $\mathcal{C}(G, X_0 \cup X_1)$. It is a minor of $G - X_0$, therefore it is outerplanar. By Proposition 3.3.7, we get that $|E_C| \leq 10 \cdot |X_1| = 10 \cdot 3c \cdot k^2$.

We can thus estimate the number of edges in $\mathcal{C}(G, X_0 \cup X_1)$ by $(7c^2 + 3c) \cdot (k+3)^3 + 30c \cdot k^2 \leq (7c^2 + 30c) \cdot (k+3)^2$. Finally, since $\mathcal{C}(G, X_0 \cup X_1)$ contains no isolated vertices, the number of vertices is at most twice the number of edges. □

3.3.2 The outerplanar decomposition

We proceed by enriching the augmented modulator further. We would like to provide additional properties at the expense of growing the modulator size to $\mathcal{O}(k^3)$. For two vertices u, v in an augmented modulator (X_0, X_1) ideally we would like to ensure that no component of $G - (X_0 \cup X_1 \cup Z)$ is adjacent to both u and v, where Z is some vertex set of size $\mathcal{O}(k^3)$. This is not always possible, but we will guarantee that in such a case any outerplanar deletion set of size at most k must contain either u or v.

Definition 3.3.9. Let $Y \subseteq V(G)$ be a vertex subset in a graph G. We say that $u, v \in Y$ are *Y-separated* if no connected component of $G - Y$ is adjacent to both u and v.

In Lemma 3.3.11 we are going to show that when G is outerplanar and $X \subseteq V(G)$, then there always exists a small set $Y \subseteq V(G)$ so that every pair from X is $(X \cup Y)$-separated. Towards that goal, we need the following proposition.

Proposition 3.3.10. *Let $X \subseteq V(G)$ be an independent set in an outerplanar graph G. Then there exists $v \in X$ and $S \subseteq V(G) \setminus X$ of size at most four, so that S is a $(v, X \setminus v)$-separator in G.*

Proof. Consider a tree decomposition (T, χ) of G of width two where T is rooted at an arbitrary node r. For a vertex $v \in V(G)$ let $t_v \in V(T)$ be the node which is closest to the root r, among those whose bag contain v. Consider $v \in X$ for which t_v is furthest from the root (if there are many, pick any of them) and let $B_v := \chi(t_v)$. By standard properties of tree decompositions, any path from v to $X \setminus v$ either goes through $B_v \setminus v$ or ends at $B_v \setminus v$.

If $(B_v \setminus v) \cap X = \emptyset$, set $S = B_v \setminus v$. If $B_v \setminus v = \{u_1, u_2\}$, where $u_1 \in X$, $u_2 \notin X$, consider a minimal (v, u_1)-separator S' and set $S = S' \cup \{u_2\}$. There cannot be three vertex-disjoint paths connecting v, u_1 as $vu_1 \notin E(G)$ and this would give a minor model of $K_{2,3}$ in G. Therefore by Menger's theorem we have $|S'| \leq 2$ and $|S| \leq 3$. Finally, suppose $(B_v \setminus v) \subseteq X$. In this case, let S be a minimal $(v, B_v \setminus v)$-separator. If there were five vertex-disjoint paths connecting v and $B_v \setminus v$ then in particular there would be three vertex-disjoint paths connecting v and some $u \in B_v \setminus v$, which would again give a minor model of $K_{2,3}$ in G. Therefore $|S| \leq 4$.

Suppose there is a path in $G \setminus S$ connecting v with some $x \in X \setminus v$. It contains a subpath connecting v with some $u \in B_v \setminus v$. If $u \notin X$, then $u \in S$, so suppose that $u \in X$. But S contains a (v, u)-separator, so such path cannot exist in $G - S$. □

Lemma 3.3.11. *There is a polynomial-time algorithm that, given a vertex set $X \subseteq V(G)$ in an outerplanar graph G, finds a vertex set $Y \subseteq V(G) \setminus X$ of size at most $4 \cdot |X|$, so that every pair $u, v \in X$ with $u \neq v$ is $(X \cup Y)$-separated.*

Proof. We can assume that X is an independent set in G because removing edges between vertices in X does not affect the neighborhood of a connected component in $G - (X \cup Y)$. Initialize $Y = \emptyset$. By Proposition 3.3.10 we can find a vertex $v \in X$ that can be separated from $X \setminus v$ by at most 4 vertices. Add these vertices to Y and repeat this operation recursively on $X \setminus v$. □

Given an augmented modulator (X_0, X_1), we would like to find a set Z of moderate size so that for each pair (u, v) from $X_0 \cup X_1$ either u, v are $(X_0 \cup X_1 \cup Z)$-separated or there exist $k + 4$ internally vertex-disjoint paths, with non-empty interior, connecting u and v in G. If the latter case occurs, then any outerplanar deletion set of size bounded by k, can intersect at most k of these paths' interiors. Therefore, this solution must remove either u or v in order to get rid of all $K_{2,3}$-minors. We remark that this property already holds if we request $k + 3$ disjoint (u, v)-paths, but in this stronger form it also holds for a graph obtained from G by an edge removal. This fact will be crucial for the safeness proof for Reduction Rule 3.3.

In order to find the set Z, we could consider all pairs (u, v) from $X_0 \cup X_1$ and, if there exists an (u, v)-separator of size at most $k + 3$, add it to Z. This however would make Z as large as $\mathcal{O}(k^5)$. We can make this process more economical by analyzing what happens for different types of pairs from Definition 3.3.3. Recall that the number of type-A pairs is at most $c(3 + c) \cdot k^2$ and the number of type-B pairs is at most $3c^2 \cdot k^3$.

Lemma 3.3.12. *There is a polynomial-time algorithm that, when given an instance (G, k) with (k, c)-augmented modulator (X_0, X_1), returns a set $Z \subseteq V(G) \setminus (X_0 \cup X_1)$ of size at most $f_2(c) \cdot (k + 3)^3$, where $f_2(c) = 4c^2 + 15c$, such that for each pair $u, v \in X_0 \cup X_1$ of distinct vertices one of the following holds:*

1. *vertices u, v are $(X_0 \cup X_1 \cup Z)$-separated, or*

2. *there are $k+4$ internally vertex-disjoint paths, with non-empty interior, connecting u and v in G.*

Proof. Initialize $Z_0 = \emptyset$. Consider all the pairs (u, v) from the augmented modulator. If (u, v) is of type A or B, compute a minimum (u, v)-separator $S_{u,v}$ with $u, v \notin S_{u,v}$ in $G - (X_0 \cup X_1 \setminus \{u, v\}) \setminus uv$, that is, we remove the edge uv if it exists. If $|S_{u,v}| \leq k+3$, add $S_{u,v}$ to Z_0. Recall from Observation 3.3.4 that if (u, v) is of type B, then the graph $G - (X_0 \cup X_1 \setminus \{u, v\})$ is outerplanar, so $|S_{u,v}| \leq 2$, as otherwise we could construct a $K_{2,3}$ minor. For pairs of type A we add at most $(k+3) \cdot c(3+c) \cdot k^2$ elements, and for pairs of type B at most $2 \cdot 3c^2 \cdot k^3$ elements. If the pair (u, v) does not satisfy condition (2), then the set Z_0 contains a set $S_{u,v}$ which forms a (u, v)-separator in $G-(X_0 \cup X_1 \setminus \{u, v\}) \setminus uv$. Therefore u, v belong to different connected components of $G - (X_0 \cup X_1 \cup Z_0 \setminus \{u, v\}) \setminus uv$ and so they are $(X_0 \cup X_1 \cup Z_0)$-separated.

To cover pairs of type C we consider the outerplanar graph $G - X_0$. By Lemma 3.3.11 we can find a vertex set $Z_1 \subseteq V(G) \setminus (X_0 \cup X_1)$ of size $4 \cdot |X_1| \leq 12c \cdot k^2$ so that all pairs $u, v \in X_1$ are $(X_1 \cup Z)$-separated in $G - X_0$. We return the set $Z_0 \cup Z_1$, which has no more than $(k+3) \cdot c(3+c) \cdot k^2 + 2 \cdot 3c^2 \cdot k^3 + 12c \cdot k^2 \leq (4c^2 + 15c) \cdot (k+3)^3$ elements. □

We would like to simplify the interface between a connected component C of $G - (X_0 \cup X_1 \cup Z)$ and the rest of the graph. Since $G - X_0$ is outerplanar it has treewidth at most two, which implies there is a tree decomposition in which each pair of distinct bags intersects in at most 2 vertices. When constructing a separator $Z' \supseteq Z$ via the LCA closure, the neighborhood of each connected component C of $G - Z'$ within the set Z' is contained in at most two bags of the decomposition. This allows us to guarantee that $|N_G(C) \cap Z'| \leq 4$.

Lemma 3.3.13. *There is a polynomial-time algorithm that, given an outerplanar graph G and $Z \subseteq V(G)$, returns a set $Z' \supseteq Z$ of size at most $6 \cdot |Z|$ such that each connected component of $G - Z'$ has at most four neighbors in Z'.*

Proof. Consider a tree decomposition (T, χ) of width two of the graph G, rooted at a node $r \in V(T)$. It can be found in linear time [18]. For a vertex v let t_v be the node which is closest to the root among those whose bags contain v. Consider the set of nodes $T_Z = \{t_v \mid v \in Z\}$. Let $T'_Z = \overline{\mathsf{LCA}}(T_Z)$ be the LCA closure of T_Z. Finally, let Z' be union of all bags in T'_Z. We have $|T'_Z| \leq 2 \cdot |T_Z|$ and $|Z'| \leq 6 \cdot |Z|$. By Lemma 3.2.10 we obtain that each connected component of $G - Z'$ has at most four neighbors. □

In order to keep the kernel size in check, we need to analyze the number of connected components of $G - (X_0 \cup X_1 \cup Z)$. We have managed to bound the size of Z by $\mathcal{O}(k^3)$ and, in Lemma 3.3.8, we have also bounded by $\mathcal{O}(k^3)$ the number of edges in the component graph $\mathcal{C}(G, X_0 \cup X_1)$. These two properties suffice to also bound the number of connected components of $G - (X_0 \cup X_1 \cup Z)$ that have

at least two neighbors in $X_0 \cup X_1 \cup Z$. It will be easier to deal with the remaining ones later.

Lemma 3.3.14. *Let (X_0, X_1) be a (k, c)-augmented modulator in G, so that the component graph $\mathcal{C}(G, X_0 \cup X_1)$ has at most s vertices and s edges, and let $Z \subseteq V(G) \setminus (X_0 \cup X_1)$. Then there are at most $3 \cdot s + 4 \cdot |Z|$ components of $G - (X_0 \cup X_1 \cup Z)$ that have two or more neighbors in $X_0 \cup X_1 \cup Z$.*

Proof. Let $X = X_0 \cup X_1$. We analyze the number of connected components of $G - (X \cup Z)$ by splitting them into three categories.

1. Components with at least two neighbors in Z. Consider a subgraph $(Z \cup Y, E)$ of $\mathcal{C}(G, X \cup Z)$ given by restricting the vertex-side to Z and the component-side to those components Y that have at least two neighbors in Z. This graph is a minor of $G - X$, so it is outerplanar. By Proposition 3.3.7, we get $|Y| \leq 4 \cdot |Z|$.

2. Components with exactly one neighbor in Z and at least one in X. We call such a component a dangling component. For a connected component H of $G - X$, consider the collection of dangling components $(C_i)_{i=1}^{\ell}$ within H. Since each dangling component has exactly one neighbor in H, removing it does not affect connectivity of H. Therefore the graph $H' = H - \bigcup_{i=1}^{\ell} C_i$ is connected. Note that H' cannot be empty since it must contain at least one vertex in Z. For each vertex $x \in X$, there are at most two dangling components in H which are adjacent to x: if there were three C_1, C_2, C_3, they would form a minor model of $K_{2,3}$ together with x and $V(H')$. By Observation 3.3.4 this would contradict outerplanarity of $G - (X \cup \{x\})$.

   Hence the number of dangling components within H is at most twice as large as $|N_G(H) \cap X|$, which is the degree of H in $\mathcal{C}(G, X)$. The total number of dangling components is thus at most 2 times the sum of degrees of the component-nodes in $\mathcal{C}(G, X)$, which equals the number of edges in $\mathcal{C}(G, X)$. We obtain a bound $2s$ on the total number of dangling components.

3. Components without any neighbors in Z. These are also components of $G - X$, so there are at most s of them. □

The previous lemma gives us a bound on the number of components outside the modulator with at least two neighbors. To bound the total number of components outside the modulator, we employ the following reduction rule to remove the remaining components with at most one neighbor.

Reduction Rule 3.2. If for some $C \subseteq V(G)$ the graph $G\langle C \rangle$ is outerplanar and it holds that $|N_G(C)| \leq 1$, then remove the vertex set C.

Safeness of this rule follows from Observation 3.2.3, which implies $\mathsf{opd}(G - C) = \mathsf{opd}(G)$.

With these properties at hand, we are able to construct the desired extension of the augmented modulator. The decomposition below is inspired by the notion of a near-protrusion [61], combined with the idea of the augmented modulator, and with an $\mathcal{O}(k^3)$ bound on the number of leftover connected components.

Definition 3.3.15. For $k, c, d \in \mathbb{N}$ a (k, c, d)-*outerplanar decomposition* of a graph G is a triple (X_0, X_1, Z) of disjoint vertex sets in G, such that:

1. (X_0, X_1) is a (k, c)-augmented modulator for (G, k),
2. for each pair $u, v \in X_0 \cup X_1$ of distinct vertices one of the following holds:
   (a) vertices u, v are $(X_0 \cup X_1 \cup Z)$-separated, or
   (b) there are $k + 4$ internally vertex-disjoint (u, v)-paths in G, each with non-empty interior.
3. for each connected component C of $G-(X_0\cup X_1\cup Z)$ it holds that $|N_G(C)\cap Z| \leq 4$,
4. $|Z| \leq d \cdot (k+3)^3$ and there are at most $d \cdot (k+3)^3$ connected components in $G - (X_0 \cup X_1 \cup Z)$.

Lemma 3.3.16. *There is a constant c, a function $f_3 \colon \mathbb{N} \to \mathbb{N}$, and a polynomial-time algorithm that, given an instance (G, k), either returns an equivalent instance (G', k'), where $k' \leq k$ and G' is subgraph of G, along with a $(k', c, f_3(c))$-outerplanar decomposition of G', or concludes that $\mathsf{opd}(G) > k$. If $\mathsf{opd}(G) \leq k$ then it holds that $\mathsf{opd}(G') = \mathsf{opd}(G) - (k - k')$. Furthermore, $c = 40$ and $f_3(c) = 3 \cdot f_1(c) + 24 \cdot f_2(c)$ (see Lemmas 3.3.8 and 3.3.12).*

Proof. Begin with Lemma 3.3.5 to either conclude $\mathsf{opd}(G) > k$ or find an equivalent instance (G'', k'), where $k' \leq k$ and G'' is a subgraph of G, along with an (k', c)-augmented modulator. Next, apply Reduction Rule 3.1 to obtain an equivalent instance (G', k') that satisfies the conditions in the statement, along with an (k', c)-augmented modulator (X_0, X_1), so that the number of vertices and edges in $\mathcal{C}(G, X_0 \cup X_1)$ is at most $f_1(c) \cdot (k' + 3)^3$ (see Lemma 3.3.8). This reduction rule may remove edges and vertices from the graph, so G' is a subgraph of G.

We find a set Z_0 of size at most $f_2(c) \cdot (k'+3)^3$ satisfying the Condition 2 with Lemma 3.3.12. Next, apply Lemma 3.3.13 to graph $G - (X_0 \cup X_1)$ and set Z_0 to compute $Z \supseteq Z_0$, $|Z| \leq 6 \cdot f_2(c) \cdot (k'+3)^3$, which satisfies the Condition 3. Observe that Condition 2 is preserved for any superset of Z_0, and hence for Z.

Now identify all connected components of $G - (X_0 \cup X_1 \cup Z)$ with only one neighbor and apply Reduction Rule 3.2 to remove them. Note that this removed only vertices disjoint from X_0, X_1, and Z, so Conditions 1, 2(a), and 3 remain satisfied. Since such a connected component only has one neighbor in $X_0 \cup X_1$, the number of (u, v)-paths in G cannot have been decreased for any distinct $u, v \in X_0 \cup X_1$, hence Condition 2(a) also remains satisfied. We complete the proof by showing that now Condition 4 holds.

First note that $|Z| \leq 6 \cdot f_2(c) \cdot (k'+3)^3 \leq f_3(c) \cdot (k'+3)^3$. It remains to show that the number of components in $G-(X_0 \cup X_1 \cup Z)$ is at most $f_3(c) \cdot (k'+3)^3$. Any such connected component has at least two neighbors since otherwise we would have applied Reduction Rule 3.2 to remove it. Any other connected component has at least two neighbors in $X_0 \cup X_1 \cup Z$, so by Lemma 3.3.14 there are at most $3 \cdot s + 4 \cdot |Z|$ of these components, where s denotes the number of edges in $\mathcal{C}(G, X_0 \cup X_1)$ which is upper bounded by $f_1(c) \cdot (k'+3)^3$ (see Lemma 3.3.8). Hence in total there are at most $3 \cdot f_1(c) \cdot (k'+3)^3 + 4 \cdot 6 \cdot f_2(c) \cdot (k'+3)^3 = f_3(c) \cdot (k'+3)^3$ components. □

As the last property of the (k, c, d)-outerplanar decomposition, we formulate the bound on the total number of connections between $X_0 \cup X_1 \cup Z$ and the leftover components, which will lead to the total kernel size $\mathcal{O}(k^4)$.

Lemma 3.3.17. *Let (X_0, X_1, Z) be a (k, c, d)-outerplanar decomposition of a graph G. Then the number of edges in the component graph $\mathcal{C}(G, X_0 \cup X_1 \cup Z)$ is at most $f_4(c, d) \cdot (k+3)^4$, where $f_4(c, d) = cd + 6c + 4d$.*

Proof. By Definition 3.3.15(4) there are at most $d \cdot (k+3)^3$ components of $G - (X_0 \cup X_1 \cup Z)$ and each can have at most $|X_0| \leq c \cdot k$ neighbors from X_0. It remains to bound the total number of edges from $X_1 \cup Z$. The graph given by restricting the vertex-side of $\mathcal{C}(G, X_0 \cup X_1 \cup Z)$ to $X_1 \cup Z$ is a minor of $G - X_0$, hence it is outerplanar and, by Observation 3.2.8, the number of edges is at most twice the number of vertices, that is, $2 \cdot (|X_1 \cup Z| + d \cdot (k+3)^3) \leq 6c \cdot k^2 + 4d \cdot (k+3)^3$. □

3.3.3 Reducing the size of the neighborhood

Given a (k, c, d)-outerplanar decomposition (X_0, X_1, Z), we will now present the final reduction rule to reduce the size of the neighborhood $N_G(X_0 \cup X_1)$ to $\mathcal{O}(k^4)$. As the size of Z is already bounded by $\mathcal{O}(k^3)$ we focus on reducing the size of $N_G(X_0 \cup X_1) \setminus Z$. We have already shown the number of edges in the component graph $\mathcal{C}(G, X_0 \cup X_1 \cup Z)$ is bounded by $\mathcal{O}(k^4)$, so it suffices to reduce the number of edges between a single modulator vertex $x \in X_0 \cup X_1$ and a connected component C of $G - (X_0 \cup X_1 \cup Z)$ to a constant. For this, we first show in the following lemma where the neighbors of x occur in C.

Lemma 3.3.18. *Suppose G is outerplanar, $x \in V(G)$, and $G - x$ is connected. Then the vertices from $N_G(x)$ lie on an induced path P in $G - x$ such that for each biconnected component B of $G - x$ and each pair of distinct vertices $u, v \in V(P) \cap V(B)$ we have that $uv \in E(G - x)$. We can find such a path in polynomial time.*

Proof. If $|N_G(x)| = 1$ this is trivially true, so we assume $|N_G(x)| \geq 2$ in the remainder of the proof.

Consider a tree T obtained from a spanning tree of $G - x$ by iteratively removing leaves that are not in $N_G(x)$. We show T is a path. If T contains a vertex y of degree at least 3 then $T - y$ contains three components containing a neighbor

of x and, since T is connected, neighbors of y. This forms a $K_{2,3}$-minor in G contradicting outerplanarity of G. Hence T is a path, and by construction both its leaves are a neighbor of x. We now describe how to obtain the desired induced path P from T. If T is not an induced path in G, there are two nonconsecutive vertices u, v in T with $uv \in E(G)$. If there is no vertex $w \in N_G(x)$ between u and v on T, then the path T' obtained from T by replacing the subpath between u and v with the edge uv is a shorter path containing all of $N_G(x)$. Exhaustively repeat this shortcutting step and call the resulting path P. Since this procedure does not affect the first and last vertices we know the first and last vertex of P are both neighbors of x. All operations to obtain P can be performed in polynomial time.

If the path P obtained after shortcutting is not an induced path in G, there are two nonconsecutive vertices u, v in P with $uv \in E(G)$. By construction of P we know that there is a vertex $w \in N_G(x)$ between u and v on P. Now G contains a K_4-minor since $\{x, u, v, w\}$ are pairwise connected by internally vertex-disjoint paths, contradicting outerplanarity of G. So P is an induced path.

Let B be an arbitrary biconnected component of $G - x$ and let $u, v \in B \cap P$ be distinct. If uv is a bridge in $G - x$, it is trivial to see that $uv \in E(G - x)$, so we can assume that B contains at least 3 vertices (so B is 2-connected). We first consider the case where u is the first vertex along P that is contained in B and v is the last. Since the first and the last vertex of P are neighbor to x, (u, x, v) forms a (u, v)-path in G. Since $u, v \in B$ and B is 2-connected, there is a cycle within B containing u and v. This gives us two internally vertex-disjoint paths from u to v within B. If $uv \notin E(G - x)$, these paths have non-empty interiors. Together with the path (u, x, v), this leads to a $K_{2,3}$-minor in G and contradicting its outerplanarity. Hence, we can assume that $uv \in E(G - x)$.

If u and v are not the first and last vertices along P contained in B, then there are two vertices u' and v' that are. Then $u'v'$ is an edge in $G - x$ and because P is an induced path, u' and v' have to be consecutive in P, contradicting existence of such a pair (u, v). □

We now investigate what happens when a modulator vertex $x \in X_0 \cup X_1$ is the only vertex in $X_0 \cup X_1$ that is adjacent to a connected component C of $G - (X_0 \cup X_1 \cup Z)$. If x has sufficiently many edges to a part of C that is not adjacent to Z, then one of these edges can be removed without affecting the outerplanar deletion number $\mathsf{opd}(G)$. We will also exploit this property for a reduction rule later in this chapter when we reduce the number of edges within a connected component of $G - (X_0 \cup X_1 \cup Z)$.

Lemma 3.3.19. *Suppose we are given a graph G, a vertex $x \in V(G)$, and five vertices $v_1, \ldots, v_5 \in N_G(x)$ that lie, in order of increasing index, on an induced path P in $G - x$ from v_1 to v_5, such that $N_G(x) \cap V(P) = \{v_1, \ldots, v_5\}$. Let C be the component of $G - \{v_1, v_5, x\}$ containing $P - \{v_1, v_5\}$. If $G\langle C\rangle$ is outerplanar, then $\mathsf{opd}(G) = \mathsf{opd}(G \setminus xv_3)$.*

Proof. Clearly for any $S \subseteq V(G)$ if $G - S$ is outerplanar, then $G \setminus xv_3 - S$ is also outerplanar, hence $\mathsf{opd}(G) \geq \mathsf{opd}(G \setminus xv_3)$. To show $\mathsf{opd}(G) \leq \mathsf{opd}(G \setminus xv_3)$, suppose $G \setminus xv_3 - S$ is outerplanar for some arbitrary $S \subseteq V(G)$. If $x \in S$ or $v_3 \in S$ then clearly $G - S$ is outerplanar, so suppose $x, v_3 \notin S$. We show $G - S'$ is outerplanar for some $S' \subseteq V(G)$ with $|S'| \leq |S|$. Consider the following cases:

1. If $|S \cap V(P)| = 0$ then $G \setminus xv_3 - S$ contains an induced cycle formed by x together with the subpath of P from v_2 to v_4. This cycle includes x and v_3, so by Lemma 3.2.6 the graph $G \setminus xv_3 - S$ remains outerplanar after adding the edge xv_3, hence $G - S$ is outerplanar.

2. If $|S \cap V(P)| \geq 2$ then let $S' := \{v_1, v_5\} \cup (S \setminus V(C))$. Since $|S'| \leq |S|$, showing that $G - S'$ is outerplanar proves the claim. Let $\overline{C} := G - V(C)$ and note that $\overline{C} - S'$ is outerplanar since it is a subgraph of $G \setminus xv_3 - S$. Also note that $G[V(C) \cup \{x\}]$ is outerplanar since it is a subgraph of $G[V(C) \cup \{v_1, v_5, x\}] = G\langle C \rangle$. Since for any connected component H of $G - S' - x$ the graph $(G - S')\langle H \rangle$ is a subgraph of $\overline{C} - S'$ or $G[V(C) \cup \{x\}]$ we have that $(G - S')\langle H \rangle$ is outerplanar. Then by Observation 3.2.3 the graph $G - S'$ is outerplanar.

3. If $|S \cap V(P)| = 1$ then let $u \in S \cap V(P)$ and assume without loss of generality that u lies on the subpath of P from v_3 to v_5, so the subpath of P from v_1 to v_3 does not contain vertices of S (recall that $v_3 \notin S$). Let $S' := \{v_5\} \cup (S \setminus V(C))$ and note that $|S'| \leq |S|$. We shall show that $G - S'$ is outerplanar. Since $x, v_1 \notin S$, we have that also $x, v_1 \notin S'$, so $xv_1 \in E(G - S')$. In order to apply Lemma 3.2.4 to $G - S'$ and xv_1 we have to show that

   - for each connected component C' of $G - S' - \{v_1, x\}$ the graph $(G - S')\langle C' \rangle$ is outerplanar, and
   - there are at most two induced internally vertex-disjoint (v_1, x)-paths in $(G - S') \setminus v_1x$.

   Because $v_5 \in S'$ we have $G - S' - \{v_1, x\} = G - \{v_1, v_5, x\} - S'$ and since C is a connected component of $G - \{v_1, v_5, x\}$ we have that all connected components of $G - S' - \{v_1, x\}$ are either a connected component of $C - S' = C$ or of $G - S' - \{v_1, x\} - V(C)$. It is given that C is connected and $G[V(C) \cup \{v_1, v_5, x\}]$ is outerplanar so then $G[V(C) \cup \{v_1, x\}] = (G - S')\langle C \rangle$ is also outerplanar. Any other connected component C' is a connected component of $G - S' - \{v_1, x\} - V(C)$, so we have that $(G - S')\langle C' \rangle$ is a subgraph of $G - S' - V(C)$. This is in turn, a subgraph of $G \setminus xv_3 - S$ which is outerplanar. Hence $(G - S')\langle C' \rangle$ is outerplanar.

   It remains to show that there are at most two induced internally vertex-disjoint (v_1, x)-paths in $(G - S') \setminus v_1x$. Suppose for contradiction that $(G - S') \setminus v_1x$ contains three induced internally vertex-disjoint (v_1, x)-paths. As shown before, C is a connected component of $G - S' - \{v_1, x\}$ adjacent

to v_1 and x, so there exists an induced (v_1, x)-path P_1 in $G - S' \setminus v_1x$ whose internal vertices all lie in C. Since $G\langle C\rangle$ is outerplanar and C is connected, by Lemma 3.2.5 the graph $G\langle C\rangle$ does not contain two internally vertex-disjoint (v_1, x)-paths with non-empty interiors. Hence there are two induced internally vertex-disjoint (v_1, x)-paths P_2, P_3 in $(G-S'\setminus v_1x)-V(C)$. Observe that P_2 and P_3 are then disjoint from $S \setminus S' \subseteq V(C)$ and do not contain xv_3. It follows that P_1, P_2 and P_3 are three induced internally vertex-disjoint (v_1, x)-paths in $G \setminus xv_3 - S$, contradicting its outerplanarity by Lemma 3.2.4. We conclude also the second condition of Lemma 3.2.4 holds for $G - S'$ and the edge v_1x, hence $G - S'$ is outerplanar. □

We now use the properties of the (k, c, d)-outerplanar decomposition to show that any solution of size at most k contains all but possibly one vertex from $(X_0 \cup X_1) \cap N_G(C)$, where C is a connected component from $G - (X_0 \cup X_1 \cup Z)$. We use this fact together with the result from Lemma 3.3.19 to identify an irrelevant edge, which leads to the following reduction rule:

Reduction Rule 3.3. Given a (k, c, d)-outerplanar decomposition (X_0, X_1, Z) of a graph G, a vertex $x \in X_0 \cup X_1$, and five vertices $v_1, \ldots, v_5 \in N_G(x) \setminus (X_0 \cup X_1)$ that lie, in order of increasing index, on an induced path P in $G - (X_0 \cup X_1)$ from v_1 to v_5, such that $N_G(x) \cap V(P) = \{v_1, \ldots, v_5\}$. Let C be the component of $G - (X_0 \cup X_1) - \{v_1, v_5\}$ containing $P - \{v_1, v_5\}$. If $V(C) \cap Z = \emptyset$ remove the edge xv_3.

Lemma 3.3.20 (Safeness)**.** *Suppose that applying Reduction Rule 3.3 to a graph G removes the edge $e = xv_3$. If $\mathsf{opd}(G) > k$ then $\mathsf{opd}(G \setminus e) > k$ and if $\mathsf{opd}(G) \leq k$ then $\mathsf{opd}(G \setminus e) = \mathsf{opd}(G)$.*

Proof. Clearly $\mathsf{opd}(G \setminus e) \leq \mathsf{opd}(G)$ so it suffices to show that $\mathsf{opd}(G \setminus e) \leq k$ implies $\mathsf{opd}(G\setminus e) = \mathsf{opd}(G)$. Suppose $G\setminus e-S$ is outerplanar for some $S \subseteq V(G)$ of size at most k; we prove $\mathsf{opd}(G) \leq |S|$. If S contains x or v_3 then the claim is trivial. Otherwise let $X := X_0 \cup X_1$ and $X_S := X \cap S$ and note that $x \notin X_S$. Since C is a connected component of $G - X - \{v_1, v_5\}$ we have that $N_G(C) \subseteq X \cup \{v_1, v_5\}$. We first show that $N_G(C) \subseteq X_S \cup \{v_1, v_5, x\}$. Suppose for contradiction that some $u \in N_G(C)$ is not contained in $X_S \cup \{v_1, v_5, x\}$, so that $u \in X \setminus (S \cup \{v\})$. Since x and u are both neighbor to C, which is connected and does not contain vertices from X or Z, we have that x and u are not $(X \cup Z)$-separated. It follows from Definition 3.3.15(2) that there are $k + 4$ internally vertex-disjoint paths, with non-empty interior, connecting x and u in G. At most one of these paths contains the edge e, so in $G \setminus e$ there are at least $k + 3$ internally vertex-disjoint paths, with non-empty interior, connecting x and u. Since $x, u \notin S$, and $|S| \leq k$ we have that $G \setminus e - S$ has at least 3 internally vertex-disjoint paths, with non-empty interior, connecting x and v. This contradicts outerplanarity of $G \setminus e - S$. Hence $N_G(C) \subseteq X_S \cup \{v_1, v_5, x\}$.

Consider the graph $G' := G - X_S$. To prove that $\mathsf{opd}(G) \leq |S|$, it suffices to prove $\mathsf{opd}(G') \leq |S| - |X_S|$. Observe that $x \in V(G')$ and $v_1, \ldots, v_5 \in N_{G'}(x)$

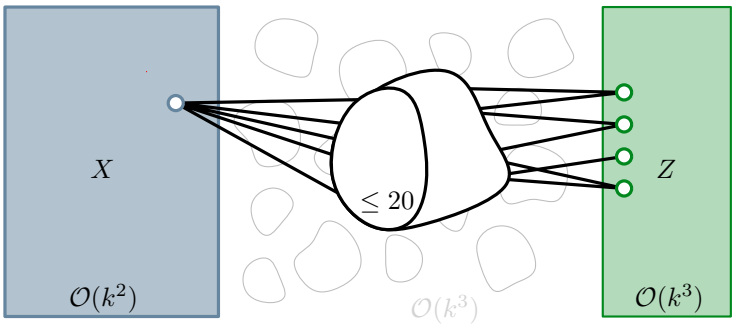


Figure 3.3 An illustration of Lemma 3.3.21. Given a (k, c, d)-outerplanar decomposition (X_0, X_1, Z) of a graph G, a vertex $x \in X = X_0 \cup X_1$ and a component C of $G - (X_0 \cup X_1 \cup Z)$, we are guaranteed that $|N(C) \cap Z| \leq 4$ and we can apply Reduction Rule 3.3 until $|N(x) \cap V(C)| \leq 20$. The expressions at the bottom bound the size of X, the number of components of $G - (X \cup Z)$, and size of Z.

lie in order of increasing index on the induced path P in $G' - x$ from v_1 to v_5, such that $N_{G'}(x) \cap P = \{v_1, \ldots, v_5\}$. Let C' be the connected component of $G' - \{v_1, v_5, x\}$ containing $P - \{v_1, v_5\}$. In order to apply Lemma 3.3.19 we show that $G'[V(C') \cup \{v_1, v_5, x\}]$ is outerplanar.

Since C is connected and $N_G(C) \subseteq X_S \cup \{v_1, v_5, x\}$ we have that C is a connected component of $G - (X_S \cup \{v_1, v_5, x\}) = G' - \{v_1, v_5, x\}$. As C' is also a connected component of $G' - \{v_1, v_5, x\}$ and both C and C' contain $P - \{v_1, v_5\}$ they are the same connected component. It follows that $G[V(C') \cup \{v_1, v_5, x\}] = G[V(C) \cup \{v_1, v_5, x\}]$, which is outerplanar by Definition 3.3.3 since it only intersects with $X_0 \cup X_1$ on the single vertex x.

This shows Lemma 3.3.19 can be applied to G' with vertex x and the path P. Since $S \setminus X_S$ forms a size-$(|S| - |X_S|)$ outerplanar deletion set for $G' \setminus e$, it follows there is a size-$(|S| - |X_S|)$ outerplanar deletion set S' for G'. Then $S' \cup X_S$ forms a size-$|S|$ outerplanar deletion set for G. □

We now show how this reduction rule can be applied to reduce the number of edges between a vertex $x \in X_0 \cup X_1$ and a connected component in $G - (X_0 \cup X_1 \cup Z)$ to a constant. This leads to an $\mathcal{O}(k^4)$ bound on the size of $N_G(X_0 \cup X_1)$; see Figure 3.3.

Lemma 3.3.21. *There is a polynomial-time algorithm that, given a (k, c, d)-outerplanar decomposition (X_0, X_1, Z) of a graph G, a vertex $x \in X_0 \cup X_1$ and a component C of $G - (X_0 \cup X_1 \cup Z)$, applies Reduction Rule 3.3 or concludes that $|N_G(x) \cap V(C)| \leq 20$.*

Proof. We first describe the algorithm and then proceed to prove its correctness.

Algorithm If $x \notin N_G(C)$ then conclude $|N(x) \cap V(C)| \leq 20$. Otherwise let $C^+ := G[V(C) \cup (N_G(C) \cap Z) \cup \{x\}]$. Apply Lemma 3.3.18 to find an induced path P in C^+ containing all of $N_{C^+}(x)$ (we will show C^+ is outerplanar). Let the vertices $N_{C^+}(x) = \{v_1, \ldots, v_\ell\}$ be indexed by the order in which they occur on P. For all $1 \leq i \leq \ell - 4$ let P_i be the subpath of P from v_i to v_{i+4} and let C_i denote the connected component of $C^+ - \{v_i, v_{i+4}, x\}$ containing $P_i - \{v_i, v_{i+4}\}$. If for some $1 \leq i \leq \ell - 4$ we have $V(C_i) \cap Z = \emptyset$ then apply Reduction Rule 3.3 with x and P_i to remove the edge xv_{i+2}. Otherwise conclude $|N(x) \cap V(C)| \leq 20$.

All operations can be performed in polynomial time.

Correctness We first show Lemma 3.3.18 is applicable. Clearly $C^+ - x = G[V(C) \cup (N_G(C) \cap Z)]$ is connected since C is connected, so it remains to show that C^+ is outerplanar. This follows from the fact that C^+ is a subgraph of $G - ((X_0 \cup X_1) \setminus \{x\})$, which is outerplanar by Observation 3.3.4. For the remainder of the proof we first establish a number of properties of the graphs defined in the algorithm.

Claim 3.3.22. Any connected component of $C^+ - (V(P) \cup \{x\})$ has at most two neighbors in $V(P)$ and they must be consecutive along P.

Proof. Consider the cycle in C^+ formed by the vertices $V(P) \cup \{x\}$ (recall the first and last vertices of P are neighbor to x). Since C^+ is outerplanar the claim follows directly from Lemma 3.2.7. ⌟

Observation 3.3.23. *For any $1 \leq i \leq \ell - 4$, since C_i is a connected component of $C^+ - \{v_i, v_{i+4}, x\}$ we have that any connected component of $C_i - V(P)$ is also a connected component of $C^+ - \{v_i, v_{i+4}, x\} - V(P) = C^+ - (V(P) \cup \{x\})$.*

Claim 3.3.24. For all $1 \leq i \leq \ell - 4$ we have $V(C_i) \cap V(P) = V(P_i - \{v_i, v_{i+4}\})$.

Proof. Let $u, v \in V(P) \cap C_i$ be distinct vertices. Since C_i is connected, there exists a path in C_i connecting u and v. Let P' be a shortest (u, v)-path in C_i. If P' contains a vertex not in P then this is a vertex in a connected component D in $C_i - V(P)$. By Observation 3.3.23 and Claim 3.3.22 we have that D has at most two neighbors in P and they are consecutive along P. Since the path P' must enter and leave D, the path visits both these neighbors, however since these neighbors are adjacent we can obtain a shorter (u, v)-path by skipping vertices in D. This contradicts that P' is a shortest (u, v)-path. It follows that the shortest path in C_i between any two vertices from P is a subpath of P. Since C_i does not contain v_i and v_{i+4} by definition, we have $V(C_i) \cap V(P) = V(P_i - \{v_i, v_{i+4}\})$. ⌟

Claim 3.3.25. For all $1 \leq i \leq \ell - 4$, if $V(C_i) \cap Z = \emptyset$ then $N_G(C_i) \subseteq X_0 \cup X_1 \cup \{v_i, v_{i+4}\}$.

Proof. Suppose for some $1 \leq i \leq \ell - 4$ that $V(C_i) \cap Z = \emptyset$ and let $v \in N_G(C_i)$. We show $v \in X_0 \cup X_1 \cup \{v_i, v_{i+4}\}$. If $v \in V(C^+)$ then since C_i is a connected component of $C^+ - \{v_i, v_{i+4}, x\}$ we have $v \in \{v_i, v_{i+4}, x\} \subseteq X_0 \cup X_1 \cup \{v_i, v_{i+4}\}$, so suppose $v \notin V(C^+) \supseteq V(C)$. Since $v \in N_G(C_i)$ there exists a vertex $u \in N_G(v) \cap V(C_i)$ and note that $u \notin Z$ because $N_G(C_i) \cap Z = \emptyset$. Since $u \in V(C_i) \subseteq V(C^+ - \{v_i, v_{i+4}, x\}) \subseteq V(C) \cup (N_G(C) \cap Z)$ and $u \notin Z$ we have $u \in V(C)$. Because u and v are neighbors we have $v \in N_G(C)$ so $v \in X_0 \cup X_1 \cup Z$ since C is a connected component of $G - (X_0 \cup X_1 \cup Z)$. Clearly if $v \in X_0 \cup X_1$ then the claim holds, so suppose $v \in Z$. However since $v \in N_G(C)$ and $v \in Z$ we have $v \in N_G(C) \cap Z$ so by definition of C^+ we have $v \in V(C^+)$, a contradiction since we assumed $v \notin V(C^+)$. ⌟

Suppose that for some $1 \leq i \leq \ell - 4$ we have $V(C_i) \cap Z = \emptyset$. In order to show that Reduction Rule 3.3 applies to x and P_i, first note that (X_0, X_1, Z) is a (k, c, d)-outerplanar decomposition of G and $x \in X_0 \cup X_1$. The vertices $v_i, \ldots, v_{i+4}$ lie on P_i, an induced path in $G - (X_0 \cup X_1)$ from v_i to v_{i+4} such that $N(x) \cap V(P_i) = \{v_i, \ldots, v_{i+4}\}$. We show that C_i is the connected component of $G - (X_0 \cup X_1) - \{v_i, v_{i+4}\}$ containing $P_i - \{v_i, v_{i+4}\}$.

Note that C_i does not contain any vertices from $X_0 \cup X_1 \cup \{v_i, v_{i+4}\}$ so C_i is a (connected) subgraph of $G - (X_0 \cup X_1) - \{v_i, v_{i+4}\}$. By Claim 3.3.25 we have $N_G(C_i) \subseteq X_0 \cup X_1 \cup \{v_1, v_{i+4}\}$. We can conclude that C_i is a connected component of $G - (X_0 \cup X_1) - \{v_i, v_{i+4}\}$, and by definition it contains $P_i - \{v_i, v_{i+4}\}$.

Finally, observe that $G[V(C_i) \cup \{v_i, v_{i+4}, x\}]$ is outerplanar as it is a subgraph of C^+. So since $V(C_i) \cap Z = \emptyset$ we have that Reduction Rule 3.3 applies.

Now suppose that the algorithm was unable to apply Reduction Rule 3.3, i.e., for all $1 \leq i \leq \ell - 4$ we have $V(C_i) \cap Z \neq \emptyset$. We show $|N(x) \cap V(C)| \leq 20$. Suppose for contradiction that $|N(x) \cap V(C)| > 20$. Then $N_{C^+}(x) > 20$, so the path P contains more than 20 neighbors of x, i.e., $\ell > 20$ so $C_1, C_5, C_9, C_{13}, C_{17}$ are defined. Since $V(C_i) \cap Z \neq \emptyset$ for all $1 \leq i \leq \ell - 4$ we know $C_1, C_5, C_9, C_{13}, C_{17}$ all contain a vertex from Z. We show $C_1, C_5, C_9, C_{13}, C_{17}$ are disjoint.

If $C_1, C_5, C_9, C_{13}, C_{17}$ are not disjoint, then there exist integers $i, j \in \{1, 5, 9, 13, 17\}$ and a vertex v such that $i < j$ and $v \in V(C_i) \cap V(C_j)$. Using Claim 3.3.24 we find that $V(P) \cap V(C_i) \cap V(C_j) \subseteq (V(C_i) \cap V(P)) \cap (V(C_j) \cap V(P)) = V(P_i - \{v_i, v_{i+4}\}) \cap V(P_j - \{v_j, v_{j+4}\}) = \emptyset$, so $v \notin V(P)$. Then v is a vertex in some connected component D of $C_i - V(P)$ and a connected component D' of $C_j - V(P)$. By Observation 3.3.23, both D and D' are connected components of $C^+ - (V(P) \cup \{x\})$, and since both contain v, they are the same connected component. Since C_i is connected, D must contain a neighbor $u_1 \in V(C_i) \cap V(P) = V(P_i - \{v_i, v_{i+4}\})$. Similarly $D' = D$ must contain a neighbor $u_2 \in V(C_j) \cap V(P) = V(P_j - \{v_j, v_{j+4}\})$. Since these two sets are disjoint we have $u_1 \neq u_2$. By Claim 3.3.22 these neighbors must be the only neighbors of D and they must be consecutive along P. However the vertex v_{i+4} lies on P between u_1 and u_2 since $i + 4 \leq j$ so u_1 and u_2 are not consecutive along P. By contradiction, $C_1, C_5, C_9, C_{13}, C_{17}$ are disjoint.

Since $C_1, C_5, C_9, C_{13}, C_{17}$ are disjoint subgraphs of C^+ and each subgraph contains a vertex from Z, we have that $|Z \cap V(C^+)| \geq 5$. By definition of C^+ we know $V(C^+) = V(C) \cup \{x\} \cup (N(C) \cap Z)$. Recall that $|N(C) \cap Z| \leq 4$ by Definition 3.3.15(3), so then $(V(C) \cup \{x\}) \cap Z \neq \emptyset$. This is a contradiction since $x \in X_0 \cup X_1$ and C is a connected component of $G - (X_0 \cup X_1 \cup Z)$, hence $|N(x) \cap V(C)| \leq 20$. □

We are going to apply Lemma 3.3.21 to a computed outerplanar decomposition in order to reduce the total neighborhood size of $X_0 \cup X_1$. This allows us to construct a final modulator L of size $\mathcal{O}(k^4)$ with a structure referred to in previous works as a protrusion decomposition. We can now proceed to proving a lemma that encapsulates application of Reduction Rule 3.3.

Lemma 3.3.26. *There exists a function $f_5 \colon \mathbb{N}^2 \to \mathbb{N}$ and a polynomial-time algorithm that, when given a (k, c, d)-outerplanar decomposition (X_0, X_1, Z) of a graph G, either applies Reduction Rule 3.2 or Reduction Rule 3.3, or outputs a set $L \subseteq V(G)$ such that*

1. *$|L| \leq f_5(c, d) \cdot (k + 3)^4$,*
2. *$|E_G(L, L)| \leq f_5(c, d) \cdot (k + 3)^4$,*
3. *there are at most $f_5(c, d) \cdot (k + 3)^4$ connected components in $G - L$, and*
4. *for each connected component C of $G - L$ the graph $G\langle C \rangle$ is outerplanar and $|N_G(C)| \leq 4$.*

Furthermore, $f_5(c, d) = 24 \cdot (20 \cdot f_4(c, d) + d + c + c^2)$ (see Lemma 3.3.17).

Proof. We first describe the algorithm and then proceed to prove its correctness.

Algorithm For all $x \in X_0 \cup X_1$ and connected components C in $G - (X_0 \cup X_1 \cup Z)$ we run the algorithm from Lemma 3.3.21 to apply Reduction Rule 3.3 or conclude that $|N_G(x) \cap V(C)| \leq 20$. If Reduction Rule 3.3 could not be applied to any x and C, we take $X = X_0 \cup X_1$ and apply Lemma 3.3.13 on the graph $G - X$ with vertex set $Z \cup N_G(X)$ to obtain a set $Z' \subseteq V(G) \setminus X$. We set $L = X \cup Z'$. If some component C of $G - L$ has at most one neighbor, we apply Reduction Rule 3.2 to remove C. Otherwise we return L.

Correctness It can easily be seen that Lemma 3.3.21 applies on all $x \in X_0 \cup X_1$ and connected components C in $G - (X_0 \cup X_1 \cup Z)$. If by calling Lemma 3.3.21 we have applied Reduction Rule 3.3 we can terminate the algorithm. Otherwise it holds that $|N_G(x) \cap V(C)| \leq 20$ for each $x \in X_0 \cup X_1$ and each connected component C of $G - (X_0 \cup X_1 \cup Z)$. Let us examine Z' and L given by the execution of the algorithm.

Clearly $G - X$ is outerplanar as it is a subgraph of $G - X_0$, which justifies that the algorithm correctly applies Lemma 3.3.13. To show Condition 1 and 2, we first prove a bound on $|E_G(X, V(G) \setminus (X \cup Z)|$.

Consider the component graph $H = \mathcal{C}(G, X \cup Z)$. For any $x \in X$ and connected component C of $G - (X \cup Z)$ if H does not contain an edge between x and the vertex representing C, then $|N_G(x) \cap V(C)| = 0$. If H contains an edge between x and the vertex representing C, then our earlier bound applies: $|N_G(x) \cap V(C)| \leq 20$. By Lemma 3.3.17 we have that H contains at most $f_4(c,d) \cdot (k+3)^4$ edges, so $|E_G(X, V(G) \setminus (X \cup Z))| \leq 20 \cdot f_4(c,d) \cdot (k+3)^4$ and $|N_G(X) \setminus Z| \leq 20 \cdot f_4(c,d) \cdot (k+3)^4$.

By Lemma 3.3.13 we have $|Z'| \leq 6 \cdot |Z \cup N_G(X)|$ and by Definition 3.3.15(4) we have $|Z| \leq d \cdot (k+3)^3$. To show Condition 1 we check that

$$|Z'| \leq 6 \cdot (20 \cdot f_4(c,d) + d)) \cdot (k+3)^4, \text{ and}$$
$$|L| = |Z'| + |X| \leq 6 \cdot (20 \cdot f_4(c,d) + d + c)) \cdot (k+3)^4.$$

Let us now bound the number of edges in $E_G(L, L)$. We group these edges into four categories: (a) edges within X_0, (b) edges between X_0 and $X_1 \cup Z$, (c) edges between X_0 and $L \setminus (X \cup Z)$, and (d) edges within $L \setminus X_0$. The number of edges in (a) is clearly at most $|X_0|^2 = c^2 \cdot k^2$. Similarly, in case (b) we obtain the bound $|X_0| \cdot |X_1 \cup Z| = ck \cdot (3c \cdot k^2 + d \cdot (k+3)^3) \leq (3c^2 + d) \cdot (k+3)^4$. To handle case (c), observe that $E_G(X_0, L \setminus (X \cup Z)) \subseteq E_G(X, V(G) \setminus (X \cup Z))$ and the size of this set has already been bounded by $20 \cdot f_4(c,d) \cdot (k+3)^4$. Finally, the subgraph of G induced by $L \setminus X_0$ is outerplanar and by Observation 3.2.8 we bound the number of edges in case (d) by $2 \cdot |L| \leq 2 \cdot 6 \cdot (20 \cdot f_4(c,d) + d + c)) \cdot (k+3)^4$. By collecting all summands we obtain that $|E_G(L, L)| \leq 13 \cdot (20 \cdot f_4(c,d) + d + c + c^2)) \cdot (k+3)^4$ and prove Condition 2.

To show Condition 4 note that $Z \cup N_G(X) \subseteq Z'$ by Lemma 3.3.13 and so $Z \cup N_G[X] \subseteq L$. Consider a connected component C of $G - L$. Since C does not contain neighbors of X we have $N_G[C] \cap X = \emptyset$. So then $G\langle C \rangle$ is a subgraph of $G - X$, hence it is outerplanar. Furthermore, by Lemma 3.3.13 we know that $|N_{G-X}(C)| \leq 4$ and so $|N_G(C)| \leq 4$.

If $|N_G(C)| \leq 1$ for some connected component C of $G - L$, we have applied Reduction Rule 3.2 and terminated the algorithm. If the algorithm is unable to apply this reduction rule, we know that all components of $G - L$ have at least two neighbors, which must belong to $L \setminus X$. The vertices representing these components in the component graph $\mathcal{C}(G - X, L)$ all have degree at least 2. Note also that this graph is bipartite (by definition) and outerplanar since it is a minor of $G - X$, which is outerplanar. It follows from Proposition 3.3.7 that $G - L$ has at most $4 \cdot |L| \leq 4 \cdot 6 \cdot (20 \cdot f_4(c,d) + d + c)) \cdot (k+3)^4$ components. This shows that Condition 3 holds. □

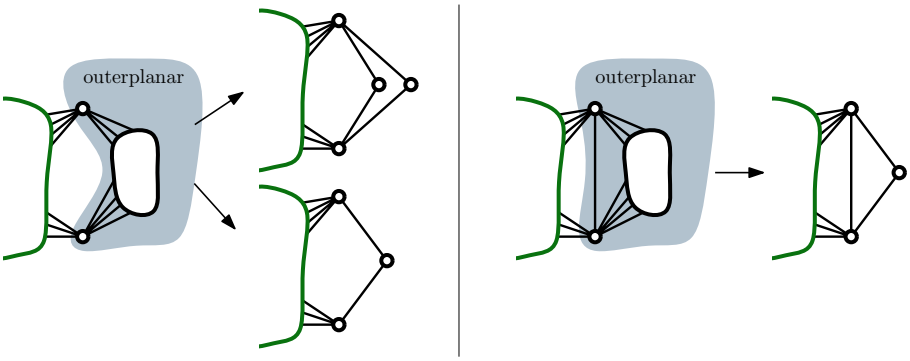


Figure 3.4 On the left a depiction of Reduction Rule 3.4, which reduces a connected subgraph to one or two vertices depending on its internal structure. On the right a depiction of Reduction Rule 3.5 which contracts a connected subgraph to a single vertex if it is outerplanar together with the two adjacent vertices that form its neighborhood.

3.4 Compressing the outerplanar subgraphs

3.4.1 Reducing the number of biconnected components

Once we arrive at the decomposition from Lemma 3.3.26, it remains to compress outerplanar subgraphs with a small boundary. First, we present a reduction to bound the number of biconnected components in such a subgraph. It will also come in useful later, for reducing the maximum size of a face in a biconnected outerplanar graph with a small boundary. Intuitively, this reduction checks whether an outerplanar subgraph with exactly two non-adjacent neighbors can supply one or two vertex-disjoint paths to the rest of the graph and replaces this subgraph with a minimal gadget with the same property, see also Figure 3.4.

Reduction Rule 3.4. Consider a graph G and vertex set $C \subseteq V(G)$ such that $N_G(C) = \{x, y\}$, $xy \notin E(G)$, $G[C]$ is connected, and $G\langle C\rangle$ is outerplanar. Let $P = (u_1, u_2, \ldots, u_m)$, $u_1 = x$, $u_m = y$ be any shortest path connecting x and y in $G\langle C\rangle$ and $D_1, D_2, \ldots, D_\ell$ be the connected components of $G\langle C\rangle - V(P)$. We consider 3 cases:

1. if there is a component D_i, for which $N_G(D_i)$ includes two non-consecutive elements of P, replace C with two vertices c_1, c_2, each adjacent to both x and y,
2. if there are two distinct components D_i, D_j, for which $|N_G(D_i) \cap N_G(D_j)| \geq 2$, replace C with two vertices c_1, c_2, each adjacent to both x and y,
3. otherwise replace C with one vertex c_1 adjacent to both x and y.

Lemma 3.4.1. *Let $x, y \in V(G)$ and $C \subseteq V(G)$ be such that Reduction Rule 3.4 applies and let G' be the graph obtained after application of the rule. Then G' is a minor of G.*

Proof. In case 1, there exists a component D_i adjacent to non-consecutive vertices u_j, u_h from $V(P)$, $j < h$. Let P_x, P_C, P_y denote the non-empty subpaths: $(u_1, \ldots, u_j), (u_{j+1}, \ldots, u_{h-1}), (u_h, \ldots, u_m)$. First, we remove all the connected components of $G[N[C]] - V(P)$ different from D_i. Next, we contract P_x into x, P_y into y, P_C into a vertex denoted c_1, and D_i into a vertex denoted c_2. By the choice of D_i we see that each of c_1, c_2 is adjacent to both x, y, therefore we have obtained G' through vertex deletions and edge contractions.

In case 2, there exist distinct components D_{i_1}, D_{i_2} both adjacent to vertices u_j, u_h from $V(P)$, with $j < h$. Let P_x, P_y denote the subpaths $(u_1, \ldots, u_j)$ and $(u_h, \ldots, u_m)$. Again, we begin by removing all the connected components of $G[N[C]] - V(P)$ different from D_{i_1}, D_{i_2}. Next, we contract P_x into x, P_y into y, D_{i_1} into a vertex denoted c_1, and D_{i_2} into a vertex denoted c_2, thus obtaining G'.

In case 3, we simply contract C into a vertex c_1. □

In order to show correctness of the reduction rule, we will prove that any outerplanar deletion set in the new instance can be turned into an outerplanar deletion set in the original instance without increasing its size. If we replaced the vertex set C with two vertices, we show that any outerplanar deletion set must break all the connections between the neighbors of C which go outside C. In the other case, when we replaced C with just one vertex, we show that we can undo the graph modification from Reduction Rule 3.4 while preserving the outerplanarity.

Lemma 3.4.2. *Let $x, y \in V(G)$ and $C \subseteq V(G)$ be such that Reduction Rule 3.4 applies and let G' be the graph obtained after application of the rule. If $S' \subseteq V(G')$ is an outerplanar deletion set in G', then there exists a set $S \subseteq V(G)$ such that $|S| \leq |S'|$ and which is an outerplanar deletion set in G.*

Proof. Let $C' \subseteq V(G')$ consist of the vertices put in place of C, that is, c_1 and, if we replaced C with two vertices, c_2. We naturally identify the elements of $V(G') \setminus C'$ with $V(G) \setminus C$. In particular, $N_{G'}(C') = N_G(C) = \{x, y\}$. We consider four cases:

- $S' \cap \{x, y\} \neq \emptyset$. We show that $G - S'$ is outerplanar. If $S' \supseteq \{x, y\}$ then this is immediate since $G[C]$ is outerplanar by assumption and forms a connected component of $G - S'$, while $(G - S') - V(C)$ is a subgraph of $G' - S'$.

  Otherwise, let $z = S' \cap \{x, y\}$ and $\overline{z} = \{x, y\} \setminus \{z\}$. As before, $(G - S') - V(C)$ is outerplanar since it is a subgraph of $G' - S'$. The graph $G - S'$ can be obtained from $(G - S') - V(C)$ by attaching $G[C]$ onto the cut vertex $\overline{z}$, and is therefore outerplanar by Observation 3.2.3.

- $S' \cap V(C') \neq \emptyset$. We define the set S as $(S' \setminus V(C')) \cup \{x\}$. It clearly holds that $|S| \leq |S'|$. Furthermore, y is a cut vertex in $G - S$. The graph $G - (S \cup$

$C \cup \{x\}$) is isomorphic with $G' - (S' \cup C' \cup \{x\})$, hence it is outerplanar. On the other hand, $G[C \cup \{y\}]$ is outerplanar by assumption. Therefore, all the components obtained by splitting $G - S$ at y are outerplanar and thus $G - S$ is outerplanar by Observation 3.2.3.

- $S' \cap N_{G'}[C'] = \emptyset$ and $C' = \{c_1, c_2\}$. We can simply write $S = S'$ as we have identified elements of $V(G') \setminus C'$ and $V(G) \setminus C$. Let $F_1, F_2 \ldots, F_\ell$ be the connected components of $G' - (S' \cup C' \cup \{x, y\})$. Observe that no F_i can be adjacent to both x, y, as otherwise x, y, c_1, c_2, F_i would form branch sets of a $K_{2,3}$-minor in $G' - S'$. The graph $G - S$ can be obtained from $G\langle C\rangle$ by appending the components $F_1, F_2 \ldots, F_\ell$ at x or y. For each $i \in [\ell]$ it holds that $G\langle F_i\rangle$ is a subgraph of $G' - S'$, so it is outerplanar. From Observation 3.2.3 we infer that $G - S$ is outerplanar.

- $S' \cap N_{G'}[C'] = \emptyset$ and $C' = \{c_1\}$. We again set $S = S'$ via vertex identification and we are going to transform $G' - S'$ into $G - S$ while preserving outerplanarity of the graph. Note that the path P contains at least one vertex from C as $xy \notin E(G)$. Subdividing a subdivided edge multiple times preserves outerplanarity, and so does replacing (x, c_1, y) with P. Let G'' denote the resulting graph.

  Since P is a shortest (x, y)-path in $G\langle C\rangle$, there are no edges in $G\langle C\rangle$ connecting non-adjacent vertices of P. Recall that $D_1, D_2, \ldots, D_\ell$ are the connected components of $G\langle C\rangle - V(P)$. Since $C' = \{c_1\}$, the conditions from cases 1 and 2 in Reduction Rule 3.4 are not satisfied. Therefore each component D_i is either adjacent to one vertex from $V(P)$ or to two vertices which are consecutive. Furthermore, for any pair of consecutive vertices on P, there can be only one component D_i adjacent to both of them.

  For each $i \in [\ell]$ it holds that $N_G[D_i] \subseteq N_G[C]$, so $G\langle D_i\rangle$ is outerplanar. If D_i has two neighbors u, v in P then any (u, v)-path in $G'' \setminus uv$ includes x or y as an internal vertex, hence there cannot be two induced internally vertex-disjoint (u, v)-paths in $G'' \setminus uv$. By Lemma 3.2.5 appending D_i to the edge uv in G'' supplies at most one more induced (u, v)-path and no other D_j, $j \in [\ell] \setminus \{i\}$, can supply a (u, v)-path in $G'' \setminus uv$, so this preserves outerplanarity due to Lemma 3.2.4 applied to the edge uv. Next, by Observation 3.2.3 the graph obtained by appending each component adjacent to a single vertex is still outerplanar. We have replaced c_1 back with C, thus transforming $G' - S'$ into $G - S$, while preserving outerplanarity of the graph, hence $G - S$ is outerplanar.

As $N_{G'}[C'] = V(C') \cup \{x, y\}$, the case distinction is exhaustive and completes the proof. □

Lemma 3.4.3. *Let G be a graph and G' be obtained from G by applying Reduction Rule 3.4. Then* $\mathsf{opd}(G') = \mathsf{opd}(G)$.

Proof. By Lemma 3.4.1 we know that G' is a minor of G, so $\mathsf{opd}(G') \leq \mathsf{opd}(G)$. On the other hand, if G' admits an outerplanar deletion set of size at most ℓ, then Lemma 3.4.2 guarantees that the same holds for G. □

We are now going to make use of Reduction Rule 3.4 to reduce the number of biconnected components in an outerplanar graph with a small boundary. Recall that the block-cut tree of a graph H has a vertex for each biconnected component of H and for each cut vertex in H. A biconnected component B and a cut vertex v are connected by an edge if $v \in B$. We will show that when the block-cut tree of $H = G\langle A\rangle$ is large then we can always find either one or two cut vertices that cut off an outerplanar subgraph which can be either removed or compressed. Recall that $\partial_G(B) = N_G(V(G) \setminus B)$ denotes the boundary of vertex set $B \subseteq V(G)$ in graph G.

Lemma 3.4.4. *For a graph G and a vertex set $A \subseteq V(G)$, such that $|N_G(A)| \leq 4$, $G[A]$ is connected, and $G\langle A\rangle$ is outerplanar, there is a polynomial-time algorithm that, given G and A satisfying the conditions above, outputs either*

1. *a block-cut tree of $G\langle A\rangle$ with at most 25 biconnected components, where each such biconnected component B satisfies $|\partial_G(B)| \leq 4$, or*
2. *a vertex set $C \subseteq A$, to which either Reduction Rule 3.2 or Reduction Rule 3.4 applies and decreases the number of vertices in the graph.*

Proof. We begin by computing the block-cut tree T of $G\langle A\rangle$ and rooting it at an arbitrary node [75]. For a node $t \in V(T)$ let $\chi(t)$ denote the vertex set represented by t, either a biconnected component, or a single vertex of t that corresponds to a cut vertex. Note that each leaf in T must represent a biconnected component. Furthermore, observe that no vertex from $N_G(A)$ can be a cut vertex in $G\langle A\rangle$, because $G[A]$ is connected. Therefore for each $v \in N_G(A)$ there is a unique biconnected component containing v. Let $t_v \in V(T)$ be the node in the block-cut tree representing this component.

First suppose that there exists a biconnected component B of $G\langle A\rangle$ such that $|\partial_G(B)| > 4$. The set $\partial_G(B)$ is a disjoint union of $N_G(A) \cap B$ and the cut vertices of $G\langle A\rangle$ lying in B. Let $d = |N_G(A) \cap B|$. Then there are at least $5-d$ cut vertices of $G\langle A\rangle$ lying in B, but at most $4-d$ vertices of $N_G(A) \setminus B$. By a counting argument, there is one cut vertex $v \in B$ of $G\langle A\rangle$ which separates $B \setminus \{v\}$ from a set $C \subseteq N_G[A]$ which does not contain any vertex from $N_G(A)$. Hence, $C \subseteq A$ and it induces a connected outerplanar subgraph of G having exactly one neighbor in G. Therefore, Reduction Rule 3.2 applies for C and removes it, decreasing the number of vertices in G.

Suppose for the rest of the proof that there are at least 26 biconnected components of $G\langle A\rangle$. Let $L \subseteq V(T)$ denote the LCA closure of the set $\{t_v \mid v \in N_G(A)\}$. By Lemma 3.2.9 we know that $|L| \leq 7$ and each connected subtree of $T - L$ then is adjacent to at most two nodes from L. It follows that if $t \in V(T) \setminus L$, then $\chi(t) \cap N_G(A) = \emptyset$.

Suppose that some component T_C of $T-L$ is adjacent to just one node from L. It is either a cut vertex or its neighbor in T_C is a cut vertex. The set $C = \bigcup_{t \in T_C} \chi(t)$ has an empty intersection with $N_G(A)$ and contains a vertex v which separates $C \setminus \{v\}$ from the rest of the graph G. Therefore we can find a subset $C' \subseteq C \setminus \{v\}$ which induces a connected subgraph of G, has exactly one neighbor v, and $C' \cup \{v\} \subseteq N_G[A]$ induces an outerplanar graph. Hence, Reduction Rule 3.2 applies for C' and removes it, decreasing the size of the graph.

A similar situation occurs when T_C has a vertex of degree at least 3. Then there exists a leaf t in T_C which represents a biconnected component and is not adjacent to L, so it is also a leaf in T. Again, $\chi(t) \cap N_G(A) = \emptyset$, so $\chi(t)$ contains a single vertex v which separates $\chi(t) \setminus \{v\}$ from the rest of the graph G. Analogously as above, Reduction Rule 3.2 applies and decreases the size of the graph.

Suppose now that the previous cases do not hold. Then each connected component of $T-L$ is adjacent to exactly two nodes from L and induces a path in T with the endpoints adjacent to L. If we contracted each such component to an edge connecting two nodes from L, we would obtain a tree with vertex set L and $|L|-1$ edges. Hence, the number of such components of $T - L$ is at most 6. Since the total number of biconnected components in $G\langle A\rangle$ is at least 26, we infer that there exists a connected component T_C of $T - L$ containing at least $\left\lceil \frac{26-7}{6} \right\rceil = 4$ nodes representing biconnected components of $G\langle A\rangle$. The set $C = \bigcup_{t \in T_C} \chi(t)$ contains two vertices u, v which together separate $C - \{u, v\}$ from the rest of $G\langle A\rangle$. Note that u, v belong to disjoint biconnected components of $G\langle A\rangle$, so $uv \notin E(G)$ and the set $C - \{u, v\}$ induces a connected subgraph of G. As $C \cap N_G(A) = \emptyset$, this implies that $C - \{u, v\}$ is a connected component of $G - \{u, v\}$. Furthermore, a union of 4 biconnected components has at least 5 vertices (the corner case occurs when they are all single edges), so $C - \{u, v\}$ has at least 3 vertices. Therefore, Reduction Rule 3.2 applies for $C - \{u, v\}$ and replaces it with at most two new vertices, therefore the total number of vertices in the graph decreases. All the described operations on the block-cut tree can be implemented to run in polynomial time. □

3.4.2 Reducing a large biconnected component

We now give the remaining reduction rules to reduce the size of a biconnected component of a protrusion. If a subgraph of a graph is outerplanar and adjacent only to two connected vertices, we can use Lemma 3.2.4 to argue that the entire subgraph can be replaced by any other outerplanar graph that is adjacent to the same two vertices. The following reduction rule exploits this by replacing such a subgraph with a single vertex, see also Figure 3.4.

Reduction Rule 3.5. Suppose that there is an edge $e = uv$ in a graph G such that $G - V(e)$ has a connected component C such that $G\langle C\rangle$ is outerplanar. Then contract C into a single vertex.

Lemma 3.4.5 (Safeness)**.** *Let G, $uv \in E(G)$, C satisfy the requirements of Reduction Rule 3.5 and let G' be obtained from G by contracting C to a new vertex c. Then* $\mathsf{opd}(G) = \mathsf{opd}(G')$.

Proof. It suffices to prove inequality $\mathsf{opd}(G) \leq \mathsf{opd}(G')$. If $|N_G(C)| = 1$, then we obtain the case already considered in Reduction Rule 3.2, which is safe due to Observation 3.2.3. Assume for the rest of the proof that $N_G(C) = \{u, v\}$.

Let $S' \subseteq V(G')$ be any outerplanar deletion set in G'; we will prove $\mathsf{opd}(G) \leq |S'|$. We first deal with two easy cases.

If $S' \cap \{u, v\} \neq \emptyset$, then we argue that $S = (S' \setminus \{c\})$ is an outerplanar deletion set in G. This follows from the fact that $G\langle C\rangle$ is outerplanar while C has at most one neighbor in $G - S'$, so that Observation 3.2.3 shows $G - S$ is outerplanar. Hence $\mathsf{opd}(G) \leq |S| \leq |S'|$.

If $c \in S'$ but $S' \cap \{u, v\} = \emptyset$, then $S = (S' \setminus \{c\}) \cup \{u\}$ is not larger than S'. To see that $G - S$ is outerplanar, we apply Observation 3.2.3 to the cut vertex v. Since $G\langle C\rangle$ is outerplanar by assumption, while $(G - S) - V(C)$ is a subgraph of $G' - S'$ and therefore outerplanar, Observation 3.2.3 ensures $G - S$ is outerplanar. Hence $\mathsf{opd}(G) \leq |S'|$.

Suppose now that $S' \cap \{u, v, c\} = \emptyset$. We show Lemma 3.2.4 applied to the graph $G - S'$ and edge uv. First, each connected component of $G - S' - \{u, v\}$ is outerplanar when considered together with its neighborhood. It remains to show that $(G - S') \setminus uv$ does not have three induced internally vertex-disjoint paths connecting u and v. Since (u, c, v) already gives such a path and $G' - S'$ is outerplanar, the graph $G' - (S' \cup \{c\}) \setminus uv = (G - (S' \cup C)) \setminus uv$ does not have two induced internally vertex-disjoint (u, v)-paths. By Lemma 3.2.5, there also cannot be two such paths in $G\langle C\rangle \setminus uv$. Therefore replacing c with C does not increase the number of induced internally vertex-disjoint (u, v)-paths and so Lemma 3.2.4 applies. We have thus shown that S' is an outerplanar deletion set in G, which concludes the proof. □

The next reduction rule addresses high degree vertices within a biconnected component. It uses the same idea as used in Reduction Rule 3.3, in fact, its safeness follows directly from Lemma 3.3.19. See also Figure 3.5.

Reduction Rule 3.6. Suppose we are given a graph G, a vertex $x \in V(G)$, and five vertices $v_1, \ldots, v_5 \in N_G(x)$ that lie, in order of increasing index, on an induced path P in $G - x$ from v_1 to v_5, such that $N_G(x) \cap V(P) = \{v_1, \ldots, v_5\}$. Let C be the component of $G - \{v_1, v_5, x\}$ containing $P - \{v_1, v_5\}$. If $G\langle C\rangle$ is outerplanar, then remove the edge xv_3.

The final reduction rule reduces the number of "internal" edges of an outerplanar biconnected graph. These are the edges whose endpoints form a separator in the graph. The previous rule addresses the case where these edges share an endpoint. The final reduction rule focuses on the case where the edges are disjoint: they form a matching.

Definition 3.4.6. For a graph G, a sequence of edges $e_1, \ldots, e_\ell \in E(G)$ is an *order-respecting matching* if the set of edges is a matching and if for all $1 \leq i < j < k \leq \ell$ we have that e_i and e_k are in different connected components of $G - V(e_j)$.

We can now formulate a property of biconnected graphs containing an order-respecting matching. This allows us to identify a number of cycles in the graph that are useful in the proof of the final reduction rule.

Lemma 3.4.7. *For an integer $\ell > 1$, if G is biconnected and $e_1, \ldots, e_\ell$ is an order-respecting matching in G then there exist vertex-disjoint paths P_x, P_y in G, such that each of P_x, P_y intersects every set $V(e_i)$, $i \in [\ell]$, and these intersections appear in order of increasing index.*

Proof. Let G' be the graph obtained from G by subdividing e_1 and e_ℓ with new vertices a and b. Since subdividing edges preserves biconnectivity, the graph G' is biconnected. So then there are two internally vertex-disjoint (a, b)-paths P_1 and P_2. Take $P_x := P_1 - \{a, b\}$ and $P_y := P_2 - \{a, b\}$. Let x_1, x_ℓ, y_1, and y_ℓ be the unique vertices in (respectively) $V(P_x) \cap V(e_1), V(P_x) \cap V(e_\ell), V(P_y) \cap V(e_1)$, and $V(P_y) \cap V(e_\ell)$. Observe that P_x is an (x_1, x_ℓ)-path in G and P_y is a (y_1, y_ℓ)-path in G and both paths are vertex disjoint.

For any $e_i \in \{e_2, \ldots, e_{\ell-1}\}$ we have by definition of order-respecting matching that x_1 and x_ℓ are in different connected components of $G - V(e_i)$, hence one of the endpoints of e_i must lie on P_x. Similarly one of the endpoints of e_i must lie on P_y. For all $1 < i < \ell$ let x_i denote the endpoint of e_i that lies on P_x and let y_i denote the endpoint of e_i that lies on P_y.

By Definition 3.4.6 we have for all $1 \leq i < j < k \leq \ell$ that x_i and x_k are in different connected components of $G - V(e_j)$. So the subpath of P_x between x_i and x_k must contain a vertex of $V(e_j)$. Since y_j lies on P_y which is disjoint from P_x we must have that x_j lies on P_x between x_i and x_k. Since $1 \leq i < j < k \leq \ell$ are arbitrary it follows that $\{x_1, \ldots, x_\ell\}$ occur in order of increasing index on the path P_x. Similarly, $\{y_1, \ldots, y_\ell\}$ occur in order of increasing index on the path P_y. □

We are now ready to formulate the final reduction rule. It applies within a biconnected outerplanar part of the graph that has multiple "internal" edges. We make use of the definition of order-respecting matching to define an order on the internal edges, so that the endpoints of the first and the last one separate the biconnected outerplanar part from the remainder of the graph. We show that in such a case the edges in the middle can be removed without affecting the outerplanar deletion number of the graph (see Figure 3.5). To advocate the safeness of such a graph modification, we observe that if any cycle in the modified outerplanar part is disjoint from a solution, then by Lemma 3.2.7 the solution must intersect all the paths connecting the endpoints of the first and the last edge of the matching on the "outside". This allows us to apply the outerplanarity criterion

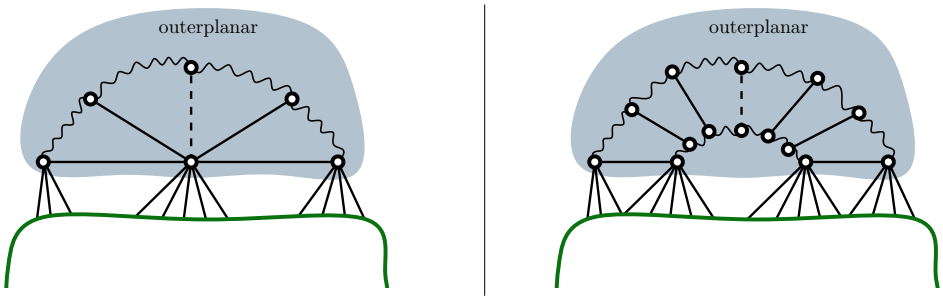


Figure 3.5 On the left a depiction of Reduction Rule 3.6 which is able to remove the middle edge of a fan structure in an outerplanar subgraph that is sufficiently isolated from the rest of the graph. On the right a depiction of Reduction Rule 3.7, which removes the middle edge of an order-respecting matching in an outerplanar subgraph that is sufficiently isolated from the rest of the graph.

from Lemma 3.2.4 to an edge on this cycle. In order to simplify the argument that such a cycle exists, we require an order-respecting matching of size 7.

Reduction Rule 3.7. Let G be a graph, $e_1, \ldots, e_7$ be a matching in G, and let C be a connected component of $G - (V(e_1) \cup V(e_7))$. If $e_1, \ldots, e_7$ is an order-respecting matching in $G\langle C\rangle$, $\{e_2, \ldots, e_6\} \subseteq E(C)$, $N_G(C) = V(e_1) \cup V(e_7)$, and $G\langle C\rangle$ is biconnected and outerplanar, then remove e_4.

Lemma 3.4.8 (Safeness)**.** *Let $e_1, \ldots, e_7$ be a matching in a graph G and let C be a connected component of $G - (V(e_1) \cup V(e_7))$. If Reduction Rule 3.7 applies to G, $e_1, \ldots, e_7$, and C, then $\mathsf{opd}(G \setminus e_4) = \mathsf{opd}(G)$.*

Proof. Clearly any solution to G is also a solution to $G \setminus e_4$ so $\mathsf{opd}(G \setminus e_4) \leq \mathsf{opd}(G)$. To show $\mathsf{opd}(G \setminus e_4) \geq \mathsf{opd}(G)$ suppose that $G \setminus e_4 - S$ is outerplanar. We will show that there is a set $S' \subseteq V(G)$ of size at most $|S|$ such that $G - S'$ is outerplanar. We first formulate the following structural property:

Claim 3.4.9. If $i \in \{2, \ldots, 6\}$ then for any induced path P in $G \setminus e_i$ between the endpoints of e_i, either P is an induced path in $G\langle C\rangle \setminus e_i$, or P has a subpath disjoint from C connecting an endpoint of e_1 to an endpoint of e_7.

Proof. Let $i \in \{2, \ldots, 6\}$ be arbitrary. Suppose that P is an induced path in $G \setminus e_i$ between the endpoints of e_i that contains a vertex v outside $G\langle C\rangle$. We show that P contains a subpath disjoint from C connecting an endpoint of e_1 to an endpoint of e_7. Consider the subpath P' of P from the last vertex x of $G\langle C\rangle$ before v, to the first vertex y of $G\langle C\rangle$ after v. By definition x, y have neighbors outside $G\langle C\rangle$ so $x, y \in N_G(C) = V(e_1) \cup V(e_7)$, showing that P' is disjoint from C. Since P is induced, so is P', hence there is no edge between x and y. It follows that one

of x, y is an endpoint of e_1 and the other an endpoint of e_7, hence P' is a subpath of P disjoint from C that connects an endpoint of e_1 to an endpoint of e_7. ⌟

We apply Lemma 3.4.7 to $G\langle C\rangle$ and the matching $(e_1, \ldots, e_7)$ to obtain vertex-disjoint paths P_x, P_y in $G\langle C\rangle$ which intersect each $V(e_i)$, $i \in [7]$, in order of increasing index. For $i \in [7]$ let $\{x_i, y_i\}$ be the endpoints of edge e_i with $x_i \in V(P_x)$ and $y_i \in V(P_y)$. For all $1 \leq p < q \leq 7$ let $C_{p,q}$ denote the cycle in $G\langle C\rangle$ as obtained by combining the edges e_p, e_q with the subpaths in P_x, P_y from $V(e_p)$ to $V(e_q)$. Observe that whenever $p_1 < q_1 < p_2 < q_2$, then $V(C_{p_1,q_1}) \cap V(C_{p_2,q_2}) = \emptyset$. For brevity let C_i denote $C_{i,i+1}$ for all $1 \leq i \leq 6$. We consider the following cases:

- If $|N_G[C] \cap S| \geq 3$ then take $S' := \{x_1, y_1, x_7\} \cup (S \setminus V(C))$ and observe that $|S'| \leq |S|$. We show $G - S'$ is outerplanar using Observation 3.2.3. We show for all connected components C' of $G - S' - y_7$ that $(G - S')\langle C'\rangle$ is outerplanar. Since $S' \cap V(C) = \emptyset$ we have that C is a connected component of $G - S' - V(e_1) - V(e_7)$. Since $(V(e_1) \cup V(e_7)) \setminus S' = \{y_7\}$ this yields that C is a connected component of $G - S' - y_7$. Clearly $(G - S')\langle C\rangle$ is outerplanar since it is a subgraph of $G\langle C\rangle$. Any other connected component C' of $G - S' - y_7$ clearly is a connected component of $G - S' - y_7 - V(C)$, so then $(G - S')\langle C'\rangle$ is a subgraph of $G - S' - V(C)$. Since both endpoints of e_4 are in C we have that $G - S' - V(C) = G \setminus e_4 - S' - V(C)$. This is a subgraph of $G \setminus e_4 - S$ since $S \subseteq S' \cup V(C)$. Because $G \setminus e_4 - S$ is outerplanar we can conclude that $(G - S')\langle C'\rangle$ is outerplanar. By Observation 3.2.3 we have that $G - S'$ is outerplanar.

- If $|N_G[C] \cap S| = 2$ and at least one of $\{C_1, C_6\}$ does not intersect S, we may assume without loss of generality (by symmetry) that C_6 does not intersect S. Take $S' := V(e_1) \cup (S \setminus V(C))$ and observe that $|S'| \leq |S|$. We show $G - S'$ is outerplanar using Lemma 3.2.4 on the edge e_7.

  We first show for all connected components C' of $(G - S') - V(e_7)$ that $(G - S')\langle C'\rangle$ is outerplanar. Since $S' \cap V(C) = \emptyset$ we have that C is a connected component of $G - S' - V(e_1) - V(e_7) = (G - S') - V(e_7)$. Clearly $(G - S')\langle C\rangle$ is outerplanar since it is a subgraph of $G\langle C\rangle$. Any other connected component C' of $G - S' - V(e_7)$ is a connected component of $G - S' - V(e_7) - V(C)$, so then $(G - S')\langle C'\rangle$ is a subgraph of $G - S' - V(C)$. Since both endpoints of e_4 are in C we have that $G - S' - V(C)$ is a subgraph of $G \setminus e_4 - S$, which is outerplanar. Hence $(G - S')\langle C'\rangle$ is outerplanar.

  It remains to show that there do not exist three induced internally vertex-disjoint paths in $(G - S') \setminus e_7$ connecting the endpoints of e_7. Since C_6 does not intersect S or S' and $V(C_6) \cap V(e_4) = \emptyset$ we have that $C_6 \setminus e_7$ is a path connecting the endpoints of e_7 in $(G - S) \setminus \{e_4, e_7\}$ and in $(G - S') \setminus e_7$. By shortcutting we obtain an induced path P^* in $(G - S) \setminus \{e_4, e_7\}$ and in $(G - S') \setminus e_7$, so that P^* connects the endpoints of e_7 and its internal vertices are all contained in C_6. Suppose for a contradiction that $(G - S') \setminus e_7$

contains three induced internally vertex-disjoint paths P_1, P_2, P_3 connecting the endpoints of e_7. Observe that if such a path P_i in $(G - S') \setminus e_7$ intersects $V(C)$, then its interior vertices belong entirely to $V(C)$, since $N_G(C) = V(e_1) \cup V(e_7)$ while $V(e_1) \subseteq S'$. If two paths P_i, P_j out of $\{P_1, P_2, P_3\}$ intersect $V(C)$, then we contradict Lemma 3.2.5 applied to the edge e_7 of the outerplanar subgraph $G[V(C) \cup V(e_7)]$ since C is a connected component of $G[V(C) \cup V(e_7)] - V(e_7)$ and would contain the interiors of two internally vertex-disjoint (u, v) paths. Hence at most one path P_i intersects $V(C)$, while the remaining two paths avoid $V(C)$ and therefore also exist in $(G - S) \setminus \{e_4, e_7\}$. But then we can replace P_i by P^* to obtain three induced internally vertex-disjoint paths in $(G - S) \setminus \{e_4, e_7\}$ connecting the endpoints of e_7, contradicting the outerplanarity of $(G - S) \setminus e_4$ by Lemma 3.2.4. This shows also the second condition of Lemma 3.2.4 is satisfied, hence $G - S'$ is outerplanar.

- Otherwise we are in one of the following cases: (1) $|N_G[C] \cap S| \leq 1$, or (2) $|N_G[C] \cap S| = 2$ and S intersects both C_1 and C_6. We show that $G - S$ is outerplanar.

  Claim 3.4.10. Suppose the preconditions of Reduction Rule 3.7 hold, $S \subseteq V(G)$, $G \setminus e_4 - S$ is outerplanar, and either $|N_G[C] \cap S| \leq 1$ or $|N_G[C] \cap S| = 2$ and S intersects both C_1 and C_6. Then any path in $G \setminus e_4 - S$ from an endpoint of e_1 to an endpoint of e_7 intersects $V(C)$.

  Proof. First we argue that at least one of $C_{1,3}, C_{3,5}, C_{5,7}$ is disjoint from S. If S intersects C_1 and C_6, then S cannot intersect $C_{3,5}$ since C_1, C_6, and $C_{3,5}$ are vertex-disjoint. If C_1 or C_6 does not intersect S then by assumption $|N_G[C] \cap S| \leq 1$ so S cannot intersect both $C_{1,3}$ and $C_{5,7}$ as they are vertex-disjoint. Hence S is disjoint from at least one of $C_{1,3}$, $C_{3,5}$, or $C_{5,7}$.

  For the sake of contradiction, suppose that there is a path P_z in $G \setminus e_4 - (S \cup C)$ which connects $z_1 \in V(e_1)$ to $z_7 \in V(e_7)$. First consider the case $V(C_{3,5}) \cap S = \emptyset$. Recall the paths P_x, P_y defined in the beginning of the proof. Let P'_x (resp. P'_y) be the subpath of P_x (resp. P_y) from $x_1 \in V(e_1)$ to $x_3 \in V(e_3)$ (resp. $y_1 \in V(e_1)$ to $y_3 \in V(e_3)$). If $|N_G[C] \cap S| = 2$, then S intersects C_6, which is disjoint from $V(P'_x) \cup V(P'_y)$. We therefore have $|(V(P'_x) \cup V(P'_y)) \cap S| \leq 1$ and by disjointness of P_x, P_y we infer that one of the paths P'_x, P'_y is disjoint from S; assume w.l.o.g. that it is P'_x. Let P''_x be P'_x if $z_1 \in V(P'_x)$ or P'_x concatenated with e_1 otherwise. Then P''_x connects $V(e_3)$ to z_1, it is internally vertex disjoint from $C_{3,5}$ and, since $z_1 \notin S$, it is vertex-disjoint from S. Using a symmetric argument we can construct a path, disjoint from S, which connects $V(e_5)$ to z_7. By concatenating these paths with P_z we obtain that there is a connected component of $(G \setminus e_4 - S) - V(C_{3,5})$ which is adjacent to at least two vertices on the cycle $C_{3,5}$: one from $V(e_3)$ and the other one from $V(e_5)$. Since $V(e_4)$ separates $V(e_3)$ from $V(e_5)$ in $G\langle C \rangle$,

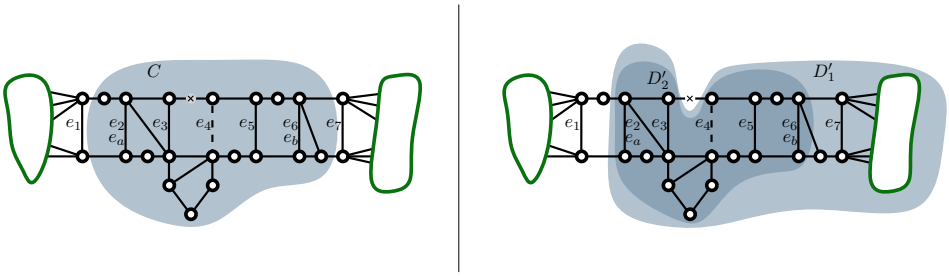


Figure 3.6 Illustration of the last case of the proof of Lemma 3.4.8. The illustration shows a situation to which Reduction Rule 3.7 is applicable to remove e_4. A solution S in $G \setminus e_4$ consisting of a singleton vertex is visualized by a cross, leading to choices for $e_a = e_2$ and $e_b = e_6$. On the right, the induced subgraph D'_1 and its subgraph D'_2 are highlighted.

these vertices are non-consecutive on the cycle $C_{3,5}$. By Lemma 3.2.7, this contradicts the assumption that $G \setminus e_4 - S$ is outerplanar.

It remains to consider the case $V(C_{1,3}) \cap S = \emptyset$, as the case $V(C_{5,7}) \cap S = \emptyset$ is symmetric. We have $V(C_1) \subseteq V(C_{1,3})$ so $V(C_1) \cap S = \emptyset$ and by the assumption $|N_G[C] \cap S| \leq 1$. Let P'_x (resp. P'_y) be the subpath of P_x (resp. P_y) from $x_3 \in V(e_3)$ to $x_7 \in V(e_7)$ (resp. $y_3 \in V(e_3)$ to $y_7 \in V(e_7)$). Recall that neither of P_x, P_y goes through e_4 as its endpoints lie on each of P_x, P_y. By disjointness of P_x, P_y we infer that one of the paths P'_x, P'_y is disjoint from S; assume w.l.o.g. that it is P'_x. Similarly as before, we define P''_x to be P'_x if $z_7 \in V(P'_x)$ or P'_x concatenated with e_7 otherwise. Then P''_x connects $V(e_3)$ to z_7 in $G\langle C \rangle \setminus e_4 - S$ and it is internally vertex disjoint with $C_{1,3}$. By concatenating P''_x with P_z we obtain that there is a connected component of $(G \setminus e_4 - S) - V(C_{1,3})$ which is adjacent to two non-consecutive vertices on the cycle $C_{1,3}$ ($z_1 \in V(e_1)$ and the other one from $V(e_3)$). By Lemma 3.2.7, this contradicts the initial assumption that $G \setminus e_4 - S$ is outerplanar. ⌟

Using this structural property we complete the proof of the third case. We start by defining two edges e_a and e_b, whose endpoints are disjoint from S, as follows; see Figure 3.6. If S intersects both C_1 and C_6 then take $e_a = e_3$ and $e_b = e_5$, otherwise we have $|N_G[C] \cap S| \leq 1$ so at least one of $e_a \in \{e_2, e_3\}$ is not hit by S. Similarly there is an edge $e_b \in \{e_5, e_6\}$ that does not intersect S. Now note that $\{e_1, e_a, e_4, e_b, e_7\}$ is an order-respecting matching in $G\langle C \rangle$, $N_G(V(e_a)) \subseteq N_G[C]$, and $N_G(V(e_b)) \subseteq N_G[C]$.

To show $G - S$ is outerplanar we use Lemma 3.2.4 on the edge e_a, i.e., we show for any connected component H of $G - S - V(e_a)$ that $(G - S)\langle H \rangle$ is outerplanar and there do not exist three induced internally vertex-disjoint paths in $(G - S) \setminus e_a$ connecting both endpoints of e_a.

To prove the latter, observe that any induced path in $(G - S) \setminus e_a$ is also an induced path in $G \setminus e_a$ and by Claim 3.4.10 they cannot contain a subpath from an endpoint of e_1 to an endpoint of e_7. Now by Claim 3.4.9 we have any such path is an induced path in $G\langle C\rangle \setminus e_a$. Since $G\langle C\rangle \setminus e_a$ is outerplanar we have by Lemma 3.2.4 that there are at most two such paths that are internally vertex-disjoint.

We now show for any connected component H of $G - S - V(e_a)$ that $(G - S)\langle H\rangle$ is outerplanar. Consider the case that H does not intersect $C_{a,b}$. Then we know that $(G - S)\langle H\rangle$ does not contain the edge e_4, hence $(G - S)\langle H\rangle$ is a subgraph of $G \setminus e_4 - S$, which is outerplanar. For the other case, let D_1 be the graph consisting of all connected components of $G - S - V(e_a)$ intersecting $C_{a,b}$. It suffices to show that $(G - S)\langle D_1\rangle$ is outerplanar.

We use Lemma 3.2.4 on the edge e_b in the graph $D_1' := (G - S)\langle D_1\rangle$, i.e., we show for any connected component H of $D_1' - V(e_b)$ that $D_1'\langle H\rangle$ is outerplanar and there are at most two induced internally vertex-disjoint paths in $D_1' \setminus e_b$ connecting the endpoints of e_b. Let D_2 be the graph consisting of all connected components of $D_1' - V(e_b)$ that intersect $C_{a,b}$. Since $(e_1, \ldots, e_7)$ is an order-respecting matching in $G\langle C\rangle$, we have that the graph $D_1'\langle D_2\rangle$ is a subgraph of $G\langle C\rangle$. Hence $D_1'\langle D_2\rangle$ is outerplanar. Any other connected component H of $D_1' - V(e_b)$ that does not intersect $C_{a,b}$ does not contain the edge e_4, hence $D_1'\langle H\rangle$ is a subgraph of $G \setminus e_4 - S$, which is outerplanar.

It remains to show that there are at most two induced internally vertex-disjoint paths in $D_1' \setminus e_b$ connecting both endpoints of e_b. By Claim 3.4.10 such paths cannot contain an endpoint of e_1. Since such a path is also an induced path in $G \setminus e_b$ so then by Claim 3.4.9 we have that such a path is an induced path in $G\langle C\rangle \setminus e_b$. Since $G\langle C\rangle \setminus e_b$ is outerplanar it follows from Lemma 3.2.4 that there are at most two such internally vertex-disjoint paths. Hence $D_1' = (G - S)\langle D_1\rangle$ is outerplanar completing the argument that $G - S$ is outerplanar.

We have shown in all cases that there exists a set S' of size at most $|S|$ such that $G - S'$ is outerplanar. □

This concludes our final reduction rule. What remains is how they can be applied in polynomial time to reduce the protrusions to a constant size. Since the number of biconnected components can be bounded by a constant (see Lemma 3.4.4), we proceed to show how to reduce the size of these biconnected components to a constant.

3.4.3 Reducible structures in biconnected components

In this section we use the weak dual of a biconnected outerplanar graph to argue that a large biconnected outerplanar graph contains a structure to which a reduction rule is applicable. We therefore need some terminology to relate objects in

a biconnected outerplanar graph G with those in its weak dual $\widehat{G}$. The following properties are well-known.

Observation 3.4.11 ([120, Corollary 6])**.** *Any biconnected outerplanar graph on at least three vertices has a unique Hamiltonian cycle.*

Observation 3.4.12 ([55])**.** *The weak dual of a biconnected outerplanar graph is a tree.*

It is justified to speak of *the* weak dual of a biconnected outerplanar graph, since all embeddings in which all vertices lie on the outer face have exactly the same set of faces. This can easily be seen by noting that the unique Hamiltonian cycle has a unique outerplanar embedding up to reversing the ordering, and that all remaining edges are chords of this cycle drawn in the interior. For a biconnected outerplanar graph G we can therefore uniquely classify its edges into *exterior edges* which lie on the outer face of an outerplanar embedding, and *interior edges* which bound two interior faces and are chords of the Hamiltonian cycle formed by the outer face.

For a biconnected outerplanar graph G we use $\widehat{G}$ to denote its weak dual. For an interior edge $e \in E(G)$ we use $\widehat{e}$ to denote its dual in $\widehat{G}$. Note that exterior edges, which lie on the outer face, do not have a dual in $\widehat{G}$ since we work with the *weak* dual. Each bounded face f of an outerplanar embedding of G corresponds to a vertex $\widehat{f}$ of $\widehat{G}$. For a vertex $\widehat{f}$ of $\widehat{G}$, we use $V_G(\widehat{f})$ to denote the vertices of G incident on face f. We extend this notation to vertex sets and subgraphs of $\widehat{G}$, so that $V_G(\widehat{G'}) = \bigcup_{\widehat{f} \in \widehat{G'}} V_G(\widehat{f})$. Similarly, for an edge $\widehat{e}$ of $\widehat{G}$ we use $V_G(\widehat{e})$ or simply $V(e)$ to denote the endpoints of the edge in G for which $\widehat{e}$ is the dual. For $\widehat{Y} \subseteq E(\widehat{G})$ we define $V_G(\widehat{Y}) := \bigcup_{\widehat{e} \in \widehat{Y}} V_G(\widehat{e})$.

It is insightful to think about the weak dual $\widehat{G}$ of a biconnected outerplanar graph G as prescribing a way in which to build graph G as follows: starting from the disjoint union of the induced cycles forming the interior faces corresponding to $V(\widehat{G})$, for each edge $\widehat{f'}\widehat{f''} = \widehat{e} \in E(\widehat{G})$ glue together the induced cycles for f' and f'' at the edge e that is common to both cycles. Since each interior face is an induced cycle and therefore biconnected, while gluing biconnected subgraphs along edges preserves biconnectivity, we observe the following.

Observation 3.4.13. *If G is a biconnected outerplanar graph and $\widehat{R}$ is a non-empty connected subtree of its weak dual $\widehat{G}$, then $G[V_G(\widehat{R})]$ is biconnected.*

Suppose we have a connected subtree $\widehat{R}$ of the weak dual $\widehat{G}$ of a biconnected outerplanar graph G. In the process of constructing G from the induced cycles formed by its faces, the subgraph $G[\widehat{R}]$ only becomes adjacent to vertices outside the subgraph by gluing faces outside $\widehat{R}$ onto faces of $\widehat{R}$. Since these are glued along edges whose dual has one endpoint inside and one endpoint outside $\widehat{R}$, we observe the following.

Observation 3.4.14. *Let G be a biconnected outerplanar graph and let $\widehat{R}$ be a connected subtree of its weak dual $\widehat{G}$. Let $\widehat{Y} \subseteq E(\widehat{G})$ denote those edges which have exactly one endpoint in $\widehat{R}$. The only vertices of $V_G(\widehat{R})$ which have a neighbor outside $V_G(\widehat{R})$ are those in $V_G(\widehat{Y})$.*

From these observations, we deduce the following lemma stating how a subtree of $\widehat{G}$ that is attached to the rest of $\widehat{G}$ at a single edge, represents a connected subgraph.

Lemma 3.4.15. *Let G be a biconnected outerplanar graph, let $\widehat{e} \in E(\widehat{G})$, and let $\widehat{R}$ be one of the two trees in $\widehat{G} \setminus \widehat{e}$. For $C = V_G(\widehat{R}) \setminus V_G(\widehat{e})$ the graph $G[C]$ is connected and $N_G(C) = V_G(\widehat{e})$.*

Proof. By Observation 3.4.13, the graph $G[V_G(\widehat{R})]$ is biconnected, so by Observation 3.4.11 it has a Hamiltonian cycle. It is easy to see that $\widehat{R}$ is the weak dual of $G[V_G(\widehat{R})]$. The latter graph contains edge e, since it is incident on both faces represented by the endpoints of $\widehat{e}$, one of which has its dual vertex in $\widehat{R}$. Since $\widehat{e} \notin E(\widehat{R})$, the edge e is an exterior edge of $G[\widehat{R}]$. So e is an edge of the unique Hamiltonian cycle of $G[\widehat{R}]$, which implies that the removal of $V(e)$ leaves that subgraph connected. Hence $V_G(\widehat{R}) \setminus V_G(\widehat{e})$ is a connected subgraph of G. By Observation 3.4.14 the only vertices of $V_G(\widehat{R})$ which have neighbors in G outside $V_G(\widehat{R})$ are those in $V_G(\widehat{e})$, which proves that $G[\widehat{R}] - V_G(\widehat{e})$ is a connected component of $G - V_G(\widehat{e})$. Its neighborhood in G is equal to $V_G(\widehat{e})$ since the predecessor and successor of the vertices of $V_G(\widehat{e})$ on the Hamiltonian cycle of $G[\widehat{R}]$ are contained in $G[\widehat{R}] - V_G(\widehat{e})$. □

The following lemma gives a condition under which removing the endpoints of a matching of two edges preserves connectivity. Recall the definition of order-respecting matching (Definition 3.4.6).

Lemma 3.4.16. *If G is a biconnected outerplanar graph and $M = \{e_1, e_2, e_3\}$ is an order-respecting matching in G such that e_1 and e_3 are exterior edges while e_2 is an interior edge, then the subgraph $G' := G - V(\{e_1, e_3\})$ is connected and $N_G(G') = V(\{e_1, e_3\})$.*

Proof. Consider the unique Hamiltonian cycle C of G. The exterior edges e_1 and e_3 lie on C while the interior edge e_2 is a chord of C. Since M is an order-respecting matching, the vertex sets $V(e_1)$ and $V(e_3)$ lie in different connected components of $G - V(e_2)$. Partition the cycle C into two vertex-disjoint $(V(e_1), V(e_3))$-paths C_1, C_2, and let C_1', C_2' be the paths formed by the interior vertices of C_1 and C_2. Since $V(e_2)$ separates $V(e_1)$ and $V(e_3)$, the paths C_1', C_2' are non-empty and both contain a vertex of $V(e_2)$. This implies that the graph $G[C_1' \cup C_2']$ is connected, since the edge e_2 connects the two paths. Since $C_1' \cup C_2'$ span all vertices of G except $V(\{e_1, e_3\})$ we have $G' = G[V(C_1') \cup V(C_2')]$, and since each vertex of $V(\{e_1, e_3\})$ is adjacent to an endpoint of C_1' or C_2', the lemma follows. □

The following shows that the order-respecting property of a matching can be deduced from its path-like structure in the dual.

Lemma 3.4.17. *Let G be a biconnected outerplanar graph. If $\widehat{P}$ is a path in $\widehat{G}$ and $M = \{e_1, \ldots, e_\ell\}$ is a matching in G such that the dual edges $\widehat{e}_1, \ldots, \widehat{e}_\ell$ appear on $\widehat{P}$ in the order as given by the indices, but not necessarily consecutively, then M is an order-respecting matching in G.*

Proof. For $1 < i < \ell$ let $\widehat{G}_i$ denote the tree in $\widehat{G} \setminus \widehat{e}_i$ that contains $\widehat{e}_1$. Since the edges $\widehat{e}_i$ lie on $\widehat{P}$ in order of increasing index, tree $\widehat{G}_i$ contains $\{\widehat{e}_j \mid j < i\}$ but no edge of $\{\widehat{e}_j \mid j > i\}$. Since the edges in the matching M are vertex-disjoint, this implies that $V_G(\widehat{G}_i)$ contains $V_G(\widehat{e}_j)$ for $j < i$ but contains no vertex of $V_G(\widehat{e}_j)$ for $j > i$.

By Lemma 3.4.15, we have $N_G(V_G(\widehat{G}_i) \setminus V_G(\widehat{e}_i)) = V_G(\widehat{e}_i)$ for all $1 < i < \ell$, which implies that the only vertices of $V_G(\widehat{G}_i)$ which have neighbors outside that set are those in $V_G(\widehat{e}_i)$. Hence the removal of $V_G(\widehat{e}_i)$ breaks all paths from endpoints of $V_G(\widehat{e}_j)$ to endpoints of $V_G(\widehat{e}_r)$ for $j < i < r$. This establishes that the matching M is order-respecting. □

The previous observation leads to the following lemma. Intuitively, it gives a property similar to that of a tree decomposition, saying that the vertices of $\widehat{G}$ representing faces containing a fixed vertex $t \in V(G)$ form a connected subtree of $\widehat{G}$.

Lemma 3.4.18. *Let G be a biconnected outerplanar graph and let $t \in V(G)$. If interior faces f and f' both contain t, then $t \in V(e)$ for each edge $\widehat{e}$ on the unique $(\widehat{f}, \widehat{f}')$-path $\widehat{P}$ in $\widehat{G}$.*

Proof. Let $\widehat{e}$ be an edge on $\widehat{P}$ and consider the two trees $\widehat{G}_1, \widehat{G}_2$ of $\widehat{G} \setminus \widehat{e}$. Since $\widehat{e}$ lies on the path between $\widehat{f}$ and $\widehat{f}'$, with $t \in V_G(\widehat{f}) \cap V_G(\widehat{f}')$, we have $t \in V_G(\widehat{G}_1) \cap V_G(\widehat{G}_2)$. By applying Observation 3.4.14 twice, once for $\widehat{G}_1$ and once for $\widehat{G}_2$, the graph $G - V_G(\widehat{e})$ has one connected component on vertex set $V_G(\widehat{G}_1) \setminus V_G(\widehat{e})$ and one connected component on vertex set $V_G(\widehat{G}_2) \setminus V_G(\widehat{e})$, which implies $V_G(\widehat{G}_1) \cap V_G(\widehat{G}_2) \subseteq V_G(\widehat{e}) = V(e)$. Since t belongs to $V_G(\widehat{G}_1) \cap V_G(\widehat{G}_2)$, we have $t \in V(e)$. □

The next lemma analyzes how a star of edges incident on a common vertex x are represented in the weak dual.

Lemma 3.4.19. *Let G be a biconnected outerplanar graph and let $\widehat{P}$ be a path in $\widehat{G}$ consisting of consecutive vertices and edges $\widehat{f}_0, \widehat{e}_1, \widehat{f}_1, \widehat{e}_2, \widehat{f}_2, \ldots, \widehat{e}_\ell, \widehat{f}_\ell$ of $\widehat{G}$, such that $x \in V(e_1) \cap V(e_\ell)$. Then $x \in V(e_i)$ for all $i \in [\ell]$, and letting v_i denote the endpoint of e_i other than x for each i, the vertices $\{v_i \mid i \in [\ell]\}$ lie in order of increasing index on an induced (v_1, v_ℓ)-path P in $G[V_G(\{\widehat{f}_1, \ldots, \widehat{f}_{\ell-1}\})] - x$ that contains no other neighbors of x.*

Proof. Since $\widehat{f_0}$ (respectively $\widehat{f_\ell}$) is incident on $\widehat{e}_1$ ($\widehat{e_\ell}$) and $x \in V(e_1) \cap V(e_\ell)$, we have $x \in V_G(\widehat{f_0}) \cap V_G(\widehat{f_\ell})$. By Lemma 3.4.18, this implies that $x \in V(e_i)$ for all $i \in [\ell]$.

Since G is a simple graph without parallel edges, the fact that $x \in V(e_i)$ for all $i \in [\ell]$ implies that the other endpoints of the edges e_i are all distinct. Let $\{v_i\} = V(e_i) \setminus \{x\}$ for each i. Since $\widehat{P}$ is a path in $\widehat{G}$, we can construct the graph $G' := G[V_G(\{\widehat{f}_1, \ldots, \widehat{f}_{\ell-1}\})]$ from disjoint induced cycles for the faces $f_1, \ldots, f_{\ell-1}$ by gluing them back-to-back along the edges $e_1, \ldots, e_\ell$, all of which are incident on x. Note that we do not use the faces f_0 and f_ℓ. Since in this construction we glue disjoint cycles together at distinct edges in a path-like sequence, and all edges along which we glue are incident on x, it follows that $G' - x$ is an induced path. It contains $v_1, \ldots, v_\ell$ in this order and no other neighbors of x. □

The last property of a weak dual we need states how removing two nonadjacent vertices incident on a common interior face separates the graph.

Lemma 3.4.20. *Let G be a biconnected outerplanar graph and let $\widehat{G}$ be its weak dual. Let $\widehat{f} \in V(\widehat{G})$ and let $u, v \in V(G)$ be nonadjacent vertices incident to f. Let E_1, E_2 be the edge sets of the two (u,v)-subpaths along the boundary of f, respectively. Let $\widehat{G}_i$ for $i \in [2]$ denote the union of all trees $\widehat{R}$ of $\widehat{G} - \widehat{f}$ for which the unique edge connecting $\widehat{R}$ to $\widehat{f}$ is the dual of an edge in E_i. Then the connected components of $G - \{u, v\}$ are $G[(V(E_i) \cup V_G(\widehat{G}_i)) \setminus \{u, v\}]$ for $i \in [2]$.*

Proof. Consider the process of constructing G from the disjoint cycles bounding its interior faces as dictated by the weak dual $\widehat{G}$. Since any interior face is an induced cycle, removing the nonadjacent vertices u, v from the induced cycle bounding f separates the cycle into exactly two paths P_1, P_2. Since weak dual witnesses that G can be constructed by gluing the subgraphs $G[\widehat{R}]$ for $\widehat{R}$ a tree of $\widehat{G} - \widehat{f}$ onto an edge of the cycle bounding f, the interior vertices of the paths P_1, P_2 belong to different connected components of $G - \{u, v\}$. Since each subgraph $G[\widehat{R}]$ is glued onto at least one interior vertices of a path P_1, P_2, the graph $G - \{u, v\}$ has exactly two connected components and their contents are as claimed; note that $V(E_i) \setminus \{u, v\}$ are exactly the interior vertices of path P_i. □

Using these properties we can now prove that a large biconnected outerplanar graph contains a reducible structure. Each of the three types of structures below can be reduced by one of our reduction rules.

Lemma 3.4.21. *Let G be a biconnected outerplanar graph and let $T \subseteq V(G)$ be a non-empty subset of vertices with $|T| \leq 4$. If $|V(G)| > 6288$, then in polynomial time we can identify one of the following* reducible structures *in G:*

- *two (possibly adjacent) vertices u, v such that there is a component C of $G - \{u, v\}$ that does not contain any vertex of T, for which $|V(C)| > 2$,*

- *a matching $e_1, \ldots, e_7$ in G, such that there is a single connected component C of $G-(V(e_1)\cup V(e_7))$ which contains $\{e_2, \ldots, e_6\}$ but no vertex from T, such that $N_G(C) = V(e_1) \cup V(e_7)$, the graph $G\langle C\rangle$ is biconnected, and $e_1, \ldots, e_7$ is an order-respecting matching in $G\langle C\rangle$, or*

- *a vertex $x \in V(G)$ and five vertices $v_1, \ldots, v_5 \in N_G(x)$ that lie, in order of increasing index, on an induced path P in $G - x$ from v_1 to v_5, such that $N_G(x) \cap V(P) = \{v_1, \ldots, v_5\}$, and such that the connected component of $G - \{v_1, v_5, x\}$ which contains $P - \{v_1, v_5\}$ contains no vertex from T.*

Proof. We will refer to vertices from T as terminals. For each $t \in T$, fix an interior face $f(t)$ of G incident to t and define $\widehat{f(T)} := \{\widehat{f(t)} \mid t \in T\}$. Let $\widehat{G}_T$ be the minimal subtree of the weak dual $\widehat{G}$ spanning $\widehat{f(T)}$. We say a vertex of $\widehat{G}_T$ is *important* if it belongs to $\widehat{f(T)}$ or has degree unequal to two in the graph $\widehat{G}_T$. By minimality of $\widehat{G}_T$ each leaf of $\widehat{G}_T$ belongs to $\widehat{f(T)}$, from which it easily follows that the edges of $\widehat{G}_T$ can be partitioned into at most five paths between important vertices, such that no interior vertex of such a path is important. The number five corresponds to the fact that any tree on at most four leaves without vertices of degree two has at most two internal vertices, so at most six vertices and hence five edges in total. The following claim shows the relevance of the set of important vertices.

Claim 3.4.22. Let $\widehat{R}$ be a connected subtree of $\widehat{G}$ and let

$$S = \{V(e) \mid \widehat{e} \text{ has exactly one endpoint in } \widehat{R}\}.$$

If $\widehat{R}$ contains no important vertex of $\widehat{G}_T$, then $V_G(\widehat{R}) \setminus S$ contains no vertex of T.

Proof. Suppose there exists $t \in T$ with $t \in V_G(\widehat{R})$, so t lies on a face f whose dual $\widehat{f}$ is in $\widehat{R}$. Since $\widehat{R}$ contains no important vertex, some tree $\widehat{R}'$ of $\widehat{G} - V(\widehat{R})$ contains the chosen face $\widehat{f(t)}$ representing t. The edge $\widehat{e}$ connecting $\widehat{R}$ to $\widehat{R}'$ lies on the path between $\widehat{f}$ and $\widehat{f(t)}$ in $\widehat{G}$ and has exactly one endpoint in $\widehat{R}$. By Lemma 3.4.18, we have $t \in V(e)$ and therefore $t \in S$. ⌟

We derive a number of claims showing how to find a reducible structure under certain conditions. After presenting these claims, we show that at least one of them guarantees the existence of a reducible structure if G is sufficiently large. The first claim shows that if any subtree of $\widehat{G}$ attaches to $\widehat{G}_T$ at a single edge and represents more than two vertices other than the attachments, we find a reducible structure of the first type.

Claim 3.4.23. Let $\widehat{R}$ be a connected component of $\widehat{G} - V(\widehat{G}_T)$, which is a tree, and let $\widehat{e}$ be the unique edge of $\widehat{G}$ connecting $\widehat{R}$ to $\widehat{G}_T$. If $|V_G(\widehat{R}) \setminus V(e)| > 2$, then G contains a reducible structure.

Proof. By Lemma 3.4.15 we know that $V_G(\widehat{R}) \setminus V(e)$ is the vertex set of a connected component of $G - V(e)$, and by Claim 3.4.22 this component contains no terminals since $\widehat{G}_T$ spans all important vertices. As the number of vertices in the connected component is larger than two, this yields a reducible structure of the first type for $\{u, v\} = V(e)$. ⌟

Next we show that if $\widehat{G}_T$ contains a long path of non-important vertices, we find a reducible structure of the second or third type.

Claim 3.4.24. Let $\widehat{P}$ be a subpath of $\widehat{G}_T$ such that no vertex of $\widehat{P}$ is important. If $|V(\widehat{P}')| > 6 \cdot 4 + 1$, then G contains a reducible structure.

Proof. Consider such a subpath $\widehat{P}$ of $\widehat{G}_T$. Let $\widehat{e}_0, \ldots, \widehat{e}_{24}$ be the first 25 edges on $\widehat{P}$. Partition these into sets $\widehat{E}_1, \ldots, \widehat{E}_6$ of four consecutive edges each, and let $\widehat{E}_7$ be a singleton set with the next edge. Observe that for any two edges $\widehat{e}_i, \widehat{e}_j$ on $\widehat{P}$ with $i < j - 1$, the subtree $\widehat{R}$ of $\widehat{G} \setminus \{\widehat{e}_i, \widehat{e}_j\}$ containing $\widehat{e}_{i+1}$ contains no important vertex since the only vertices of $\widehat{G}_T$ it contains belong to $\widehat{P}$; here we exploit the fact that all vertices of degree three or more in $\widehat{G}_T$ are important. Hence by Claim 3.4.22 the set $V_G(\widehat{R}) \setminus V(\{e_i, e_j\})$ contains no vertex of T. We will use this property below to find a reducible structure of the second or third type via a case distinction.

Suppose first that there exists $i \in [6]$ such that $V_G(\widehat{e}_{4i}) \cap V_G(\widehat{e}_{4(i+1)}) \neq \emptyset$, that is, some vertex $x \in V(G)$ is a common endpoint of e_{4i} and $e_{4(i+1)}$. Let $\widehat{E}'_i := \widehat{E}_i \cup \{\widehat{e}_{4(i+1)}\}$ and let $E'_i := \{e \mid \widehat{e} \in \widehat{E}'_i\}$. Let $\widehat{P}'$ be the four vertices of $\widehat{G}$ which are incident to two edges of $\widehat{E}_i$, that is, the four internal vertices of the path formed by the five edges $\widehat{E}_i$. By applying Lemma 3.4.19 to the path formed by $\widehat{E}_i$ we find $x \in V(e)$ for each $e \in E'_i$ while the other endpoints $v_1, \ldots, v_5$ of the edges in E'_i are all distinct and lie on an induced (v_1, v_5)-path P in $G[V_G(\widehat{P}')] - x$ containing no other neighbors of x. Let $\widehat{R}$ be the tree of $\widehat{G} \setminus \{\widehat{e}_{4i}, \widehat{e}_{4(i+1)}\}$ containing $\widehat{e}_{4i+1}$ and note that $V_G(\{\widehat{e}_{4i}, \widehat{e}_{4(i+1)}\}) = \{v_1, v_5, x\}$ and $V_G(\widehat{R}) \supseteq V_G(\widehat{P}')$. By Observation 3.4.14, the only vertices of $V_G(\widehat{R})$ which have neighbors outside $V_G(\widehat{R})$ are those in $V_G(\{\widehat{e}_{4i}, \widehat{e}_{4(i+1)}\}) = \{v_1, v_5, x\}$. Consider the connected component C of $G - \{v_1, v_5, x\}$ that contains $P - \{v_1, v_5\}$. By the previous argument, $V(C) \subseteq V_G(\widehat{R})$ while the construction ensures $V(C) \cap \{v_1, v_5, x\} = \emptyset$. By the argument in the first paragraph of this claim, $V_G(\widehat{R}) \setminus V(\{e_i, e_j\}) = V_G(\widehat{R}) \setminus \{v_1, v_5, x\}$ contains no vertex of T, implying that no vertex of T belongs to C. Hence G contains a reducible structure of the third type.

Now suppose the previous case does not apply; then the set $M = \{e_{4i} \mid 0 \leq i \leq 6\}$ is a matching in G. Since $\widehat{M}$ lies on the path $\widehat{P}$ in $\widehat{G}$ in the relative order given by the indices, by Lemma 3.4.17 the set M is an order-respecting matching in G, which implies it is order-respecting in all subgraphs containing this matching. Let $\widehat{R}$ be the tree of $\widehat{G} \setminus \{\widehat{e}_0, \widehat{e}_{24}\}$ containing the rest of $\widehat{M}$. By Observation 3.4.13, the graph $G[V_G(\widehat{R})]$ is biconnected. Since for each $0 \leq i \leq 6$

the tree $\widehat{R}$ contains a vertex incident on $\widehat{e}_{4i}$, which corresponds to a face in G incident on e_{4i}, it follows that $G[V_G(\widehat{R})]$ contains all edges of M. Since $\widehat{R}$ is the weak dual of $G[V_G(\widehat{R})]$, while $\widehat{R}$ does not contain the edges $\widehat{e}_0$ and $\widehat{e}_{24}$, the edges e_0, e_{24} are exterior edges of $G[V_G(\widehat{R})]$; since $\widehat{e}_4$ is contained in $E(\widehat{R})$ the edge e_4 is an interior edge of $G[V_G(\widehat{R})]$. By applying Lemma 3.4.16 to the order-respecting matching $\{e_0, e_4, e_{24}\}$ in the biconnected graph $G[V_G(\widehat{R})]$, we find that $C = G[V_G(\widehat{R})] - V(\{e_0, e_{24}\})$ is connected and contains a vertex adjacent to each member of $V(\{e_0, e_{24}\})$, so that $G\langle C\rangle = G[V_G(\widehat{R})]$ is biconnected and contains the order-respecting matching M. By the argument in the first paragraph, the set $V_G(\widehat{R}) \setminus V(\{e_0, e_{24}\})$ contains no vertex of T, so that C contains no vertex of T. Since Observation 3.4.14 shows that no vertex of C has a neighbor outside $V(\{e_0, e_{24}\})$ it follows that C is a connected component of $G - V(\{e_0, e_{24}\})$. Hence we find a reducible structure of the second type. ⌟

As the last ingredient, we show that if any interior face of G contains more than 16 vertices, we find a reducible structure.

Claim 3.4.25. If G contains an interior face f that is incident to more than 16 vertices, then G contains a reducible structure of the first type.

Proof. Let f be a bounded face in G and consider the cycle bounding f on the edge set $E(f)$. We call an edge e that lies on f a portal if the tree in $\widehat{G} \setminus \{\widehat{e}\}$ that does not contain $\widehat{f}$ contains a vertex of $\widehat{f(T)}$. Since $|f(T)| \leq 4$ at most four edges on f are portals. By Lemma 3.4.18, if a terminal t lies on f but $f(t) \neq \widehat{f}$, then the edge e for which $\widehat{e}$ lies on the path from $\widehat{f}$ to $\widehat{f(t)}$ is a portal.

Let $T' := (T \cap V(f)) \cup \{V(e) \mid e \in E(f) \text{ is a portal}\}$. Since each terminal t for which $\widehat{f(t)} \neq \widehat{f}$ contributes two adjacent vertices to T', while each terminal t with $\widehat{f(t)} = \widehat{f}$ contributes one vertex, it follows that T' can be partitioned into at most four sets of two vertices each, such that the vertices in each subset are consecutive along the face. Since f is incident to more than 16 vertices, this implies that there is a subpath P of the boundary of f consisting of five vertices, whose three interior vertices do not belong to T'. Let $\{u, v\}$ be the endpoints of P. Let $\widehat{G}_P$ denote the union of all trees $\widehat{R}$ of $\widehat{G} - \widehat{f}$ for which the unique edge $\widehat{e}$ connecting $\widehat{R}$ to $\widehat{f}$ satisfies $e \in E(P)$. By Lemma 3.4.20 there is a connected component C of $G - \{u, v\}$ on vertex set $(V(P) \cup V_G(\widehat{G}_P)) \setminus \{u, v\}$. This implies that all three interior vertices of P belong to C.

We conclude the proof by showing that C contains no vertex of T. To see this, note first that $V(P) \setminus \{u, v\}$ contains no vertex of T by choice of P. Since all endpoints of portals belong to T', while no interior vertex of P belongs to T', it follows that no edge of P is a portal. Consequently, for each edge $e \in E(P)$ the tree of $\widehat{G} \setminus \widehat{e}$ that does not contain $\widehat{f}$ does not contain any vertex of $\widehat{f(T)}$. We exploit this in the following argument. Suppose there exists $t \in T$ with $t \in V_G(\widehat{G}_P)$, and let e be the edge of P such that $\widehat{G} \setminus \widehat{e}$ contains a tree $\widehat{R}$ with $t \in V_G(\widehat{R})$.

Since e is not a portal, $\widehat{f(t)}$ does not belong to $\widehat{R}$. Hence $\widehat{f(t)}$ is either equal to $\widehat{f}$, or belongs to some tree $\widehat{R}'$ of $\widehat{G} - \widehat{f}$ for which the edge $\widehat{e}'$ connecting $\widehat{R'}$ to $\widehat{f}$ satisfies $e' \notin E(P)$. We consider these cases separately.

- If $\widehat{f(t)} = \widehat{f}$, then $t \in V(f)$ and therefore $t \in T'$. By construction, no interior vertex of P belongs to T'. Since all vertices that belong both to $V(f)$ and to $V_G(\widehat{G}_P)$ belong to P, it follows that t is an endpoint of P, so $t \in \{u, v\}$. This shows that t does not belong to the component C on vertex set $(V(P) \cup V_G(\widehat{G}_P)) \setminus \{u, v\}$, as required.
- If $\widehat{f(t)} \neq \widehat{f}$, then $\widehat{f(t)}$ lies in some tree $\widehat{R}'$ of $\widehat{G} - \widehat{f}$ for which the edge $\widehat{e}'$ connecting $\widehat{R}'$ to $\widehat{f}$ satisfies $e' \in E(f) \setminus E(P)$, where $E(f)$ are the edges of G bounding face f. Since both $\widehat{e}$ and $\widehat{e}'$ lie on the path in $\widehat{G}$ between $\widehat{f(t)}$ and a vertex of $\widehat{R}$ representing a face containing t, by Lemma 3.4.18 we have $t \in V(e') \cap V(e)$. So t is simultaneously an endpoint of the edge e that lies on P and the edge e' that lies on face f but not in P. Consequently, t is an endpoint of P and therefore $t \in \{u, v\}$, showing that t does not belong to the component C on vertex set $(V(P) \cup V_G(\widehat{G}_P)) \setminus \{u, v\}$.

The preceding argument established that there is a component of $G - \{u, v\}$ containing at least three vertices and no terminals, which forms a reducible structure of the first kind and completes the proof. ⌟

We can now complete the proof of Lemma 3.4.21 by combining the claims above. Recall from the beginning of the proof that $\widehat{G}_T$ is a tree containing at most six important vertices, such that each vertex of $\widehat{f(T)}$ and each vertex whose degree in $\widehat{G}_T$ is unequal to two is important. Removing the important vertices from $\widehat{G}_T$ splits it into at most five paths $\widehat{P}_1, \ldots, \widehat{P}_\ell$ of non-important vertices. If any of these paths has more than 25 vertices, we find a reducible structure via Claim 3.4.24. Assume that this is not the case; then $\widehat{G}_T$ consists of at most $5 \cdot 25$ non-important and 6 important vertices. If there exists $\widehat{f} \in \widehat{G}_T$ such that face f contains more than 16 vertices, we find a reducible structure via Claim 3.4.25. If not, then the faces represented by $\widehat{G}_T$ span at most $(5 \cdot 25 + 6) \cdot 16$ edges. All remaining vertices of G lie on faces whose duals are not contained in $\widehat{G}_T$, and therefore each such vertex lies on a face f for which $\widehat{f}$ belongs to some tree $\widehat{R}$ of $\widehat{G} - V(\widehat{G}_T)$. When $\widehat{e}$ is the edge connecting $\widehat{R}$ to $\widehat{G}_T$ then e lies on a face represented by $\widehat{G}_T$. If for any such tree $\widehat{R}$ the number of vertices $|\widehat{R} \setminus V(e)|$ which are not already accounted for is larger than two, then Claim 3.4.23 yields a reducible structure. If not, then since the number of attachment edges is bounded by $(5 \cdot 25 + 6) \cdot 16$, while each attached tree contributes at most two additional vertices, the total number of vertices in G is bounded by $(5 \cdot 25 + 6) \cdot 16 + 2 \cdot (5 \cdot 25 + 6) \cdot 16 = 3(5 \cdot 25 + 6) \cdot 16 = 6288$. Hence any biconnected outerplanar graph with more than this number of vertices contains a reducible structure. The proof above easily turns into a polynomial-time algorithm to find such a structure. □

3.5 Wrapping up

Finally, we combine the decomposition from Lemma 3.3.26 with the rules that reduce protrusions. If A is sufficiently large, $G\langle A\rangle$ is outerplanar, and $|N_G(A)| \leq 4$, we explicitly detect a reducible structure within $G\langle A\rangle$. First, we use Reduction Rule 3.4 to reduce the number of biconnected components in $G\langle A\rangle$. Next, we apply Lemma 3.4.21 to a biconnected component B of $G\langle A\rangle$ and the set $T = \partial_G(B)$. It provides us with one of several reducible structures which match our reduction rules. It is crucial that when C is a subgraph of B and $C \cap \partial_G(B) = \emptyset$, then the neighborhood of C in both B and G is the same.

We remark that the following lemma can be turned into an iterative procedure that maintains the decomposition from Lemma 3.3.26 without the need to recompute the set L. However, we state it in the simplest form as our analysis does not keep track of the polynomial in the running time.

Lemma 3.5.1. *Consider a graph G and a vertex set $A \subseteq V(G)$, such that $|A| > 25 \cdot 6288$, $|N_G(A)| \leq 4$, $G[A]$ is connected, and $G\langle A\rangle$ is outerplanar. There is a polynomial-time algorithm that, given G and A satisfying the conditions above, outputs a proper minor G' of G, so that $\mathsf{opd}(G') = \mathsf{opd}(G)$.*

Proof. Within this proof, we say that replacing graph G with G' is safe when $\mathsf{opd}(G') = \mathsf{opd}(G)$. With G and A as specified, we can execute the algorithm from Lemma 3.4.4. Suppose that it outputs a vertex set $C \subseteq S$ to which either Reduction Rule 3.2 or Reduction Rule 3.4 applies and shrinks the graph. They are safe due to Observation 3.2.3 and Lemma 3.4.3. In this case we can terminate the algorithm.

Otherwise Lemma 3.4.4 provides us with a block-cut tree of $G\langle A\rangle$ with at most 25 biconnected components, where each such biconnected component B satisfies $|\partial_G(B)| \leq 4$. By a counting argument we can choose one biconnected component B with more than 6288 vertices. We execute the algorithm from Lemma 3.4.21 for the graph B and the set $T = \partial_G(B)$. Note that B is a subgraph of $G\langle A\rangle$, which is outerplanar by the assumption. Depending on the type of returned structure, we select an appropriate reduction rule.

- Case 1. We find two (possibly adjacent) vertices u, v such that there is a component C of $B - \{u, v\}$ that does not contain any vertex of $\partial_G(B)$, for which $|V(C)| > 2$. Clearly $N_G(C) \subseteq \{u, v\}$ and $G\langle C\rangle$ is outerplanar as a subgraph of B. If $uv \in E(G)$, then we apply Reduction Rule 3.5 to contract C into a single vertex. This reduction is safe due to Lemma 3.4.5.

  If $uv \notin E(G)$, we apply Reduction Rule 3.4. The criterion in the statement of the rule can be easily checked in polynomial time. By Lemma 3.4.1, the replacement operation is equivalent to a series of edge contractions. The safeness follows from Lemma 3.4.3. Since $|V(C)| > 2$, we always perform at least one contraction.

- Case 2. We obtain a matching $e_1, \ldots, e_7$ in B, such that there is a single connected component C of $B - (V(e_1) \cup V(e_7))$ which contains $\{e_2, \ldots, e_6\}$ but no vertex from $\partial_G(B)$, such that $N_B(C) = V(e_1) \cup V(e_7)$, the graph $B\langle C\rangle$ is biconnected, and $e_1, \ldots, e_7$ is an order-respecting matching in $B\langle C\rangle$. Since $C \cap \partial_G(B) = \emptyset$, we get that $N_G(C) = N_B(C) = V(e_1) \cup V(e_7)$. This allows us to apply Reduction Rule 3.7 and remove the edge e_4. This is safe thanks to Lemma 3.4.8.

- Case 3. We find a vertex $x \in V(B)$ and five vertices $v_1, \ldots, v_5 \in N_B(x)$ that lie, in order of increasing index, on an induced path P in $B - x$ from v_1 to v_5, such that $N_B(x) \cap V(P) = \{v_1, \ldots, v_5\}$, and such that the connected component C of $B - \{v_1, v_5, x\}$ which contains $P - \{v_1, v_5\}$ contains no vertex from $\partial_G(B)$. This means that C is also the connected component of $G - \{v_1, v_5, x\}$ which contains $P - \{v_1, v_5\}$. Furthermore, the path P is also induced in the graph $G - x$ because B is an induced subgraph of G. The graph $G\langle C\rangle$ is outerplanar as a subgraph of B and thus Reduction Rule 3.6 applies so we can remove the edge xv_3. This operation is safe due to Lemma 3.3.19.

In each case we are able to perform a contraction or removal operation while preserving the outerplanar deletion number of the graph. The claim follows. □

It is important that the graph is guaranteed to shrink at each step, so after polynomially many invocations of Lemma 3.5.1 we must arrive at an irreducible instance. We are now ready to prove the main theorem with the final bound on the size of compressed graph $2 \cdot (25 \cdot 6288 + 5) \cdot f_5(c, f_3(c)) \cdot (k_2 + 3)^4$ (see Lemmas 3.3.16 and 3.3.26), where $c = 40$. Recall that instances (G, k) and (G', k') are equivalent if $\mathsf{opd}(G) \leq k \Leftrightarrow \mathsf{opd}(G') \leq k'$.

Theorem (3.1.1, restated)**.** *The* Outerplanar Deletion *problem admits a polynomial-time kernelization algorithm that, given an instance (G, k), outputs an equivalent instance (G', k'), such that $k' \leq k$, graph G' is a minor of G, and G' has $\mathcal{O}(k^4)$ vertices and edges. Furthermore, if $\mathsf{opd}(G) \leq k$, then $\mathsf{opd}(G') = \mathsf{opd}(G) - (k - k')$.*

Proof of Theorem 3.1.1. We use Lemma 3.3.16 to either conclude that $\mathsf{opd}(G) > k$ or to find an equivalent instance (G_1, k_1), where G_1 is a subgraph of G and $k_1 \leq k$. In the first case, either K_4 or $K_{2,3}$ must be a minor of G. We check it in polynomial time and depending on the result we output an instance $(H, 0)$, where $H = K_4$ or $H = K_{2,3}$, which is then equivalent to (G, k). Otherwise we are guaranteed that $\mathsf{opd}(G) \leq k$ implies $\mathsf{opd}(G_1) = \mathsf{opd}(G) - (k - k_1)$ and we also obtain a (k_1, c, d)-outerplanar decomposition of G_1, where $d = f_3(c)$, which we supply to the algorithm from Lemma 3.3.26. In the first scenario, it applies Reduction Rule 3.3 or Reduction Rule 3.2 to the instance (G_1, k_1). These rules may remove vertices or edges, but do not change the value of the parameter and transform

the instance into an equivalent one due to Lemma 3.3.20 and Observation 3.2.3. Moreover, when $\mathsf{opd}(G_1) \leq k_1$, then the outerplanar deletion number also stays intact. If this is the case, we shrink the graph and rerun the algorithm from scratch on the smaller graph, starting from recomputing the outerplanar decomposition.

Since the graph shrinks at each step, at some point the routine described in Lemma 3.3.26 terminates with the set L as the outcome. Let (G_2, k_2) be the instance equivalent to (G, k) obtained so far. Then for the returned set $L \subseteq V(G_2)$ we have that $f_5(c,d) \cdot (k_2+3)^4$ upper bounds each of: $|L|$, $|E_G(L,L)|$, and the number of connected components in $G_2 - L$. Furthermore, for each connected component A of $G_2 - L$ it holds that $G_2\langle A\rangle$ is outerplanar and $|N_{G_2}(A)| \leq 4$. If any of these components has more than $25 \cdot 6288$ vertices, then Lemma 3.5.1 applies and produces an equivalent instance (G_3, k_2), where G_3 is a proper minor of G_2. This reduction does not affect the parameter nor the outerplanar deletion number. We can thus again rerun the algorithm from scratch on the smaller graph.

Otherwise, each component of $G_2 - L$ has bounded size and (G_2, k_2) is the outcome of the kernelization algorithm. We check that G_2 has at most $(25 \cdot 6288 + 1) \cdot f_5(c,d) \cdot (k_2+3)^4$ vertices. The number of edges in G_2 can be upper bounded by $|E_G(L,L)|$ plus the sum of edges in each $G\langle A\rangle$, where A is connected component of $G_2 - L$. Note that this also takes into account the edges with just one endpoint in L. We estimate $|V(G\langle A\rangle)| \leq 25 \cdot 6288 + 4$ and by Observation 3.2.8 we have that $|E(G\langle A\rangle)| \leq 2 \cdot |V(G\langle A\rangle)|$. Therefore, $|E(G_2)|$ is at most $2 \cdot (25 \cdot 6288 + 5) \cdot f_5(c,d) \cdot (k_2+3)^4$. □

As a consequence of the theorem above, we obtain the first concrete bounds on the sizes of minor-minimal obstructions to having an outerplanar vertex deletion set of size k.

Corollary (3.1.2, restated)**.** *If G is a graph such that $\mathsf{opd}(G) > k$ but each proper minor G' of G satisfies $\mathsf{opd}(G') \leq k$, then G has $\mathcal{O}(k^4)$ vertices and edges.*

Proof of Corollary 3.1.2. Let $p\colon \mathbb{N} \to \mathbb{N}$ be a function such that for each instance (G,k) of Outerplanar Deletion there is an equivalent instance (G',k') where G' is a minor of G on at most $p(k)$ vertices and at most $p(k)$ edges. Theorem 3.1.1 provides such a function with $p(k) \in \mathcal{O}(k^4)$. In the remainder of the proof, we refer to a vertex set S whose removal makes a graph outerplanar as a *solution*, regardless of its size.

Let (G,k) be a pair satisfying the preconditions to the corollary. Note that we have $\mathsf{opd}(G) \leq k+1$ since the graph G' obtained after removing an arbitrary vertex v satisfies $\mathsf{opd}(G') \leq k$ and therefore has a solution S of size at most k, so that $S \cup \{v\}$ is a solution in G of size at most $k+1$.

Next, we argue that for each vertex $v \in V(G)$ there is a solution of size $k+1$ in G without v. If v is an isolated vertex in G then this is trivial. Otherwise, let $uv \in E(G)$ be an arbitrary edge incident on v. Since $G' := G \setminus uv$ is a proper minor of G, it has a solution S' of size k by assumption. We have $v \notin S'$: if $v \in S'$, then $G' - S = G - S$ which would imply that G has a solution of size k,

contradicting $\mathsf{opd}(G) > k$. Now observe that $S' \cup \{u\}$ is a solution in G, because the graph $G - (S' \cup \{u\})$ is a minor of $G' - S$, as removing vertex u also removes the edge uv. Hence $S' \cup \{u\}$ is a solution of size $k+1$ in G that does not contain v.

Consider the effect of applying the kernelization algorithm of Theorem 3.1.1 to the instance $(G, k+1)$, resulting in an instance (G', k'). Since $\mathsf{opd}(G) = k+1$, we have that $\mathsf{opd}(G') = \mathsf{opd}(G) - (k+1-k')$. We also know that G' is a minor of G and the number of vertices and edges in G' is at most $p(k+1)$. We are going to show that $G' = G$. Suppose otherwise and consider the series of graph modifying reductions applied by the kernelization algorithm, resulting in successive instances $(G, k+1) = (G_1, k_1), (G_2, k_2), \ldots, (G_\ell, k_\ell) = (G', k')$. We consider two cases: $k_2 < k_1$ and $k_2 = k_1$. In the first case, the only reduction rule that may decrease the value of the parameter is the one from Lemma 3.3.5. However, as $k_1 = k+1$ and for each vertex $v \in V(G)$ there is a solution of size $k+1$ in G without v, this reduction must produce a new instance with $k_2 = k_1$; a contradiction. In the second case, the graph G_2 is a proper minor of $G_1 = G$ and $\mathsf{opd}(G_2) = \mathsf{opd}(G) = k+1$ because no reduction can change the outerplanar deletion number unless it is greater than k_1 or $k_2 < k_1$. But each proper minor of G has a solution of size at most k by assumption, which again leads to a contradiction. This implies that $G' = G$ and hence the number of vertices and edges in G is at most $p(k+1) \in \mathcal{O}(k^4)$. □

3.6 Conclusion

We presented a number of elementary reduction rules for Outerplanar Deletion that can be applied in polynomial time to obtain a kernel of $\mathcal{O}(k^4)$ vertices and edges. This kernel does not use protrusion replacement and the constants hidden by the $\mathcal{O}$-notation can be derived easily. This is the first concrete kernel for Outerplanar Deletion, and a step towards more concrete kernelization bounds for Planar-$\mathcal{F}$ Deletion.

In earlier work Dell and Van Melkebeek [40, Theorem 3] have shown that there is no kernel for Outerplanar Deletion of bitsize $\mathcal{O}(k^{2-\varepsilon})$ unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. This naturally leads to the question, can these two bounds be brought closer together?

Our work exploits the fact that $K_{2,3}$-minor free graphs cannot have many disjoint paths between two vertices. Previous work [60] used a similar observation to derive a kernel for θ_c-Minor-Free Deletion. Since the appearance of the extended abstract of our work, a concrete kernelization bound was given for $\{K_4\}$-Minor-Free Deletion by Schols [118], showing that a similar approach can also work for $\mathcal{F}$-Minor-Free Deletion when $\mathcal{F}$-minor free graphs may contain many disjoint paths between two vertices. Both for $\{K_4, K_{2,3}\}$-Minor-Free Deletion and $\{K_4\}$-Minor-Free Deletion it is possible to obtain near-protrusions that behave nicely in the following sense: there is a set Z of $\mathcal{O}(1)$ vertices such that for any optimal solution S all but one of the neighbors of the near-protrusion

which are not in S, belong to Z except for at most one vertex. This can be exploited to give explicit concrete kernelizations. This leads to the question, is there something special about $\{K_4, K_{2,3}\}$ and $\{K_4\}$ that makes this possible or can these techniques also be used to obtain explicit concrete polynomial kernels for all Planar-$\mathcal{F}$ Deletion problems?

4

A Turing Kernelization Dichotomy for Finding $\mathcal{F}$-Minor Free Graphs

4.1 Introduction

In Chapter 3 we gave a kernelization for the Outerplanar Deletion problem parameterized by the solution size, often referred to as the natural parameter. Using the natural parameter is a common approach in kernelization [5, 61, 76, 90], and it often leads to meaningful results when we can expect the solution to be small. However, this method does not give any nontrivial guarantees when the solution size is known to be proportional to the total size of the input. For that reason, there is an alternative line of research [25, 36, 59, 69, 79, 80, 81, 123] that focuses on parameterizations based on a measure of nontriviality of the instance (cf. [104]). One formal way to capture nontriviality of a graph problem is to measure how many vertex-deletions are needed to reduce the input graph to a graph class in which the problem can be solved in polynomial time. Since many graph problems can be solved in polynomial time on trees and forests, the structural graph parameter *feedback vertex number* (the minimum number of vertex deletions needed to make the graph acyclic, i.e., a forest) is a relevant measure of the distance of the input to a trivially solvable one.

Previous research has shown that for the Vertex Cover problem, there is a polynomial kernel parameterized by the feedback vertex number [79]. This preprocessing algorithm guarantees that inputs which are large with respect to their feedback vertex number can be efficiently reduced. Similarly, the Feedback Vertex Set problem parameterized by the feedback vertex number (its natural parameter in this case) admits a polynomial kernel [23, 76, 121]. The Vertex Cover and Feedback Vertex Set problems are arguably the simplest of the $\mathcal{F}$-Minor-Free Deletion problems (taking $\mathcal{F} = \{K_2\}$ and $\mathcal{F} = \{K_3\}$ respectively) which leads to the following question: Do *all* $\mathcal{F}$-Minor-Free Deletion problems admit a polynomial kernel when parameterized by the feedback vertex number?

To our initial surprise, we prove that the answer to this question is *no* (under the assumption that $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$, which we tacitly assume throughout the informal discussion in this introduction). We show that some $\mathcal{F}$-Minor-Free Deletion problems parameterized by the feedback vertex number do *not* admit a polynomial kernel. The non-existence of a polynomial kernel does not necessarily imply that any size-reduction strategy will fail. *Turing* kernelization [54] is a relaxation of the traditional form of kernelization. Some problems that do *not* admit polynomial kernelizations, do admit polynomial Turing kernelizations [17, 78, 83, 95, 126]. Recall that where a kernelization algorithm is required to find a single polynomial-size problem instance whose solution can be converted into a solution of the original problem, a Turing kernelization may use the results of multiple polynomial-size problem instances to derive the solution to the original problem. A formal definition of Turing kernelization is given in Section 2.4.

Given that not all $\mathcal{F}$-Minor-Free Deletion problems parameterized by the feedback vertex number admit a polynomial kernel, one may wonder if in those cases a polynomial Turing kernel exists instead. Using the framework of Hermelin et al. [72], our argument for ruling out a traditional polynomial kernel, also rules out the existence of polynomial-size Turing kernelizations under a certain hardness assumption.

Results While it is known that the parameterization by feedback vertex number admits polynomial kernels for $\mathcal{F} = \{K_2\}$, for $\mathcal{F} = \{K_3\}$, and for any set $\mathcal{F}$ containing a planar graph[1] but no forests [61], we show there are also cases that do not admit polynomial kernels. For example, the case of $\mathcal{F}$ consisting of a single graph P_3 that forms a path on three vertices does not admit a polynomial kernel. This lower bound for $\mathcal{F} = \{P_3\}$ follows from a more general theorem that we state below.

Recall that a graph is a forest if and only if its treewidth is one [19]. Hence the feedback vertex number is exactly the minimum number of vertex deletions needed

[1] If $\mathcal{F}$ contains no forests, then any acyclic graph is $\mathcal{F}$-minor free, implying the size of an optimal solution is at most the size of a feedback vertex set. Hence the kernelization for the solution-size parameterization yields a kernel of size bounded polynomially in the feedback vertex number.

to obtain a graph of treewidth one. Our lower bound also holds for $\mathcal{F}$-Subgraph-Free Deletion, which is the related problem that asks whether there is a vertex set S of size at most k such that $G - S$ contains no graph $H \in \mathcal{F}$ as a *subgraph*. Let $\min \mathrm{tw}(\mathcal{F}) := \min_{H \in \mathcal{F}} \mathsf{tw}(H)$. We prove the following.

Theorem 4.1.1. *Let $\mathcal{F}$ be a finite set of graphs, such that each graph in $\mathcal{F}$ has a connected component on at least three vertices. Then $\mathcal{F}$-Minor-Free Deletion and $\mathcal{F}$-Subgraph-Free Deletion do not admit polynomial kernels when parameterized by the vertex-deletion distance to a graph of treewidth $\min \mathrm{tw}(\mathcal{F})$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

To see that Theorem 4.1.1 implies the claimed lower bound for $\mathcal{F} = \{P_3\}$, observe that whenever $\mathcal{F}$ contains an acyclic graph with at least one edge we have $\min \mathrm{tw}(\mathcal{F}) = 1$ and therefore the vertex-deletion distance to a graph of treewidth $\min \mathrm{tw}(\mathcal{F})$ equals the feedback vertex number. The theorem also generalizes earlier results of Cygan et al. [36, Theorem 13], who investigated the problem of *losing treewidth*. They proved that for each fixed $1 \leq \eta < \rho$, the η-Transversal problem (delete at most ℓ vertices to get a graph of treewidth at most η) does not have a polynomial kernel when parameterized by the vertex-deletion distance to treewidth ρ. Since the treewidth of a graph does not increase when taking minors, there is a finite set $\mathcal{F}_\eta$ of forbidden minors (cf. [113]) that characterize the graphs of treewidth at most η. As the members of the obstruction set for $\eta \geq 1$ are easily seen to be connected, have treewidth $\eta + 1$, and at least three vertices, the lower bound of Theorem 4.1.1 encompasses the theorem of Cygan et al. and generalizes it to arbitrary $\mathcal{F}$-Minor-Free Deletion problems.

Theorem 4.1.1 is obtained through a polynomial-parameter transformation from the cnf-sat problem parameterized by the number of variables, for which a superpolynomial kernelization lower bound is known [40, 63]. The main technical contribution in the hardness proof consists of the design of a gadget that acts as a clause checker. A certain budget of vertex deletions is available to break all $\mathcal{F}$-minors present in the gadget, and this is possible if and only if one of the neighboring vertices in a variable gadget is removed by the solution. This removal encodes that the variable is set in a way that satisfies the clause. The intricate part of the construction is to design the gadget knowing only that $\mathcal{F}$ has a graph with a connected component of at least three vertices. Here we extensively rely on the fact that minimal minor models of biconnected graphs live in biconnected subgraphs, together with the fact that the treewidth of a graph does not increase when attaching structures along cut vertices in a tree-like manner.

The reduction proving Theorem 4.1.1 also proves the non-existence of polynomial-size Turing kernelizations, unless all parameterized problems in the complexity class $\mathsf{MK}[2]$ defined by Hermelin et al. [72] have polynomial Turing kernels. (The cnf-sat problem with clauses of unbounded length, parameterized by the number of variables, is $\mathsf{MK}[2]$-complete [72, Theorem 1, cf. Theorem 10] and widely believed *not* to admit polynomial-size Turing kernels.)

Motivated by the general form of the lower bound statement in Theorem 4.1.1, we also investigate upper bounds and derive a complexity dichotomy. For any $\mathcal{F}$ that does not meet the criterion of Theorem 4.1.1, we obtain a polynomial Turing kernel.

Theorem 4.1.2. *Let $\mathcal{F}$ be a finite family of graphs, such that some $H \in \mathcal{F}$ has no connected component of three or more vertices. Then* $\mathcal{F}$-Minor-Free Deletion *and* $\mathcal{F}$-Subgraph-Free Deletion *admit polynomial Turing kernels when parameterized by the vertex-deletion distance to a graph of treewidth* $\min \mathrm{tw}(\mathcal{F})$.

The main insight in the Turing kernelization is the following. If $H \in \mathcal{F}$ has no connected component of three or more vertices, then H consists of disjoint edges and isolated vertices. If H only has isolated vertices, then $\mathcal{F}$-Minor-Free Deletion is polynomial-time solvable because the leftover graph has less than $|V(H)| \in \mathcal{O}(1)$ vertices, for which we can search by brute force. Otherwise, H is a matching of size $t \geq 1$ plus potentially some isolated vertices. The isolated vertices turn out only to make a difference if the solution $\mathcal{F}$-free graph has constant size. In the interesting case, we can focus on $H \in \mathcal{F}$ being a matching of size t. Then a graph that is $\mathcal{F}$-minor-free does not admit a matching of size t, and therefore has a vertex cover of size at most $2t$. Hence a solution to $\mathcal{F}$-Minor-Free Deletion can be extended to a vertex cover by including $\mathcal{O}(1)$ additional vertices. Using the Tutte-Berge formula, we can make the relation between $\mathcal{F}$-Minor-Free Deletion and the vertex cover precise, and use it to reduce an instance of $\mathcal{F}$-Minor-Free Deletion parameterized by deletion distance to $\min \mathrm{tw}(\mathcal{F})$, to the logical OR of a polynomial number of instances of Vertex Cover parameterized by deletion distance to $\min \mathrm{tw}(\mathcal{F})$. If $\mathcal{F}$ has a graph with no component of size at least three, then $\min \mathrm{tw}(\mathcal{F}) = 1$, implying that the parameter is the feedback vertex set size. This allows us to use the polynomial kernel for Vertex Cover parameterized by feedback vertex set on each generated instance. We query the resulting instances of size $k^{\mathcal{O}(1)}$ to the oracle to find the answer.

Organization We present preliminaries on graphs and kernelization in Section 4.2. Section 4.3 develops the lower bounds on (Turing) kernelization when all graphs in $\mathcal{F}$ have a connected component with at least three vertices. In Section 4.4 we show that in all other cases, a polynomial Turing kernelization exists.

4.2 Preliminaries

If a graph H is a minor of a graph G we denote this as $H \preceq G$. We say a graph H is a *component-wise minor* of a graph G, denoted as $H \precsim G$, when every connected component of H is a minor of G.

Observation 4.2.1. *For graphs H and G, if $H \preceq G$ and $G \preceq H$ then H and G are isomorphic.*

Since the treewidth of a graph H is the maximum treewidth of its connected components, we have that if $H \preceq\!\!\sim G$ for some graph G, then each connected component H' of H is a minor of G, hence $\mathsf{tw}(H') \leq \mathsf{tw}(G)$. This leads to the following observation:

Observation 4.2.2. *For graphs H and G, if $H \preceq\!\!\sim G$ then $\mathsf{tw}(H) \leq \mathsf{tw}(G)$.*

Definition 4.2.3. Let $\mathcal{F}$ be a family of graphs and let $G \in \mathcal{F}$. For every relation $\trianglelefteq \in \{\preceq, \preceq\!\!\sim\}$ we define minimal and maximal elements as follows:

- G is said to be $\trianglelefteq$-*minimal* in $\mathcal{F}$ when for all graphs $H \in \mathcal{F}$ we have $H \trianglelefteq G \Rightarrow G \trianglelefteq H$.
- G is said to be $\trianglelefteq$-*maximal* in $\mathcal{F}$ when for all graphs $H \in \mathcal{F}$ we have $G \trianglelefteq H \Rightarrow H \trianglelefteq G$.

Definition 4.2.4. We call a connected component C of a graph G a $\preceq$-*maximal component* of G when C is $\preceq$-maximal in the set of graphs that form the connected components of G.

For *type* $\in \{\text{minor}, \text{subgraph}\}$ and a finite family of graphs $\mathcal{F}$, we define:

$\mathcal{F}$-*type*-Free Deletion
Input: A graph G and an integer ℓ.
Parameter: Vertex-deletion distance of G to a graph of treewidth $\min \mathrm{tw}(\mathcal{F})$.
Question: Is there a set $X \subseteq V(G)$ of at most ℓ vertices such that $G - X$ does not contain any $H \in \mathcal{F}$ as a *type*?

For any integer α, a graph G is called α-*robust* when $|V(G)| \geq \alpha$ and no vertex $v \in V(G)$ exists such that $G - v$ contains a connected component with strictly less than $\alpha - 1$ vertices.

Proposition 4.2.5. *Any graph G has a unique maximal α-robust subgraph. Any α-robust subgraph of G is a subgraph of the maximal α-robust subgraph of G.*

Proof. The proposition follows straightforwardly from the fact that if $G[A]$ and $G[B]$ are α-robust, then so is $G[A \cup B]$. We now prove this fact.

Consider two vertex sets $A, B \subseteq V(G)$, such that $G[A]$ and $G[B]$ are α-robust. We show that $G[A \cup B]$ is α-robust. Since $G[A]$ is α-robust we have $|A| \geq \alpha$ so then $|A \cup B| \geq \alpha$. Suppose for contradiction that there exists a vertex $v \in A \cup B$ such that $G[A \cup B] - v$ contains a connected component of size smaller than $\alpha - 1$. Let C be the vertices of this connected component. We know C contains vertices of at least one of A and B. Assume, without loss of generality, $A \cap C \neq \emptyset$. Then $G[A \cap C]$ is a connected component of size less than $\alpha - 1$ in $G[A] - v$. If $v \in A$ this directly contradicts α-robustness of $G[A]$, so assume $v \notin A$. Now $G[A]$ contains a connected component with less than $\alpha - 1$ vertices. Since $|C| < \alpha \leq |A|$ there exists a vertex $u \in A \setminus C$, so then $G[A] - u$ contains a connected component with less than $\alpha - 1$ vertices, which contradicts α-robustness of $G[A]$. □

Recall that a biconnected component of a graph is a maximal subgraph that is biconnected. We define a *leaf-block* of a graph G as a biconnected component of G that contains at most one vertex v that is a cut vertex in G. The size of a leaf-block H is $|V(H)|$. The size of a smallest leaf-block of a graph G is denoted as $\lambda(G)$. Observe that G is α-robust if and only if $\lambda(G) \geq \alpha$. For any graph G and integer α, let α-prune(G) denote the unique maximal α-robust subgraph of G, which may be empty. Note that α-prune(G) can be obtained from G by repeatedly removing interior vertices of leaf blocks of size less than α.

4.3 Lower bound

In this section we consider the case where all graphs in $\mathcal{F}$ contain a connected component of at least three vertices and give a polynomial-parameter transformation from CNF-SAT parameterized by the number of variables. In order to construct a clause gadget G for a clauses with more than two literals, our construction relies on the presence of a connected component $H_\uparrow$ with at least three vertices in a $\precsim$-minimal graph in $\mathcal{F}$. Such a graph only exists when *all* graphs in $\mathcal{F}$ have a connected component on at least 3 vertices. Intuitively, the reason we need at least three vertices is as follows. The gadget for a clause C_i is constructed based on the sequence of its literals $\ell_1, \ldots, \ell_{|C_i|}$. For each literal ℓ_i, there is a corresponding part of the gadget which checks three things: whether a literal before ℓ_i is satisfied, whether ℓ_i itself is satisfied, and whether a literal after ℓ_i is satisfied. To implement this check, we need that H has at least three vertices.

4.3.1 Properties of biconnected and robust subgraphs

Our construction exploits the way in which biconnected components of H and the clause gadget G restrict the options for an H-model to exist in G. We therefore first derive some relevant properties.

Proposition 4.3.1. *If H is an α-robust graph and φ is a minimal H-model in a graph G, then $G[\varphi(H)]$ is α-robust.*

Proof. Since H is α-robust we have $|V(H)| \geq \alpha$. So $|\varphi(H)| \geq |V(H)| \geq \alpha$ and it remains to verify that $G[\varphi(H)] - v$ does not have any connected components smaller than $\alpha - 1$ for any v. Take an arbitrary vertex $v \in \varphi(H)$ and let $u \in V(H)$ be such that $v \in \varphi(u)$. Since $H - u$ does not have connected components smaller than $\alpha - 1$, the graph $G[\varphi(H)] - \varphi(u)$ cannot have connected components smaller than $\alpha - 1$. Consider a spanning tree of $G[\varphi(u)]$. Each leaf of this spanning tree must be connected to a vertex in a different branch set, otherwise φ is not minimal. We know every connected component in $G[\varphi(u)] - v$ contains at least one leaf of this spanning tree, hence every connected component of $G[\varphi(u)] - v$ is connected to $G[\varphi(H)] - \varphi(u)$. So $G[\varphi(H)] - v$ does not contain a connected component smaller than $\alpha - 1$. Since v was arbitrary, the graph $G[\varphi(H)]$ is α-robust. □

Proposition 4.3.2. *If φ is an H-model in G and B is a biconnected component of H, then $G[\varphi(B)]$ contains a biconnected subgraph on at least $|B|$ vertices.*

Proof. Since $G[\varphi(B)]$ clearly contains B as a minor, there is a minimal B-model φ' in G with $\varphi'(B) \subseteq \varphi(B)$, so that $G[\varphi'(B)]$ is a subgraph of $G[\varphi(B)]$. It suffices to show that $G[\varphi'(B)]$ contains a biconnected component on at least $|V(B)|$ vertices. Since B is biconnected, it is $|V(B)|$-robust so by Proposition 4.3.1 we know $G[\varphi'(B)]$ is $|V(B)|$-robust. Hence $G[\varphi'(B)]$ contains a biconnected component on at least $|V(B)|$ vertices. □

Observation 4.3.3. *For any graph H, which may be the null graph, and integers $\alpha \geq \beta$ we have α-prune(β-prune(H)) = α-prune(H).*

Proposition 4.3.4. *For graphs H and G we have $H \preceq G \Rightarrow \alpha$-prune($H$) $\preceq \alpha$-prune(G) for any integer α.*

Proof. Clearly α-prune(H) $\preceq H$ so since $H \preceq G$ we have α-prune(H) $\preceq G$. For simplicity let $H' := \alpha$-prune(H). Let φ be a minimal H'-model in G and observe $H' \preceq G[\varphi(H')]$. From Proposition 4.3.1 we know $G[\varphi(H')]$ is α-robust. Since $G[\varphi(H')]$ is an α-robust subgraph of G it is also a subgraph of α-prune(G) by Proposition 4.2.5. Hence α-prune(H) $\preceq G[\varphi(H')] \preceq \alpha$-prune($G$). □

Proposition 4.3.5. *For any $\trianglelefteq \in \{\preceq, \precsim\}$, two integers $\alpha \geq \beta$, and graphs H and G we have that $H \trianglelefteq G \Rightarrow \alpha$-prune($H$) $\trianglelefteq \beta$-prune(G).*

Proof. Suppose $H \preceq G$, then

$$\begin{aligned} \alpha\text{-prune}(H) &\preceq \alpha\text{-prune}(G) && \text{by Proposition 4.3.4} \\ &= \alpha\text{-prune}(\beta\text{-prune}(G)) && \text{by Observation 4.3.3} \\ &\preceq \beta\text{-prune}(G). \end{aligned}$$

Alternatively, suppose $H \precsim G$. If H' is a connected component of α-prune(H), then there exists a connected component H'' of H such that $H' \preceq H''$. Since $H \precsim G$ we have $H'' \preceq G$ so then $H' \preceq G$. As shown above, this implies α-prune(H') $\preceq \beta$-prune(G). Note that α-prune(H') $= H'$ since H' is a connected component of α-prune(H). It follows that α-prune(H) $\precsim \beta$-prune(G). □

4.3.2 Clause gadget construction

We proceed to construct a clause gadget to be used in the polynomial-parameter transformation from CNF-SAT.

Lemma 4.3.6. *For any connected graph H with at least three vertices there exists a polynomial-time algorithm that, given an integer $n \geq 1$, outputs a graph G and a vertex set $S \subseteq V(G)$ of size n such that all of the following are true:*

1. $\mathsf{tw}(G) \leq \mathsf{tw}(H)$,

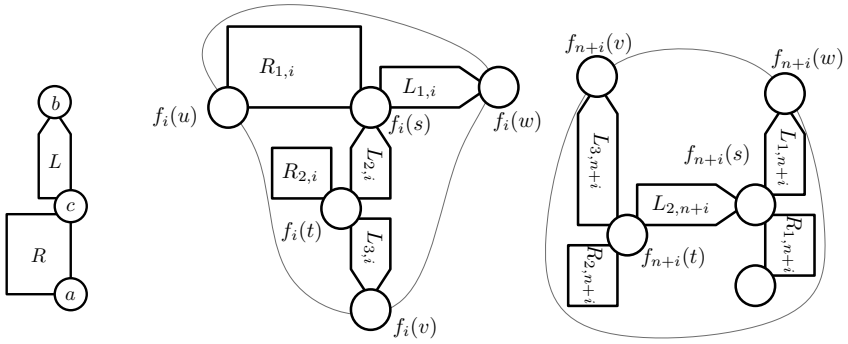
2. *G contains a packing of $3n-1$ vertex-disjoint H-subgraphs,*
3. *$G-S$ contains a packing of $3n-2$ vertex-disjoint H-subgraphs, and*
4. *$\forall v \in S$ there exists $X \subseteq V(G)$ of size $3n-1$ s.t. all of the following are true:*
   (a) *$v \in X$,*
   (b) *$G-X$ is H-minor free,*
   (c) *$\lambda(H)$-prune$(G-X) \precsim H$, and*
   (d) *for all connected components G_c of $G-X$ that contain a vertex of S we have $|V(G_c)| < \lambda(H)$ and G_c contains exactly one vertex of S.*

Proof. Before describing the construction of G and S, we define a few subgraphs and vertices. Let L be a smallest leaf-block of H. Let R be the graph obtained from H by removing all vertices of L that are not cut vertices in H. Note that when H is biconnected, $L = H$ and R is the null graph. We distinguish three distinct vertices a, b, c in H. Vertices c and b are both part of L, where c is the cut vertex (if there is one) and b is any other vertex in L. Finally vertex a is any vertex in H that is not c or b. See Figure 4.1a. In the construction of G we will combine copies of H such that a, b, and c form cut vertices in G and are part of two different H-subgraphs. Intuitively this choice of b and c ensures that removing either one from a copy of H in G means no vertex from the L-subgraph of this copy of H can be used in a minimal H-model in G. In the remainder of this proof we use $f_{K \to K'} \colon V(K) \to V(K')$ for isomorphic graphs K and K' to denote a fixed isomorphism.

Construction Take two copies of H, call them H_1 and H_2. Let R_1 and L_1 denote the subgraphs of H_1 related to R and L, respectively, by the isomorphism between H and H_1. Similarly let R_2 and L_2 denote the subgraphs of H_2. Take a copy of L which we call L_3. Let M be the graph obtained from the disjoint union of H_1, H_2, and L_3 by identifying the pair $f_{H \to H_1}(c)$ and $f_{H \to H_2}(b)$ into a single vertex s, and identifying the pair $f_{H \to H_2}(c)$ and $f_{L \to L_3}(c)$ into a single vertex t. We label $f_{H \to H_1}(a)$, $f_{H \to H_1}(b)$, and $f_{L \to L_3}(b)$ as u, w, and v respectively.

This construction is motivated by the fact that the graphs $M - \{v, s\}$, $M - \{u, t\}$, and $M - \{w, t\}$ are all H-minor free, which we will exploit in the formal correctness argument later. We will connect copies of M to each other via the vertices u, v, and w such that, although two vertices need to be removed in every copy of M, one such vertex can always be in two copies of M at the same time.

Now take $2n-1$ copies of M, call them $M_1, \ldots, M_{2n-1}$. For readability we denote $f_{M \to M_i}$ as f_i for all $1 \leq i \leq 2n-1$. For all $1 \leq i < n$ we identify $f_i(w)$ and $f_{n+i}(v)$, and we identify $f_{n+i}(w)$ and $f_{i+1}(u)$. Let this graph be G, and let S be the set of vertices $f_i(v)$ for all $1 \leq i \leq n$. Let $H_{1,i}$, $H_{2,i}$, $R_{1,i}$, $R_{2,i}$, $L_{1,i}$, $L_{2,i}$, and $L_{3,i}$ denote the subgraphs in M_i that correspond to the subgraphs H_1, H_2, R_1, R_2, L_1, L_2, and L_3 in M. See Figures 4.1b and 4.1c.



(a) Graph H **(b)** Graph M_i for $1 \leq i \leq n$ **(c)** Graph M_{n+i} for $1 \leq i < n$

Figure 4.1 We show the situation where a is contained in R. Note that a can always be chosen such that it is contained in R when H is not biconnected. Note that the graphs in Figures 4.1b and 4.1c are isomorphic but drawn differently.

This concludes the description of graph G and set S, see Figure 4.2 for an illustration. In Figure 4.5 the subgraphs W_1, W_2, and W_3 form a concrete example of G with the choice $H = P_3$ and $n = 4$, $n = 3$, and $n = 3$ respectively. It is easily seen that G and S can be constructed in polynomial time.

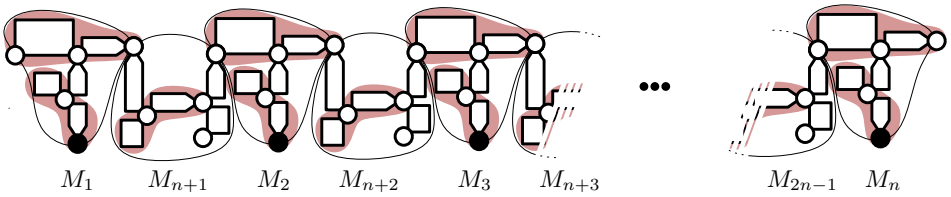
Correctness It remains to verify that all conditions of the lemma statement are met.

Condition 1: Since we connected copies of L and R in a treelike fashion along cut vertices, we did not introduce any new biconnected components. The treewidth of a graph is equal to the maximum treewidth over all its biconnected components so we know that $\mathsf{tw}(G) \leq \max\{\mathsf{tw}(R), \mathsf{tw}(L)\} = \mathsf{tw}(H)$.

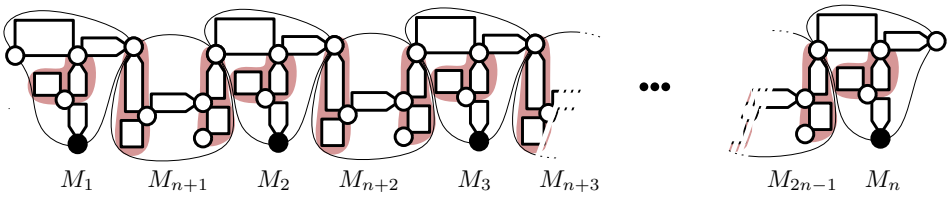
Condition 2: For each $1 \leq i \leq n$ we can distinguish two H-subgraphs in M_i, namely $H_{1,i}$ and $L_{3,i} \cup R_{2,i}$. This gives us $2n$ H-subgraphs in G. Note that since all $M_1, \ldots, M_n$ are vertex-disjoint, these $2n$ H-subgraphs are also vertex-disjoint in G. For each $n < i \leq 2n - 1$ we distinguish one H-subgraph, namely $H_{2,i}$. Note that since $H_{2,i}$ is vertex-disjoint from all $M_1, \ldots, M_{i-1}$, $M_{i+1}, \ldots, M_{2n-1}$ we have a total of $2n + n - 1 = 3n - 1$ vertex-disjoint H-subgraphs in G. This packing is shown in Figure 4.2a.

Condition 3: Alternatively, for each $1 \leq i \leq n$ we can distinguish one H-subgraph in M_i, namely $H_{2,i}$. For each $n < i \leq 2n - 1$ we distinguish two H-subgraphs in M_i, namely $H_{1,i}$ and $L_{3,i} \cup R_{2,i}$. Again these H-subgraphs are vertex-disjoint, and since they also do not contain any vertices of S, they form a packing of $n + 2(n - 1) = 3n - 2$ vertex-disjoint H-subgraphs in $G - S$. See Figure 4.2b.

Condition 4: Finally we prove that for all $v \in S$ there exists a set $X \subseteq V(G)$ of size $3n-1$ such that the four parts listed in Condition 4 are true. For this purpose



(a) A packing of $3n-1$ vertex-disjoint H-subgraphs in G



(b) A packing of $3n-2$ vertex-disjoint H-subgraphs in $G-S$

Figure 4.2 Two packings of vertex-disjoint H-subgraphs in G and $G-S$. Vertices in S are marked black.

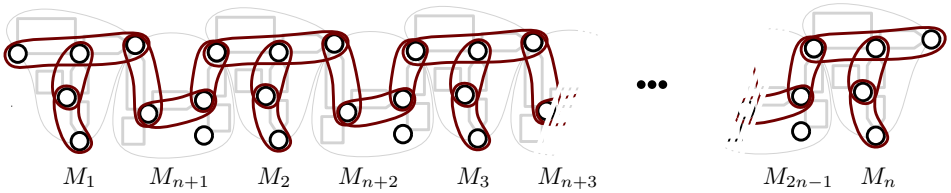


Figure 4.3 Vertex sets in $\mathcal{Q}$ are encircled.

we first identify a family $\mathcal{Q}$ of vertex sets such that any H-model in G spans at least one vertex set in $\mathcal{Q}$. Let $\mathcal{Q}$ be defined as follows: (see Figure 4.3)

$$\mathcal{Q} = \{\{f_i(v), f_i(t)\} \mid 1 \leq i \leq 2n-1\} \cup \{\{f_i(t), f_i(s)\} \mid 1 \leq i \leq 2n-1\}$$
$$\cup \{\{f_i(s), f_i(w)\} \mid n+1 \leq i \leq 2n-1\} \cup \{\{f_i(u), f_i(s), f_i(w)\} \mid 1 \leq i \leq n\}.$$

Claim 4.3.7. If φ is an H-model in G, then $\varphi(H) \supseteq Q$ for some $Q \in \mathcal{Q}$.

Proof. Let φ be an arbitrary H-model in G. We know from Proposition 4.3.2 that $G[\varphi(L)]$ contains a biconnected subgraph on at least $|L|$ vertices. Let B be such a biconnected subgraph in G. Subgraph B must be fully contained in a biconnected component of G. Such a biconnected component must contain at least $|B| \geq |L|$ vertices. We make a case distinction over all biconnected components in G with size at least $|L|$, and prove that if B is contained in them, then $\varphi(H) \supseteq Q$ for some $Q \in \mathcal{Q}$.

- $L_{2,i}$ for any $1 \leq i \leq 2n-1$: We know $|L_{2,i}| = |L|$ so then $B = L$, hence $f_i(t), f_i(s) \in \varphi(L)$.
- $L_{3,i}$ for any $1 \leq i \leq 2n-1$: We know $|L_{3,i}| = |L|$ so then $B = L$, hence $f_i(v), f_i(t) \in \varphi(L)$.
- $L_{1,i}$ for any $n+1 \leq i \leq 2n-1$: We know $|L_{1,i}| = |L|$ so then $B = L$, hence $f_i(s), f_i(w) \in \varphi(L)$.
- $L_{1,i}$ for any $1 \leq i \leq n$: We know $|L_{1,i}| = |L|$ so $B = L$, hence $f_i(s), f_i(w) \in \varphi(L)$. If $f_i(t) \in \varphi(H)$ or $f_i(u) \in \varphi(H)$ then clearly $\mathcal{Q} \ni \{f_i(t), f_i(s)\} \subseteq \varphi(H)$ or $\mathcal{Q} \ni \{f_i(u), f_i(s), f_i(w)\} \subseteq \varphi(H)$. If $f_{n+i}(t) \in \varphi(H)$ then $\mathcal{Q} \ni \{f_{n+i}(v), f_{n+i}(t)\} \subseteq \varphi(H)$, since $f_i(w) = f_{n+i}(v)$. Suppose $\varphi(H)$ does not contain $f_i(t)$, $f_i(u)$, or $f_{n+i}(t)$, then φ must be an H-model in the graph $G' := (H_{1,i} - f_i(u)) \cup (L_{2,i} - f_i(t)) \cup (L_{3,n+i} - f_i(t))$, so $H \preceq G'$. By Proposition 4.3.4 we know that $|L|$-prune$(H) \preceq |L|$-prune(G'). Clearly we have $|L|$-prune$(H) = H$. The graph G' contains at least two leaf blocks that are smaller than $|L|$, namely $L_{2,i} - f_i(t)$ and $L_{3,n+i} - f_i(t)$, so $|L|$-prune(G') is a subgraph of $H_{1,i} - f_i(u)$. But then $|V(|L|\text{-prune}(G'))| < |V(H)|$ so $|L|$-prune(G') cannot contain an H-model. Contradiction.
- There can be biconnected components of size at least $|L|$ in $R_{2,i}$ for any $1 \leq i \leq 2n-1$. Suppose $f_i(t) \notin \varphi(H)$, then φ must be an H-model in the graph $R_{2,i} - f_i(t)$. Clearly this is not possible since $|V(R_{2,i} - f_i(t))| < |V(H)|$, so $f_i(t) \in \varphi(H)$. If $f_i(v) \in \varphi(H)$ or $f_i(s) \in \varphi(H)$ then $\mathcal{Q} \ni \{f_i(v), f_i(t)\} \subseteq \varphi(H)$ or $\mathcal{Q} \ni \{f_i(t), f_i(s)\} \subseteq \varphi(H)$. Suppose $\varphi(H)$ does not contain $f_i(v)$ or $f_i(s)$, then φ must be an H-model in the graph $G' := R_{2,i} \cup (L_{3,i} - f_i(v)) \cup (L_{2,i} - f_i(s))$, so $H \preceq G'$. By Proposition 4.3.4 we know that $|L|$-prune$(H) \preceq |L|$-prune(G'), so $H \preceq |L|$-prune$(G') = R_{2,i}$. This is a contradiction since $R_{2,i}$ cannot contain H as a minor.
- There can be biconnected components of size at least $|L|$ in $R_{1,i}$ for any $n+1 \leq i \leq 2n-1$. Suppose $f_i(s) \notin \varphi(H)$, then φ must be an H-model in the graph $R_{1,i} - f_i(s)$. As before this is not possible since $|V(R_{1,i} - f_i(s))| < |V(H)|$, so $f_i(s) \in \varphi(H)$. If $f_i(t) \in \varphi(H)$ or $f_i(w) \in \varphi(H)$ then $\mathcal{Q} \ni \{f_i(t), f_i(s)\} \subseteq \varphi(H)$ or $\mathcal{Q} \ni \{f_i(s), f_i(w)\} \subseteq \varphi(H)$. Suppose $\varphi(H)$ does not contain $f_i(t)$ or $f_i(w)$, then φ must be an H-model in the graph $G' := R_{1,i} \cup (L_{2,i} - f_i(t)) \cup (L_{1,i} - f_i(w))$, so $H \preceq G'$. As before, by Proposition 4.3.4 it follows that $H \preceq R_{1,i}$ which is a contradiction.
- There can be biconnected components of size at least $|L|$ in $R_{1,i}$ for any $1 < i < n$. Suppose $f_{n+i-1}(s) \in \varphi(H)$ then $f_i(u) = f_{n+i-1}(w) \in \varphi(H)$ since any path in G connecting $f_{n+i-1}(s)$ to any vertex in $R_{1,i}$ includes $f_i(u)$. So $\mathcal{Q} \ni \{f_{n+i-1}(s), f_{n+i-1}(w)\} \subseteq \varphi(H)$. Similarly if $f_i(t) \in \varphi(H)$ then $\mathcal{Q} \ni \{f_i(t), f_i(s)\} \subseteq \varphi(H)$ and if $f_{n+i}(t) \in \varphi(G)$ then $\mathcal{Q} \ni \{f_{n+i}(v), f_{n+i}(t)\} \subseteq \varphi(H)$. Suppose $\varphi(H)$ does not contain $f_{n+i-1}(s)$, $f_i(t)$, or $f_{n+i}(t)$, then φ

must be an H-model in the graph $G' := H_{1,i} \cup (L_{1,n+i-1} - f_{n+i-1}(s)) \cup (L_{2,i} - f_i(t)) \cup (L_{3,n+i} - f_{n+i}(t))$. If $\mathcal{Q} \ni \{f_i(u), f_i(s), f_i(w)\} \subseteq \varphi(H)$, then the claim holds, so suppose $\{f_i(u), f_i(s), f_i(w)\} \not\subseteq \varphi(H)$, then for some $p \in \{f_i(u), f_i(s), f_i(w)\}$ we have that φ is an H-model in $G' - p$. Therefore $H \preceq G' - p$ and by Proposition 4.3.4 we know $|L|$-prune$(H) \preceq |L|$-prune$(G' - p)$, so $H \preceq |L|$-prune$(G' - p) = |L|$-prune$(H_{1,i} - p)$. However $|L|$-prune$(H_{1,i} - p)$ has at most $|V(H_{1,i} - p)| = |V(H)| - 1$ vertices, so it cannot contain an H-model. Contradiction.

- There can be biconnected components of size at least $|L|$ in $R_{1,1}$. As in the previous case we can assume that $\varphi(H)$ does not contain $f_1(t)$ or $f_{n+1}(t)$, so then φ must be an H-model in $G' := H_{1,1} \cup (L_{2,1} - f_i(t)) \cup (L_{3,n+1} - f_{n+1}(t))$. Like in the previous case, by Proposition 4.3.4 this results in a contradiction.

- There can be biconnected components of size at least $|L|$ in $R_{1,n}$. As above we can assume that $\varphi(H)$ does not contain $f_{2n-1}(s)$ or $f_1(t)$, so then φ must be an H-model in $G' := H_{1,1} \cup (L_{1,2n-1} - f_{2n-1}(s)) \cup (L_{2,1} - f_i(t))$. Again, by Proposition 4.3.4 this results in a contradiction.

This concludes the proof of Claim 4.3.7. ⌟

We now proceed to prove Condition 4 of the lemma statement. Let $f_j(v) \in S$ be an arbitrary vertex in S, implying $1 \leq j \leq n$, and let X be defined as:

$$\bigcup_{1 \leq i < j} \{f_i(t), f_i(w), f_{i+n}(s)\} \cup \{f_j(v), f_j(s)\} \cup \bigcup_{j < i \leq n} \{f_i(t), f_i(u), f_{i+n-1}(t)\}.$$

In Figure 4.4 the vertices in X are shown in graph G as a cross. Observe that $|X| = 3n - 1$ and $f_j(v) \in X$. Furthermore X contains at least one element from each set in $\mathcal{Q}$, hence $G - X$ is H-minor free by Claim 4.3.7. This shows Parts 4a and 4b of the lemma statement hold. We proceed to show Parts 4c and 4d.

Part 4c: Consider the graph $G' := \lambda(H)$-prune$(G - X)$. Figure 4.4 shows a supergraph of G' in red for the case that $a \in V(R)$. Every connected component in G' must contain a biconnected component with at least $\lambda(H) = |L|$ vertices. Consider all biconnected components in $G - X$ containing at least $|L|$ vertices. These can only be contained in the following subgraphs of G: $R_{2,i}$ for any $1 \leq i \leq 2n - 1$, $H_{1,i}$ for any $1 \leq i \leq n$, and $R_{1,i}$ for any $n + 1 \leq i \leq 2n - 1$. Note that any path from a vertex of one of these subgraphs to a vertex of another contains at least one vertex in X, hence any connected component in G' contains vertices of at most one of these subgraphs. Since all other biconnected components in $G - X$ have size less than $|L|$ we know that each connected component in $|L|$-prune$(G - X)$ is a subgraph of $R_{1,i}$, $R_{2,i}$ or $H_{1,i}$ for some i, hence $|L|$-prune$(G - X) \precsim H$.

Part 4d: Finally we show that all connected components in $G - X$ that contain a vertex of S have size less than $|L|$. Since we have $f_j(v) \in X$, there is no connected component in $G - X$ containing $f_j(v)$. For all $i \neq j$ we have $f_i(t) \in X$ so the connected components in $G - X$ containing a vertex from S are $L_{3,i} - f_i(t)$ for

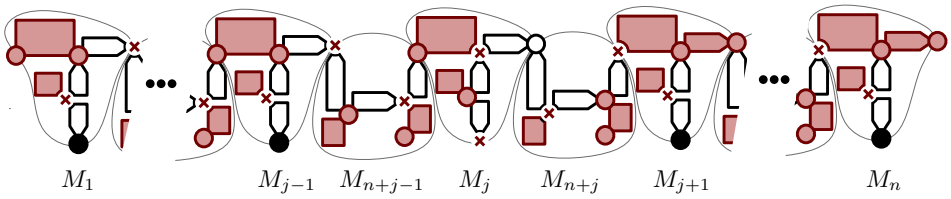


Figure 4.4 The graph $G - X$. Vertices in X that are removed from the graph are marked by a cross. Vertices in S are marked black. A supergraph of $\lambda(H)$-prune$(G - X)$ is shown in red. Note that when $|V(R)| = |V(L)|$, not all subgraphs and vertices marked red are necessarily part of $\lambda(H)$-prune$(G - X)$. Note that the subgraphs W_1, W_2, and W_3 (shaded in red) include some vertices from G_{var}.

all $1 \leq i < j$ or $j < i \leq n$. These all have size $|L| - 1$ and contain exactly one vertex of S. □

4.3.3 Reduction for connected graphs H

Using the clause gadget described in Lemma 4.3.6 we give a polynomial-parameter transformation for the case where $\mathcal{F}$ contains a single, connected graph H.

Lemma 4.3.8. *For any connected graph H with at least three vertices there exists a polynomial-time algorithm that, given a* CNF*-formula Φ with k variables, outputs a graph G and an integer ℓ such that all of the following are true:*

1. *there is a set $S \subseteq V(G)$ of at most $2k$ vertices such that $\mathsf{tw}(G-S) \leq \mathsf{tw}(H)$,*
2. *G contains ℓ vertex-disjoint H-subgraphs,*
3. *if Φ is not satisfiable then there does not exist a set $X \subseteq V(G)$ of size at most ℓ such that $G - X$ is H-subgraph free,*
4. *if Φ is satisfiable then there exists a set $X \subseteq V(G)$ of size at most ℓ such that $G - X$ is H-minor free, $\lambda(H)$-prune$(G - X) \precsim H$, and $\mathsf{tw}(G - X) \leq \mathsf{tw}(H)$.*

Proof. Let $x_1, \ldots, x_k$ denote the variables of Φ, let $C_1, \ldots, C_m$ denote the sets of literals in each clause of Φ, and let n denote the total number of occurrences of literals in Φ, i.e., $n = \sum_{1 \leq j \leq m} |C_j|$.

Construction Let $H_1, \ldots, H_k$ be copies of H. In each copy H_i we arbitrarily label one vertex v_{x_i} and another $v_{\neg x_i}$. Let G_{var} be the graph obtained from the disjoint union of $H_1, \ldots, H_k$. For each clause C_j of Φ we create a graph called W_j and vertex set $S_j \subseteq V(W_j)$ by invoking Lemma 4.3.6 with H and $|C_j|$. Let G be the

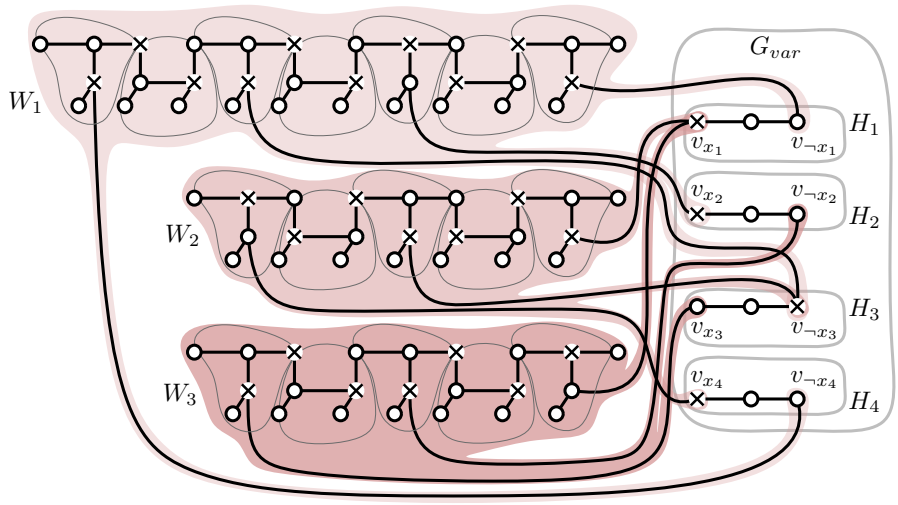


Figure 4.5 The graph G as obtained with $H = P_3$ and $\Phi = (\neg x_4 \vee \neg x_3 \vee x_2 \vee \neg x_1) \wedge (x_4 \vee \neg x_3 \vee x_1) \wedge (x_3 \vee \neg x_2 \vee x_1)$. Vertices in a solution corresponding to the satisfying assignment $x_1 = \mathsf{True}, x_2 = \mathsf{True}, x_3 = \mathsf{False}, x_4 = \mathsf{True}$ are marked with a cross.

graph obtained from the disjoint union of $W_1, \ldots, W_m$ and G_{var} where we identify the vertices in S_j with the appropriate v_{x_i} or $v_{\neg x_i}$ as follows: For each clause C_j let $s_1, \ldots, s_{|C_j|}$ be the vertices in S_j in some arbitrary order, and let $c_1, \ldots, c_{|C_j|}$ be the literals in C_j, then we identify s_i and v_{c_i} for each $1 \leq i \leq |C_j|$. Finally let $\ell = k+3n-2m$ and $S = \bigcup_{1 \leq i \leq k} \{v_{x_i}, v_{\neg x_i}\}$. Note that $S_j \subseteq S$ for all $1 \leq j \leq m$. This concludes the description of G, ℓ, and S. See Figure 4.5 for an example.

Correctness It is easy to see they can be constructed in polynomial time. We proceed to show that all conditions in the lemma statement are met.

Condition 1: Clearly $|S| = 2k$ and since every connected component in $G - S$ is a subgraph of $H_1, \ldots, H_k$ or $W_1, \ldots, W_m$, it follows from Lemma 4.3.6(1) that $\mathsf{tw}(G - S) \leq \mathsf{tw}(H)$.

Condition 2: For all $1 \leq j \leq m$ we know from Lemma 4.3.6(3) that $W_j - S$ contains a packing of $3|C_j| - 2$ H-subgraphs. Since $W_j - S$ and $W_i - S$ are vertex-disjoint for $j \neq i$ we can combine these packings to obtain a packing in $G - S$ of $\sum_{1 \leq j \leq m}(3|C_j| - 2) = 3n - 2m$ vertex-disjoint H-subgraphs. Note that this packing does not contain vertices from $H_1, \ldots, H_k$, so we can add these to the packing and obtain a packing of $k + 3n - 2m = \ell$ vertex disjoint H-subgraphs in G.

Condition 3: We now show that if Φ is not satisfiable, then there does not exist a set $X \subseteq V(G)$ of size at most ℓ such that $G - X$ is H-subgraph free. Suppose there

exists such a set X. Since there is a packing of ℓ vertex-disjoint H-subgraphs in G, we know that X contains exactly one vertex from each H-subgraph in the packing. Since v_{x_i} and $v_{\neg x_i}$ belong to the same subgraph, they cannot both be contained in X. Consider the variable assignment where x_i is assigned *true* if $v_{x_i} \in X$ or *false* otherwise. Since we assumed Φ is not satisfiable, there is at least one clause in Φ that evaluates to *false* with this variable assignment. Let C_j denote such a clause. Since C_j evaluates to *false*, all of its literals must be *false*, so for all variables x_i that are not negated in C_j we have $x_i = \mathit{false}$ and therefore $v_{x_i} \notin X$. For all negated variables x_i in C_j we know $x_i = \mathit{true}$ meaning $v_{x_i} \in X$, so $v_{\neg x_i} \notin X$. This means that $\emptyset = X \cap S_j = X \cap V(W_j) \cap V(G_{var})$, but since G_{var} contains k vertex-disjoint H-subgraphs we have $|X \cap V(G_{var})| \geq k$, so then $|X \cap (V(G_{var}) \setminus V(W_j))| \geq k$. For all i there is a packing of $3|C_i| - 2$ vertex-disjoint H-subgraphs in $W_i - S = W_i - V(G_{var})$, so in the graph $G - V(W_j)$ there are $k + \sum_{i \neq j}(3|C_i| - 2)$ vertex-disjoint H-subgraphs. This means that $|X \cap (V(G) \setminus V(W_j))| \geq k + \sum_{i \neq j}(3|C_i| - 2)$, and since $|X| = k + \sum_{1 \leq i \leq m}(3|C_i| - 2)$ we know that $|X \cap V(W_j)| \leq 3|C_j| - 2$. However W_j contains $3|C_j| - 1$ vertex-disjoint H-subgraphs, so $G - X$ cannot be H-subgraph free. Contradiction.

Condition 4: Finally we show that if Φ is satisfiable then there exists a set $X \subseteq V(G)$ of size at most ℓ such that $G - X$ is H-minor free, $\lambda(H)$-prune$(G - X) \preceq H$, and $\mathsf{tw}(G - X) \leq \mathsf{tw}(H)$. Since Φ is satisfiable there exists a variable assignment such that each clause contains at least one literal that is true. Consider the set X' consisting of all vertices v_{x_i} when x_i is *true* and $v_{\neg x_i}$ when x_i is *false*. Since every clause contains one literal that is true, we know for each $1 \leq j \leq m$ that W_j contains at least one vertex from X'. So for each $1 \leq j \leq m$ we have $X' \cap S_j \neq \emptyset$. Take an arbitrary vertex $v_j \in X' \cap S_j$ and let $X_j \subseteq V(W_j)$ be the vertex set containing v_j obtained from Condition 4 of Lemma 4.3.6. Let $X = X' \cup \bigcup_{1 \leq j \leq m} X_j$. For all $1 \leq j \leq m$ we know $|X' \cap X_j| \geq 1$ since $v_j \in X' \cap X_j$. So $|X| \leq |X'| + \sum_{1 \leq j \leq m}(3|C_j| - 2) = k + 3n - 2m = \ell$.

By Lemma 4.3.6(4b) we have that $W_j - X_j$ is H-minor free for all $1 \leq j \leq m$, so clearly $W_j - X$ is also H-minor free. Consider an arbitrary connected component G' of $G - X$. If G' is also a connected component of $W_j - X$ for some $1 \leq j \leq m$, then we have that G' is H-minor free, $\lambda(H)$-prune$(G') \preceq H$ (by Lemma 4.3.6(4c)), and $\mathsf{tw}(G') \leq \mathsf{tw}(W_j) \leq \mathsf{tw}(H)$. If G' is not a connected component of $W_j - X$ for any $1 \leq j \leq m$, then it contains a connected component of $H_i - X$ as a subgraph, for some $1 \leq i \leq k$. When G' does not contain any vertices of S we know that G' must be a subgraph of H_i, so G' is H-minor free, $\lambda(H)$-prune$(G') \preceq G' \preceq H$, and $\mathsf{tw}(G') \leq \mathsf{tw}(H_i) = \mathsf{tw}(H)$.

Suppose on the other hand G' does contain a vertex $v \in S$. No connected component of $W_j - X_j$ contains more than one vertex from S and each connected component of G_{var} contains exactly two vertices of S, one of which is in X. So v is the only vertex in G' that is contained in S. Moreover, since S is the only overlap between the graphs G_{var} and W_j for all $1 \leq j \leq m$, we have that v is a cut vertex in G', such that for some $1 \leq i \leq k$, each biconnected component of G' is a

subgraph of $H_i - X$ or $W_j - X$ for any $1 \leq j \leq m$. So each of these biconnected components of G' has treewidth at most $\mathsf{tw}(H)$, hence $\mathsf{tw}(G') \leq \mathsf{tw}(H)$. Also, each biconnected component in G' that is a subgraph of $W_j - X = W_j - X_j$ for some $1 \leq j \leq m$ contains a vertex from S and therefore has size at most $\lambda(H) - 1$ by Lemma 4.3.6(4d) on the choice of X_j. So we have that $\lambda(H)$-prune(G') is a subgraph of H_i, hence $\lambda(H)$-prune$(G') \preceq G' \preceq H$. Additionally since H_i contains at least one vertex that is not contained in G' we have $H \not\preceq \lambda(H)$-prune(G'). Because $H = \lambda(H)$-prune(H) we can conclude by Proposition 4.3.5 that G' is H-minor free. Since H is connected, and all connected components of $G - X$ are H-minor free, $G - X$ must also be H-minor free. We also know for all connected components G' of $G - X$ that $\lambda(H)$-prune$(G') \preceq H$, so $\lambda(H)$-prune$(G - X) \precsim H$. Finally since $\mathsf{tw}(G') \leq \mathsf{tw}(H)$ for each connected component G' of $G - X$ we have that $\mathsf{tw}(G - X) \leq \mathsf{tw}(H)$. □

The construction from Lemma 4.3.8 can directly be used to give a polynomial-parameter transformation from CNF-SAT parameterized by the number of variables. Observe that if $G - X$ is $\mathcal{F}$-minor free, then $G - X$ is also $\mathcal{F}$-subgraph free. Similarly, if $G - X$ contains an H-subgraph for all $X \subseteq V(G)$ with $|X| \leq \ell$, then $G - X$ also contains an H-minor. Therefore, for any *type* $\in \{\text{minor}, \text{subgraph}\}$ and $\mathcal{F}$ consisting of one connected graph on at least three vertices, Lemma 4.3.8 gives a polynomial-parameter transformation from CNF-SAT parameterized by the number of variables to $\mathcal{F}$-*type*-Free Deletion parameterized by deletion distance to $\min \mathrm{tw}(\mathcal{F})$.

4.3.4 Reduction for families of disconnected graphs

When $\mathcal{F}$ contains multiple graphs, each containing a connected component of at least three vertices, it is possible to select a connected component H of one of the graphs in $\mathcal{F}$ such that the construction described in Lemma 4.3.8 forms the main ingredient for a polynomial-parameter transformation. This will formally be argued in the next lemma. To aid the intuition for this technical construction, we describe a simple special case. If $\mathcal{F}$ is a family of connected graphs, each on at least three vertices, and we choose $H \in \mathcal{F}$ as a $\precsim$-minimal graph in $\mathcal{F}$ with $\mathsf{tw}(H) = \min \mathrm{tw}(\mathcal{F})$, we may safely apply the construction of Lemma 4.3.8, to reduce the satisfiability of a CNF-formula Φ to $\mathcal{F}$-Minor-Free Deletion on a graph G. For a deletion set $X \subseteq V(G)$ corresponding to a satisfiable assignment, the graph $G - X$ is guaranteed to be H-minor free by Lemma 4.3.8, and $\mathsf{tw}(G - X) \leq \mathsf{tw}(H)$. The latter implies that $G - X$ also does not contain any graphs $F \in \mathcal{F}$ with $\mathsf{tw}(F) > \min \mathrm{tw}(\mathcal{F})$ as a minor; and since H is connected and $\precsim$-minimal among the treewidth-minimal graphs in $\mathcal{F}$, the fact that $G - X$ is H-minor free implies that $G - X$ does not contain any other treewidth-minimal graph in $\mathcal{F}$ as a minor either. Hence our choice of H ensures that $G - X$ is not only H-minor free, but also $\mathcal{F}$-minor free. The next lemma introduces a more sophisticated choice of H that also works when $\mathcal{F}$ contains disconnected graphs.

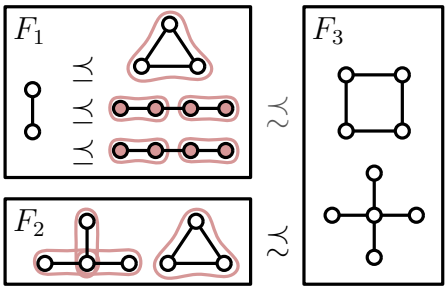


Figure 4.6 In $\mathcal{F} = \{F_1, F_2, F_3\}$ there are two graphs (F_1 and F_2) that are $\precsim$-minimal, in this case both with treewidth $2 = \min \mathrm{tw}(\mathcal{F})$, hence $\mathcal{F}_\downarrow = \{F_1, F_2\}$. Together, the graphs in $\mathcal{F}_\downarrow$ contain five $\preceq$-maximal components. The leaf-blocks of these components are circled in red. Observe that this leaves three candidates for $H_\uparrow$, namely those with a leaf-block of size 2. Suppose we select $H_\uparrow = P_4$, so $H = F_1$, then $c = 2$ since $H_\uparrow$ occurs twice in H. Vertices in Y are colored red.

Lemma 4.3.9. *For any fixed finite set of graphs $\mathcal{F}$, all with a connected component of at least 3 vertices, there exists a polynomial time algorithm that, given a* CNF*-formula Φ with k variables, outputs a graph G and integer ℓ such that all of the following are true:*

1. *there exists a set $S \subseteq V(G)$ of at most $k^{\mathcal{O}(1)}$ vertices such that $\mathsf{tw}(G - S) \leq \min \mathrm{tw}(\mathcal{F})$,*
2. *if Φ is not satisfiable then there does not exist a set $X \subseteq V(G)$ of size at most ℓ such that $G - X$ is $\mathcal{F}$-subgraph free, and*
3. *if Φ is satisfiable then there exists a set $X \subseteq V(G)$ of size at most ℓ such that $G - X$ is $\mathcal{F}$-minor free.*

Proof. Before describing the construction of G and ℓ we define some graphs and sets based on $\mathcal{F}$.

Note that as a consequence of Observation 4.2.2, there is a graph $F \in \mathcal{F}$ that is $\precsim$-minimal with $\mathsf{tw}(F) = \min \mathrm{tw}(\mathcal{F})$. Let $\mathcal{F}_\downarrow \subseteq \mathcal{F}$ denote the set of all $\precsim$-minimal graphs in $\mathcal{F}$ that have treewidth $\min \mathrm{tw}(\mathcal{F})$. We select a $\preceq$-maximal component $H_\uparrow$ of a graph $H \in \mathcal{F}_\downarrow$ such that $\lambda(H_\uparrow) \leq \lambda(H'_\uparrow)$ for all $\preceq$-maximal components $H'_\uparrow$ of any $H' \in \mathcal{F}_\downarrow$. Note that $H_\uparrow$ contains at least 3 vertices since otherwise $H_\uparrow$ would be a minor of at least one connected component of H containing at least 3 vertices, which contradicts $H_\uparrow$ being a $\preceq$-maximal component of H. Let $c \geq 1$ denote the number of connected components in H isomorphic to $H_\uparrow$ and let Y denote the set of vertices contained in these connected components, i.e., $H[Y]$ is isomorphic to $c \cdot H_\uparrow$. See Figure 4.6 for an example of the choices of $\mathcal{F}_\downarrow$, H, $H_\uparrow$, and c for a concrete $\mathcal{F}$.

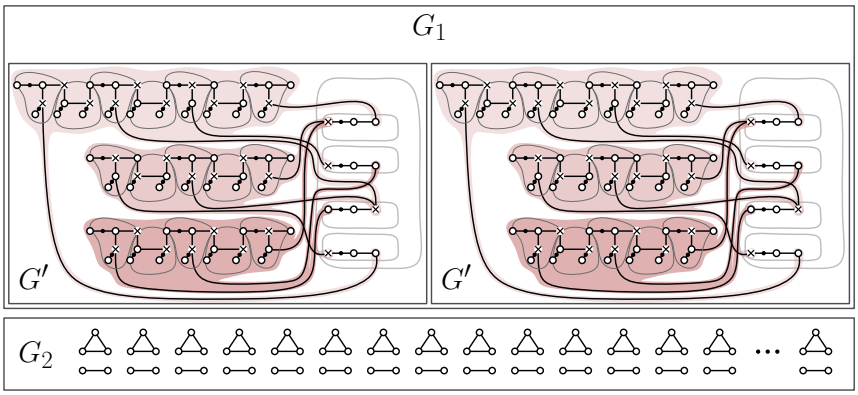


Figure 4.7 Based on the choices of H, $H_\uparrow$, c, and Y in Fig. 4.6 and the CNF-formula Φ as in Fig. 4.5 we obtain the graph $G = G_1 \cup G_2$ depicted above.

Construction We take the algorithm from Lemma 4.3.8 for the graph $H_\uparrow$ and apply it to Φ to construct a graph G' and integer ℓ'. Let $S' \subseteq V(G')$ be the vertex set obtained from Lemma 4.3.8(1). Let $G_1 := (2c-1) \cdot G'$, and let the set S be the union of all $2c-1$ corresponding copies of S'. Take $\ell := (2c-1) \cdot \ell'$ and let $G_2 := (\ell+1) \cdot (H - Y)$ and $G := G_2 \cup G_1$. See Figure 4.7 for a concrete example of G.

Before proving the conditions of the lemma statement hold for G and ℓ we prove some properties of G_2.

Claim 4.3.10. G_2 has the following four properties: (1) $G_2 \precsim H$, (2) $\mathsf{tw}(G_2) \leq \mathsf{tw}(H)$, (3) G_2 is H-minor free, and (4) G_2 is $\mathcal{F}$-minor free.

Proof. Property (1) follows directly from the construction and Property (2) follows directly from Property (1). To show Property (3), we show that G_2 is $H_\uparrow$-minor free. Suppose for contradiction that G_2 contains $H_\uparrow$ as minor then, since $H_\uparrow$ is connected, there is a connected component H' of G_2 that contains $H_\uparrow$ as minor. H' is also a connected component of H. Since $H_\uparrow$ is a $\preceq$-maximal component of H and $H_\uparrow \preceq H'$ we know $H' \preceq H_\uparrow$, and it follows from Observation 4.2.1 that H' is isomorphic to $H_\uparrow$. This is a contradiction since G_2 contains only connected components of H that are not isomorphic to $H_\uparrow$.

Having shown that G_2 is $H_\uparrow$-minor free, Property (4) is easily shown by contradiction. Suppose G_2 is not $\mathcal{F}$-minor free, then there exists a graph $B \in \mathcal{F}$ such that $B \preceq G_2$. It follows from $G_2 \precsim H$ that $B \precsim H$ and since H is $\precsim$-minimal in $\mathcal{F}$ we have that $H \precsim B \preceq G_2$, but then $H_\uparrow \preceq G_2$. This is a contradiction since G_2 is $H_\uparrow$-minor free. ⌟

Correctness We show all conditions of the lemma statement hold for G and ℓ.

Condition 1.: Observe that $|S| = (2c-1)\cdot 2k \in k^{\mathcal{O}(1)}$. By Lemma 4.3.8(1) that $\mathsf{tw}(G_1 - S) \leq \mathsf{tw}(H_\uparrow) \leq \mathsf{tw}(H)$ and since $\mathsf{tw}(G_2) \leq \mathsf{tw}(H)$ by Claim 4.3.10, we obtain $\mathsf{tw}(G-S) \leq \mathsf{tw}(H) = \min \mathrm{tw}(\mathcal{F})$.

Condition 2.: Suppose Φ is not satisfiable, and take an arbitrary $X \subseteq V(G)$ of size at most ℓ. We prove $G - X$ is not $\mathcal{F}$-subgraph free by showing that $G - X$ contains an H-subgraph. First note that $G_2 - X$ contains at least one copy of $H - Y = H - c \cdot H_\uparrow$, so it remains to show that $G_1 - X$ contains c vertex-disjoint $H_\uparrow$-subgraphs. Recall that G_1 is the disjoint union of $2c-1$ copies of G'. Consider the subgraph $\hat{G}_1$ of G_1 consisting of the G'-subgraphs in G_1 that contain at most ℓ' vertices of X. Since Φ is not satisfiable, G' leaves at least one $H_\uparrow$-subgraph when ℓ' or fewer vertices are removed, so each G'-subgraph in $\hat{G}_1$ leaves at least one $H_\uparrow$-subgraph in $G_1 - X$. When $\hat{G}_1$ contains at least c vertex-disjoint G'-subgraphs, we know that there are at least c vertex-disjoint $H_\uparrow$-subgraphs in $G_1 - X$, concluding the proof. Suppose instead that $\hat{G}_1$ contains less than c vertex-disjoint G'-subgraphs. Let x be the number of G'-subgraphs in $G_1 - V(\hat{G}_1)$. Since G_1 contains $2c-1$ vertex-disjoint G'-subgraphs we have $x \geq c$. Each of the G'-subgraphs in $G_1 - V(\hat{G}_1)$ contains at least $\ell'+1$ vertices of X, so $\hat{G}_1$ contains at most $\ell - x(\ell'+1)$ vertices of X. We also know $\hat{G}_1$ contains $\ell'((2c-1)-x)$ vertex-disjoint $H_\uparrow$-subgraphs since G' contains ℓ' vertex-disjoint $H_\uparrow$-subgraphs (by Lemma 4.3.8(2)) and there are $(2c-1)-x$ vertex-disjoint G'-subgraphs in $\hat{G}_1$. We conclude that the number of vertex-disjoint $H_\uparrow$-subgraphs in $\hat{G}_1 - X$, and therefore also in $G_1 - X$, is at least

$$\begin{aligned}
\ell'((2c-1)-x) - (\ell - x(\ell'+1)) &= \ell'((2c-1)-x) - (\ell'(2c-1) - \ell' x - x) \\
&= \ell'((2c-1)-x) - \ell'((2c-1)-x) + x \\
&= x \geq c.
\end{aligned}$$

This concludes the proof of Condition 2..

Condition 3.: When Φ is satisfiable we know that there exists a set $X' \subseteq V(G')$ of size at most ℓ' such that $G' - X'$ is $H_\uparrow$-minor free and $\lambda(H_\uparrow)$-prune$(G' - X') \precsim H_\uparrow$. So then there exists a set $X \subseteq V(G_1)$ of size at most $(2c-1)\cdot \ell' = \ell$ such that $G_1 - X$ is $H_\uparrow$-minor free and $\lambda(H_\uparrow)$-prune$(G_1 - X) \precsim H_\uparrow$. Since G_2 is also $H_\uparrow$-minor free we know that $G - X$ is $H_\uparrow$-minor free and therefore also H-minor free. We now show that $G - X$ is also $\mathcal{F}$-minor free.

First observe the following:

$$\lambda(H_\uparrow)\text{-prune}(G_2 - X) \preceq G_2 - X \preceq G_2 \precsim H, \text{ and} \tag{4.1}$$

$$\lambda(H_\uparrow)\text{-prune}(G_1 - X) \precsim H_\uparrow \preceq H. \tag{4.2}$$

We now deduce

$$\begin{aligned}
\lambda(H_\uparrow)\text{-prune}(G - X) &= \lambda(H_\uparrow)\text{-prune}((G_2 - X) \cup (G_1 - X)) \\
&= \lambda(H_\uparrow)\text{-prune}(G_2 - X) \cup \lambda(H_\uparrow)\text{-prune}(G_1 - X)) \\
&\precsim H \qquad \text{(by Equations (4.1) and (4.2))}
\end{aligned}$$

Suppose $G-X$ is not $\mathcal{F}$-minor free, then for some $H' \in \mathcal{F}$ we have $H' \preceq G-X$. There must exist a graph $B \in \mathcal{F}$ such that B is $\precsim$-minimal in $\mathcal{F}$ and $B \precsim G-X$ since if H' is $\precsim$-minimal in $\mathcal{F}$ then H' forms such a graph B, and if on the other hand H' is not $\precsim$-minimal in $\mathcal{F}$ then there exists a graph $H'' \in \mathcal{F}$ such that $H'' \precsim H'$ and H'' is $\precsim$-minimal in $\mathcal{F}$, meaning H'' forms such a graph B.

Since $B \precsim G-X$ we know by Observation 4.2.2 that $\mathsf{tw}(B) \leq \mathsf{tw}(G-X)$. Recall that $\mathsf{tw}(G-X) \leq \min \mathrm{tw}(\mathcal{F})$ so then $B \in \mathcal{F}_{\downarrow}$. Because of how we chose $H_{\uparrow}$, we know for all $\preceq$-maximal components $B_{\uparrow}$ of B that $\lambda(B_{\uparrow}) \geq \lambda(H_{\uparrow})$. Therefore

$$\begin{aligned} B &\precsim \lambda(H_{\uparrow})\text{-prune}(B) && \text{since } B = \lambda(H_{\uparrow})\text{-prune}(B) \\ &\precsim \lambda(H_{\uparrow})\text{-prune}(G-X) && \text{by Proposition 4.3.5 since } B \precsim G-X \\ &\precsim H. \end{aligned}$$

Since H is $\precsim$-minimal in $\mathcal{F}$, it follows that $H \precsim B$. By definition of $\precsim$ we have $H_{\uparrow} \preceq B \precsim G-X$. Since $H_{\uparrow}$ is connected we conclude $H_{\uparrow} \preceq G-X$. This is a contradiction since $G-X$ is $H_{\uparrow}$-minor free. □

We conclude that a polynomial-parameter transformation exists for all *type* $\in \{\text{minor}, \text{subgraph}\}$ and $\mathcal{F}$ containing only graphs with a connected component on at least three vertices. Together with the fact that CNF-SAT is $\mathsf{MK}[2]$-hard and does not admit a polynomial kernel unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ (cf. [72, Lemma 9]), this proves the following generalization of Theorem 4.1.1.

Theorem 4.3.11. *For type* $\in \{\text{minor}, \text{subgraph}\}$ *and a set* $\mathcal{F}$ *of graphs, all with a connected component of at least three vertices,* $\mathcal{F}$*-type*-Free Deletion *parameterized by vertex-deletion distance to a graph of treewidth* $\min \mathrm{tw}(\mathcal{F})$ *is* $\mathsf{MK}[2]$*-hard and does not admit a polynomial kernel unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

4.4 A polynomial Turing kernelization

In this section we consider the case where $\mathcal{F}$ contains a graph with no connected component of more than two vertices; or in short $\mathcal{F}$ contains a P_3-subgraph-free graph. This graph consists of isolated vertices and disjoint edges. Let $\mathsf{isol}(G)$ denote the set of isolated vertices in a graph G, i.e., $\mathsf{isol}(G) = \{v \in V(G) \mid \deg(v) = 0\}$. We first show that the removal of all isolated vertices from all graphs in $\mathcal{F}$ only changes the answer to $\mathcal{F}$-Minor-Free Deletion and $\mathcal{F}$-Subgraph-Free Deletion when the input is of constant size.

Lemma 4.4.1. *For type* $\in \{\text{minor}, \text{subgraph}\}$ *and any family of graphs* $\mathcal{F}$ *containing a* P_3*-subgraph-free graph, let* $\mathcal{F}' = \{F - \mathsf{isol}(F) \mid F \in \mathcal{F}\}$. *For any graph* G*, if* G *is* $\mathcal{F}$*-type free but not* $\mathcal{F}'$*-type free, then* $|V(G)| < \max_{F \in \mathcal{F}}(|V(F)| + 2|V(F)|^3)$.

Proof. We first prove the lemma for *type* = subgraph. Suppose G is $\mathcal{F}$-subgraph free but not $\mathcal{F}'$-subgraph free, so G contains an H'-subgraph for some $H' \in \mathcal{F}'$.

This subgraph consists of $|V(H')|$ vertices. Let $H \in \mathcal{F}$ be the graph for which $H' = H - \mathsf{isol}(H)$. The graph G cannot contain $|\mathsf{isol}(H)|$ vertices in addition to the vertices in the H'-subgraph because otherwise G trivially contains an $\mathcal{F}$-subgraph. Hence $|V(G)| < |V(H')| + |\mathsf{isol}(H)| = |V(H)| \leq \max_{F \in \mathcal{F}} |V(F)|$.

Next, we show the lemma holds for *type* = minor. If some graph G is $\mathcal{F}$-minor free but not $\mathcal{F}'$-minor free then for some graph $H \in \mathcal{F}$ we have $H' \preceq G$ but not $H \preceq G$ where $H' = H - \mathsf{isol}(H)$. Let φ be a minimal H'-model in G. The graph G has less than $|V(\mathsf{isol}(H))|$ vertices that are not in any branch set of φ, since otherwise an H-model could be constructed in G by taking the branch sets of φ and adding $|V(\mathsf{isol}(H))|$ branch sets consisting of a single vertex.

The number of vertices in G that are contained in a branch set of φ can also be limited. For an arbitrary vertex $v \in V(H')$ consider a spanning tree T of $G[\varphi(v)]$. If $\varphi(v)$ contains multiple vertices then for each leaf p of T, there must be a vertex $u \in N_{H'}(v)$ and $q \in N_G(p) \cap \varphi(u)$, such that p is the only vertex from $\varphi(v)$ that is adjacent to $\varphi(u)$; otherwise, removing leaf p from the branch set $\phi(v)$ would yield a smaller H'-model in G. Hence there can only be $\max\{1, \deg_{H'}(v)\}$ leaves in T.

To give a bound on the size of each branch set consider a smallest graph $D \in \mathcal{F}'$ that is P_3-subgraph free. Take $\ell = |V(D)|$ and note that $D \preceq P_\ell$. Since we know that G is $\mathcal{F}'$-minor free, G must also be P_ℓ-subgraph-free, therefore T is also P_ℓ-subgraph free. Consider an arbitrary vertex r in T. Since T is a tree, there is exactly one path from r to each leaf of T and every vertex of T lies on at least one path from r to a leaf of T. Since there are no more than $\max\{1, \deg_{H'}(v)\}$ leaves in T there are at most $\max\{1, \deg_{H'}(v)\}$ such paths, and all these paths contain less than ℓ vertices since T is P_ℓ-subgraph free, hence in total T contains less than $\deg_{H'}(v) \cdot \ell$ vertices. We can now give a bound on the total number of vertices in G as follows:

$$\begin{aligned}
|V(G)| &< |\mathsf{isol}(H)| + \sum_{v \in H - \mathsf{isol}(H)} |\varphi(v)| \\
&\leq |\mathsf{isol}(H)| + \sum_{v \in H - \mathsf{isol}(H)} (\deg_{H'}(v) \cdot \ell) \\
&\leq |\mathsf{isol}(H)| + 2 \cdot |E(H)| \cdot \ell \\
&\leq |V(H)| + 2 \cdot |V(H)|^2 \cdot |V(D)| \\
&\leq \max_{F \in \mathcal{F}} (|V(F)| + 2|V(F)|^3)
\end{aligned}$$

This concludes the proof. □

After the removal of isolated vertices in $\mathcal{F}$ to obtain $\mathcal{F}'$, we know that $\mathcal{F}'$ contains a graph consisting entirely of disjoint edges, i.e., this graph is isomorphic to $c \cdot P_2$ for some integer $c \geq 0$. If $c = 0$ then $\mathcal{F}$-*type*-free graphs have constant size and the problem is polynomial-time solvable. We proceed assuming $c \geq 1$. Let

the matching number of a graph G, denoted as $\nu(G)$, be the size of a maximum matching in G. We make the following observation.

Observation 4.4.2. *For all $c \geq 1$, a graph G is $c \cdot P_2$-subgraph free if and only if $\nu(G) \leq c - 1$.*

We give a characterization of graphs with bounded matching number, based on an adaptation of the Tutte-Berge formula [13]. We use $\mathsf{odd}(G)$ to denote the number of connected components in G that consist of an odd number of vertices.

Lemma 4.4.3. *For any graph G and integer m we have $\nu(G) \leq m$ if and only if $V(G)$ can be partitioned into three disjoint sets U, R, S such that all of the following are true:*

- *all connected components in $G[R]$ have an odd number of at least 3 vertices,*
- *$G[S]$ is independent,*
- *$N_G(S) \subseteq U$, and*
- *$|U| + \frac{1}{2}(|R| - \mathsf{odd}(G[R])) \leq m$.*

Proof. Consider the Tutte-Berge formula [13] (cf. [119, Chapter 24]):

$$\nu(G) = \tfrac{1}{2} \min_{U \subseteq V(G)} (|V(G)| - \mathsf{odd}(G - U) + |U|).$$

Suppose $\nu(G) \leq m$. It follows from the Tutte-Berge formula that there exists a $U_1 \subseteq V(G)$ such that $\frac{1}{2}(|V(G)| - \mathsf{odd}(G - U_1) + |U_1|) = \nu(G) \leq m$. From each connected component H in $G - U_1$ on an even number of vertices, select a vertex that is not a cut vertex of H (any leaf of a spanning tree of H suffices) and add the selected vertices to a set U_2. Now take $U = U_1 \cup U_2$. Note that $G - U$ contains only connected components with an odd number of vertices and $\mathsf{odd}(G - U) = \mathsf{odd}(G - U_1) + |U_2|$. Let S be the set of isolated vertices in $G - U$ and let $R = V(G) \setminus (U \cup S)$. Observe that U, R, and S satisfy the first three conditions in the lemma statement: $G[R]$ contains only connected components with an odd number of at least 3 vertices, $G[S]$ is independent, and $N_G(S) \subseteq U$. Note that this implies that $\mathsf{odd}(G[R]) + |S| = \mathsf{odd}(G[R \cup S]) = \mathsf{odd}(G - U)$. The last requirement follows from the Tutte-Berge formula as follows:

$$\begin{aligned}
|U| + \tfrac{1}{2}(|R| - \mathsf{odd}(G[R])) &= \tfrac{1}{2}(2|U| + |S| - |S| + |R| - \mathsf{odd}(G[R]) \\
&= \tfrac{1}{2}((|U| + |S| + |R|) - (\mathsf{odd}(G[R]) + |S|) + |U|) \\
&= \tfrac{1}{2}(|V(G)| - \mathsf{odd}(G - U) + |U|) \\
&= \tfrac{1}{2}(|V(G)| - (\mathsf{odd}(G - U_1) + |U_2|) + |U_1| + |U_2|) \\
&= \tfrac{1}{2}(|V(G)| - \mathsf{odd}(G - U_1) + |U_1|) \\
&= \nu(G) \leq m
\end{aligned}$$

For the reverse direction of the proof, suppose $V(G)$ can be partitioned into disjoint sets U, R, S as described in the lemma statement. A maximum matching in $G[R]$ has size at most $\frac{1}{2}(|R| - \mathsf{odd}(G[R])$ since at least one vertex in each odd component remains unmatched and every matching edge covers two vertices. Since $N_G(S) \subseteq U$ we know that S is isolated in $G - U$, so $\nu(G - U) = \nu(G[R]) \leq \frac{1}{2}(|R| - \mathsf{odd}(G[R])$. Since a matching in G is at most $|U|$ edges larger than a matching in $G - U$ we conclude $\nu(G) \leq |U| + \frac{1}{2}(|R| - \mathsf{odd}(G[R]) \leq m$. □

Let us showcase how Lemma 4.4.3 can be used to attack $\mathcal{F}$-Minor-Free Deletion when $\mathcal{F}$ consists of a single graph $c \cdot P_2$, so that the problem is to find a set $X \subseteq V(G)$ of size at most ℓ such that $G - X$ has matching number less than c.

Theorem 4.4.4. *For any constant c, the $\{c \cdot P_2\}$-Minor-Free Deletion problem parameterized by the size k of a feedback vertex set, can be solved in polynomial time using an oracle that answers* Vertex Cover *instances with $\mathcal{O}(k^3)$ vertices.*

Proof. If an instance (G, ℓ) admits a solution X, then Lemma 4.4.3 guarantees that $V(G-X)$ can be partitioned into U, R, S satisfying the four conditions for $m = c - 1$. We try all relevant options for the sets U and R in the partition, of which there are only polynomially many since $|U| + \frac{1}{3}|R| \leq m \in \mathcal{O}(1)$.

For given sets $U, R \subseteq V(G)$, we can decide whether there is a solution X of size at most ℓ for which U, R, and $S := V(G) \setminus (U \cup R \cup X)$ form the partition witnessing that $G - X$ has matching number at most m, as follows. If some component of $G[R]$ has an even number of vertices or less than three vertices, we reject outright. Similarly, if $|U| + \frac{1}{2}(|R| - \mathsf{odd}(G[R])) > m$, we reject. Now, if U and R were guessed correctly, then Lemma 4.4.3 guarantees that the only neighbors of R in the graph $G - X$ belong to U. Hence we infer that all vertices of $X' := N_G(R) \setminus U$ must belong to the solution X. Note that since S is an independent set in $G - X$, the solution X forms a vertex cover of $G - (U \cup R)$, so that $X'' := X \setminus X'$ is a vertex cover of $G' := G - (U \cup R \cup X')$. On the other hand, for every vertex cover X'' of G', the graph $G - (X' \cup X'')$ will have matching number at most m, as witnessed by the partition. Hence the problem of finding a minimum solution X whose corresponding graph $G - X$ has U and R as two of the classes in its witness partition, reduces to finding a minimum vertex cover of the graph G'. In terms of the decision problem, this means G has a solution of size at most ℓ with U and R as witness partite sets, if and only if G' has a vertex cover of size at most $\ell - |X'|$. Since $\mathsf{fvs}(G') \leq \mathsf{fvs}(G)$, we can apply the known [79] kernel for Vertex Cover parameterized by the feedback vertex number to reduce $(G', \ell - |X'|)$ to an equivalent instance with $\mathcal{O}(\mathsf{fvs}(G)^3)$ vertices, which is queried to the oracle. If the oracle answers positively to any query, then (G, ℓ) has answer yes; otherwise the answer is no. □

We remark that by using the polynomial-time reduction guaranteed by NP-completeness, the queries to the oracle can be posed as instances of the original $\mathcal{F}$-Minor-Free Deletion problem, rather than Vertex Cover. The following

lemma formalizes this and will be used as a black box for our general Turing kernelization.

Lemma 4.4.5. *Let $\mathcal{F}$ be a finite set of graphs such that each graph in $\mathcal{F}$ contains at least one edge, and let type $\in \{minor, subgraph\}$. There is a polynomial-time algorithm that, given a graph G and integer ℓ, decides whether G has a vertex cover of size at most ℓ, using an oracle that answers $\mathcal{F}$-type-*Free Deletion *instances with $\mathsf{fvs}(G)^{\mathcal{O}(1)}$ vertices.*

Proof. For a fixed $\mathcal{F}$ and *type*, the following procedure solves the Vertex Cover instance (G, ℓ) in polynomial time using an oracle for $\mathcal{F}$-*type*-Free Deletion instances with $\mathsf{fvs}(G)^{\mathcal{O}(1)}$ vertices.

1. Compute a 2-approximate feedback vertex set S on G in polynomial time, for example using the algorithm by Bafna et al. [11].

2. Apply the kernelization by Jansen and Bodlaender [79] for Vertex Cover parameterized by feedback vertex set to the instance (G, ℓ) and the approximate feedback vertex set S. This takes polynomial time, and results in an instance (G_1, ℓ_1) of Vertex Cover on $\mathcal{O}(|S|^3) \leq \mathcal{O}(\mathsf{fvs}(G)^3)$ vertices that is equivalent to (G, ℓ).

3. Since every graph in $\mathcal{F}$ contains at least one edge, the $\mathcal{F}$-*type*-Free Deletion problem is NP-complete [93]. The Vertex Cover problem is also known to be NP-complete, hence there exists a polynomial-time algorithm that transforms the Vertex Cover instance (G_1, ℓ_1) into an equivalent $\mathcal{F}$-*type*-Free Deletion instance (G_2, ℓ_2). Since this algorithm runs in polynomial time and the size of its input is $\mathsf{fvs}(G)^{\mathcal{O}(1)}$, the number of vertices in G_2 is upper-bounded by $\mathsf{fvs}(G)^{\mathcal{O}(1)}$.

4. Query the instance (G_2, ℓ_2) of size $\mathsf{fvs}(G)^{\mathcal{O}(1)}$ to the $\mathcal{F}$-*type*-Free Deletion oracle, and output the oracle's answer as the decision on the Vertex Cover instance (G, ℓ). □

We point out that in Lemma 4.4.5, the oracle that answers $\mathcal{F}$-*type*-Free Deletion instances with $\mathsf{fvs}(G)^{\mathcal{O}(1)}$ vertices may be replaced with an oracle that answers Vertex Cover instances on $\mathcal{O}(\mathsf{fvs}(G)^3)$ vertices, due to the application of the Vertex Cover kernelization in Step 2. Hence when using an oracle for Vertex Cover, the query size can be bounded uniformly and does not depend on $\mathcal{F}$.

We now present our general (non-adaptive) Turing kernelization for the minor-free and subgraph-free deletion problems for all families $\mathcal{F}$ containing a P_3-subgraph-free graph, combining three ingredients. Lemma 4.4.1 allows us to focus on families whose graphs have no isolated vertices. The guessing strategy of Theorem 4.4.4 is the second ingredient. The final ingredient is required to deal with the fact that a solution subgraph $G - X$ that is $c \cdot P_2$-minor free for some $c \cdot P_2 \in \mathcal{F}$, may still

have one of the other graphs in $\mathcal{F}$ as a minor. To cope with this issue, we show in Lemma 4.4.7 that if $G - X$ has no matching of size c (i.e., $G - X$ has a vertex cover of size at most $2c$), but does contain a minor model of some graph in $\mathcal{F}$, then there is such a minor model of constant size. By employing a more expensive (but still polynomially bounded) guessing step, this allows us to complete the Turing kernelization. In the following lemmas $\Delta(G)$ will denote the maximum degree of G.

Proposition 4.4.6 ([59, Proposition 1])**.** *If G contains H as a minor, then there is a subgraph G^* of G containing an H-minor such that $\Delta(G^*) \leq \Delta(H)$ and $|V(G^*)| \leq |V(H)| + \mathsf{vc}(G^*) \cdot (\Delta(H) + 1)$.*

Lemma 4.4.7. *For any type $\in \{minor, subgraph\}$, let $\mathcal{F}$ be a family of graphs, let G be a graph with vertex cover C, and let $S = V(G - C)$. If G contains an $\mathcal{F}$-type, then there exists $S' \subseteq S$ such that $G[C \cup S']$ contains an $\mathcal{F}$-type and $|S'| \leq \max_{H \in \mathcal{F}} |V(H)| + |C| \cdot (\Delta(H) + 1)$.*

Proof. Suppose *type* = minor, then by Proposition 4.4.6 we know that if G contains $H \in \mathcal{F}$ as a minor, then there is a subgraph G^* of G containing an H-minor such that $|V(G^*)| \leq |V(H)| + \mathsf{vc}(G^*) \cdot (\Delta(H) + 1)$. Take $S' = V(G^*) \cap S$, then $G[C \cup S'] = G[C \cup V(G^*)]$ contains an $\mathcal{F}$-minor and

$$\begin{aligned} |S'| &\leq |V(G^*)| \\ &\leq |V(H)| + \mathsf{vc}(G^*) \cdot (\Delta(H) + 1) \\ &\leq |V(H)| + |C| \cdot (\Delta(H) + 1) \\ &\leq \max_{H \in \mathcal{F}} |V(H)| + |C| \cdot (\Delta(H) + 1). \end{aligned}$$

On the other hand, when *type* = subgraph then G contains an H-subgraph for some $H \in \mathcal{F}$, and trivially there exists a set $X \subseteq V(G)$ of $|V(H)|$ vertices such that $G[X]$ contains an H-subgraph. Take $S' = X - C$ and clearly $G[C \cup S']$ contains an H-subgraph. □

Armed with Lemma 4.4.7 we now present the proof of the general Turing kernelization.

Theorem 4.1.2. *Let $\mathcal{F}$ be a finite family of graphs, such that some $H \in \mathcal{F}$ has no connected component of three or more vertices. Then $\mathcal{F}$-*Minor-Free Deletion *and $\mathcal{F}$-*Subgraph-Free Deletion *admit polynomial Turing kernels when parameterized by the vertex-deletion distance to a graph of treewidth* $\min \mathrm{tw}(\mathcal{F})$.

Proof. Fix some *type* $\in \{\text{minor}, \text{subgraph}\}$. First, consider input instances (G, ℓ) for which $|V(G)| - \ell \leq \max_{F \in \mathcal{F}}(|V(F)| + 2|V(F)|^3)$. If $|V(G)| - \ell < 0$, there is a trivial solution. Otherwise, there exists a vertex set X of size at most ℓ such that $G - X$ is $\mathcal{F}$-*type* free if and only if there exists a set X' of size exactly ℓ such that $G - X'$ is $\mathcal{F}$-*type* free, since $\mathcal{F}$-*type*-free graphs are hereditary and X' be obtained by adding sufficiently many vertices to X. Such a set X' exists if and only if

there exists a vertex set Y of size exactly $|V(G)|-\ell \leq \max_{F\in\mathcal{F}}(|V(F)|+2|V(F)|^3)$ such that $G[Y]$ is $\mathcal{F}$-*type* free. Since there are only polynomially many such vertex sets Y, and for each Y we can check in polynomial time whether $G[Y]$ contains an $\mathcal{F}$-*type* [112], we can apply brute force to solve the instance in polynomial time.

So from now on we only consider instances (G,ℓ) for which $|V(G)| - \ell > \max_{F\in\mathcal{F}}(|V(F)| + 2|V(F)|^3)$. This means that for any vertex set X of size at most ℓ, the graph $G-X$ contains more than $\max_{F\in\mathcal{F}}(|V(F)|+2|V(F)|^3)$ vertices. Take $\mathcal{F}' = \{F - \mathsf{isol}(F) \mid F \in \mathcal{F}\}$ and we obtain from Lemma 4.4.1 that if $G-X$ is $\mathcal{F}$-*type* free, it is also $\mathcal{F}'$-*type* free, and clearly if $G-X$ contains an $\mathcal{F}$-*type* it also contains an $\mathcal{F}'$-*type*. Hence the $\mathcal{F}$-*type*-Free Deletion instance (G,ℓ) is equivalent to the $\mathcal{F}'$-*type*-Free Deletion instance (G,ℓ). Note that if $\mathcal{F}$ contains an edgeless graph then $\mathcal{F}'$ contains the null graph. In this case the instance is trivially false since every graph contains the null graph as a subgraph. In the rest of the algorithm we assume each graph in $\mathcal{F}'$ contains at least one edge.

Since every graph in $\mathcal{F}$ contains an edge and at least one graph in $\mathcal{F}$ has no component of three vertices or more, we have $\min \mathrm{tw}(\mathcal{F}) = 1$. Therefore the parameter, the deletion distance to treewidth $\min \mathrm{tw}(\mathcal{F})$, is equal to $\mathsf{fvs}(G)$.

To complete the Turing kernelization for $\mathcal{F}$-*type*-Free Deletion, it suffices to give a polynomial-time algorithm solving $\mathcal{F}'$-*type*-Free Deletion using an oracle that can solve $\mathcal{F}$-Minor-Free Deletion instances (G',ℓ') for which $|V(G')| \leq \mathsf{fvs}(G)^{\mathcal{O}(1)}$ and $\mathsf{fvs}(G') \leq \mathsf{fvs}(G)^{\mathcal{O}(1)}$. Note that the latter condition on (G',ℓ') is redundant since $\mathsf{fvs}(G') < |V(G')|$ for any graph G'.

Our Turing kernelization will use the algorithm described in Lemma 4.4.5 to solve Vertex Cover instances (G'',ℓ'') for induced subgraphs G'' of G. This algorithm requires an oracle for $\mathcal{F}$-*type*-Free Deletion instances with $\mathsf{fvs}(G'')^{\mathcal{O}(1)}$ vertices. Note that since G'' is an induced subgraph of G, we have $\mathsf{fvs}(G'') \leq \mathsf{fvs}(G)$, hence our $\mathcal{F}$-*type*-Free Deletion oracle for instances with $\mathsf{fvs}(G)^{\mathcal{O}(1)}$ suffices. We will refer to this algorithm as `VCoracle`.

Using this subroutine, the Turing kernelization algorithm is given in Algorithm 1. The high-level idea is as follows. Let M be the smallest graph in $\mathcal{F}'$ that has no component of three or more vertices, or equivalently, which is P_3-subgraph free; then M consists of isolated edges. The Turing kernelization first guesses the sets U and R as per Lemma 4.4.3 witnessing that the graph $G-X$ obtained after removing the unknown solution X does not have a matching of $|E(M)|$ edges (i.e., that $G-X$ does not contain $M \in \mathcal{F}'$ as both a minor and a subgraph). Since Lemma 4.4.3 guarantees that in the graph $G-X$ we have $N_{G-X}(R) \subseteq U$, it follows that $N_G(R) \setminus U$ must belong to the unknown solution X if this guess was correct. The algorithm then considers the remaining vertices $Q := V(G) \setminus (U \cup R \cup N_G(R))$ and classifies them into $2^{|U|}$ types based on their adjacency to U. An additional guessing step attempts to guess up to α (line 2) vertices of each type in $G-X$, which will be part of the set S in the partition of Lemma 4.4.3, by taking them into the range of the function f (line 8). The algorithm tests whether the graph $G[f(2^U) \cup U \cup R]$ is $\mathcal{F}'$-*type* free. If not, then the guess was incorrect.

If so, then for each type of which fewer than α vertices were guessed to remain behind in $G - X$, the algorithm collects the remaining vertices of that type in a set Q' to be added to the solution X, and a Vertex Cover instance is formulated on the remaining vertices of Q. For types of which α vertices remained behind, no vertices have to be added to Q' or the solution X in this step, because using Lemma 4.4.7 it can be guaranteed that having more vertices of that type will not lead to an $\mathcal{F}'$-*type*. The algorithm returns *true* if the formulated instance of Vertex Cover has a solution that yields a set of size at most ℓ when combined with the vertices of $N_G(R) \setminus U$ and Q'.

Algorithm 1: Solving $\mathcal{F}'$-*type*-Free Deletion instances using `VCoracle` with $\mathcal{F}'$ containing a P_3-subgraph-free graph and no edgeless graphs.

input : A graph G and an integer ℓ
output: `true` if there exists a set X of size at most ℓ such that $G - X$ is $\mathcal{F}'$-*type*-free, or `false` otherwise.

$m := |E(M)| - 1$ where M is a smallest P_3-subgraph-free graph in $\mathcal{F}'$
$\alpha := \max_{H \in \mathcal{F}'} |V(H)| + 3m(\Delta(H) + 1)$
forall $U \subseteq V(G)$ with $|U| \leq m$ **do**
  forall $R \subseteq V(G - U)$ such that
    all connected components in $G[R]$ have an odd number of at least 3 vertices and $|U| + \frac{1}{2}(|R| - \mathsf{odd}(G[R])) \leq m$
  do
    $Q := V(G) \setminus (U \cup R \cup N_G(R))$
    forall functions $f\colon 2^U \to 2^Q$ such that
      $G[f(2^U)]$ is independent and
        ▷ `Recall that` $f(2^U) = \bigcup_{Y \subseteq U} f(Y)$
      $G[f(2^U) \cup U \cup R]$ is $\mathcal{F}'$-*type* free and
      $\forall_{Y \subseteq U} |f(Y)| \leq \alpha$ and
      $\forall_{Y \subseteq U} \forall_{v \in f(Y)} N_G(v) \cap U = Y$
    do
      $Q' := \{v \in Q \setminus f(2^U) \mid |f(N_G(v) \cap U)| < \alpha\}$
      if `VCoracle`$(G[Q] - Q', \ell - |(N_G(R) \setminus U) \cup Q'|)$ **then**
        return `true`
return `false`

Soundness When the algorithm returns *true*, then consider the values of U, R, Q, f, and Q' at the time that *true* is returned. There exists a vertex cover X' of size at most $\ell - |(N_G(R) \setminus U) \cup Q'|$ in $G[Q] - Q'$. Let $X = X' \cup (N_G(R) \setminus U) \cup Q'$, which has size at most ℓ. The set X is a vertex cover in $G - (U \cup R)$, since $G - (U \cup R) - X =$

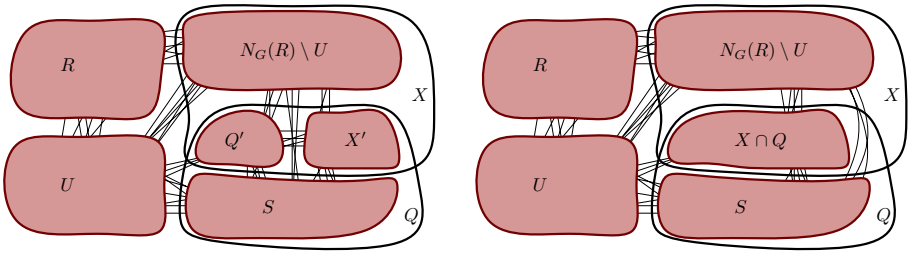


Figure 4.8 We show two partitions of G. Figure 4.8a shows a partition of G given that Algorithm 1 returns *true*, while Figure 4.8b shows a partition of G given that $G - X$ is $\mathcal{F}$-*type* free. Note that in both cases there can be no edges between R and Q.

$(G[Q] - Q') - X'$. Hence $S := V(G - (U \cup R)) \setminus X$ is an independent set, even in G. The sets U, R, S, X form a partition of $V(G)$. See Figure 4.8a for a visual representation of these sets. We will show that X is a solution to $\mathcal{F}'$-*type*-Free Deletion on G.

Consider an arbitrary vertex $v \in S$. Note that since $N_G(R) \subseteq U \cup X$ we have $S = V(G) \setminus (U \cup R \cup X) \subseteq V(G) \setminus (U \cup R \cup N_G(R)) = Q$, so $v \in Q$. By definition of X we know $Q' \subseteq X$ so $v \notin Q'$. Then by definition of Q' on line 14 we observe the following:

Observation 4.4.8. *For all $v \in S$ we have $v \in f(2^U)$ or $|f(N_G(v) \cap U)| \geq \alpha$.*

Assume for a contradiction that $G - X = G[U \cup R \cup S]$ contains an $\mathcal{F}'$-*type*. Since $G[S]$ is independent, $U \cup R$ is a vertex cover in $G - X$, and by Lemma 4.4.7 there exists a set $S' \subseteq S$ with $|S'| \leq \max_{H \in \mathcal{F}'} |V(H)| + |U \cup R| \cdot (\Delta(H) + 1)$ such that $G[U \cup R \cup S']$ contains an $\mathcal{F}'$-*type*. Note that $|R| - 3\,\mathsf{odd}(G[R]) \geq 0$ since every connected component in $G[R]$ contains at least 3 vertices, so then

$$\begin{aligned}
|S'| &\leq \max_{H \in \mathcal{F}'} |V(H)| + |U \cup R| \cdot (\Delta(H) + 1) \\
&\leq \max_{H \in \mathcal{F}'} |V(H)| + (|U| + |R| + \frac{1}{2}(|R| - 3\,\mathsf{odd}(G[R]))) \cdot (\Delta(H) + 1) \\
&\leq \max_{H \in \mathcal{F}'} |V(H)| + 3(|U| + \frac{1}{2}(|R| - \mathsf{odd}(G[R]))) \cdot (\Delta(H) + 1) \\
&\leq \max_{H \in \mathcal{F}'} |V(H)| + 3m(\Delta(H) + 1) \\
&= \alpha.
\end{aligned}$$

Claim 4.4.9. The graph $G[U \cup R \cup S']$ is isomorphic to a subgraph of $G[U \cup R \cup f(2^U)]$.

Proof. Observe that S contains no neighbors of R, and since $G[S]$ is independent, we know for all $v \in S$ that $N_G(v) \subseteq U \cup X$ and therefore $N_{G-X}(v) = N_G(v) \cap U$. From Observation 4.4.8 it follows for all $v \in S'$ that $v \in f(2^U)$ or $|f(N_G(v) \cap U)| \geq \alpha$. In the latter case v is a false twin of any vertex $u \in f(N_G(v) \cap U)$ in $G - X$ since by definition of f we have $N_G(u) \cap U = N_G(v) \cap U$ for all vertices $u \in f(N_G(v) \cap U)$. We have $|S'| \leq |f(N_G(v) \cap U)|$ for all $v \in S'$, so there exists a bijection that maps all vertices $v \in S'$ to a vertex $u \in f(2^U)$ that is a false twin of v in $G - X$. Any two false twins in $G - X$ are interchangeable in $G - X$, hence $G[U \cup R \cup S']$ is isomorphic to a subgraph of $G[U \cup R \cup f(2^U)]$. ⌟

Since f is chosen such that $G[U \cup R \cup f(2^U)]$ is $\mathcal{F}'$-*type* free on line 10, Claim 4.4.9 leads to a contradiction with the fact that $G[U \cup R \cup S']$ contains an $\mathcal{F}'$-*type*. We conclude that if the algorithm returns *true* a set X of size ℓ exists such that $G - X$ is $\mathcal{F}'$-*type* free.

Completeness Next, we consider the reverse direction. We show that the algorithm returns *true* when there exists a set X of size at most ℓ such that $G - X$ is $\mathcal{F}'$-*type* free. Let $m = |E(M)| - 1$ where M is the smallest P_3-subgraph-free graph in $\mathcal{F}'$, i.e., M is isomorphic to $(m+1) \cdot P_2$ since no graph in $\mathcal{F}'$ contains isolated vertices. The graph $G - X$ is $\mathcal{F}'$-*type* free so it is also $(m+1) \cdot P_2$-subgraph free, and by Observation 4.4.2 we know $\nu(G - X) \leq m$. Therefore by Lemma 4.4.3 there exists a partition U', R', S of $V(G - X)$ such that all of the following are true:

- all connected components in $(G - X)[R'] = G[R']$ have an odd number of at least 3 vertices,
- $(G - X)[S] = G[S]$ is independent,
- $N_{G-X}(S) \subseteq U$ or equivalently $N_G(S) \subseteq U \cup X$, and
- $|U'| + \frac{1}{2}(|R'| - \mathsf{odd}(G[R'])) \leq m$.

Clearly U' and R' are such that there is an iteration in the algorithm where $U = U'$ and $R = R'$. Let Q be the set as defined on line 7 in this iteration, see Figure 4.8b. Let $g \colon 2^U \to 2^S$ be defined as $g(Y) = \{v \in S \mid Y = N_G(v) \cap U\}$ for all $Y \subseteq U$. We define a function $f' \colon 2^U \to 2^S$ that maps any $Y \subseteq U$ to an arbitrary subset of $g(Y)$ of size $\min\{|g(Y)|, \alpha\}$. We make the following observations:

- Since $N_G(S) \subseteq U \cup X$ we have $N_G(R) \cap S = \emptyset$ so $S = S \setminus N_G(R) = V(G) \setminus (U \cup R \cup X \cup N_G(R)) \subseteq V(G) \setminus (U \cup R \cup N_G(R)) = Q$, so $f' \colon 2^U \to 2^Q$.
- $G[S]$ is independent, so $G[f'(2^U)]$ is also independent because $f'(2^U) \subseteq S$.
- $G[U \cup R \cup f'(2^U)]$ is a subgraph of $G[U \cup R \cup S] = G - X$, and since $G - X$ is $\mathcal{F}'$-*type* free, $G[U \cup R \cup f'(2^U)]$ is also $\mathcal{F}'$-*type* free.

- Clearly $\forall_{Y \subseteq U} |f'(Y)| \leq \alpha$, and
- $\forall_{Y \subseteq U} \forall_{v \in f'(Y)} N_G(v) \cap U = Y$.

Hence f' satisfies all conditions stated in line 8 of the algorithm, so there is an iteration of the algorithm where $f = f'$. Let Q' be the set as defined on line 14 in this iteration. We now show that there exists a vertex cover of size at most $\ell - |(N_G(R) \setminus U) \cup Q'|$ in $G[Q] - Q'$.

Since $G[S]$ is independent, X is a vertex cover in $G[X \cup S] = G - (U \cup R)$. Then clearly $X \setminus (N_G(R) \setminus U)$ is a vertex cover in $G - (U \cup R \cup (N_G(R) \setminus U))$ and since $N_G(R) \setminus U \subseteq X$ we have $|X \setminus (N_G(R) \setminus U)| \leq \ell - |N_G(R) \setminus U|$. Similarly consider the set $A = (N_G(R) \setminus U) \cup Q'$. Clearly $X \setminus A$ is a vertex cover in $G - (U \cup R \cup A)$ and $|X \setminus A| \leq \ell - |A|$ if $A \subseteq X$. We will show that $A \subseteq X$. We know $N_G(R) \setminus U \subseteq X$ so it remains to be shown that $Q' \subseteq X$. Consider an arbitrary $v \in Q'$ and suppose $v \notin X$. Since $Q' \subseteq Q$ we obtain from the definition of Q that $v \notin U$ and $v \notin R$, so then $v \in S$. We also note from the definition of Q' that $|f(N_G(v) \cap U)| < \alpha$. Since $f = f'$ we have $|f'(N_G(v) \cap U)| < \alpha$, and from the definition of f' we know that if $|f'(Y)| < \alpha$ for some $Y \subseteq U$, then $f'(Y) = g(Y)$. By definition of g we have $v \in g(N_G(v) \cap U)$, so then $v \in f(N_G(v) \cap U) \subseteq f(2^U)$. This is a contradiction since $v \notin f(2^U)$ by definition of Q'.

Now we have shown that $X \setminus A$ is a vertex cover of size at most $\ell - |A| = \ell - |(N_G(R) \setminus U) \cup Q'|$ in $G - (U \cup R \cup A) = G[Q] - Q'$, hence the `VCoracle` should report that a vertex cover exists on line 8.

Running time and query size The sets U and R have a maximum size of m and $2m$ respectively, so there are at most $\binom{|V(G)|}{m} \leq |V(G)|^m$ and $\binom{|V(G)|}{2m} \leq |V(G)|^{2m}$ possibilities for U and R respectively. The function f maps all subsets of U to subsets of Q with a maximum size of α, so there are at most $2^{|U|} \cdot \binom{|Q|}{\alpha} \leq 2^{2m} \cdot |V(G)|^\alpha$ possible functions f. From the definition of m and α on lines 1 and 2 it can be determined that $m \in \mathcal{O}(\max_{H \in \mathcal{F}} |V(H)|)$ and $\alpha \in \mathcal{O}(\max_{H \in \mathcal{F}} |V(H)|^2)$. It can now be seen that the total number of calls to `VCoracle` is bounded by $|V(G)|^{\mathcal{O}(\max_{H \in \mathcal{F}} |V(H)|^2)}$. Since $\mathcal{F}$ is fixed and `VCoracle` runs in polynomial time, this yields a polynomial bound on the running time of Algorithm 1.

The `VCoracle` subroutine (Lemma 4.4.5) is invoked on induced subgraphs G'' of G which therefore have a feedback vertex number of at most $\mathsf{fvs}(G)$. Hence Lemma 4.4.5 only queries the oracle for instances with $\mathsf{fvs}(G'')^{\mathcal{O}(1)} \leq \mathsf{fvs}(G)^{\mathcal{O}(1)}$ vertices. □

4.5 Conclusion

Earlier work [23, 76, 79, 121] has shown that several $\mathcal{F}$-Minor-Free Deletion problems admit polynomial kernelizations when parameterized by the feedback vertex number. In this chapter we showed that when $\mathcal{F}$ contains a forest and each

graph in $\mathcal{F}$ has a connected component of at least three vertices, the $\mathcal{F}$-Minor-Free Deletion and $\mathcal{F}$-Subgraph-Free Deletion problems do *not* admit such a polynomial kernel unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. This lower bound generalizes to any $\mathcal{F}$ where each graph has a connected component of at least three vertices, when we consider the vertex-deletion distance to treewidth $\min \mathrm{tw}(\mathcal{F})$ as parameter.

For all other choices of $\mathcal{F}$ we showed that a polynomial Turing kernelization exists for $\mathcal{F}$-Minor-Free Deletion and $\mathcal{F}$-Subgraph-Free Deletion parameterized by the feedback vertex number. The size of the Vertex Cover queries generated by the Turing kernelization does not depend on $\mathcal{F}$: the Turing kernelization can be shown to be *uniformly polynomial* (cf. [65]) by a further analysis of the polynomial-time reduction referred to in Step 3 of the procedure described in Lemma 4.4.5. However, it remains unknown whether the *running time* can be made uniformly polynomial, and whether the Turing kernelization can be improved to a traditional kernelization. Due to the large degree of the polynomial running time, the algorithm is mainly of theoretical interest.

In this chapter we discussed $\mathcal{F}$-Minor-Free Deletion and $\mathcal{F}$-Subgraph-Free Deletion. We leave open the case of $\mathcal{F}$-Induced-Subgraph-Free Deletion where a vertex set S is a solution for a graph G if $G - S$ does not contain any graph in $\mathcal{F}$ as *induced subgraph*. Although a number of our lower bound results also apply to this problem, the Turing kernelization we present cannot easily be generalized to $\mathcal{F}$-Induced-Subgraph-Free Deletion. This is mainly because we make use of a characterization of graphs that do not have a size-c matching. A similar characterization for graphs that do not have a size-c induced matching is unlikely to exist since finding a maximum induced matching is NP-complete while finding a maximum matching is not.

Our results leave open the possibility that all $\mathcal{F}$-Minor-Free Deletion problems admit a polynomial kernel when parameterized by the vertex-deletion distance to a *linear forest*, i.e., a collection of paths. Resolving this question may be an interesting direction for future work.

5

Finding Antler Structures to Solve Feedback Vertex Set

5.1 Introduction

In the previous chapters we have presented methods to reduce the task of solving large problem instances to solving problem instances that are small (bounded by a polynomial function of the parameter). In this chapter we look at preprocessing from a new perspective with the goal to open up a new research direction aimed at understanding the power of preprocessing in speeding up algorithms that solve NP-hard problems exactly [52, 70]. In a nutshell, this new perspective can be summarized as: how can an algorithm identify part of an optimal solution in an efficient preprocessing phase? We explore this question for the classic [87] Feedback Vertex Set problem on undirected graphs, leading to a new graph structure called *antler* which reveals vertices that belong to an optimal feedback vertex set.

We start by motivating the need for a new direction in the theoretical analysis of preprocessing. The use of preprocessing, often via the repeated application of reduction rules, has long been known [3, 4, 108] to speed up the solution of algorithmic tasks in practice. The notion of kernelization made it possible to also analyze the power of preprocessing theoretically. A substantial framework has been built around the definition of kernelization [35, 48, 56, 62, 70]. It includes

deep techniques for obtaining kernelization algorithms [21, 61, 89, 106], as well as tools for ruling out the existence of small kernelizations [22, 40, 49, 63, 72] under complexity-theoretic hypotheses. This body of work gives a good theoretical understanding of polynomial-time data compression for NP-hard problems.

However, we argue that these results on kernelization *do not* explain the often exponential speed-ups (e.g. [3], [6, Table 6]) caused by applying effective preprocessing steps to non-trivial algorithms. Why not? A kernelization algorithm guarantees that the input *size* is reduced to a function of the parameter; but the running time of modern parameterized algorithms for NP-hard problems is not exponential in the total input size. Instead, the running time of an FPT algorithm scales polynomially with the input size and is exponential only in the parameter. Hence an exponential speed-up of such algorithms cannot be explained by merely a decrease in input size, but only by a decrease in the *parameter*!

We therefore propose the following novel research direction: to investigate how preprocessing algorithms can decrease the parameter value of FPT algorithms, in a theoretically sound way. It is nontrivial to phrase meaningful formal questions in this direction. To illustrate this difficulty, note that strengthening the definition of kernelization to "a preprocessing algorithm that is guaranteed to always output an equivalent instance of the same problem with a strictly smaller parameter" is useless. Under minor technical assumptions, such an algorithm would allow the problem to be solved in polynomial time by repeatedly reducing the parameter, and solving the problem using an FPT or XP algorithm once the parameter value becomes constant. Hence NP-hard problems do not admit such parameter-decreasing algorithms. To formalize a meaningful line of inquiry, we take our inspiration from the Vertex Cover problem, the fruit fly of parameterized algorithms.

A rich body of theoretical and applied algorithmic research has been devoted to the exact solution of the Vertex Cover problem [6, 50, 73, 74]. A standard 2-way branching algorithm can test whether a graph G has a vertex cover of size k in time $\mathcal{O}(2^k(n+m))$, which can be improved by more sophisticated techniques [31]. The running time of the algorithm scales linearly with the input size, and exponentially with the size k of the desired solution. This running time suggests that to speed up the algorithm by a factor 1000, one either has to decrease the input size by a factor 1000, or decrease k by $\log_2(1000) \approx 10$.

It turns out that state-of-the-art preprocessing strategies for Vertex Cover indeed often *succeed* in decreasing the size of the solution that the follow-up algorithm has to find, by means of crown-reduction [2, 32, 51], or the intimately related Nemhauser-Trotter reduction based on the linear-programming relaxation [103]. Observe that if $H \subseteq V(G)$ is a set of vertices with the property that there exists a minimum vertex cover of G containing all of H, then G has a vertex cover of size k if and only if $G-H$ has a vertex cover of size $k-|H|$. Therefore, if a preprocessing algorithm can identify a set of vertices H which are guaranteed to belong to an optimal solution, then it can effectively reduce the parameter of the problem by restricting to a search for a solution of size $k-|H|$ in $G-H$.

A *crown decomposition* (cf. [1, 32, 51], [35, §2.3], [62, §4]) of a graph G serves exactly this purpose. It consists of two disjoint vertex sets $(\mathit{head}, \mathit{crown})$, such that *crown* is a non-empty independent set whose neighborhood is contained in *head*, and such that the graph $G[\mathit{head} \cup \mathit{crown}]$ has a matching M of size $|\mathit{head}|$. As *crown* is an independent set, the matching M assigns to each vertex of *head* a private neighbor in *crown*. It certifies that any vertex cover in G contains at least $|\mathit{head}|$ vertices from $\mathit{head} \cup \mathit{crown}$, and as *crown* is an independent set with $N_G(\mathit{crown}) \subseteq \mathit{head}$, a simple exchange argument shows there is indeed an optimal vertex cover in G containing all of *head* and none of *crown*. Since there is a polynomial-time algorithm to find a crown decomposition if one exists [2, Theorems 11–12], this yields the following preprocessing guarantee for Vertex Cover: if the input instance (G, k) has a crown decomposition $(\mathit{head}, \mathit{crown})$, then a polynomial-time algorithm can reduce the problem to an equivalent one with parameter at most $k - |\mathit{head}|$, thereby giving a formal guarantee on reduction in the parameter based on the structure of the input.[1]

As a first step towards preprocessing from the perspective of parameter reduction, we present a graph decomposition for Feedback Vertex Set which can identify vertices S that belong to an optimal solution and which therefore facilitate a reduction from finding a solution of size k in graph G, to finding a solution of size $k - |S|$ in $G - S$. While there has been a significant amount of work on kernelization for Feedback Vertex Set [23, 26, 76, 84, 121], the corresponding preprocessing algorithms do not succeed in finding vertices that belong to an optimal solution, other than those for which there is a self-loop or those which form the center of a flower (consisting of $k+1$ otherwise vertex-disjoint cycles [23, 26, 121], or a technical relaxation of this notion [76]). In particular, apart from the trivial self-loop rule, earlier preprocessing algorithms can only conclude a vertex v belongs to all optimal solutions (of a size k which must be given in advance) if they find a suitable packing of cycles witnessing that solutions without v must have size larger than k. In contrast, our argumentation will be based on *local* exchange arguments, which can be applied independently of the global solution size k, and can be used to detect vertices which are in some but not necessarily all optimal solutions.

We therefore introduce a new graph decomposition for preprocessing Feedback Vertex Set. To motivate it, we distill the essential features of a crown decomposition. Effectively, a crown decomposition of G certifies that G has a minimum vertex cover containing all of *head*, because (i) any vertex cover has to pick at least $|\mathit{head}|$ vertices from $\mathit{head} \cup \mathit{crown}$, as the matching M certifies

[1] The effect of the crown reduction rule can also be theoretically explained by the fact that interleaving basic 2-way branching with exhaustive crown reduction yields an algorithm whose running time is only exponential in the *gap* between the size of a minimum vertex cover and the cost of an optimal solution to its linear-programming relaxation [97]. However, this type of result cannot be generalized to Feedback Vertex Set since it is already NP-complete to determine whether there is a feedback vertex set whose size matches the cost of the linear-programming relaxation (Corollary 5.3.6).

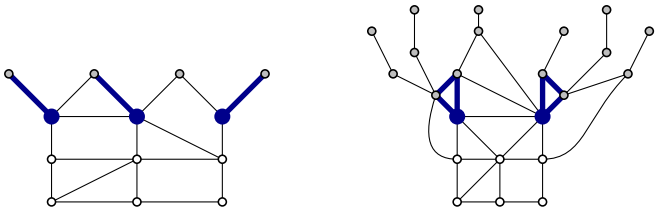


Figure 5.1 Graph structures showing there is an optimal solution containing all blue vertices and no gray vertices, certified by the blue subgraph. Left: Crown decomposition for Vertex Cover. Right: Antler for Feedback Vertex Set. For legibility, the number of edges in the drawing has been restricted. It therefore has vertices of degree at most 2, which makes the graph it reducible by standard reduction rules; but adding all possible edges between gray and blue vertices leads to a structure of minimum degree at least three which is still a 1-antler.

that $\mathsf{vc}(G[\mathit{head} \cup \mathit{crown}]) \geq |\mathit{head}|$, while (ii) any minimum vertex cover S' in $G - (\mathit{head} \cup \mathit{crown})$ yields a minimum vertex cover $S' \cup \mathit{head}$ in G, since $N_G(\mathit{crown}) \subseteq \mathit{head}$ and *crown* is an independent set. To obtain similar guarantees for Feedback Vertex Set, we need a decomposition to supply disjoint vertex sets $(\mathit{head}, \mathit{antler})$ such that

1. any minimum feedback vertex set in G contains at least $|\mathit{head}|$ vertices from $\mathit{head} \cup \mathit{antler}$, and
2. any minimum feedback vertex set S' in $G - (\mathit{head} \cup \mathit{antler})$ yields a minimum feedback vertex set $S' \cup \mathit{head}$ in G.

To achieve Condition 1, it suffices for $G[\mathit{head} \cup \mathit{antler}]$ to contain a set of $|\mathit{head}|$ vertex-disjoint cycles (implying that each cycle contains exactly one vertex of *head*); to achieve Condition 2, it suffices for $G[\mathit{antler}]$ to be acyclic, with each tree T of the forest $G[\mathit{antler}]$ connected to the remainder $V(G) \setminus (\mathit{head} \cup \mathit{antler})$ by at most one edge (implying that all cycles through *antler* intersect *head*). We call such a decomposition a 1-antler. Here *antler* refers to the shape of the forest $G[\mathit{antler}]$, which no longer consists of isolated spikes of a crown (see Figure 5.1). The prefix 1 indicates it is the simplest type of antler; we present a generalization later. An antler is *non-empty* if $\mathit{head} \cup \mathit{crown} \neq \emptyset$, and the *width* of the antler is defined to be $|\mathit{head}|$.

Unfortunately, assuming $\mathsf{P} \neq \mathsf{NP}$ there is no *polynomial-time* algorithm that always outputs a non-empty 1-antler if one exists. We prove this in Section 5.3.1. However, for the purpose of making a preprocessing algorithm that reduces the target solution size, we can allow FPT time in a parameter such as $|\mathit{head}|$ to find a decomposition. Each fixed choice of $|\mathit{head}|$ would then correspond to a reduction rule which identifies a small ($|\mathit{head}|$-sized) part of an optimal feedback vertex set, for which there is a simple certificate for it being part of an optimal solution. Such

a reduction rule can then be iterated in the preprocessing phase, thereby potentially decreasing the target solution size by an arbitrarily large amount. Hence we consider the parameterized complexity of testing whether a graph admits a non-empty 1-antler with $|head| \leq k$, parameterized by k. On the one hand, we show this problem to be W[1]-hard in Section 5.3.2. This hardness turns out to be a technicality based on the forced bound on $|head|$, though. We provide the following FPT algorithm which yields a search-space reducing preprocessing step.

Theorem 5.1.1. *There is an algorithm that runs in $2^{\mathcal{O}(k^5)} \cdot n^{\mathcal{O}(1)}$ time that, given a multigraph G on n vertices and integer k, either correctly determines that G does not admit a non-empty 1-antler of width at most k, or outputs a set S of at least k vertices such that there exists an optimal feedback vertex set in G containing all vertices of S.*

Hence if the input graph admits a non-empty 1-antler of width at most k, the algorithm is guaranteed to find at least k vertices that belong to an optimal feedback vertex set, thereby reducing the search space.

Based on this positive result, we go further and generalize our approach beyond 1-antlers. For a 1-antler $(head, antler)$ in G, the set of $|head|$ vertex-disjoint cycles in $G[head \cup antler]$ forms a very simple certificate that any feedback vertex set of G contains at least $|head|$ vertices from $head \cup antler$. We can generalize our approach to identify part of an optimal solution, by allowing more complex certificates of optimality. The following interpretation of a 1-antler is the basis of the generalization: for a 1-antler $(head, antler)$ in G, there is a subgraph G' of $G[head \cup antler]$ (formed by the $|head|$ vertex-disjoint cycles) such that $V(G') \supseteq head$ and $head$ is an optimal feedback vertex set of G'; and furthermore this subgraph G' is simple because all its connected components, being cycles, have a feedback vertex set of size 1. For an arbitrary integer z, we therefore define a z-antler in a multigraph G as a pair of disjoint vertex sets $(head, antler)$ such that

1. any minimum feedback vertex set in G contains at least $|head|$ vertices from $head \cup antler$, as witnessed by the fact that $G[head \cup antler]$ has a subgraph G' for which $head$ is an optimal feedback vertex set and with each component of G' having a feedback vertex set of size at most z; and
2. the graph $G[antler]$ is acyclic, with each tree T of the forest $G[antler]$ connected to the remainder $V(G) \setminus (head \cup antler)$ by at most one edge. This second condition is not changed compared to a 1-antler.

See Figure 5.2 for an example of a 3-antler of width 5. Our main result is the following.

Theorem 5.1.2. *There is an algorithm that runs in $2^{\mathcal{O}(k^5 z^2)} \cdot n^{\mathcal{O}(z)}$ time that, given a multigraph G on n vertices and integers $k \geq z \geq 0$, either correctly determines that G does not admit a non-empty z-antler of width at most k, or outputs a set S of at least k vertices such that there exists an optimal feedback vertex set in G containing all vertices of S.*

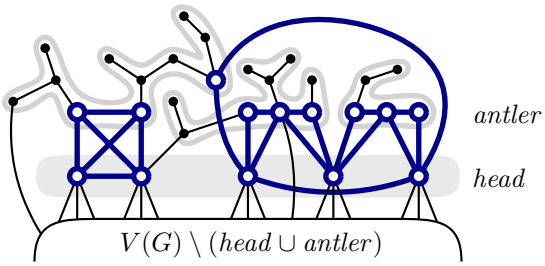


Figure 5.2 An example of a 3-antler. The subgraph G', marked in blue, has a feedback vertex number of 5 showing that any feedback vertex set of G contains at least $|head| = 5$ from $head \cup antler$. Each connected component of G' has a feedback vertex set of at most 3. The subgraph $G[antler]$ is acyclic and each of its connected components has at most one edge to $V(G) \setminus (head \cup antler)$.

In fact, we prove a slightly stronger statement. If a graph G can be reduced to a graph G' by iteratively removing z-antlers, each of width at most k, and the sum of the widths of this sequence of antlers is t, then we can find in time $f(k,z) \cdot n^{\mathcal{O}(z)}$ a subset of at least t vertices of G that belong to an optimal feedback vertex set. This implies that if a complete solution to Feedback Vertex Set can be assembled by iteratively combining $\mathcal{O}(1)$-antlers of width at most k, then the entire solution can be found in time $f'(k) \cdot n^{\mathcal{O}(1)}$. Hence our work uncovers a new parameterization in terms of the complexity of the solution structure, rather than its size, in which Feedback Vertex Set is fixed-parameter tractable.

Our algorithmic results are based on a combination of graph reduction and color coding. We use reduction steps inspired by the kernelization algorithms [23, 121] for Feedback Vertex Set to bound the size of *antler* in the size of *head*. After such reduction steps, we use color coding [7] to help identify antler structures. A significant amount of effort goes into proving that the reduction steps preserve antler structures and the optimal solution size.

5.2 Preliminaries

In this section we give a number of preliminary definitions. First we describe multigraphs and related notation, which are used extensively throughout this chapter. Unless mentioned otherwise, all graphs in this chapter are multigraphs. After the definitions of notation for multigraphs, we formally define antlers and related concepts.

Multigraphs Multigraphs are a generalization of a simple graphs in that they can have multiple edges with the same endpoints and edges with only one endpoint, called a *self-loop*. We formally model a multigraph using an edge-incidence

relation ι as follows: we define a *multigraph* G as a tuple consisting of a vertex set $V(G)$, an edge set $E(G)$, and a function $\iota\colon E(G) \to 2^{V(G)}$ where $\iota(e)$ is the set of one or two vertices incident to e for all $e \in E(G)$.

Most notation we use for multigraphs is similar to notation used for simple graphs. We formally define the relevant terms here. For a multigraph G and vertex set $S \subseteq V(G)$, let $G[S]$ denote the multigraph induced by S which consists of the vertex set S, the edge set $E' = \{e \in E(G) \mid \iota(e) \subseteq S\}$, and the function $\iota'\colon E' \to 2^S$ defined as $\iota'(e) = \iota(e)$ for all $e \in E'$. For a set of vertices and/or edges $X \subseteq V(G) \cup E(G)$ we define $G - X$ as the graph obtained from G after removing all vertices and edges in X, more formally, $G - X$ is the multigraph consisting of the vertex set $V(G) \setminus X$, the edge set $E' = \{e \in E(G) \setminus X \mid \iota(e) \subseteq V(G) \setminus X\}$, and the function $\iota'\colon E' \to 2^{V(G)\setminus S}$ defined as $\iota'(e) = \iota(e)$ for all $e \in E'$. If X is a singleton set consisting of a single vertex v or edge e we may write $G - v$ or $G - e$ as shorthand for $G - \{v\}$ or $G - \{e\}$.

The open neighborhood of a vertex v in a multigraph G, denoted $N_G(v)$, is the set of all vertices $u \neq v$ for which there is an edge between u and v, i.e., $N_G(v) = \{u \in V(G)\setminus\{v\} \mid \exists e \in E(G)\colon \{u, v\} = \iota(e)\}$. As in simple graphs, we define $N_G(S) = \bigcup_{v\in S} N_G(v)$ as the open neighborhood of a vertex set $S \subseteq V(G)$, the closed neighborhood of v is defined as $N_G[v] = N_G(v) \cup \{v\}$, and the closed neighborhood of a vertex set $S \subseteq V(G)$ is defined as $N_G[S] = N_G(S) \cup S$.

The *degree* $\deg_G(v)$ of a vertex v in a multigraph G is the number of edge-incidences to v where a self-loop contributes two edge-incidences. Formally, we define $\deg_G(v) = |\{e \in E(G) \mid v \in \iota(e)\}| + |\{e \in E(G) \mid \{v\} = \iota(e)\}|$. Note that the *degree-sum theorem*, $\sum_{v\in V(G)} \deg_G(v) = 2|E(G)|$, remains true in multigraphs. Like in simple graphs, we define $E_G(A, B)$ as the set of edges with one endpoint in the vertex set $A \subseteq V(G)$ and another in the vertex set $B \subseteq V(G)$, i.e., $E_G(A, B) = \{e \in E(G) \mid \iota(e) = \{a, b\} \text{ for some } a \in A, b \in B\}$. We define $e_G(A, B) = |E_G(A, B)|$.

As before with simple graphs, in all notation using the graph as subscript, we may omit the subscript if the graph is clear from the context, and to simplify the presentation, in expressions taking one or more vertex sets as parameter such as $N_G(..)$ and $E_G(.., ..)$, we sometimes use a subgraph H of G as argument as a shorthand for the vertex set $V(H)$ that is formally needed.

For two multigraphs G_1 and G_2, the graph $G_1 \cap G_2$ is the multigraph on vertex set $V' = V(G_1) \cap V(G_2)$, edge set $E' = E(G_1) \cap E(G_2)$, and function $\iota'\colon E' \to 2^{V'}$ defined as $\iota'(e) = \iota(e)$ for all $e \in E'$.

We assume the multigraphs are stored such that the number of edges between any two vertices can be retrieved and modified in constant time so that ensuring there are at most two edges between any two vertices (and hence $m \in \mathcal{O}(n^2)$) can be done without overhead in the asymptotic runtime of our algorithms.

Feedback vertex cuts and antlers We now introduce antlers and related structures. A *feedback vertex cut* (FVC) in a multigraph G is a pair of disjoint

vertex sets (C, F) such that $C, F \subseteq V(G)$, $G[F]$ is a forest, and for each tree T in $G[F]$ we have $e(T, G - (C \cup F)) \leq 1$. The *width* of a FVC (C, F) is $|C|$, and (C, F) is *empty* if $|C \cup F| = 0$.

Observation 5.2.1. *If (C, F) is a FVC in G then any cycle in G containing a vertex from F also contains a vertex from C. The set C is a FVS in $G[C \cup F]$, hence $|C| \geq \mathsf{fvs}(G[C \cup F])$.*

Observation 5.2.2. *If (C, F) is a FVC in G then for any $X \subseteq V(G)$ we have that $(C \setminus X, F \setminus X)$ is a FVC in $G - X$. Additionally, for any $Y \subseteq E(G)$ we have that (C, F) is a FVC in $G - Y$.*

We now present one of the main concepts for this work. An *antler* in a multigraph G is a FVC (C, F) in G such that $|C| \leq \mathsf{fvs}(G[C \cup F])$. Then by Observation 5.2.1 the set C is a minimum FVS in $G[C \cup F]$ and no cycle in $G - C$ contains a vertex from F. We observe:

Observation 5.2.3. *If (C, F) is an antler in G, then $\mathsf{fvs}(G) = |C| + \mathsf{fvs}(G - (C \cup F))$.*

For a multigraph G and vertex set $C \subseteq V(G)$, a *C-certificate* is a subgraph H of G such that C is a minimum FVS in H. We say a C-certificate has *order* z if for each component H' of H we have $\mathsf{fvs}(H') = |C \cap V(H')| \leq z$. For an integer $z \geq 0$, a *z-antler* in G is an antler (C, F) in G such that $G[C \cup F]$ contains a C-certificate of order z. Note that a 0-antler has width 0. In Figure 5.2 we depicted a feedback vertex cut $(\mathit{head}, \mathit{antler})$ of width 5 where $G[\mathit{head} \cup \mathit{antler}]$ contains a *head*-certificate of order 3 (marked in blue), hence $(\mathit{head}, \mathit{antler})$ is a 3-antler of width 5.

Observation 5.2.4. *If (C, F) is a z-antler in G for some $z \geq 0$, then for any $X \subseteq C$, we have that $(C \setminus X, F)$ is a z-antler in $G - X$. See Figure 5.3 for an example.*

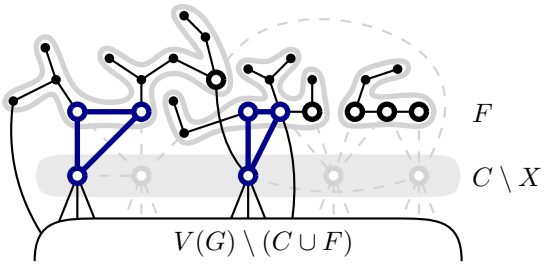


Figure 5.3 Consider the 3-antler $(\mathit{head}, \mathit{antler}) = (C, F)$ from Figure 5.2. The pair $(C \setminus X, F)$ remains a 3-antler after removing a subset $X \subseteq C$ from G.

5.3 Hardness results

To motivate the use of FPT algorithm to find antlers we present the hardness results presented in the introduction here. As these results apply to the simplest types of antlers this also forms an introduction to their properties. The hardness results presented in this section apply to the type of antlers as discussed in Section 5.1 consisting of two vertex sets *head* and *antler*. This type of antler is formally defined in Section 5.2 as 1-antlers, the simplest type of antler, consisting of the vertex sets C and F corresponding to *head* and *antler* respectively. For convenience we give a self contained definition of a 1-antler below.

Definition 5.3.1. A *1-antler* in an multigraph G is a pair of disjoint vertex sets (C, F) such that:

1. $G[F]$ is acyclic,
2. each tree T of the forest $G[F]$ is connected to $V(G) \setminus (C \cup F)$ by at most one edge, and
3. the subgraph $G[C \cup F]$ contains $|C|$ vertex-disjoint cycles.

A 1-antler is called *non-empty* if $H \cup C \neq \emptyset$.

Observe that the combination of Properties 1 and 2 is equivalent to stating that (C, F) is a FVC, and Property 3 describes the existence of a C-certificate of order 1 in $G[C \cup F]$. Will use the following easily verified consequence of this definition.

Observation 5.3.2. *If (C, F) is a 1-antler in an multigraph G, then for each vertex $c \in C$ the subgraph $G[\{c\} \cup F]$ contains a cycle.*

5.3.1 NP-hardness of finding 1-antlers

For the hardness results in this section we use a reduction from CNF-SAT. We can use the same construction as in Chapter 4 which is captured in Lemma 4.3.8. For convenience we give a simplified version of the lemma below, which is sufficient to show the results in this section.

Lemma 5.3.3 (Lemma 4.3.8 for $H = K_3$)**.** *There is a polynomial-time algorithm that, given a CNF formula Φ, outputs a graph G and a collection $\mathcal{H} = \{H_1, \ldots, H_\ell\}$ of vertex-disjoint cycles in G, such that Φ is satisfiable if and only if G has a feedback vertex set of size ℓ.*

We now show that finding non-empty 1-antlers is NP-hard.

Theorem 5.3.4. *Assuming $\mathsf{P} \neq \mathsf{NP}$, there is no polynomial-time algorithm that, given a simple graph G, outputs a non-empty 1-antler in G or concludes that no non-empty 1-antler exists.*

Proof. We need the following simple claim.

Claim 5.3.5. If G contains a packing of $\ell \geq 1$ vertex-disjoint cycles and a feedback vertex set of size ℓ, then G admits a non-empty 1-antler (C, F).

Proof. Let C be a feedback vertex set in G of size ℓ and let $F := V(G) \setminus C$. ⌟

Now suppose there is a polynomial-time algorithm to find a non-empty 1-antler decomposition, if one exists. We use it to solve CNF-SAT in polynomial time. Given an input formula Φ for CNF-SAT, use Lemma 5.3.3 to produce in polynomial time a graph G and a packing $\mathcal{H}$ of ℓ vertex-disjoint cycles in G, such that Φ is satisfiable if and only if $\mathsf{fvs}(G) = \ell$. The following recursive polynomial-time algorithm correctly tests, given a graph G and a packing $\mathcal{H}$ of some $\ell \geq 0$ vertex-disjoint cycles in G, whether $\mathsf{fvs}(G) = \ell$.

1. If $\ell = 0$, then output YES if and only if G is acyclic.
2. If $\ell > 0$, run the hypothetical algorithm to find a non-empty 1-antler (C, F).
   (a) If a non-empty 1-antler (C, F) is returned, then let $\mathcal{H}'$ consist of those cycles in the packing not intersected by C, and let $\ell' := |\mathcal{H}'|$. Return the result of recursively running the algorithm on $G' := G - (C \cup F)$ and $\mathcal{H}'$ to test whether G' has a feedback vertex set of size $|\mathcal{H}'|$.
   (b) Otherwise, return NO.

The claim shows that the algorithm is correct when it returns NO. Observation 5.2.3 shows that if we recurse, we have $\mathsf{fvs}(G) = \ell$ if and only if $\mathsf{fvs}(G - (C \cup H)) = \ell'$; hence the result of the recursion is the correct output. Since the number of vertices in the graph reduces by at least one in each iteration, the overall running time is polynomial assuming the hypothetical algorithm to compute a non-empty 1-antler. Hence using Lemma 5.3.3 we can decide CNF-SAT in polynomial time. □

Next, we show that determining whether the cost of a solution to the linear programming relaxation of Feedback Vertex Set on G is equal to the feedback vertex number of G is NP-complete.

Corollary 5.3.6. *For a simple graph G, determining whether $\mathsf{fvs}(G) = \mathsf{fvs}_{\mathsf{LP}}(G)$ is* NP*-complete. Here $\mathsf{fvs}_{\mathsf{LP}}(G)$ denotes the minimum cost of a solution to the linear programming relaxation of* Feedback Vertex Set *on G.*

Proof. Membership in NP is trivial; we prove hardness. Suppose such an algorithm exists. As above, we use it to solve CNF-SAT in polynomial time. Given an input Φ for CNF-SAT, use Lemma 5.3.3 to produce in polynomial time a graph G and packing $\mathcal{H}$ of ℓ vertex-disjoint cycles in G, such that Φ is satisfiable if and only if $\mathsf{fvs}(G) = \ell$.

Compute the cost c of an optimal solution to the linear programming relaxation of Feedback Vertex Set on G, using the ellipsoid method. By the properties

of a relaxation, if $c > \ell$ then $\mathsf{fvs}(G) > \ell$, and hence we can safely report that Φ is unsatisfiable. If $c \leq \ell$, then the existence of ℓ vertex-disjoint cycles in G implies that $c = \ell$. Run the hypothetical algorithm to test whether $\mathsf{fvs}(G) = \mathsf{fvs}_{\mathsf{LP}}(G)$. If the answer is YES, then G has a feedback vertex set of size ℓ and hence Φ is satisfiable; if not, then Φ is unsatisfiable. □

5.3.2 W[1]-hardness of finding bounded-width 1-antlers

We consider the following parameterized problem.

BOUNDED-WIDTH 1-ANTLER DETECTION
Input: A multigraph G and an integer k.
Parameter: k.
Question: Does G admit a non-empty 1-antler (C, F) with $|C| \leq k$?

We prove that BOUNDED-WIDTH 1-ANTLER DETECTION is W[1]-hard by a reduction from MULTICOLORED CLIQUE, which is defined as follows.

MULTICOLORED CLIQUE
Input: A simple graph G, an integer k, and a partition of $V(G)$ into sets $V_1, \ldots, V_k$.
Parameter: k.
Question: Is there a clique S in G such that for each $1 \leq i \leq k$ we have $|S \cap V_i| = 1$?

The sets V_i are referred to as *color classes*, and a solution clique S is called a *multicolored clique*. It is well-known that MULTICOLORED CLIQUE is W[1]-hard (cf. [35, Theorem 13.25]). Our reduction is inspired by the W[1]-hardness of detecting a Hall set [35, Exercise 13.28].

Theorem 5.3.7. BOUNDED-WIDTH 1-ANTLER DETECTION *is* W[1]*-hard.*

Proof. We give a parameterized reduction (Definition 2.4.1) from the MULTICOLORED CLIQUE problem. By inserting isolated vertices if needed, we may assume without loss of generality that for the input instance $(G, k, V_1, \ldots, V_k)$ we have $|V_1| = |V_2| = \ldots = |V_k| = n$ for some $n \geq k(k-1) + 4$. For each $i \in [k]$, fix an arbitrary labeling of the vertices in V_i as $v_{i,j}$ for $j \in [n]$. Given this instance, we construct an input (G', k') for BOUNDED-WIDTH 1-ANTLER DETECTION as follows.

1. For each $i \in [k]$, for each $j \in [n]$, create a set $U_{i,j} = \{u_{i,j,\ell} \mid \ell \in [k] \setminus \{i\}\}$ of vertices in G' to represent $v_{i,j}$. Intuitively, vertex $u_{i,j,\ell}$ represents the connection that the jth vertex from the ith color class should have to the neighbor in the ℓth color class chosen in the solution clique.

2. Define $\mathcal{U} := \bigcup_{i \in [k]} \bigcup_{j \in [n]} U_{i,j}$. Insert (single) edges to turn $\mathcal{U}$ into a clique in G'.

3. For each edge e in G between vertices of different color classes, let $e = \{v_{i,j}, v_{i',j'}\}$ with $i < i'$, and insert two vertices into G' to represent e:
   - Insert a vertex w_e, add an edge from w_e to each vertex in $U_{i,j} \cup U_{i',j'}$, and then add a second edge between w_e and $u_{i,j,i'}$.
   - Insert a vertex $w_{e'}$, add an edge from $w_{e'}$ to each vertex in $U_{i,j} \cup U_{i',j'}$, and then add a second edge between $w_{e'}$ and $u_{i',j',i}$.

   Let W denote the set of vertices of the form $w_e, w_{e'}$ inserted to represent an edge of G. Observe that W is an independent set in G.
4. Finally, insert a vertex u^* into G'. Add a single edge from u^* to all other vertices of G', to make u^* into a universal vertex.

This concludes the construction of G'. Note that G' contains double-edges, but no self-loops. We set $k' := k(k-1)$, which is appropriately bounded for a parameterized reduction. It is easy to see that the reduction can be performed in polynomial time. It remains to show that G has a multicolored k-clique if and only if G' has a non-empty 1-antler of width at most k. To illustrate the intended behavior of the reduction, we first prove the forward implication.

Claim 5.3.8. If G has a multicolored clique of size k, then G' has a non-empty 1-antler of width k'.

Proof. Suppose S is a multicolored clique of size k in G. Choose indices $j_1, \ldots, j_k$ such that $S \cap V_i = \{v_{i,j_i}\}$ for all $i \in [k]$. Define a 1-antler (C, F) in G' as follows:

$$C = \bigcup_{i \in [k]} U_{i,j_i},$$

$$F = \{w_e, w_{e'} \mid e \text{ is an edge in } G \text{ between distinct vertices of } S\}.$$

Since each set U_{i,j_i} has size $k-1$, it follows that $|C| = k(k-1) = k'$. Since $F \subseteq W$ is an independent set in G', it also follows that $G'[F]$ is acyclic. Each tree T in the forest $G'[F]$ consists of a single vertex w_e or $w_{e'}$. By construction, there is exactly one edge between T and $V(G') \setminus (C \cup F)$; this is the edge to the universal vertex u^*. It remains to verify that $G'[C \cup F]$ contains $|C|$ vertex-disjoint cycles, each containing exactly one vertex of C. Consider an arbitrary vertex $u_{i,j_i,\ell}$ in C; we show we can assign it a cycle in $G'[C \cup F]$ so that all assigned cycles are vertex-disjoint. Since S is a clique, there is an edge e in G between v_{i,j_i} and v_{ℓ,j_ℓ}, and the corresponding vertices $w_e, w_{e'}$ are in F. If $i < \ell$, then $w_e \in F$ and there are two edges between $u_{i,j_i,\ell}$ and w_e, forming a cycle on two vertices. If $i > \ell$, then there is a cycle on two vertices $u_{i,j_i,\ell}$ and $w_{e'}$. Since for any vertex of the form w_e or $w_{e'}$ there is a unique vertex of C that it has a double-edge to, the resulting cycles are indeed vertex-disjoint. This proves that (C, F) is a 1-antler of width k'. ⌟

Before proving reverse implication, we establish some structural claims about the structure of 1-antlers in G'.

Claim 5.3.9. If (C, F) is a non-empty 1-antler in G' with $|C| \leq k'$, then all of the following conditions holds:

1. $\mathcal{U} \cap F = \emptyset$.
2. $u^* \notin C \cup F$.
3. $W \cap C = \emptyset$.
4. $C \subseteq \mathcal{U}$, $F \subseteq W$, and each tree of the forest $G'[F]$ consists of a single vertex.
5. For each vertex $w \in F$ we have $N_{G'}(w) \cap \mathcal{U} \subseteq C$.
6. $F \neq \emptyset$.

Proof. Condition 1: Assume for a contradiction that there is a vertex $u_{i,j,\ell} \in \mathcal{U} \cap F$. Since $G'[F]$ is a forest by Property 1 of Definition 5.3.1, while $\mathcal{U}$ is a clique in G', it follows that $|F \cap \mathcal{U}| \leq 2$. By Property 2, for a vertex in F, there is at most one of its neighbors that belongs to neither F nor C. Since $|\mathcal{U}| \geq n \geq k(k-1)+4$, and $u_{i,j,\ell} \in F$ is adjacent to all other vertices of $\mathcal{U}$ since that set forms a clique, the fact that $|F \cap \mathcal{U}| \leq 2$ implies that $|C \cap \mathcal{U}| \geq |\mathcal{U}| - 2 - 1 \geq k(k-1)+4-3 > k'$. So $|C| > k'$, which contradicts that (C, F) is a 1-antler with $|C| \leq k'$.

Condition 2: Since u^* is a universal vertex in G', the set $\mathcal{U} \cup \{u^*\}$ is a clique and the preceding argument shows that $u^* \notin F$. To prove the claim we show that additionally, $u^* \notin C$. Assume for a contradiction that $u^* \in C$. By Observation 5.3.2, the graph $G'[\{u^*\} \cup F]$ contains a cycle. Since W is an independent set in G' and u^* is not incident on any double-edges, the graph $G'[\{u^*\} \cup W]$ is acyclic. Hence to get a cycle in $G'[\{u^*\} \cup F]$, the set F contains at least one vertex that is not in W and not u^*; hence this vertex belongs to $\mathcal{U}$. So $\mathcal{U} \cap F \neq \emptyset$; but this contradicts Condition 1.

Condition 3: Assume for a contradiction that $w \in W \cap C$. Again by Observation 5.3.2, there is a cycle in $G'[\{w\} \cup F]$, and since G' does not have any self-loops this implies $N_{G'}(w) \cap F \neq \emptyset$. But by construction of G' we have $N_{G'}(w) \subseteq \mathcal{U} \cup \{u^*\}$, so F contains a vertex of either $\mathcal{U}$ or u^*. But this contradict either Condition 1 or Condition 2.

Condition 4: Since the sets $\mathcal{U}, W, \{u^*\}$ form a partition of $V(G')$, the preceding claims imply $C \subseteq \mathcal{U}$ and $F \subseteq W$. Since W is an independent set in G', this implies that each tree T of the forest $G'[W]$ consists of a single vertex.

Condition 5: Consider a vertex $w \in F$, which by itself forms a tree in $G'[F]$. Since $u^* \notin C \cup F$, the edge between T and u^* is the unique edge connecting T to a vertex of $V(G') \setminus (C \cup F)$, and therefore all neighbors of T other than u^* belong to $C \cup F$. Since a vertex $w \in W$ has $N_{G'}(w) \subseteq \mathcal{U} \cup \{u^*\}$, it follows that $N_{G'}(w) \cap \mathcal{U} \subseteq C$.

Condition 6: By the assumption that (C, F) is non-empty, we have $C \cup F \neq \emptyset$. This implies that $F \neq \emptyset$: if C would contain a vertex c while $F = \emptyset$, then by Observation 5.3.2 the graph $G'[\{c\} \cup F] = G'[\{c\}]$ would contain a cycle, which is not the case since G' has no self-loops. Hence $F \neq \emptyset$. ⌟

With these structural insights, we can prove the remaining implication.

Claim 5.3.10. If G' has a non-empty 1-antler (C, F) with $|C| \leq k'$, then G has a multicolored clique of size k.

Proof. Let (C, F) be a non-empty 1-antler in G' with $|C| \leq k'$. By Claim 5.3.9 we have $C \subseteq \mathcal{U}$, while $F \subseteq W$ and $F \neq \emptyset$. Consider a fixed vertex $w \in F$. Since $F \subseteq W$, vertex w is of the form w_e or $w_{e'}$ constructed in Step 3 to represent some edge e of G. Choose $i^* \in [k], j^* \in [n]$ such that $v_{i^*,j^*} \in V_{i^*}$ is an endpoint of edge e in G. By construction we have $U_{i^*,j^*} \subseteq N_{G'}(w)$ and therefore Claim 5.3.9(5) implies $U_{i^*,j^*} \subseteq C$.

Consider an arbitrary $\ell \in [k] \setminus \{i^*\}$. Then $u_{i^*,j^*,\ell} \in U_{i^*,j^*} \subseteq C$, so by Observation 5.3.2 the graph $G'[\{u_{i^*,j^*,\ell}\} \cup F]$ contains a cycle. Since $G'[F]$ is an independent set and G' has no self-loops, this cycle consists of two vertices joined by a double-edge. By construction of G', such a cycle involving $u_{i^*,j^*,\ell}$ exists only through vertices w_e or $w_{e'}$ where e is an edge of G connecting v_{i^*,j^*} to a neighbor in class V_ℓ. Consequently, F contains a vertex w that represents such an edge e. Let v_{ℓ,j_ℓ} denote the other endpoint of e. Then $N_{G'}(w) \supseteq U_{i^*,j^*} \cup U_{\ell,j_\ell}$, and by Claim 5.3.9(5) we therefore have $U_{\ell,j_\ell} \subseteq C$.

Applying the previous argument for all $\ell \in [k] \setminus \{i^*\}$, together with the fact that $U_{i^*,j^*} \subseteq C$, we find that for each $i \in [k]$ there exists a value j_i such that $U_{i,j_i} \subseteq C$. Since $|C| \leq k(k-1)$ while each such set U_{i,j_i} has size $k-1$, it follows that the choice of j_i is uniquely determined for each $i \in [k]$, and that there are no other vertices in C. To complete the proof, we argue that the set $S = \{v_{i,j_i} \mid i \in [k]\}$ is a clique in G.

Consider an arbitrary pair of distinct vertices $v_{i,j_i}, v_{i',j_{i'}}$ in S, and choose the indices such that $i < i'$. We argue that G contains an edge e between these vertices, as follows. Since $u_{i,j_i,i'} \in U_{i,j_i} \subseteq C$, by Observation 5.3.2 the graph $G'[\{u_{i,j_i,i'}\} \cup F]$ contains a cycle. As argued above, the construction of G' and the fact that $F \subseteq W$ ensure that this cycle consists of $u_{i,j_i,i'}$ joined to a vertex in F by a double-edge. By Step 3 and the fact that $i < i'$, this vertex is of the form w_e for an edge e in G connecting v_{i,j_i} to a vertex $v_{i',j'}$ in $V_{i'}$. By construction of G' we have $U_{i',j'} \subseteq N_{G'}(w_e)$, and then $w_e \in F$ implies by Claim 5.3.9(5) that $U_{i',j'} \subseteq C$. Since we argued above that for index i' there is a unique choice $j_{i'}$ with $U_{i',j_{i'}} \subseteq C$, we must have $j' = j_{i'}$. Hence the vertex w_e contained in F represents the edge of G between v_{i,j_i} and $v_{i',j_{i'}}$ in G, which proves in particular that the edge exists. As the choice of vertices was arbitrary, this shows that S is a clique in G. As it contains exactly one vertex from each color class, graph G has a multicolored clique of size k. ⌟

Claims 5.3.8 and 5.3.10 show that the instance (G, k) of Multicolored Clique is equivalent to the instance (G', k') of Bounded-Width 1-Antler Detection. This concludes the proof of Theorem 5.3.7. □

Observe that the proof of Theorem 5.3.7 shows that the variant of Bounded-Width 1-Antler Detection where we ask for the existence of a 1-antler of width *exactly* k, is also W[1]-hard.

5.4 Structural properties of antlers

In this section we give a number of structural properties of antlers. While antlers may intersect in non-trivial ways, the following proposition relates the sizes of the cross-intersections.

Proposition 5.4.1. *If (C_1, F_1) and (C_2, F_2) are antlers in G, then $|C_1 \cap F_2| = |C_2 \cap F_1|$.*

Proof. We show $\mathsf{fvs}(G[F_1 \cup F_2]) = |C_1 \cap F_2|$. First we show $\mathsf{fvs}(G[F_1 \cup F_2]) \geq |C_1 \cap F_2|$ by showing $(C_1 \cap F_2, F_1)$ is an antler in $G[F_1 \cup F_2]$. Clearly (C_1, F_1) is an antler in $G[F_1 \cup F_2 \cup C_1]$, so then by Observation 5.2.4 $(C_1 \cap F_2, F_1)$ is an antler in $G[F_1 \cup F_2 \cup C_1] - (C_1 \setminus F_2) = G[F_1 \cup F_2]$.

Second we show $\mathsf{fvs}(G[F_1 \cup F_2]) \leq |C_1 \cap F_2|$ by showing $G[F_1 \cup F_2] - (C_1 \cap F_2)$ is acyclic. Note that $G[F_1 \cup F_2] - (C_1 \cap F_2) = G[F_1 \cup F_2] - C_1$. Suppose $G[F_1 \cup F_2] - C_1$ contains a cycle. We know this cycle does not contain a vertex from C_1, however it does contain at least one vertex from F_1 since otherwise this cycle exists in $G[F_2]$ which is a forest. We know from Observation 5.2.1 that any cycle in G containing a vertex from F_1 also contains a vertex from C_1. Contradiction. The proof for $\mathsf{fvs}(G[F_1 \cup F_2]) = |C_2 \cap F_1|$ is symmetric. It follows that $|C_1 \cap F_2| = \mathsf{fvs}(G[F_1 \cup F_2]) = |C_2 \cup F_1|$. □

Lemma 5.4.2 shows that what remains of a z-antler (C_1, F_1) when removing a different antler (C_2, F_2), again forms a smaller z-antler. We will rely on this lemma repeatedly to ensure that after having found and removed an incomplete fragment of a width-k z-antler, the remainder of that antler persists as a z-antler to be found later.

Lemma 5.4.2. *For any integer $z \geq 0$, if a multigraph G has a z-antler (C_1, F_1) and another antler (C_2, F_2), then $(C_1 \setminus (C_2 \cup F_2), F_1 \setminus (C_2 \cup F_2))$ is a z-antler in $G - (C_2 \cup F_2)$.*

Before we prove Lemma 5.4.2, we prove a weaker claim:

Proposition 5.4.3. *If (C_1, F_1) and (C_2, F_2) are antlers in G, then $(C_1 \setminus (C_2 \cup F_2), F_1 \setminus (C_2 \cup F_2))$ is an antler in $G - (C_2 \cup F_2)$.*

Proof. For brevity let $C_1' := C_1 \setminus (C_2 \cup F_2)$ and $F_1' := F_1 \setminus (C_2 \cup F_2)$ and $G' := G - (C_2 \cup F_2)$ (see Figure 5.4). First note that (C_1', F_1') is a FVC in G' by Observation 5.2.2. We proceed to show that $\mathsf{fvs}(G'[C_1' \cup F_1']) \geq |C_1'|$. By Observation 5.2.4 $(\emptyset, F_2)$ is an antler in $G - C_2$, so then by Observation 5.2.2 we have $(\emptyset, F_2 \cap (C_1 \cup F_1))$ is a FVC in $G[C_1 \cup F_1] - C_2$. Since a FVC of width 0

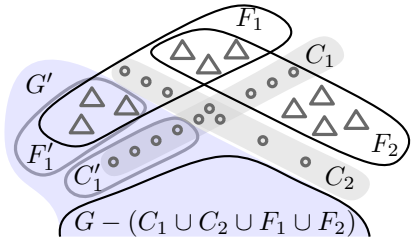


Figure 5.4 A diagram of the different vertex sets used in the proof of Proposition 5.4.3. The triangles represent induced trees.

is an antler we can apply Observation 5.2.3 and obtain $\mathsf{fvs}(G[C_1 \cup F_1] - C_2) = \mathsf{fvs}(G[C_1 \cup F_1] - (C_2 \cup F_2)) = \mathsf{fvs}(G'[C'_1 \cup F'_1])$. We derive

$$\begin{aligned}
\mathsf{fvs}(G'[C'_1 \cup F'_1]) &= \mathsf{fvs}(G[C_1 \cup F_1] - C_2) \\
&\geq \mathsf{fvs}(G[C_1 \cup F_1]) - |C_2 \cap (C_1 \cup F_1)| \\
&= |C_1| - |C_2 \cap C_1| - |C_2 \cap F_1| && \text{Since } C_1 \cap F_1 = \emptyset \\
&= |C_1| - |C_2 \cap C_1| - |C_1 \cap F_2| && \text{By Proposition 5.4.1} \\
&= |C_1| - |(C_2 \cap C_1) \cup (C_1 \cap F_2)| && \text{Since } C_2 \cap F_2 = \emptyset \\
&= |C_1 \setminus (C_2 \cup F_2)| = |C'_1|. && \square
\end{aligned}$$

We can now prove Lemma 5.4.2.

Proof. For brevity let $C'_1 := C_1 \setminus (C_2 \cup F_2)$ and $F'_1 := F_1 \setminus (C_2 \cup F_2)$ and $G' := G - (C_2 \cup F_2)$. By Proposition 5.4.3 we know (C'_1, F'_1) is an antler, so it remains to show that $G'[C'_1 \cup F'_1]$ contains a C'_1-certificate of order z. Since (C_1, F_1) is a z-antler in G, we have that $G[C_1 \cup F_1]$ contains a C_1-certificate of order z. Let H denote this C_1-certificate and let $\overline{H}$ be the set of all edges and vertices in $G'[C'_1 \cup F'_1]$ that are not in H. Now (C_1, F_1) is a z-antler in $G'' := G - \overline{H}$ since it is a FVC by Observation 5.2.2 and $G''[C_1 \cup F_1]$ contains a C_1-certificate of order z since H is a subgraph of G''. Note that (C_2, F_2) is also an antler in G'' since $\overline{H}$ does not contain vertices or edges from $G[C_2 \cup F_2]$. It follows that (C'_1, F'_1) is an antler in G'' by Proposition 5.4.3, so $G''[C'_1 \cup F'_1]$ is a C'_1-certificate in G''. Clearly this is a C'_1-certificate of order z since $G''[C'_1 \cup F'_1]$ is a subgraph of H. Since $G''[C'_1 \cup F'_1]$ is a subgraph of $G'[C'_1 \cup F'_1]$ it follows that $G'[C'_1 \cup F'_1]$ contains a C'_1-certificate of order z. $\square$

Lemma 5.4.4 shows that we can consider consecutive removal of two z-antlers as the removal of a single z-antler.

Lemma 5.4.4. *For any integer $z \geq 0$, if a multigraph G has a z-antler (C_1, F_1) and $G - (C_1 \cup F_1)$ has a z-antler (C_2, F_2) then $(C_1 \cup C_2, F_1 \cup F_2)$ is a z-antler in G.*

Proof. Since (C_1, F_1) is a z-antler in G we know $G[C_1 \cup F_1]$ contains a C_1-certificate of order z, similarly $(G - (C_1 \cup F_1))[C_2 \cup F_2]$ contains a C_2-certificate of order z. The union of these certificates forms a $(C_1 \cup C_2)$-certificate of order z in $G[C_1 \cup C_2 \cup F_1 \cup F_2]$. It remains to show that $(C_1 \cup C_2, F_1 \cup F_2)$ is a FVC in G.

First we show $G[F_1 \cup F_2]$ is acyclic. Suppose for contradiction that $G[F_1 \cup F_2]$ contains a cycle $\mathcal{C}$. Since (C_1, F_1) is a FVC in G, any cycle containing a vertex from F_1 also contains a vertex from C_1, hence $\mathcal{C}$ does not contain vertices from F_1. Therefore $\mathcal{C}$ can only contain vertices from F_2. This is a contradiction with the fact that $G[F_2]$ is acyclic.

Finally we show that for each tree T in $G[F_1 \cup F_2]$ we have $e(T, G - (C_1 \cup C_2 \cup F_1 \cup F_2)) \leq 1$. If $V(T) \subseteq F_2$ this follows directly from the fact that (C_2, F_2) is a FVC in $G - (C_1 \cup F_1)$. Similarly if $V(T) \subseteq F_1$ this follows directly from the fact that (C_1, F_1) is a FVC in G. So suppose T is a tree that contains vertices from both F_1 and F_2. Since T is connected, each tree in $T[F_1]$ contains a neighbor of a vertex in a tree in $T[F_2]$. Hence no tree in $T[F_1]$ contains a neighbor of $V(G - (C_1 \cup C_2 \cup F_1 \cup F_2))$, so $e(V(T) \cap F_1, G - (C_1 \cup C_2 \cup F_1 \cup F_2)) = 0$. To complete the proof we show $e(V(T) \cap F_2, G - (C_1 \cup C_2 \cup F_1 \cup F_2)) \leq 1$. Recall each tree in $G[F_2]$ has at most 1 edge to $G - (C_1 \cup C_2 \cup F_1 \cup F_2)$, so it suffices to show that $T[F_2]$ is connected. Suppose $T[F_2]$ is not connected, then let $u, v \in F_2$ be vertices from different components of $T[F_2]$. Since T is connected, there is a path from u to v. This path must use a vertex $w \in V(T - F_2) \subseteq F_1$. Let T' denote the tree in $T[F_1]$ that contains this vertex. Since (C_1, F_1) is a FVC in G we have that $e(T', F_2) \leq e(T', G - (C_1 \cup F_1)) \leq 1$ hence no vertex in T' can be part of a path from u to v in T. This contradicts our choice of T'. □

The last structural property of antlers, given in Lemma 5.4.6, derives a bound on the number of trees of a forest $G[F]$ needed to witness that C is an optimal FVS of $G[C \cup F]$. Lemma 5.4.6 is a corollary to the following lemma.

Lemma 5.4.5. *If a multigraph G contains a C-certificate H of order $z \geq 0$ for some $C \subseteq V(G)$, then H contains a C-certificate $\hat{H}$ of order z such that $\hat{H} - C$ has at most $\frac{|C|}{2}(z^2 + 4z - 1)$ trees.*

Proof. Consider a tree T in $H - C$, we show that $\mathsf{fvs}(H - V(T)) = \mathsf{fvs}(H)$ if

1. for all $v \in C$ such that $H[V(T) \cup \{v\}]$ has a cycle, $H - V(T)$ contains k cycles whose vertex sets only intersect in v, and
2. for all $\{u, v\} \in \binom{N_H(T)}{2}$ there are at least $z+1$ other trees in $H - C$ adjacent to u and v.

Consider the component H' of H containing T. It suffices to show that $\mathsf{fvs}(H' - V(T)) = \mathsf{fvs}(H')$. Clearly $\mathsf{fvs}(H' - V(T)) \leq \mathsf{fvs}(H')$ so it remains to show that $\mathsf{fvs}(H' - V(T)) \geq \mathsf{fvs}(H')$. Assume $\mathsf{fvs}(H' - V(T)) < \mathsf{fvs}(H')$, then let X be a FVS in $H' - V(T)$ with $|X| < \mathsf{fvs}(H') = |C \cap V(H')| \leq z$. For any $v \in C \cap V(H')$ such that $H[V(T) \cup \{v\}]$ has a cycle we know from Condition 1 that $H' - V(T)$

has $z > |X|$ cycles that intersect only in v, hence $v \in X$. By Condition 2 we have that all but possibly one vertex in $N_G(T)$ must be contained in X, since if there are two vertices $x, y \in N_G(T) \setminus X$ then $H - V(T) - X$ has at least $z+1-|X| \geq 2$ internally vertex-disjoint paths between x and y forming a cycle and contradicting our choice of X. Since there is at most one vertex $v \in N_G(T) \setminus X$ and $H[T \cup \{v\}]$ does not have a cycle, we have that $H' - X$ is acyclic, a contradiction since $|X| < \mathsf{fvs}(H')$.

The desired C-certificate $\hat{H}$ can be obtained from H by iteratively removing trees from $H - C$ for which both conditions hold. We show that if no such tree exists, then $H - C$ has at most $\frac{|C|}{2}(z^2 + 2z - 1)$ trees. Each tree T for which Condition 1 fails can be charged to a vertex $v \in C$ that witnesses this, i.e., $H[T \cup \{v\}]$ has a cycle and there are at most z trees T' such that $T' \cup v$ has a cycle. Clearly each vertex $v \in C$ can be charged at most z times, hence there are at most $z \cdot |C|$ trees violating Condition 1. Similarly each tree T for which Condition 2 fails can be charged to a pair of vertices $\{u, v\} \in \binom{N_H(T)}{2}$ for which at most $z + 1$ trees in $H - T$ are adjacent to u and v. Clearly each pair of vertices can be charged at most $z + 1$ times. Additionally each pair consists of vertices from the same component of H. Let $H_1, \ldots, H_\ell$ be the components in H, then the number of such pairs is at most

$$\begin{aligned}\sum_{1 \leq i \leq \ell} \binom{|C \cap V(H_i)|}{2} &= \sum_{1 \leq i \leq \ell} \frac{1}{2}|C \cap V(H_i)|(|C \cap V(H_i)| - 1) \\ &\leq \sum_{1 \leq i \leq \ell} \frac{1}{2}|C \cap V(H_i)|(z - 1) \\ &= \frac{|C|}{2}(z - 1)\end{aligned}$$

Thus $H - C$ has at most $z \cdot |C| + (z + 1) \cdot \frac{|C|}{2}(z - 1) = \frac{|C|}{2}(z^2 + 2z - 1)$ trees violating Condition 2.

To conclude, there are at most $z \cdot |C| + \frac{|C|}{2}(z^2 + 2z - 1)$ trees of $H - C$ that violate one or both conditions and any tree in $H - C$ that satisfies both conditions can be excluded from $\hat{H}$. We obtain a final bound on the number of trees in $\hat{H}$ of $z \cdot |C| + \frac{|C|}{2}(z^2 + 2z - 1) = \frac{|C|}{2}(z^2 + 4z - 1)$. □

We can now give an upper bound on the number of trees in $G[F]$ required for a z-antler (C, F).

Lemma 5.4.6. *Let (C, F) be a z-antler in a multigraph G for some $z \geq 0$. There exists an $F' \subseteq F$ such that (C, F') is a z-antler in G and $G[F']$ has at most $\frac{|C|}{2}(z^2 + 4z - 1)$ trees.*

Proof. Since (C, F) is a z-antler, $G[C \cup F]$ contains a C-certificate H or order z and by Lemma 5.4.5 we know H contains a C-certificate $\hat{H}$ of order z such that $\hat{H} - C$ has at most $\frac{|C|}{2}(z^2 + 4z - 1)$ components. Take $F' := V(H - C)$ then $G[F']$ has at

most $\frac{|C|}{2}(z^2+4z-1)$ components and $\hat{H}$ is a subgraph of $G[C \cup F']$, meaning (C, F') is a z-antler. □

5.5 Finding antlers

5.5.1 Finding feedback vertex cuts

As described in Section 5.1, our algorithm aims to identify vertices in antlers uses color coding. To allow a relatively small family of colorings to identify an entire antler structure (C, F) with $|C| \leq k$, we need to bound $|F|$ in terms of k as well. We therefore use several graph reduction steps. In this section, we show that if there is a width-k antler whose forest F is significantly larger than k, then we can identify a reducible structure in the graph. To identify a reducible structure we will also use color coding. In Section 5.5.2 we show how to reduce such a structure while preserving antlers and optimal feedback vertex sets.

Define the function $f_r \colon \mathbb{N} \to \mathbb{N}$ as $f_r(x) = 2x^3 + 3x^2 - x$. We say a FVC (C, F) is *reducible* if $|F| > f_r(|C|)$, and we say (C, F) is a *single-tree* FVC if $G[F]$ is connected.

Definition 5.5.1. A FVC (C, F) is *simple* if $|F| \leq 2f_r(|C|)$ and one of the following holds: (a) $G[F]$ is connected, or (b) all trees in $G[F]$ have a common neighbor v and there exists a single-tree FVC (C, F_2) with $v \in F_2 \setminus F$ and $F \subseteq F_2$.

The algorithm we will present can identify a reducible FVC when it is also simple. First we show that such a FVC always exists when the graph contains a single-tree reducible FVS.

Lemma 5.5.2. *If a multigraph G contains a reducible single-tree FVC (C, F) there exists a simple reducible FVC (C, F') with $F' \subseteq F$.*

Proof. We use induction on $|F|$. If $|F| \leq 2f_r(|C|)$ then (C, F) is simple by condition (a). Assume $|F| > 2f_r(|C|)$. Since (C, F) is a FVC and $G[F]$ is connected there is at most one vertex $v \in F$ that has a neighbor in $V(G) \setminus (C \cup F)$. If no such vertex exists, take $v \in F$ to be any other vertex. Observe that $(C, F \setminus \{v\})$ is a FVC. Consider the following cases:

- All trees in $G[F] - v$ contain at most $f_r(|C|)$ vertices. Let F' be the vertices of an inclusion minimal set of trees of $G[F] - v$ such that $|F'| > f_r(|C|)$. Clearly $|F'| \leq 2f_r(|C|)$ since otherwise the set is not inclusion minimal. Each tree in $G[F']$ contains a neighbor of v and $F' \subseteq F$, hence (C, F') is simple by condition (b), and (C, F') is reducible since $|F'| > f_r(|C|)$.

- There is a tree T in $G[F] - v$ that contains more than $f_r(|C|)$ vertices. Now $(C, V(T))$ is a single-tree reducible FVC with $|V(T)| < |F|$, so the induction hypothesis applies. □

We proceed to show how a simple reducible FVC can be found using color coding. A vertex coloring of G is a function $\chi\colon V(G) \to \{\dot{\mathsf{C}}, \dot{\mathsf{F}}\}$. We say a simple FVC (C, F) is *properly colored* by a coloring χ if $F \subseteq \chi^{-1}(\dot{\mathsf{F}})$ and $C \cup N_G(F) \subseteq \chi^{-1}(\dot{\mathsf{C}})$.

Lemma 5.5.3. *Given a multigraph G and coloring χ of G that properly colors a simple reducible FVC (C, F), a reducible FVC (C', F') can be found in $\mathcal{O}(n^3)$ time.*

Proof. If (C, F) is simple by condition (a), i.e., $G[F]$ is connected, it is easily verified that we can find a reducible FVC in $\mathcal{O}(n^3)$ time as follows: Consider the set $\mathcal{T}$ of all trees in $G[\chi^{-1}(\dot{\mathsf{F}})]$. For each tree $T \in \mathcal{T}$, if there is a vertex $u \in N_G(T)$ such that $e(\{u\}, T) = 1$ take $C' := N_G(T) \setminus \{u\}$, otherwise take $C' := N_G(T)$. If $|V(T)| > f_r(|C'|)$ return $(C', V(T))$.

In the remainder of the proof we assume that (C, F) is simple by condition (b).

Algorithm For each vertex $u \in \chi_V^{-1}(\dot{\mathsf{C}})$ consider the set $\mathcal{T}$ of all trees T in $G[\chi_V^{-1}(\dot{\mathsf{F}})]$ such that $e(\{u\}, T) = 1$. Let $C' \subseteq \chi_V^{-1}(\dot{\mathsf{C}}) \setminus \{u\}$ be the set of vertices (excluding u) with a neighbor in at least two trees in $\mathcal{T}$ and let $\mathcal{T}_1$ be the set of trees $T \in \mathcal{T}$ for which $N_G(T) \subseteq C' \cup \{u\}$. Now consider the set of trees $\mathcal{T}_2 = \mathcal{T} \setminus \mathcal{T}_1$ as a set of objects for a 0-1 knapsack problem where we define for each $T \in \mathcal{T}_2$ its weight as $|N_G(T) \setminus (C' \cup \{u\})|$ and its value as $|V(T)|$. Using the dynamic programming algorithm [122] for the 0-1 knapsack problem we compute for all $0 \leq b \leq |N_G(V(\mathcal{T}_2)) \setminus (C' \cup \{u\})|$ a set of trees $\mathcal{T}_2^b \subseteq \mathcal{T}_2$ with a combined weight $\sum_{T \in \mathcal{T}_2^b} |N_G(T) \setminus (C' \cup \{u\})| \leq b$ such that the combined value $\sum_{T \in \mathcal{T}_2^b} |V(T)|$ is maximized. If for any such b we have $|V(\mathcal{T}_1)| + |V(\mathcal{T}_2^b)| > f_r(|C'| + b)$ then take $\hat{C} := C' \cup N_G(V(\mathcal{T}_2^b)) \setminus \{u\}$ and $\hat{F} := V(\mathcal{T}_1) \cup V(\mathcal{T}_2^b)$ and return $(\hat{C}, \hat{F})$.

Correctness To show that $(\hat{C}, \hat{F})$ is a FVC, first note that $G[\hat{F}]$ is a forest. For each tree T in this forest we have $e(T, \{u\}) = 1$ and $N_G(T) \subseteq C' \cup \{u\} \cup N_G(V(\mathcal{T}_2^b)) = \hat{C} \cup \{u\}$. It follows that $e(T, G - (\hat{C} \cup \hat{F})) = e(T, \{u\}) = 1$. To show that $(\hat{C}, \hat{F})$ indeed reducible observe that $\sum_{T \in \mathcal{T}_2^b} |N_G(T) \setminus (C' \cup \{u\})| = |\bigcup_{T \in \mathcal{T}_2^b} N_G(T) \setminus (C' \cup \{u\})|$ since if two trees $T_1, T_2 \in \mathcal{T}_2^b$ have a common neighbor that is not u, it must be in C' by definition, hence the neighborhoods only intersect on $C' \cup \{u\}$. We can now deduce $|\hat{F}| = |V(\mathcal{T}_1)| + |V(\mathcal{T}_2^b)| > f_r(|C'| + b) \geq f_r(|C'| + \sum_{T \in \mathcal{T}_2^b} |N_G(T) \setminus (C' \cup \{u\})|) = f_r(|C' \cup N_G(V(\mathcal{T}_2^b)) \setminus \{u\}|) = f_r(|\hat{C}|)$.

It remains to show that if χ properly colors a simple reducible FVC (C, F) then for some $u \in \chi_V^{-1}(\dot{\mathsf{C}})$ there exists a b such that $|V(\mathcal{T}_1)| + |V(\mathcal{T}_2^b)| \geq f_r(|C'| + b)$. Recall that we assumed (C, F) is simple by condition (b), i.e., all trees in $G[F]$ have a common neighbor v and there exists a single-tree FVC (C, F_2) with $v \in F_2 \setminus F$ and $F \subseteq F_2$. Since (C, F) is properly colored we know $v \in \chi^{-1}(\dot{\mathsf{C}})$, so in some iteration we will have $u = v$. Consider the sets $\mathcal{T}$, $\mathcal{T}_1$, $\mathcal{T}_2$, and C' as defined in this iteration. We first show $C' \subseteq C$. If $w \in C'$ then w has a neighbor in two trees

in $\mathcal{T}$. This means there are two internally vertex disjoint paths between v and w, forming a cycle. Since $v \in F_2$ we have by Observation 5.2.1 for the FVC (C, F_2) that this cycle must contain a vertex in C which is therefore different from v. Recall that (C, F) is properly colored, hence all vertices in C have color $\dot{\mathsf{C}}$. Note that the internal vertices of these paths all have color $\dot{\mathsf{F}}$ because they are vertices from trees in $G[\chi_V^{-1}(\dot{\mathsf{F}})]$. Hence $w \in C$ and therefore $C' \subseteq C$. To complete the proof we show

Claim 5.5.4. There exists a value b such that $|V(\mathcal{T}_1)| + |V(\mathcal{T}_2^b)| \geq f_r(|C'| + b)$.

Proof. Recall that we assumed existence of a properly colored FVC (C, F) that is reducible and simple by condition (b) witnessed by the FVC (C, F_2). Consider the set $\mathcal{T}'$ of trees in $G[F]$. Note that any tree T' in $\mathcal{T}'$ is a tree in $G[\chi^{-1}(\dot{\mathsf{F}})]$ since (C, F) is properly colored and note also that T' contains a neighbor of v. If $e(T', \{v\}) > 1$ then $G[F_2]$ contains a cycle, contradicting that (C, F_2) is a FVC in G, hence $e(T', \{v\}) = 1$. It follows that $T' \in \mathcal{T}$, meaning $\mathcal{T}' \subseteq \mathcal{T}$. Take $\mathcal{T}_2' = \mathcal{T}' \setminus \mathcal{T}_1 = \mathcal{T}' \cap \mathcal{T}_2$ and $b = \sum_{T \in \mathcal{T}_2'} |N_G(V(T)) \setminus (C' \cup \{v\}|$. Clearly $\mathcal{T}_2'$ is a candidate solution for the 0-1 knapsack problem with capacity b, hence $|V(\mathcal{T}_2^b)| \geq |V(\mathcal{T}_2')|$. We deduce

$$
\begin{aligned}
|V(\mathcal{T}_1)| + |V(\mathcal{T}_2^b)| \geq |V(\mathcal{T}_1)| + |V(\mathcal{T}_2')| &\geq |V(\mathcal{T}')| = |F| \\
&> f_r(|C|) && \text{since } (C, F) \text{ is reducible} \\
&= f_r(|C' \cup C|) && \text{since } C' \subseteq C \\
&= f_r(|C' \cup (N_G(F) \setminus \{v\}) \cup C|) && \text{since } N_G(\mathcal{T}_2') \setminus \{v\} \subseteq C \\
&\geq f_r(|C' \cup (N_G(\mathcal{T}_2') \setminus \{v\})|) && \text{since } f_r \text{ is non-decreasing} \\
&= f_r(|C'| + |N_G(\mathcal{T}_2') \setminus (C' \cup \{v\})|) && \text{since } |A \cup B| = |A| + |B \setminus A| \\
&> f_r(|C'| + b)
\end{aligned}
$$

⌟

Running time For each $u \in \chi_V^{-1}(\dot{\mathsf{C}})$ we perform a number of $\mathcal{O}(n+m)$-time operations and run the dynamic programming algorithm for a problem with $\mathcal{O}(n)$ objects and a capacity of $\mathcal{O}(n)$ yielding a run time of $\mathcal{O}(n^2)$ for each u or $\mathcal{O}(n^3)$ for the algorithm as a whole. □

Using the previous lemmas the problem of finding a reducible single-tree FVC reduces to finding a coloring that properly colors a simple reducible FVC. We generate a set of colorings that is guaranteed to contain at least one such coloring. To generate this set we use the concept of a universal set. For some set D of size n and integer s with $n \geq s$ an (n, s)-*universal set* for D is a family $\mathcal{U}$ of subsets of D such that for all $S \subseteq D$ of size at most s we have $\{S \cap U \mid U \in \mathcal{U}\} = 2^S$.

Theorem 5.5.5 ([102, Theorem 6], cf. [35, Theorem 5.20])**.** *For any set D and integers n and s with $|D| = n \geq s$, an (n, s)-universal set $\mathcal{U}$ for D with $|\mathcal{U}| = 2^{\mathcal{O}(s)} \log n$ can be created in $2^{\mathcal{O}(s)} n \log n$ time.*

It can be shown that whether a simple FVC of width k is properly colored is determined by at most $1 + k + 2f_r(k) = \mathcal{O}(k^3)$ relevant vertices. By creating an $(n, \mathcal{O}(k^3))$-universal set for $V(G)$ using Theorem 5.5.5, we can obtain in $2^{\mathcal{O}(k^3)} \cdot n \log n$ time a set of $2^{\mathcal{O}(k^3)} \cdot \log n$ colorings that contains a coloring for each possible assignment of colors for these relevant vertices. By applying Lemma 5.5.3 for each coloring we obtain the following lemma:

Lemma 5.5.6. *There exists an algorithm that, given a multigraph G and an integer k, outputs a (possibly empty) FVC (C, F) in G. If G contains a reducible single-tree FVC of width at most k then (C, F) is reducible. The algorithm runs in time $2^{\mathcal{O}(k^3)} \cdot n^3 \log n$.*

Proof. Take $s = 2f_r(k)+k+1$. By Theorem 5.5.5 an (n, s)-universal set $\mathcal{U}$ for $V(G)$ of size $2^{\mathcal{O}(s)} \log n$ can be created in $2^{\mathcal{O}(s)} n \log n$ time. For each $Q \in \mathcal{U}$ let χ_Q be the coloring of G with $\chi_Q^{-1}(\dot{\mathsf{C}}) = Q$. Run the algorithm from Lemma 5.5.3 on χ_Q for every $Q \in \mathcal{U}$ and return the first reducible FVC. If no reducible FVC was found return $(\emptyset, \emptyset)$. We obtain an overall run time of $2^{\mathcal{O}(s)} \cdot n^3 \log n = 2^{\mathcal{O}(k^3)} \cdot n^3 \log n$.

To prove correctness assume G contains a reducible single-tree FVC (C, F) with $|C| \leq k$. Then by Lemma 5.5.2 we know G contains a simple reducible FVC (C, F'). Coloring χ properly colors (C, F') if all vertices in $F' \cup C \cup N_G(F')$ are assigned the correct color. Hence at most $|F'| + |C + N_G(F')| \leq 2f_r(k) + k + 1 = s$ vertices need to have the correct color. By construction of $\mathcal{U}$, there is a $Q \in \mathcal{U}$ such that χ_Q assigns the correct colors to these vertices. Hence χ_Q properly colors (C, F') and by Lemma 5.5.3 a reducible FVC is returned. □

5.5.2 Reducing feedback vertex cuts

We introduce reduction rules inspired by existing kernelization algorithms [23, 121] and apply them on the subgraph $G[C \cup F]$ for a FVC (C, F) in G. We give 5 reduction rules and show at least one is applicable if $|F| > f_r(|C|)$. The rules reduce the number of vertices $v \in F$ with $\deg_G(v) < 3$ or reduce $e(C, F)$. The following lemma shows that this is sufficient to reduce the size of F.

Lemma 5.5.7. *Let G be a multigraph with minimum degree at least 3 and let (C, F) be a FVC in G. We have $|F| \leq e(C, F)$.*

Proof. We first show that the claim holds if $G[F]$ is a tree. For all $i \geq 0$ let $V_i := \{v \in F \mid \deg_{G[F]}(v) = i\}$. Note that since $G[F]$ is connected, $V_0 \neq \emptyset$ if and only if $|F| = 1$ and the claim is trivially true, so suppose $V_0 = \emptyset$. We first show $|V_{\geq 3}| < |V_1|$.

$$2|E(G[F])| = \sum_{v \in F} \deg_{G[F]}(v) \geq |V_1| + 2|V_2| + 3|V_{\geq 3}|$$
$$2|E(G[F])| = 2(|V(G[F])| - 1) = 2|V_1| + 2|V_2| + 2|V_{\geq 3}| - 2$$

We obtain $|V_1| + 2|V_2| + 3|V_{\geq 3}| \leq 2|V_1| + 2|V_2| + 2|V_{\geq 3}| - 2$ hence $|V_{\geq 3}| < |V_1|$. We know all vertices in F have degree at least 3 in G, so $e(V(G) \setminus F, F) \geq 2|V_1| + |V_2| > |V_1| + |V_2| + |V_{\geq 3}| = |F|$. By definition of FVC there is at most one vertex in F that has an edge to $V(G) \setminus (C \cup F)$, all other edges must be between C and F. We obtain $1 + e(C, F) > |F|$.

If $G[F]$ is a forest, then let $F_1, \ldots, F_\ell$ be the vertex sets for each tree in $G[F]$. Since (C, F_i) is a FVC in G for all $1 \leq i \leq \ell$, we know $e(C, F_i) \geq |F_i|$ for all $1 \leq i \leq \ell$, and since $F_1, \ldots, F_\ell$ is a partition of F we conclude $e(C, F) = \sum_{1 \leq i \leq \ell} e(C, F_i) \geq \sum_{1 \leq i \leq \ell} |F_i| = |F|$. □

Next, we give the reduction rules. These rules apply to a multigraph G and yield a new multigraph G' and vertex set $S \subseteq V(G) \setminus V(G')$. A reduction rule with output G' and S is *FVS-safe* if for any minimum feedback vertex set S' of G', the set $S \cup S'$ is a minimum feedback vertex set of G. A reduction rule is *antler-safe* if for all $z \geq 0$ and any z-antler (C, F) in G, there exists a z-antler (C', F') in G' with $C' \cup F' = (C \cup F) \cap V(G')$ and $|C'| = |C| - |(C \cup F) \cap S|$.

Reduction Rule 5.1. If $u, v \in V(G)$ are connected by more than two edges, remove all but two of these edges to obtain G' and take $S := \emptyset$.

Reduction Rule 5.2. If $v \in V(G)$ has degree exactly 2 and no self-loop, obtain G' by removing v from G and adding an edge e with $\iota(e) = N_G(v)$. Take $S := \emptyset$.

Reduction Rules 5.1 and 5.2 are well established and FVS-safe. Additionally Reduction Rule 5.1 can easily be seen to be antler-safe. To see that Reduction Rule 5.2 is antler-safe, consider a z-antler (C, F) in G for some $z \geq 0$. If $v \notin C$ it is easily verified that $(C, F \setminus \{v\})$ is a z-antler in G'. If $v \in C$ pick a vertex $u \in N_G(v) \cap F$ and observe that $(\{u\} \cup C \setminus \{v\}, F \setminus \{u\})$ is a z-antler in G'.

Reduction Rule 5.3. If (C, F) is an antler in G, then $G' := G - (C \cup F)$ and $S := C$.

Lemma 5.5.8. *Reduction Rule 5.3 is FVS-safe and antler-safe.*

Proof. To show Reduction Rule 5.3 is FVS-safe, let Z be a minimum FVS of G'. Now $(Z, V(G' - Z))$ is an antler in $G' = G - (C \cup F)$ so then G contains the antler $(Z \cup C, V(G' - Z) \cup F) = (Z \cup S, V(G - (Z \cup S)))$ by Lemma 5.4.4. It follows that $Z \cup S$ is a minimum FVS of G.

To show Reduction Rule 5.3 is antler-safe, let $z \geq 0$ and let $(\hat{C}, \hat{F})$ be an arbitrary z-antler in G, then by Lemma 5.4.2 $(\hat{C} \setminus (C \cup F), \hat{F} \setminus (C \cup F)$ is a z-antler

in $G' = G - (C \cup F)$. We deduce:

$$\begin{aligned}
|\hat{C} \setminus (C \cup F)| &= |\hat{C}| - |\hat{C} \cap C| - |\hat{C} \cap F| && \text{since } C \cap F = \emptyset \\
&= |\hat{C}| - |\hat{C} \cap C| - |C \cap \hat{F}| && \text{by Proposition 5.4.1} \\
&= |\hat{C}| - |(\hat{C} \cap C) \cup (C \cap \hat{F})| && \text{since } \hat{C} \cap \hat{F} = \emptyset \\
&= |\hat{C}| - |(\hat{C} \cup \hat{F}) \cap C| \\
&= |\hat{C}| - |(\hat{C} \cup \hat{F}) \cap \hat{S}'|.
\end{aligned}$$

□

The following reduction rule uses a graph structure called flower. For a vertex $v \in V(G)$ and an integer k, a *v-flower* of *order* k is a collection of k cycles in G whose vertex sets only intersect in v.

Reduction Rule 5.4. If (C, F) is a FVC in G and for some $v \in C$ the subgraph $G[F \cup \{v\}]$ contains a v-flower of order $|C|+1$, then $G' := G - v$ and $S := \{v\}$.

Lemma 5.5.9. *Reduction Rule 5.4 is FVS-safe and antler-safe.*

Proof. We first show that any minimum FVS in G contains v. Let X be a minimum FVS in G. If $v \notin X$ then $|F \cap X| > |C|$ since $G[F \cup \{v\}]$ contains a v-flower of order $|C| + 1$. Take $X' := C \cup (X \setminus F)$, clearly $|X'| < |X|$ so $G - X'$ must contain a cycle since X was minimum. This cycle must contain a vertex from $X \setminus X' \subseteq F$, so by Observation 5.2.1 this cycle must contain a vertex from C, but $C \subseteq X'$. Contradiction.

To show Reduction Rule 5.4 is FVS-safe, suppose Z is a minimum FVS of $G' = G - v$. Clearly $Z \cup \{v\}$ is a FVS in G. To show that $Z \cup \{v\}$ is minimum suppose Z' is a smaller FVS in G. We know $v \in Z$ so $Z' \setminus \{v\}$ is a FVS in $G - v$, but $|Z' \setminus \{v\}| < |Z|$ contradicting optimality of Z.

To show Reduction Rule 5.4 is antler-safe, suppose $(\hat{C}, \hat{F})$ is a z-antler in G for some $z \geq 0$. We show $(\hat{C} \setminus \{v\}, \hat{F})$ is a z-antler in G'. If $v \in \hat{C}$ then this follows directly from Observation 5.2.4, so suppose $v \notin \hat{C}$. Note that $v \in \hat{F}$ would contradict that any minimum FVS in G contains v by Observation 5.2.3. So $G[\hat{C} \cup \hat{F}] = G'[\hat{C} \cup \hat{F}]$ and $(\hat{C} \setminus \{v\}, \hat{F}) = (\hat{C}, \hat{F})$ is a FVC in $G' = G - v$ by Observation 5.2.2, hence $(\hat{C} \setminus \{v\}, \hat{F})$ is a z-antler in G'. □

Reduction Rule 5.5. If (C, F) is a FVC in G, $v \in C$, and $X \subseteq F$ such that $G[F \cup \{v\}] - X$ is acyclic, and if T is a tree in $G[F] - X$ containing a vertex $w \in N_G(v)$ such that for each $u \in N_G(T) \setminus \{v\}$ there are more than $|C|$ other trees $T' \neq T$ in $G[F] - X$ for which $\{u, v\} \subseteq N_G(T')$, then take $S := \emptyset$ and obtain G' by removing the unique edge between v and w, and adding double-edges between v and u for all $u \in N_G(V(T)) \setminus \{v\}$.

Lemma 5.5.10. *Reduction Rule 5.5 is FVS-safe and antler-safe.*

Proof. Suppose $(\hat{C}, \hat{F})$ is a z-antler in G for some $z \geq 0$. We first prove the following claim:

Claim 5.5.11. For all $u \in N_G(T)\setminus\{v\}$ we have $v \in \hat{F} \Rightarrow u \in \hat{C}$ and $u \in \hat{F} \Rightarrow v \in \hat{C}$.

Proof. Each tree of $G[F] - X$ supplies a path between u and v, hence there are more than $|C|+1$ internally vertex-disjoint paths between u and v. Suppose $v \in \hat{F}$, we show $u \in \hat{C}$. The proof of the second implication is symmetric. Suppose for contradiction that $u \notin \hat{C}$. All except possibly one of the disjoint paths between u and v must contain a vertex in $\hat{C}$ by Observation 5.2.1 since any two disjoint paths form a cycle containing a vertex from $\hat{F}$. Let $Y \subseteq \hat{C}$ be the set of vertices in $\hat{C}$ that are in a tree of $G[F] - X$ with neighbors of u and v, so $|Y| > |C|$. Then $|C \cup \hat{C} \setminus Y| < |\hat{C}|$ we derive a contradiction by showing $G[\hat{C} \cup \hat{F}] - (C \cup \hat{C} \setminus Y)$ is acyclic. We know $Y \subseteq F$, so any cycle in G containing a vertex from Y also contains a vertex from C by Observation 5.2.1. So if $G[\hat{C} \cup \hat{F}] - (C \cup \hat{C} \setminus Y)$ contains a cycle, then so does $G[\hat{C} \cup \hat{F}] - (C \cup \hat{C})$ which contradicts that $\hat{C}$ is a (minimum) FVS in $G[\hat{C} \cup \hat{F}]$ since $(\hat{C}, \hat{F})$ is an antler in G. ⌟

To prove Reduction Rule 5.5 is antler-safe, we show that $(\hat{C}, \hat{F})$ is also a z-antler in G'. Suppose $v \notin \hat{C} \cup \hat{F}$, then $G[\hat{C} \cup \hat{F}] = G'[\hat{C} \cup \hat{F}]$ as G and G' only differ on edges incident to v. It remains to show that for each tree T' in $G'[\hat{F}]$ we have $e(T', G' - (\hat{C} \cup \hat{F})) \leq 1$. Suppose T' is a tree in $G'[\hat{F}]$ with $e(T', G' - (\hat{C} \cup \hat{F})) > 1$. Since $e(T', G - (\hat{C} \cup \hat{F})) \leq 1$ we know that at least one of the edges added between v and some $u \in N_G(T)$ has an endpoint in $V(T') \subseteq \hat{F}$. Since $v \notin \hat{F}$ we have $u \in \hat{F}$, so $v \in \hat{C}$ by Claim 5.5.11 contradicting our assumption $v \notin \hat{C} \cup \hat{F}$.

Suppose $v \in \hat{C} \cup \hat{F}$, we first show that $(\hat{C}, \hat{F})$ is a FVC in G'. If $v \in \hat{C}$ this is clearly the case, so suppose $v \in \hat{F}$. From Claim 5.5.11 it follows that $N_G(T)\setminus\{v\} \subseteq \hat{C}$, so all edges added in G' are incident to vertices in $\hat{C}$ hence $(\hat{C}, \hat{F})$ is still a FVC in G'. We now show that $G'[\hat{C} \cup \hat{F}]$ contains an $\hat{C}$-certificate of order z. We know $G[\hat{C} \cup \hat{F}]$ contains a $\hat{C}$-certificate of order z. Let H be an arbitrary component of this certificate. Take $Y := V(H) \cap \hat{C}$, so Y is a minimum FVS in H. It suffices to show that Y is also a minimum FVS of $G' \cap H$. This is easily seen to be true when $v \notin V(H)$, so suppose $v \in V(H)$. First we argue that $(G' \cap H) - Y$ is acyclic. This is easily seen to be true when $v \in Y$ since G and G' only differ in edges incident to v, so suppose $v \notin Y$. Then $v \notin \hat{C}$ hence $v \in \hat{F}$ and by Claim 5.5.11 $N_G(T) \setminus \{v\} \subseteq \hat{C}$. It follows that $V(H) \cap (N_G(T) \setminus \{v\}) \subseteq Y$ so clearly $(G' \cap H) - Y$ is acyclic since all edges in $G' \cap H$ that are not in H are incident to a vertex in Y.

To show Y is a minimum FVS, suppose there is a FVS Y' of $G' \cap H$ with $|Y'| < |Y|$. Since $H - v$ is a subgraph of $G' \cap H$ we know $H - (Y' \cup \{v\})$ is acyclic, but since $|Y'| < |Y| = \mathsf{fvs}(H)$ we also know $H - Y'$ contains a cycle. This cycle must contain the edge $\{v, w\}$ since otherwise this cycle is also present in $(G' \cap H) - Y'$. Then there must be some $u \in N_G(T) \setminus \{v\}$ on the cycle, so $u, v \notin Y'$. But G' contains a double-edge between u and v so $(G' \cap H) - Y'$ contains a cycle, contradicting that Y' is a FVS in $G' \cap H$.

We finally show Reduction Rule 5.5 is FVS-safe. Let Z' be a minimum FVS in G', and suppose Z' is not a FVS in G. Then $G - Z'$ contains a cycle. This

cycle contains the edge $\{v, w\}$ since otherwise $G' - Z'$ also contains this cycle. Since G' contains double-edges between v and all $u \in N_G(T) \setminus \{v\}$ and $v \notin Z'$, it follows that $N_G(T) \setminus \{v\} \subseteq Z'$, but then no cycle in $G - Z'$ can intersect T and $\{v, w\}$ is not part of a cycle in $G - Z'$. We conclude by contradiction that Z' is a FVS in G. To prove optimality, consider a minimum FVS Z in G and observe that $(Z, V(G - Z))$ is an antler in G. Since Reduction Rule 5.5 is antler-safe we know G' contains an antler (C', F') with $C' \cup F' = (Z \cup V(G-Z)) \cap V(G') = V(G')$ and $|C'| = |Z| - |(Z \cup V(G - Z)) \cap S| = |Z|$. Since $C' \cup F' = V(G')$ we know C' is a FVS in G', hence $\mathsf{fvs}(G') \leq |C'| = |Z|$, hence $\mathsf{fvs}(G) = |Z| \geq \mathsf{fvs}(G') = |Z'|$. □

Before showing one of these reduction rules can be applied efficiently given a reducible FVC, we give the following lemma, which helps in particular with the application of Reduction Rule 5.4.

Lemma 5.5.12 (Cf. [110, Lemma 3.9])**.** *If v is a vertex in a multigraph G such that v does not have a self-loop and $G - v$ is acyclic, then we can find in $\mathcal{O}(n)$ time a set $X \subseteq V(G) \setminus \{v\}$ such that $G - X$ is acyclic and G contains a v-flower of order $|X|$.*

Proof. We prove the existence of such a set X and v-flower by induction on $|V(G)|$. The inductive proof can easily be translated into a linear-time algorithm. If G is acyclic, output $X = \emptyset$ and a v-flower of order 0. Otherwise, since v does not have a self-loop there is a tree T of the forest $G - v$ such that $G[V(T) \cup \{v\}]$ contains a cycle. Root T at an arbitrary vertex and consider a deepest node x in T for which the subgraph $G[V(T_x) \cup \{v\}]$ contains a cycle C. Then any feedback vertex set of G that does not contain v, has to contain at least one vertex of T_x; and the choice of x as a deepest vertex implies that x lies on all cycles of G that intersect T_x. By induction on $G' := G - V(T_x)$ and v, there is a feedback vertex set $X' \subseteq V(G') \setminus \{v\}$ of G' and a v-flower in G' of order $|X'|$. We obtain a v-flower of order $|X'|+1$ in G by adding C, while $X := X' \cup \{x\} \subseteq V(G) \setminus \{v\}$ is a feedback vertex set of size $|X'| + 1$. □

Finally we show that when we are given a reducible FVC (C, F) in G, then we can find and apply a rule in $\mathcal{O}(n^2)$ time. With a more careful analysis better running time bounds can be shown, but this does not affect the final running time of the main algorithm.

Lemma 5.5.13. *Given a multigraph G and a reducible FVC (C, F) in G, we can find and apply a rule in $\mathcal{O}(n^2)$ time.*

Proof. Note that if a vertex $v \in V(G)$ has a self-loop then $(\{v\}, \emptyset)$ is an antler in G and we can apply Reduction Rule 5.3. If a vertex v has degree 0 or 1 then $(\emptyset, \{v\})$ is an antler in G. Hence Reduction Rules 5.1, 5.2 and 5.3 can always be applied if the graph contains a self-loop, a vertex with degree less than 3, or more than 2 edges between two vertices. So assume G is a graph with no self-loops, minimum degree at least 3, and at most two edges between any pair of vertices.

By Lemma 5.5.7 we have $e(C,F) \geq |F| > 2|C|^3 + 3|C|^2 - |C|$ so then there must be a vertex v in C with more than $\frac{1}{|C|} \cdot (2|C|^3 + 3|C|^2 - |C|) = 2|C|^2 + 3|C| - 1$ edges to F. By Lemma 5.5.12 we can find a set $X \subseteq F$ such that $G[F \cup \{v\}] - X$ is acyclic and $G[F \cup \{v\}]$ contains a v-flower of order $|X|$. Hence if $|X| \geq |C| + 1$ Reduction Rule 5.4 can be applied, so assume $|X| \leq |C|$. For each $u \in X \cup C \setminus \{v\}$ that is not connected to v by a double-edge, mark up to $|C|+1$ trees T' in $G[F] - X$ for which $\{u, v\} \in N_G(T)$. Note that we marked at most $(|C|+1) \cdot |X \cup C \setminus \{v\}| \leq (|C|+1) \cdot (2|C|-1) = 2|C|^2 + |C| - 1$ trees. Since v has exactly one edge to each marked tree ($G[F \cup \{v\}] - X$ is acyclic) and at most two edges to each vertex in X, this accounts for at most $2|C|^2 + 3|C| - 1$ edges from v to F, so there must be at least one more edge from v to a vertex $w \in F$, hence Reduction Rule 5.5 applies.

It can easily be verified that all operations described can be performed in $\mathcal{O}(n^2)$ time. □

5.5.3 Finding and removing antlers

We will find antlers making use of color coding, using coloring functions of the form $\chi\colon V(G) \cup E(G) \to \{\dot{\mathsf{F}}, \dot{\mathsf{C}}, \dot{\mathsf{R}}\}$. For all $c \in \{\dot{\mathsf{F}}, \dot{\mathsf{C}}, \dot{\mathsf{R}}\}$ let $\chi_V^{-1}(c) = \chi^{-1}(c) \cap V(G)$. For any integer $z \geq 0$, a z-antler (C, F) in a multigraph G is *z-properly colored* by a coloring χ if all of the following hold:

(i) $F \subseteq \chi_V^{-1}(\dot{\mathsf{F}})$,

(ii) $C \subseteq \chi_V^{-1}(\dot{\mathsf{C}})$,

(iii) $N_G(F) \setminus C \subseteq \chi_V^{-1}(\dot{\mathsf{R}})$, and

(iv) $G[C \cup F] - \chi^{-1}(\dot{\mathsf{R}})$ is a C-certificate of order z.

Recall that $\chi^{-1}(\dot{\mathsf{R}})$ can contain edges as well as vertices so for any subgraph H of G the multigraph $H - \chi^{-1}(\dot{\mathsf{R}})$ is obtained from H by removing both vertices and edges. It can be seen that if (C, F) is a z-antler, then there exists a coloring that z-properly colors it. Consider for example a coloring where a vertex v is colored $\dot{\mathsf{C}}$ (resp. $\dot{\mathsf{F}}$) if $v \in C$ (resp. $v \in F$), all other vertices are colored $\dot{\mathsf{R}}$, and for some C-certificate H of order z in $G[C \cup F]$ all edges in H have color $\dot{\mathsf{F}}$ and all other edges have color $\dot{\mathsf{R}}$. The property in of a properly colored z-antler described in Lemma 5.5.14 will be useful to prove correctness of the color coding algorithm.

Lemma 5.5.14. *For any $z \geq 0$, if a z-antler (C, F) in multigraph G is z-properly colored by a coloring χ and H is a component of $G[C \cup F] - \chi^{-1}(\dot{\mathsf{R}})$ then each component H' of $H - C$ is a component of $G[\chi_V^{-1}(\dot{\mathsf{F}})] - \chi^{-1}(\dot{\mathsf{R}})$ with $N_{G - \chi^{-1}(\dot{\mathsf{R}})}(H') \subseteq C \cap V(H)$.*

Proof. Note that since $C \cap \chi_V^{-1}(\dot{\mathsf{F}}) = \emptyset$ we have that $N_{G - \chi^{-1}(\dot{\mathsf{R}})}(H') \subseteq C \cap V(H)$ implies that $N_{G[\chi_V^{-1}(\dot{\mathsf{F}})] - \chi^{-1}(\dot{\mathsf{R}})}(H') = \emptyset$ and hence that H' is a component of $G[\chi_V^{-1}(\dot{\mathsf{F}})] - \chi^{-1}(\dot{\mathsf{R}})$. We show $N_{G - \chi^{-1}(\dot{\mathsf{R}})}(H') \subseteq C \cap V(H)$.

Suppose $v \in N_{G-\chi^{-1}(\dot{\mathsf{R}})}(H')$ and let $u \in V(H')$ be a neighbor of v in $G - \chi^{-1}(\dot{\mathsf{R}})$. Since $V(H') \subseteq F$ we know $u \in F$. Since (C, F) is z-properly colored we also have $N_G(F) \setminus C = \chi^{-1}(\dot{\mathsf{R}})$, hence $N_G(u) \subseteq C \cup F \cup \chi^{-1}(\dot{\mathsf{R}})$ so then $N_{G-\chi^{-1}}(u) \subseteq C \cup F$. By choice of u we have $v \in N_{G-\chi^{-1}}(u) \subseteq C \cup F$. So since $u, v \in C \cup F$, and u and v are neighbors in $G - \chi^{-1}(\dot{\mathsf{R}})$ we know u and v are in the same component of $G[C \cup F] - \chi^{-1}(\dot{\mathsf{R}})$, hence $v \in V(H)$.

Suppose $v \notin C$, so $v \in F$. Since also $u \in F$ we know that u and v are in the same component of $G[F] - \chi^{-1}(\dot{\mathsf{R}})$, so $v \in H'$, but then $v \notin N_{G-\chi^{-1}(\dot{\mathsf{R}})}(H')$ contradicting our choice of v. It follows that $v \in C$ hence $v \in C \cap V(H)$. Since $v \in N_{G-\chi^{-1}(\dot{\mathsf{R}})}(H')$ was arbitrary $N_{G-\chi^{-1}(\dot{\mathsf{R}})}(H') \subseteq C \cap V(H)$. □

We now show that a z-antler can be obtained from a suitable coloring of the graph.

Lemma 5.5.15. *An $n^{\mathcal{O}(z)}$ time algorithm exists taking as input an integer $z \geq 0$, a multigraph G, and a coloring χ and producing as output a z-antler (C, F) in G, such that for any z-antler $(\hat{C}, \hat{F})$ that is z-properly colored by χ we have $\hat{C} \subseteq C$ and $\hat{F} \subseteq F$.*

Proof. We define a function $W_\chi \colon 2^{\chi_V^{-1}(\dot{\mathsf{C}})} \to 2^{\chi_V^{-1}(\dot{\mathsf{F}})}$ as follows: for any $C \subseteq \chi_V^{-1}(\dot{\mathsf{C}})$ let $W_\chi(C)$ denote the set of all vertices that are in a component H of $G[\chi_V^{-1}(\dot{\mathsf{F}})] - \chi^{-1}(\dot{\mathsf{R}})$ for which $N_{G-\chi^{-1}(\dot{\mathsf{R}})}(H) \subseteq C$. The algorithm we describe updates the coloring χ and recolors any vertex or edge that is not part of a z-properly colored antler to color $\dot{\mathsf{R}}$.

1. Recolor all edges to color $\dot{\mathsf{R}}$ when one of its endpoints has color $\dot{\mathsf{R}}$.
2. For each component H of $G[\chi_V^{-1}(\dot{\mathsf{F}})]$ we recolor all vertices of H and their incident edges to color $\dot{\mathsf{R}}$ if H is not a tree or $e(H, \chi_V^{-1}(\dot{\mathsf{R}})) > 1$.
3. For each subset $C \subseteq \chi_V^{-1}(\dot{\mathsf{C}})$ of size at most z, mark all vertices in C if $\mathsf{fvs}(G[C \cup W_\chi(C)] - \chi^{-1}(\dot{\mathsf{R}})) = |C|$.
4. If $\chi_V^{-1}(\dot{\mathsf{C}})$ contains unmarked vertices we recolor them to color $\dot{\mathsf{R}}$, remove markings made in Step 3 and repeat from Step 1.
5. If all vertices in $\chi_V^{-1}(\dot{\mathsf{C}})$ are marked in Step 3, return $(\chi_V^{-1}(\dot{\mathsf{C}}), \chi_V^{-1}(\dot{\mathsf{F}}))$.

Running time The algorithm will terminate after at most n iterations since in every iteration the number of vertices in $\chi_V^{-1}(\dot{\mathsf{R}})$ increases. Steps 1, 2, 4 and 5 can easily be seen to take no more than $\mathcal{O}(n^2)$ time. Step 3 can be performed in $\mathcal{O}(4^z \cdot n^{z+1})$ time by checking for all $\mathcal{O}(n^z)$ subsets $C \in \chi_V^{-1}(\dot{\mathsf{C}})$ of size at most z whether the subgraph $G[C \cup W_\chi(C)] - \chi^{-1}(\dot{\mathsf{R}})$ has feedback vertex number z. This can be done in time $\mathcal{O}(4^z \cdot n)$ [77]. Hence the algorithm runs in time $n^{\mathcal{O}(z)}$.

Correctness We show first that any z-properly colored antler prior to executing the algorithm remains z-properly colored after termination and then that, in Step 5, $(\chi_V^{-1}(\dot{\mathsf{C}}), \chi_V^{-1}(\dot{\mathsf{F}}))$ is a z-antler in G. Since $(\chi_V^{-1}(\dot{\mathsf{C}}), \chi_V^{-1}(\dot{\mathsf{F}}))$ contains all properly colored antlers this proves correctness.

Claim 5.5.16. All z-antlers $(\hat{C}, \hat{F})$ that are z-properly colored by χ prior to executing the algorithm are also z-properly colored by χ after termination of the algorithm.

Proof. To show the algorithm preserves properness of the coloring, we show that every individual recoloring preserves properness, that is, if an arbitrary z-antler is z-properly colored prior to the recoloring, it is also z-properly colored after the recoloring.

Suppose an arbitrary z-antler $(\hat{C}, \hat{F})$ is z-properly colored by χ. An edge is only recolored when one of its endpoints has color $\dot{\mathsf{R}}$. Since these edges are not in $G[\hat{C} \cup \hat{F}]$ its color does change whether $(\hat{C}, \hat{F})$ is colored z-properly. All other operations done by the algorithm are recolorings of vertices to color $\dot{\mathsf{R}}$. We show that any time a vertex v is recolored we have that $v \notin \hat{C} \cup \hat{F}$, meaning $(\hat{C}, \hat{F})$ remains colored z-properly.

Suppose v is recolored in Step 2, then we know $\chi(v) = \dot{\mathsf{F}}$, and v is part of a component H of $G[\chi_V^{-1}(\dot{\mathsf{F}})]$. Since χ z-properly colors $(\hat{C}, \hat{F})$ we have $\hat{F} \subseteq \chi_V^{-1}(\dot{\mathsf{F}})$ but $N_G(\hat{F}) \cap \chi_V^{-1}(\dot{\mathsf{F}}) = \emptyset$, so since H is a component of $G[\chi_V^{-1}(\dot{\mathsf{F}})]$ we know either $V(H) \subseteq \hat{F}$ or $V(H) \cap \hat{F} = \emptyset$. If $V(H) \cap \hat{F} = \emptyset$ then clearly $v \notin \hat{C} \cup \hat{F}$. So suppose $V(H) \subseteq \hat{F}$, then H is a tree in $G[\hat{F}]$. Since v was recolored and H is a tree it must be that $e(H, \chi_C^{-1}(\dot{\mathsf{R}})) > 1$ but this contradicts that $(\hat{C}, \hat{F})$ is a FVC.

Suppose v is recolored in Step 4, then we know v was not marked during Step 3 and $\chi(v) = \dot{\mathsf{C}}$, so $v \notin \hat{F}$. Suppose that $v \in \hat{C}$. We derive a contradiction by showing that v was marked in Step 3. Since $(\hat{C}, \hat{F})$ is z-properly colored, we know that $G[\hat{C} \cup \hat{F}] - \chi^{-1}(\dot{\mathsf{R}})$ is a $\hat{C}$-certificate of order z, so if H is the component of $G[\hat{C} \cup \hat{F}] - \chi^{-1}(\dot{\mathsf{R}})$ containing v then $\mathsf{fvs}(H) = |\hat{C} \cap V(H)| \leq z$. Since $\hat{C} \cap V(H) \subseteq \hat{C} \subseteq \chi_V^{-1}(\dot{\mathsf{C}})$ we know that in some iteration in Step 3 we have $C = \hat{C} \cap V(H)$. To show that v was marked, we show that $\mathsf{fvs}(G[C \cup W_\chi(C))] - \chi^{-1}(\dot{\mathsf{R}})) = |C|$. We know $G[W_\chi(C)] - \chi^{-1}(\dot{\mathsf{R}})$ is a forest since it is a subgraph of $G[\chi_V^{-1}(\dot{\mathsf{F}})]$ which is a forest by Step 2, so we have that $\mathsf{fvs}(G[C \cup W_\chi(C)] - \chi^{-1}(\dot{\mathsf{R}})) \leq |C|$. To show $\mathsf{fvs}(G[C \cup W_\chi(C))] - \chi^{-1}(\dot{\mathsf{R}})) \geq |C|$ we show that H is a subgraph of $G[C \cup W_\chi(C))] - \chi^{-1}(\dot{\mathsf{R}})$. By Lemma 5.5.14 we have that each component H' of $H - \hat{C}$ is also a component of $G[\chi_V^{-1}(\dot{\mathsf{F}})] - \chi^{-1}(\dot{\mathsf{R}})$ with $N_{G - \chi^{-1}(\dot{\mathsf{R}})}(H') \subseteq \hat{C} \cap V(H) = C$. Hence $V(H - \hat{C}) = V(H - C) \subseteq W_\chi(C)$ so H is a subgraph of $G[C \cup W_\chi(C)]$. Since H is also a subgraph of $G[\hat{C} \cup \hat{F}] - \chi^{-1}(\dot{\mathsf{R}})$ we conclude that H is a subgraph of $G[C \cup W_\chi(C))] - \chi^{-1}(\dot{\mathsf{R}})$ and therefore $\mathsf{fvs}(G[C \cup W_\chi(C))] - \chi^{-1}(\dot{\mathsf{R}})) \geq \mathsf{fvs}(H) = |C|$. ⌟

Claim 5.5.17. In Step 5, $(\chi_V^{-1}(\dot{\mathsf{C}}), \chi_V^{-1}(\dot{\mathsf{F}}))$ is a z-antler in G.

Proof. We know $(\chi_V^{-1}(\dot{\mathsf{C}}), \chi_V^{-1}(\dot{\mathsf{F}}))$ is a FVC in G because each connected component of $G[\chi_V^{-1}(\dot{\mathsf{F}})]$ is a tree and has at most one edge to a vertex not in $\chi_V^{-1}(\dot{\mathsf{C}})$ by Step 2. It remains to show that $G[\chi_V^{-1}(\dot{\mathsf{C}}) \cup \chi_V^{-1}(\dot{\mathsf{F}})]$ contains a $\chi_V^{-1}(\dot{\mathsf{C}})$-certificate of order z. Note that in Step 5 the coloring χ is the same as in the last execution of Step 3. Let $\mathcal{C} \subseteq 2^{\chi_V^{-1}(\dot{\mathsf{C}})}$ be the family of all subsets $C \subseteq \chi_V^{-1}(\dot{\mathsf{C}})$ that have been considered in Step 3 and met the conditions for marking all vertices in C, i.e., $\mathsf{fvs}(G[C \cup W_\chi(C)] - \chi^{-1}(\dot{\mathsf{R}})) = |C| \leq z$. Since all vertices in $\chi_V^{-1}(\dot{\mathsf{C}})$ have been marked during the last execution of Step 3 we know $\bigcup_{C \in \mathcal{C}} C = \chi_V^{-1}(\dot{\mathsf{C}})$.

Let $C_1, \ldots, C_{|\mathcal{C}|}$ be the sets in $\mathcal{C}$ in an arbitrary order and define $D_i := C_i \setminus C_{<i}$ for all $1 \leq i \leq |\mathcal{C}|$. Observe that $D_1, \ldots, D_{|\mathcal{C}|}$ is a partition of $\chi_V^{-1}(\dot{\mathsf{C}})$ with $|D_i| \leq z$ and $C_i \subseteq D_{\leq i}$ for all $1 \leq i \leq |\mathcal{C}|$. Note that D_i may be empty for some i. We now show that $G[\chi_V^{-1}(\dot{\mathsf{C}}) \cup \chi_V^{-1}(\dot{\mathsf{F}})]$ contains a $\chi_V^{-1}(\dot{\mathsf{C}})$-certificate of order z. We do this by showing there are $|\mathcal{C}|$ vertex disjoint subgraphs of $G[\chi_V^{-1}(\dot{\mathsf{C}}) \cup \chi_V^{-1}(\dot{\mathsf{F}})]$, call them $G_1, \ldots, G_{|\mathcal{C}|}$, such that $\mathsf{fvs}(G_i) = |D_i| \leq z$ for each $1 \leq i \leq |\mathcal{C}|$. Take $G_i := G[D_i \cup (W_\chi(D_{\leq i}) \setminus W_\chi(D_{<i}))] - \chi^{-1}(\dot{\mathsf{R}})$ for all $1 \leq i \leq |\mathcal{C}|$. First we show that for any $i \neq j$ the multigraphs G_i and G_j are vertex disjoint. Clearly $D_i \cap D_j = \emptyset$. We can assume $i < j$, so $D_{\leq i} \subseteq D_{<j}$ and then $W_\chi(D_{\leq i}) \subseteq W_\chi(D_{<j})$. By successively dropping two terms, we deduce

$$\begin{aligned}(W_\chi(D_{\leq i}) \setminus W_\chi(D_{<i})) \cap (W_\chi(D_{\leq j}) \setminus W_\chi(D_{<j})) &\\ \subseteq W_\chi(D_{\leq i}) \cap (W_\chi(D_{\leq j}) \setminus W_\chi(D_{<j})) &\\ \subseteq W_\chi(D_{\leq i}) \setminus W_\chi(D_{<j}) = \emptyset.&\end{aligned}$$

We complete the proof by showing $\mathsf{fvs}(G_i) = |D_i|$ for all $1 \leq i \leq \ell$. Recall that $D_i = C_i \setminus C_{<i}$. Since $C_i \in \mathcal{C}$ we know C_i is an optimal FVS in $G[C_i \cup W_\chi(C_i)] - \chi^{-1}(\dot{\mathsf{R}})$, so then clearly D_i is an optimal FVS in $G[C_i \cup W_\chi(C_i)] - \chi^{-1}(\dot{\mathsf{R}}) - C_{<i} = G[D_i \cup W_\chi(C_i)] - \chi^{-1}(\dot{\mathsf{R}})$. We know that $C_i \subseteq D_{\leq i}$ so then also $W_\chi(C_i) \subseteq W_\chi(D_{\leq i})$. It follows that D_i is an optimal FVS in $G[D_i \cup W_\chi(D_{\leq i})] - \chi^{-1}(\dot{\mathsf{R}})$. In this multigraph, all vertices in $W_\chi(D_{<i})$ must be in a component that does not contain any vertices from D_i, so this component is a tree and we obtain $|D_i| = \mathsf{fvs}(G[D_i \cup W_\chi(D_{\leq i})] - \chi^{-1}(\dot{\mathsf{R}})) = \mathsf{fvs}(G[D_i \cup W_\chi(D_{\leq i})] - \chi^{-1}(\dot{\mathsf{R}}) - W_\chi(D_{<i})) = \mathsf{fvs}(G[D_i \cup (W_\chi(D_{\leq i}) \setminus W_\chi(D_{<i}))] - \chi^{-1}(\dot{\mathsf{R}})) = \mathsf{fvs}(G_i)$. ⌟

It can be seen from Claim 5.5.16 that for any z-properly colored antler $(\hat{C}, \hat{F})$ we have $\hat{C} \subseteq \chi_V^{-1}(\dot{\mathsf{C}})$ and $\hat{F} \subseteq \chi_V^{-1}(\dot{\mathsf{F}})$. Claim 5.5.17 completes the correctness argument. □

If a multigraph G contains a reducible single-tree FVC of width at most k then we can find and apply a reduction rule by Lemma 5.5.6 and Lemma 5.5.13. If G does not contain such a FVC, but G does contain a non-empty z-antler (C, F) of

width at most k, then using Lemma 5.4.6 we can prove that whether (C, F) is z-properly colored is determined by the color of at most $41k^5z^2$ relevant vertices and edges. Similar to the algorithm from Lemma 5.5.6, we can use two $(n+m, 41k^5z^2)$-universal sets to create a set of colorings that is guaranteed to contain a coloring that z-properly colors (C, F). Using Lemma 5.5.15 we find a non-empty z-antler and apply Reduction Rule 5.3. We obtain the following:

Lemma 5.5.18. *Given a multigraph G and integers $k \geq z \geq 0$. If G contains a non-empty z-antler of width at most k we can find and apply a reduction rule in $2^{\mathcal{O}(k^5z^2)} \cdot n^{\mathcal{O}(z)}$ time.*

Proof. Consider the following algorithm: First we use Lemma 5.5.6 to obtain a FVC (C_1, F_1) in $2^{\mathcal{O}(k^3)} \cdot n^3 \log n$ time. If (C_1, F_1) is reducible we can find and apply a reduction rule in $\mathcal{O}(n^2)$ time by Lemma 5.5.13 so assume (C_1, F_1) is not reducible. Create two $(n + m, 41k^5z^2)$-universal sets $\mathcal{U}_1$ and $\mathcal{U}_2$ for $V(G) \cup E(G)$ using Theorem 5.5.5. Define for each pair $(Q_1, Q_2) \in \mathcal{U}_1 \times \mathcal{U}_2$ the coloring χ_{Q_1,Q_2} of G that assigns all vertices and edges in Q_1 color $\dot{\mathsf{C}}$, all vertices and edges in $Q_2 \setminus Q_1$ color $\dot{\mathsf{F}}$, and all vertices and edges not in $Q_1 \cup Q_2$ color $\dot{\mathsf{R}}$. For each $(Q_1, Q_2) \in \mathcal{U}_1 \times \mathcal{U}_2$ obtain in $n^{\mathcal{O}(z)}$ time a z-antler (C_2, F_2) by running the algorithm from Lemma 5.5.15 on G and χ_{Q_1,Q_2}. If (C_2, F_2) is not empty, apply Reduction Rule 5.3 to remove (C_2, F_2), otherwise report G does not contain a z-antler of width at most k.

Running time By Theorem 5.5.5, the sets $\mathcal{U}_1$ and $\mathcal{U}_2$ have size $2^{\mathcal{O}(k^5z^2)} \cdot \log n$ and can be created in $2^{\mathcal{O}(k^5z^2)} \cdot n \log n$ time. It follows that there are $|\mathcal{U}_1 \times \mathcal{U}_2| = 2^{\mathcal{O}(k^5z^2)} \cdot \log^2 n$ colorings for which we apply the $n^{\mathcal{O}(z)}$ time algorithm from Lemma 5.5.15. We obtain an overall running time of $2^{\mathcal{O}(k^5z^2)} \cdot n^{\mathcal{O}(z)}$. Since a z-antler has width at least z, we can assume $k \geq z$, hence $2^{\mathcal{O}(k^5z^2)} \cdot n^{\mathcal{O}(z)} \leq 2^{\mathcal{O}(k^7)} \cdot n^{\mathcal{O}(z)}$.

Correctness Suppose G contains a z-antler (C, F) of width at most k, we show the algorithm finds a reduction rule to apply. By Lemma 5.4.6 we know that there exists an $F' \subseteq F$ such that (C, F') is a z-antler where $G[F']$ has at most $\frac{|C|}{2}(z^2 + 4z - 1)$ trees. For each tree T in $G[F']$ note that $(C, V(T))$ is a single-tree FVC of width $|C| \leq k$. If for some tree T in G the FVC $(C, V(T))$ is reducible, then (C_1, F_1) is reducible by Lemma 5.5.6 and we find a reduction rule using Lemma 5.5.13, so suppose for all trees T in $G[F']$ that $|V(T)| \leq f_r(|C|)$. So then $|F'| \leq \frac{|C|}{2}(z^2 + 4z - 1) \cdot f_r(|C|)$. We show that in this case there exists a pair $(Q_1, Q_2) \in \mathcal{U}_1 \times \mathcal{U}_2$ such that χ_{Q_1,Q_2} z-properly colors (C, F').

Whether a coloring z-properly colors (C, F') is only determined by the colors of $C \cup F' \cup N_G(F') \cup E(G[C \cup F'])$.

Claim 5.5.19. $|C \cup F' \cup N_G(F') \cup E(G[C \cup F'])| \leq 41k^5z^2$.

Proof. Note that $|N_G(F') \setminus C| \leq \frac{|C|}{2}(z^2+4z-1)$ since no tree in $G[F']$ can have more than one neighbor outside C. Additionally we have

$$\begin{aligned}
|E(G[C \cup F'])| &\leq |E(G[C])| + |E(G[F'])| + e(C, F') \\
&\leq |E(G[C])| + |F'| + |C| \cdot |F'| && \text{since } G[F'] \text{ is a forest} \\
&\leq |C|^2 + (|C|+1) \cdot |F'| \\
&\leq |C|^2 + (|C|+1) \cdot \frac{|C|}{2}(z^2+4z-1) \cdot f_r(|C|) \\
&\leq k^2 + (k+1) \cdot \frac{k}{2}(z^2+4z-1) \cdot (2k^3+3k^2-k) \\
&\leq k^2 + \frac{z^2+4z-1}{2} \cdot (k^2+k) \cdot (2k^3+3k^2-k) \\
&\leq k^2 + \frac{z^2+4z-1}{2} \cdot 2k^2 \cdot 5k^3 && \text{since } k = 0 \text{ or } k \geq 1 \\
&\leq k^2 + \frac{5z^2}{2} \cdot 10k^5 && \text{since } z = 0 \text{ or } z \geq 1 \\
&\leq k^2 + 25k^5z^2 \leq 26k^5z^2,
\end{aligned}$$

hence

$$\begin{aligned}
&|C \cup F' \cup N_G(F') \cup E(G[C \cup F'])| \\
&\quad = |C| + |F'| + |N_G(F') \setminus C| + |E(G[C \cup F'])| \\
&\quad \leq |C| + \frac{|C|}{2}(z^2+4z-1) \cdot f_r(|C|) + \frac{|C|}{2}(z^2+4z-1) + 26k^5z^2 \\
&\quad = |C| + \frac{|C|}{2}(z^2+4z-1) \cdot (f_r(|C|)+1) + 26k^5z^2 \\
&\quad \leq k + \frac{k}{2}(5z^2) \cdot (f_r(k)+1) + 26k^5z^2 \\
&\quad \leq k + \frac{5}{2}(kz^2) \cdot (2k^3+3k^2-k+1) + 26k^5z^2 \\
&\quad \leq k + \frac{5}{2}(kz^2) \cdot (6k^3) + 26k^5z^2 \\
&\quad \leq k + \frac{30}{2}z^2k^4 + 26k^5z^2 \\
&\quad \leq 41k^5z^2.
\end{aligned}$$

⌟

By construction of $\mathcal{U}_1$ and $\mathcal{U}_2$ there exist sets $Q_1 \in \mathcal{U}_1$ and $Q_2 \in \mathcal{U}_2$ such that χ_{Q_1,Q_2} z-properly colors (C, F'). Hence the algorithm from Lemma 5.5.15 returns a non-empty z-antler for χ_{Q_1,Q_2} and Reduction Rule 5.3 can be executed. □

Note that applying a reduction rule reduces the number of vertices or increases the number of double-edges. Hence by repeatedly using Lemma 5.5.18 to apply a reduction rule we obtain, after at most $\mathcal{O}(n^2)$ iterations, a graph in which no rule applies. By Lemma 5.5.18 this graph does not contain a non-empty z-antler of width at most k. We show that this method reduces the solution size at least as much as iteratively removing z-antlers of width at most k. We first describe the behavior of such a sequence of antlers. For integer $k \geq 0$ and $z \geq 0$, we say a sequence of disjoint vertex sets $C_1, F_1, \ldots, C_\ell, F_\ell$ is a *z-antler-sequence* for a multigraph G if for all $1 \leq i \leq \ell$ the pair (C_i, F_i) is a z-antler in $G - (C_{<i} \cup F_{<i})$. The *width* of a z-antler-sequence is defined as $\max_{1 \leq i \leq \ell} |C_1|$.

Proposition 5.5.20. *If $C_1, F_1, \ldots, C_\ell, F_\ell$ is a* z-antler-sequence *for some multigraph G, then the pair $(C_{\leq i}, F_{\leq i})$ is a z-antler in G for any $1 \leq i \leq \ell$.*

Proof. We use induction on i. Clearly the statement holds for $i = 1$, so suppose $i > 1$. By induction $(C_{<i}, F_{<i})$ is a z-antler in G, and since (C_i, F_i) is a z-antler in $G - (C_{<i} \cup F_{<i})$ we have by Lemma 5.4.4 that $(C_{<i} \cup C_i, F_{<i} \cup F_i) = (C_{\leq i}, F_{\leq i})$ is a z-antler in G. □

The following theorem describes that repeatedly applying Lemma 5.5.18 reduces the solution size at least as much as repeatedly removing z-antlers of width at most k. By taking $t = 1$ we obtain Theorem 5.1.2.

Theorem 5.5.21. *Given as input a multigraph G and integers $k \geq z \geq 0$ we can find in $2^{\mathcal{O}(k^5 z^2)} \cdot n^{\mathcal{O}(z)}$ time a vertex set $S \subseteq V(G)$ such that*

1. *there is a minimum FVS in G containing all vertices of S, and*
2. *if $C_1, F_1, \ldots, C_t, F_t$ is a z-antler sequence of width at most k then $|S| \geq |C_{\leq t}|$.*

Proof. We first describe the algorithm.

Algorithm We use Lemma 5.5.18 to apply a reduction rule in G and obtain the resulting multigraph G' and vertex set S. If no applicable rule was found return an empty vertex set $S := \emptyset$. Otherwise we recursively call our algorithm on G' with integers z and k to obtain a vertex set S' and return the vertex set $S \cup S'$.

Running time Note that since every reduction rule reduces the number of vertices or increases the number of double-edges, after applying at most $\mathcal{O}(n^2)$ reduction rules we obtain a graph where no rules can be applied. Therefore after at most $\mathcal{O}(n^2)$ recursive calls the algorithm terminates. We obtain a running time of $2^{\mathcal{O}(k^5 z^2)} \cdot n^{\mathcal{O}(z)}$.

Correctness We prove correctness by induction on the recursion depth, which is shown the be finite by the run time analysis.

First consider the case that no reduction rule was found. Clearly Condition 1 holds for $G' := G$ and $S := \emptyset$. To show Condition 2 suppose $C_1, F_1, \ldots, C_t, F_t$ is a z-antler-sequence of width at most k for G. The first non-empty antler in this sequence is a z-antler of width at most k in G. Since no reduction rule was found using Lemma 5.5.18 it follows that G does not contain a non-empty z-antler of width at most k. Hence all antlers in the sequence must be empty and $|C_{\leq t}| = 0$, so Condition 2 holds for $G' := G$ and $S := \emptyset$.

For the other case, suppose G' and S are obtained by applying a reduction rule, then since this rule is FVS-safe we know for any minimum FVS S'' of G' that $S \cup S''$ is a minimum FVS in G. Since S' is obtained from a recursive call there is a minimum FVS in G' containing all vertices of S'. Let S'' be such a FVS in G', so $S' \subseteq S''$ then we know $S \cup S''$ is a minimum FVS in G. It follows that there is a minimum FVS in G containing all vertices of $S \cup S'$, proving Condition 1.

To prove Condition 2 suppose $C_1, F_1, \ldots, C_t, F_t$ is a z-antler-sequence of width at most k for G. We first prove the following:

Claim 5.5.22. There exists a z-antler-sequence $C'_1, F'_1, \ldots, C'_t, F'_t$ of width at most k for G' such that

1. $C'_{\leq t} \cup F'_{\leq t} = V(G') \cap (C_{\leq t} \cup F_{\leq t})$ and
2. $|C'_{\leq t}| = \sum_{1 \leq i \leq t} |C_i| - |(C_i \cup F_i) \cap S|$.

Proof. We use induction on t. Since G' and S are obtained through an antler-safe reduction rule and (C_1, F_1) is a z-antler in G, we know that G' contains a z-antler (C'_1, F'_1) such that $C'_1 \cup F'_1 = (C_1 \cup F_1) \cap V(G')$ and $|C'_1| = |C_1| - |(C_1 \cup F_1) \cap S|$. The claim holds for $t = 1$.

For the induction step, consider $t > 1$. By applying induction to the length-$(t-1)$ prefix of the sequence, there is a z-antler sequence $C'_1, F'_1, \ldots, C'_{t-1}, F'_{t-1}$ of width at most k for G' such that both conditions hold. We have by Proposition 5.5.20 that $(C_{\leq t}, F_{\leq t})$ is a z-antler in G. Since G' and S are obtained through an antler-safe reduction rule from G there is a z-antler (C', F') in G' such that $C' \cup F' = V(G') \cap (C_{\leq t} \cup F_{\leq t})$ and $|C'| = |C_{\leq t}| - |S \cap (C_{\leq t} \cup F_{\leq t})|$. Take $C'_t := C' \setminus (C'_{<t} \cup F'_{<t})$ and $F'_t := F' \setminus (C'_{<t} \cup F'_{<t})$. By Lemma 5.4.2 we have that (C'_t, F'_t) is a z-antler in $G' - (C'_{<t} \cup F'_{<t})$, it follows that $C'_1, F'_1, \ldots, C'_t, F'_t$ is a z-antler-sequence for G'. We first show Condition 1.

$$\begin{aligned} C'_{\leq t} \cup F'_{\leq t} &= C'_t \cup F'_t \cup C'_{<t} \cup F'_{<t} \\ &= C' \cup F' && \text{by choice of } C'_t \text{ and } F'_t \\ &= V(G') \cap (C_{\leq t} \cup F_{\leq t}) && \text{by choice of } C' \text{ and } F'. \end{aligned}$$

To prove Condition 2 and the z-antler-sequence $C'_1, F'_1, \ldots, C'_t, F'_t$ has width at most k we first show $|C'_t| = |C_t| - |(C_t \cup F_t) \cap S|$. Observe that $(C'_{\leq t}, F'_{\leq t})$ is an

antler in G' by Proposition 5.5.20.

$$\begin{aligned}
|C'_t| &= |C'_{\leq t}| - |C'_{<t}| && \text{since } C'_i \cap C'_j = \emptyset \text{ for all } i \neq j \\
&= \mathsf{fvs}(G'[C'_{\leq t} \cup F'_{\leq t}]) - |C'_{<t}| && \text{by the above} \\
&= \mathsf{fvs}(G'[V(G') \cap (C_{\leq t} \cup F_{\leq t})]) - |C'_{<t}| && \text{by Condition 1} \\
&= \mathsf{fvs}(G'[C' \cup F']) - |C'_{<t}| && \text{by choice of } C' \text{ and } F' \\
&= |C'| - |C'_{<t}| && \text{since } (C', F') \text{ is an antler in } G' \\
&= |C'| - \sum_{1 \leq i < t} (|C_i| - |(C_i \cup F_i) \cap S|) && \text{by induction} \\
&= |C_{\leq t}| - |S \cap (C_{\leq t} \cup F_{\leq t})| - \sum_{1 \leq i < t} (|C_i| - |(C_i \cup F_i) \cap S|) \\
&= \sum_{1 \leq i \leq t} (|C_i| - |S \cap (C_i \cup F_i)|) - \sum_{1 \leq i < t} (|C_i| - |(C_i \cup F_i) \cap S|) \\
& && \text{since } C_1, F_1, \ldots, C_t, F_t \text{ are pairwise disjoint} \\
&= |C_t| - |(C_t \cup F_t) \cap S|
\end{aligned}$$

We know the z-antler-sequence $C'_1, F'_1, \ldots, C_{t-1}, F_{t-1}$ has width at most k, so to show that this z-antler-sequence has width at most k it suffices to prove that $|C'_t| \leq k$. Indeed $|C'_t| = |C_t| - |(C_t \cup F_t) \cap S| \leq |C_t| \leq k$.

To complete the proof of Claim 5.5.22 we now derive Condition 2:

$$\begin{aligned}
|C'_{\leq t}| &= |C'_t| + |C'_{<t}| && \text{since } C'_t \cap C'_{<t} = \emptyset \\
&= |C_t| - |(C_t \cup F_t) \cap S| + |C'_{<t}| \\
&= |C_t| - |(C_t \cup F_t) \cap S| + \sum_{1 \leq i \leq t-1} (|C_i| - |(C_i \cup F_i) \cap S|) && \text{by induction} \\
&= \sum_{1 \leq i \leq t} (|C_i| - |(C_i \cup F_i) \cap S|).
\end{aligned}$$

⌟

To complete the proof of Condition 2 from Theorem 5.5.21 we show $|S \cup S'| \geq |C_{\leq t}|$. By Claim 5.5.22 we know a z-antler-sequence $C'_1, F'_1, \ldots, C'_t, F'_t$ of width at most k for G' exists. Since S' is obtained from a recursive call we have $|S'| \geq |C'_{\leq t}|$,

so then

$$\begin{aligned}
|S \cup S'| &= |S| + |S'| \\
&\geq |S| + |C'_{\leq t}| \\
&= |S| + \sum_{1 \leq i \leq t} (|C_i| - |(C_i \cup F_i) \cap S|) && \text{by Claim 5.5.22} \\
&= |S| + \sum_{1 \leq i \leq t} |C_i| - \sum_{1 \leq i \leq t} |(C_i \cup F_i) \cap S| \\
&= |S| + |C_{\leq t}| - |S \cap (C_{\leq t} \cup F_{\leq t})| && \text{since } C'_1, F'_1, \ldots, C'_t, F'_t \text{ are disjoint} \\
&= |S| + |C_{\leq t}| - |S| \\
&\geq |C_{\leq t}|. && \square
\end{aligned}$$

As a corollary to this theorem, we obtain a new type of parameterized-tractability result for Feedback Vertex Set. For an integer z, let the *z-antler complexity* of G be the minimum number k for which there exists a (potentially long) sequence $C_1, F_1, \ldots, C_t, F_t$ of disjoint vertex sets such that for all $1 \leq i \leq t$, the pair (C_i, F_i) is a z-antler of width at most k in $G - (C_{<i} \cup F_{<i})$, and such that $G - (C_{\leq t} \cup F_{\leq t})$ is acyclic (implying that $C_{\leq t}$ is a feedback vertex set in G). If no such sequence exists, the z-antler complexity of G is $+\infty$.

Intuitively, Corollary 5.5.23 states that optimal solutions can be found efficiently when they are composed out of small pieces, each of which has a low-complexity certificate for belonging to some optimal solution.

Corollary 5.5.23. *There is an algorithm that, given a multigraph G, returns an optimal feedback vertex set in time $f(k^*) \cdot n^{\mathcal{O}(z^*)}$, where (k^*, z^*) is any pair of integers such that the z^*-antler complexity of G is at most k^*.*

Proof. Let (k^*, z^*) be such that the z^*-antler complexity of G is at most k^*. Let $p_1 \in \mathcal{O}(k^5 z^2), p_2 \in \mathcal{O}(z)$ be concrete functions such that the running time of Theorem 5.5.21 is bounded by $2^{p_1(k,z)} \cdot n^{p_2(z)}$. Consider the pairs $\{(k', z') \in \mathbb{N}^2 \mid 1 \leq z' \leq k' \leq n\}$ in order of increasing value of the running-time guarantee $2^{p_1(k,z)} \cdot n^{p_2(z)}$. For each such pair (k', z'), start from the multigraph G and invoke Theorem 5.5.21 to obtain a vertex set S which is guaranteed to be contained in an optimal solution. If $G - S$ is acyclic, then S itself is an optimal solution and we return S. Otherwise we proceed to the next pair (k', z').

Correctness The correctness of Theorem 5.5.21 and the definition of z-antler complexity ensure that for $(k', z') = (k^*, z^*)$, the set S is an optimal solution. In particular, if $C_1, F_1, \ldots, C_t, F_t$ is a sequence of vertex sets witnessing that the z^*-antler complexity of G is at most k^*, then Theorem 5.5.21 is guaranteed by Condition 2 to output a set S of size at least $\sum_{1 \leq i \leq t} |C_i|$, which is equal to the size of an optimal solution on G by definition.

Running time For a fixed choice of (k', z') the algorithm from Theorem 5.5.21 runs in time $2^{\mathcal{O}((k')^5(z')^2)} \cdot n^{\mathcal{O}(z')} \leq 2^{\mathcal{O}((k^*)^5(z^*)^2)} \cdot n^{\mathcal{O}(z^*)}$ because we try pairs (k', z') in order of increasing running time. As we try at most n^2 pairs before finding the solution, the corollary follows. □

To conclude, we reflect on the running time of Corollary 5.5.23 compared to running times of the form $2^{\mathcal{O}(\mathsf{fvs}(G))} \cdot n^{\mathcal{O}(1)}$ obtained by FPT algorithms for the parameterization by solution size. If we exhaustively apply Lemma 5.5.13 with the FVC $(C, V(G) \setminus C)$, where C is obtained from a 2-approximation algorithm [11], then this gives an *antler-safe* kernelization: it reduces the graph as long as the graph is larger than $f_r(|C|)$. This opening step reduces the instance size to $\mathcal{O}(\mathsf{fvs}(G)^3)$ without increasing the antler complexity. As observed before, after applying $\mathcal{O}(n^2)$ reduction rules we obtain a graph in which no rules can be applied. This leads to a running time of $\mathcal{O}(n^4)$ of the kernelization. Running Theorem 5.5.21 to solve the reduced instance yields a total running time of $2^{\mathcal{O}(k^5 z^2)} \mathsf{fvs}(G)^{\mathcal{O}(z)} + \mathcal{O}(n^4)$. This is asymptotically faster than $2^{\mathcal{O}(\mathsf{fvs}(G))}$ when $z \leq k \in o(\sqrt[7]{\mathsf{fvs}(G)})$ and $\mathsf{fvs}(G) \in \omega(\log n)$, which captures the intuitive idea sketched above that our algorithmic approach has an advantage when there is an optimal solution that is large but composed of small pieces for which there are low-complexity certificates.

5.6 Conclusion

We have taken the first steps into a new direction for preprocessing which aims to investigate how and when a preprocessing phase can guarantee to identify parts of an optimal solution to an NP-hard problem, thereby reducing the running time of the follow-up algorithm. Aside from the technical results concerning antler structures for Feedback Vertex Set and their algorithmic properties, we consider the conceptual message of this research direction an important contribution of our theoretical work on understanding the power of preprocessing and the structure of solutions to NP-hard problems.

This line of investigation opens up a host of opportunities for future research. For combinatorial problems such as Vertex Cover, Odd Cycle Transversal, and Directed Feedback Vertex Set, which kinds of substructures in inputs allow parts of an optimal solution to be identified by an efficient preprocessing phase? Is it possible to give preprocessing guarantees not in terms of the size of an optimal solution, but in terms of measures of the stability [8, 10, 38] of optimal solutions under small perturbations? Some questions also remain open concerning the concrete technical results in this chapter. Can the running time of Theorem 5.1.2 be improved to $f(k) \cdot n^{\mathcal{O}(1)}$? We conjecture that it cannot, but have not been able to prove this. A related question applies to Vertex Cover: Is there an algorithm running in time $f(k) \cdot n^{\mathcal{O}(1)}$ that, given a graph G which has disjoint vertex sets (C, H) such that $N_G(C) \subseteq H$ and H of size k is an optimal

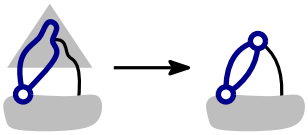


Figure 5.5 Standard reduction rules for Feedback Vertex Set reduce any 1-antler of width 1 to a pair of vertices with two edges between them, one of which has degree 3. Hence we can reduce the graph until all 1-antlers of width 1 are removed with the addition of the following reduction rule: If vertices u and v are connected by a double edge and $\deg(v) = 3$ then remove u and v from the graph and decrease the solution size by one. These reduction rules can be exhaustively applied in linear time.

vertex cover in $G[C \cup H]$, outputs a set of size at least k that is part of an optimal vertex cover in G? (Note that this is an easier target than computing such a decomposition of width k if one exists, which can be shown to be W[1]-hard.)

To apply the theoretical ideas on antlers in the practical settings that motivated their investigation, it would be interesting to determine which types of antler can be found in *linear* time. A slight extension of the standard reduction rules [35, FVS.1–FVS.5] for Feedback Vertex Set can be used to detect 1-antlers of width 1 in linear time (see Figure 5.5). Can the running time of Theorem 5.1.1 be improved to $f(k) \cdot (n + m)$? It would also be interesting to investigate which types of antlers are present in practical inputs.

6

Finding Secluded Sparse Graphs

6.1 Introduction

This thesis studies a number of problems that can be described as finding large sparse subgraphs. For the problems studied in the previous chapters the subgraphs to be found are sparse because they exclude a set of forbidden minors, and they are large because they can be obtained from the input graph by removing only a small number of vertices. We have explored a number of different techniques for various sets of forbidden minors. In this final chapter we consider a secluded variant of the problem. Rather than requiring that the subgraph can be obtained by removing a small number of vertices from the input graph, we require the subgraph to be connected to only a small set of vertices of the remainder of the graph. More formally, in the Connected Secluded Π-Subgraph problem we are given a graph G and an integer k and the task is to find a maximum size k-secluded subgraph belonging to the target graph class Π, where an induced subgraph H of G is k-secluded if $|N_G(H)| \leq k$. Note that this is not strictly speaking a decision problem as discussed in Section 2.4 since the answer is a subgraph rather than yes or no. This is because in this chapter we also consider the problem of enumerating and counting all possible solutions. The Connected Secluded Π-Subgraph problem has been studied before by Golovach et al. [67]. In this chapter we investigate the case where Π is the graph class consisting of trees, i.e., we consider the following problem.

LARGE SECLUDED TREE (LST)
Input: A graph G, a non-negative integer k, and a weight function $w\colon V(G) \to \mathbb{N}^+$.
Parameter: k.
Task: Find a k-secluded tree H of G of maximum weight, or report that no such H exists.

Secluded versions of several classic optimization problems have been studied intensively in recent years [14, 15, 29, 58, 98], many of which are discussed in Till Fluschnik's PhD thesis [57]. Golovach et al. [67] mention that finding a maximum size k-secluded tree is FPT when parameterized by k and can be solved in time $2^{2^{\mathcal{O}(k \log k)}} \cdot n^{\mathcal{O}(1)}$ using the recursive understanding technique, the details of which can be found in the arXiv version [66]. For the case where Π is characterized by a finite set of forbidden induced subgraphs $\mathcal{F}$, they show that the problem is FPT with a triple-exponential dependency on the parameter k. They pose the question whether it is possible to avoid these double- and triple-exponential dependencies on the parameter. They give some examples of Π for which this is the case, namely for Π being a clique, a star, a d-regular graph, or an induced path.

Another motivation for studying k-secluded trees comes from the work in the previous chapter. The graph structures defined in Chapter 5 have a close resemblance to the concept of k-secluded trees. The key difference is that while a k-secluded tree is not allowed to have any neighbors outside of a set of k vertices, a feedback vertex cut (C, F) of width k is allowed to have a number of edges between F and the remainder of the graph, $G - (C \cup F)$. For the case where $G[F]$ is connected, i.e., (C, F) is a single-tree feedback vertex cut, there can be at most one such edge, so $G[F]$ is a $(k+1)$-secluded tree in G. Conversely, observe that if H is a k-secluded tree in G then $(N_G(H), V(H))$ is a single-tree feedback vertex cut of width k in G.

Chapter 5 describes two color coding procedures to find certain types of feedback vertex cuts (C, F). These procedures do not necessarily attempt to maximize the size of F. While for the antlers found using Lemma 5.5.18 the size of F is irrelevant, for the feedback vertex cuts found using Lemma 5.5.6 the size of F needs to be sufficiently large compared to C. In fact, Lemma 5.5.6 is used for the purpose of reducing the size of single-tree feedback vertex cuts. It achieves this by finding a sufficiently large part of a single-tree feedback vertex cut, which can be reduced using a reduction rule. Although in this chapter we focus on the more traditional problem of finding maximum k-secluded trees, the acute reader may observe that the algorithm we present in this chapter can in fact be used to find single-tree feedback vertex cuts of maximum size, which achieves the same goal as Lemma 5.5.6.

Results Our main result is an algorithm for LARGE SECLUDED TREE that takes $2^{\mathcal{O}(k \log k)} \cdot n^4$ time. This answers the question of Golovach et al. [67] affirmatively for the case of trees. We solve a more general version of the problem,

where a set of vertices is given that should be part of the k-secluded tree. Our algorithm goes one step further by allowing us to find all maximum-weight solutions. As we will later argue, it is not possible to output all such solutions directly in the promised running time. Instead, the output consists of a bounded number of solution descriptions such that each maximum-weight solution can be constructed from one such description. This is similar in spirit to the work of Guo et al. [68], who enumerate all size-k solutions to the Feedback Vertex Set problem in $2^{\mathcal{O}(k)} \cdot m$ time. They do so by giving a list of *compact representations*, a set $\mathcal{C}$ of pairwise disjoint vertex subsets such that choosing exactly one vertex from every set results in a minimal feedback vertex set. Our descriptions allow us to *count* the number of maximum-weight k-secluded trees containing a specified vertex in the same running time.

Techniques Rather than using recursive understanding, our algorithm is based on bounded-depth branching with a non-trivial progress measure. Similarly to existing algorithms to compute spanning trees with many leaves [88], our algorithm iteratively grows the vertex set of a k-secluded tree T. If we select a vertex v in the neighborhood of the current tree T, then for any k-secluded supertree T' of T there are two possibilities: either v belongs to the neighborhood of T', or it is contained in T'; the latter case can only happen if v has exactly one neighbor in T. Solutions of the first kind can be found by deleting v from the graph and searching for a $(k-1)$-secluded supertree of T. To find solutions of the second kind we can include v in T, but since the parameter does not decrease in this case we have to be careful that the recursion depth stays bounded. Using a reduction rule to deal with degree-1 vertices, we can essentially ensure that v has at least three neighbors (exactly one of which belongs to T), so that adding v to T strictly increases the open neighborhood size $|N(T)|$. Our main insight to obtain an FPT algorithm is a structural lemma showing that, whenever $|N(T)|$ becomes sufficiently large in terms of k, we can identify a vertex u that belongs to the open neighborhood of any k-secluded supertree $T' \supseteq T$. At that point, we can remove u and decrease k to make progress.

Organization We introduce our enumeration framework in Section 6.2. We present our algorithm that enumerates maximum-weight k-secluded trees and present its correctness and running time analysis in Section 6.3. We give some conclusions in Section 6.4.

6.2 Framework for enumerating secluded trees

In this section we introduce the framework for enumerating secluded trees, including our notion of a description which captures a potentially exponential number of secluded trees. To see why such a notion is required we first argue that enumerating all maximum-weight k-secluded trees individually cannot be done in FPT

time. Consider the graph in Figure 6.1. In this graph there are $\mathcal{O}(k \cdot (\frac{n}{k})^k)$ maximum-weight k-secluded trees, which cannot all be listed in time $f(k) \cdot n^{\mathcal{O}(1)}$. However, it is possible to give one short description for such an exponential number of k-secluded trees.

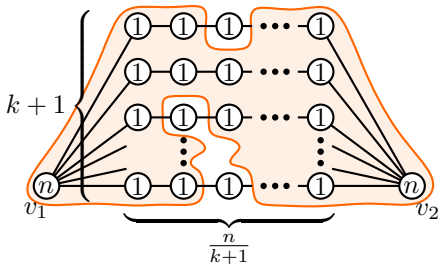


Figure 6.1 A vertex-weighted graph G on $n+2$ vertices. All vertices have weight 1 except for v_1 and v_2 which have weight n. There are $k+1$ v_1v_2-paths, each with $\frac{n}{k+1}$ internal vertices. There are $\mathcal{O}(k \cdot (\frac{n}{k})^k)$ maximum-weight k-secluded trees consisting of all vertices except one vertex out of exactly k paths. One such k-secluded tree is indicated with orange.

Definition 6.2.1. For a graph G, a *description* is a pair $(r, \mathcal{X})$ consisting of a vertex $r \in V(G)$ and a set $\mathcal{X}$ of pairwise disjoint subsets of $V(G-r)$ such that for any set S consisting of exactly one vertex from each set $X \in \mathcal{X}$, the connected component H of $G - S$ containing r is acyclic and $N(H) = S$, i.e., H is a $|\mathcal{X}|$-secluded tree in G. The *order* of a description is equal to $|\mathcal{X}|$. We say that a k-secluded tree H is *described by* a description $(r, \mathcal{X})$ if $N(H)$ consists of exactly one vertex of each $X \in \mathcal{X}$ and $r \in V(H)$.

Definition 6.2.2. For a graph G, a set of descriptions $\mathfrak{X}$ of maximum order k is called *redundant* for G if there is a k-secluded tree H in G such that H is described by two distinct descriptions in $\mathfrak{X}$. We say $\mathfrak{X}$ is *non-redundant* for G otherwise.

Definition 6.2.3. For a graph G and a set of descriptions $\mathfrak{X}$ of maximum order k, let $\mathcal{T}_G(\mathfrak{X})$ denote the set of all k-secluded trees in G described by a description in $\mathfrak{X}$.

Observation 6.2.4. *For a graph G and two sets of descriptions $\mathfrak{X}_1, \mathfrak{X}_2$ we have:*

$$\mathcal{T}_G(\mathfrak{X}_1) \cup \mathcal{T}_G(\mathfrak{X}_2) = \mathcal{T}_G(\mathfrak{X}_1 \cup \mathfrak{X}_2).$$

Observation 6.2.5. *Consider a graph G, a set of descriptions $\mathfrak{X}$, and vertex sets X_1, X_2 disjoint from $\bigcup_{(r,\mathcal{X})\in\mathfrak{X}}(\{r\} \cup \bigcup_{X\in\mathcal{X}} X)$. The set $\mathcal{T}_G(\{(r, \mathcal{X} \cup \{X_1 \cup X_2\}) \mid (r, \mathcal{X}) \in \mathfrak{X})\}$ equals:*

$$\mathcal{T}_G(\{(r, \mathcal{X} \cup \{X_1\}) \mid (r, \mathcal{X}) \in \mathfrak{X})\} \cup (r, \mathcal{X} \cup \{X_2\}) \mid (r, \mathcal{X}) \in \mathfrak{X})\}).$$

For an induced subgraph H of G and a set $F \subseteq V(G)$, we say that H is a supertree of F if H induces a tree and $F \subseteq V(H)$. Let $\mathcal{S}_G^k(F)$ be the set of all k-secluded supertrees of F in G. For a set X of subgraphs of G let $\text{maxset}_w(X) := \{H \in X \mid w(H) \geq w(H') \text{ for all } H' \in X\}$ (recall from Section 2.2 that $w(H) = w(V(H)) = \bigcup_{v \in V(H)}$). We focus our attention to the following version of the problem, where some parts of the tree are already given.

Enumerate Large Secluded Supertrees (ELSS)
Input: A graph G, a non-negative integer k, non-empty vertex sets $T \subseteq F \subseteq V(G)$ such that $G[T]$ is connected, and a weight function $w\colon V(G) \to \mathbb{N}^+$.
Parameter: k
Task: Find a non-redundant set $\mathfrak{X}$ of descriptions such that $\mathcal{T}_G(\mathfrak{X}) = \text{maxset}_w(\mathcal{S}_G^k(F))$.

In the end we solve the general enumeration problem by solving ELSS with $F = T = \{v\}$ for each $v \in V(G)$ and reporting only those k-secluded trees of maximum weight. Intuitively, our algorithm for ELSS finds k-secluded trees that "grow" out of T. In order to derive some properties of the types of descriptions we compute, we may at certain points demand that certain vertices non-adjacent to T need to end up in the k-secluded tree. For this reason the input additionally has a set F, rather than just T.

Our algorithm solves smaller instances recursively. We use the following abuse of notation: in an instance with graph G and weight function $w\colon V(G) \to \mathbb{N}^+$, when solving the problem recursively for an instance with induced subgraph G' of G, we keep the weight function w instead of restricting the domain of w to $V(G')$.

Observation 6.2.6. *For a graph G, a vertex $v \in V(G)$, and an integer $k \geq 1$, if H is a $(k-1)$-secluded tree in $G - v$, then H is a k-secluded tree in G. Consequently, $\mathcal{S}_{G-v}^{k-1}(F) \subseteq \mathcal{S}_G^k(F)$ for any $F \subseteq V(G)$.*

Observation 6.2.7. *For a graph G, a vertex $v \in V(G)$, and an integer $k \geq 1$, if H is a k-secluded tree in G with $v \in N_G(H)$, then H is a $(k-1)$-secluded tree in $G-v$. Consequently, $\{H \in \mathcal{S}_G^k(F) \mid v \in N_G(H)\} \subseteq \mathcal{S}_{G-v}^{k-1}(F)$ for any $F \subseteq V(G)$.*

6.3 Enumerate large secluded supertrees

Section 6.3.1 proves the correctness of a few subroutines used by the algorithm. Section 6.3.2 describes the algorithm to solve ELSS. In Section 6.3.3 we prove its correctness and in Section 6.3.4 we analyze its time complexity. In Section 6.3.5 we show how the algorithm for ELSS can be used to count and enumerate maximum-weight k-secluded trees containing a specified vertex.

6.3.1 Subroutines for the algorithm

Similar to the Feedback Vertex Set algorithm given by Guo et al. [68], we aim to get rid of degree-1 vertices. In our setting there is one edge case however. The

reduction rule is formalized as follows.

Definition 6.3.1. For an ELSS instance (G, k, F, T, w) with a degree-1 vertex v in G such that $F \neq \{v\}$, *contracting v into* its neighbor u yields the ELSS instance $(G - v, k, F', T', w')$ where the weight of u is increased by $w(v)$ and:

$$F' = \begin{cases} (F \setminus \{v\}) \cup \{u\} & \text{if } v \in F \\ F & \text{otherwise} \end{cases} \qquad T' = \begin{cases} (T \setminus \{v\}) \cup \{u\} & \text{if } v \in T \\ T & \text{otherwise.} \end{cases}$$

We prove the correctness of the reduction rule, that is, the descriptions of the reduced instance form the desired output for the original instance.

Lemma 6.3.2. *Let $I = (G, k, F, T, w)$ be an* ELSS *instance. Suppose G contains a degree-1 vertex v such that $\{v\} \neq F$. Let $I' = (G - v, k, F', T', w')$ be the instance obtained by contracting v into its neighbor u. If $\mathfrak{X}$ is a non-redundant set of descriptions for $G - v$ such that $\mathcal{T}_{G-v}(\mathfrak{X}) = \text{maxset}_{w'}(\mathcal{S}^k_{G-v}(F'))$, then $\mathfrak{X}$ is a non-redundant set of descriptions for G such that $\mathcal{T}_G(\mathfrak{X}) = \text{maxset}_w(\mathcal{S}^k_G(F))$.*

Proof. We first argue that $\mathfrak{X}$ is a valid set of descriptions for the graph G. For any $(r, \mathcal{X}) \in \mathfrak{X}$, we have $r \in V(G - v)$ and $\mathcal{X}$ consists of disjoint subsets of $V(G - \{v, r\})$, which trivially implies that $r \in V(G)$ and that $\mathcal{X}$ consists of disjoint subsets of $V(G - r)$. Consider any set S consisting of exactly one vertex from each $X \in \mathcal{X}$. The connected component H of $(G - v) - S$ containing r is acyclic and $N_{G-v}(H) = S$ since $\mathfrak{X}$ is a description for $G - v$. Let H' be the connected component of $G - S$ containing r. Note that $V(H) \subseteq V(H') \subseteq V(H) \cup \{v\}$. Clearly H' is acyclic as it is obtained from H by possibly adding a degree-1 vertex. We argue that $N_G(H') = S$. By construction of H' we have $N_G(H') \subseteq S$. For the sake of contradiction suppose that there is some $p \in S \setminus N_G(H')$. Since $N_{G-v}(H) = S$, there is some vertex $q \in V(H)$ such that $p \in N_{G-v}(q)$. But since $q \in V(H')$, we have that $p \in N_G(q)$ and so $p \in N_G(H')$; a contradiction to the containment of $p \in S \setminus N_G(H')$. It follows that $N_G(H') = S$.

Observe that any maximum-weight k-secluded supertree H of F in G containing u, contains its neighbor v as well: adding v to an induced tree subgraph containing u does not introduce cycles since v has degree one, does not increase the size of the neighborhood, and strictly increases the weight since $w(v) > 0$ by definition. Conversely, any (maximum-weight) k-secluded supertree H of F in G that contains v also contains u: tree H contains all vertices of the non-empty set F and $F \neq \{v\}$, so H contains at least one vertex other than v, which implies by connectivity that it contains the unique neighbor u of v. Hence a maximum-weight k-secluded supertree H of F in G contains u if and only if it contains v.

Using this fact, we relate the sets $\text{maxset}_w(\mathcal{S}^k_G(F))$ and $\text{maxset}_{w'}(\mathcal{S}^k_{G-v}(F'))$. For any $H \in \text{maxset}_w(\mathcal{S}^k_G(F))$, there is a k-secluded supertree of F' in $G - v$ of the same weight under w':

- If $v \in H$, then $u \in H$ as argued above. Now observe that $H - v$ is a k-secluded tree in $G - v$ that contains u. Since the weight of u was increased by $w(v)$

in the transformation, we have $w(H) = w'(H - v)$. Since H is a supertree of F and $H - v$ contains u, the latter is a supertree of $F' \subseteq (F \setminus \{v\}) \cup \{u\}$.

- If $v \notin H$, then $u \notin H$ and therefore $w(H) = w'(H)$ and $N_G(H) = N_{G-v}(H)$. As $v \notin H$ while H is a supertree of $F \supseteq T$ we have $v \notin F \cup T$, which shows $F' = F$ and $T' = T$ so that H is a k-secluded supertree of F' in $G - v$.

Conversely, for any k-secluded supertree H' of F' in $G-v$, there is a k-secluded supertree of F in G of the same weight under w: if $u \notin H'$ then H' itself is such a tree, otherwise $G[V(H') \cup \{v\}]$ is such a tree.

These transformations imply that the maximum w-weight of trees in $\mathcal{S}_G^k(F)$ is identical to the maximum w'-weight of trees in $\mathcal{S}_{G-v}^k(F')$. Since any $H \in \mathrm{maxset}_w(\mathcal{S}_G^k(F))$ contains u if and only if it contains v, they also show that an induced subgraph H of G containing either both u and v or neither belongs to $\mathrm{maxset}_w(\mathcal{S}_G^k(F))$ if and only if $H - v \in \mathrm{maxset}_{w'}(\mathcal{S}_{G-v}^k(F'))$; note that this holds regardless of whether $v \in H$. To show that $\mathcal{T}_G(\mathfrak{X}) = \mathrm{maxset}_w(\mathcal{S}_G^k(F))$ it now suffices to observe that if $(r, \mathcal{X}) \in \mathfrak{X}$ describes a tree $H - v \in \mathrm{maxset}_{w'}(\mathcal{S}_{G-v}^k(F'))$ via the set S containing exactly one vertex of each $X \in \mathcal{X}$, such that $H - v$ is the connected component of $(G - v) - S$ containing r, then the connected component of $G - S$ containing r is exactly H, which again holds regardless of whether $v \in H$. Hence $\mathcal{T}_G(\mathfrak{X}) = \mathrm{maxset}_w(\mathcal{S}_G^k(F))$, and the set of descriptions is non-redundant for G since $\mathfrak{X}$ is non-redundant for $G - v$. □

We say an instance is *almost leafless* if the lemma above cannot be applied, that is, if G contains a vertex v of degree 1, then $F = \{v\}$.

Lemma 6.3.3. *There is an algorithm that, given an almost leafless* ELSS *instance* (G, k, F, T, w) *such that* $k > 0$ *and* $|N_G(T)| > k(k+1)$*, runs in time* $\mathcal{O}(k \cdot n^3)$ *and either:*

1. *finds a vertex* $v \in V(G) \setminus F$ *such that any* k*-secluded supertree* H *of* F *in* G *satisfies* $v \in N_G(H)$*, or*

2. *concludes that* G *does not contain a* k*-secluded supertree of* F*.*

Proof. We aim to find a vertex $v \in V(G) \setminus F$ with $k+2$ distinct paths $P_1, \ldots, P_{k+2}$ from $N_G(T)$ to v that intersect only in v and do not contain vertices from T. We first argue that such a vertex v satisfies the first condition, if it exists. Consider some k-secluded supertree H of F. Since the paths $P_1, \ldots, P_{k+2}$ are disjoint apart from their common endpoint v while $|N_G(H)| \leq k$, there are two paths P_i, P_j with $i \neq j \in [k+2]$ for which $P_i \setminus \{v\}$ and $P_j \setminus \{v\}$ do not intersect $N_G(H)$. Since they start in $N_G(T)$, the paths $P_i \setminus \{v\}$ and $P_j \setminus \{v\}$ are contained in H. As P_i and P_j form a cycle together with a path through the connected set T, which cannot be contained in the acyclic graph H, this implies $v \in N_G(H)$.

Next we argue that if G has a k-secluded supertree H of $F \supseteq T$, then there exists such a vertex v. Consider an arbitrary such H and root it at a vertex $t \in T$.

For each vertex $u \in N_G(T)$, we construct a path P_u disjoint from T that starts in u and ends in $N_G(H)$, as follows.

- If $u \notin H$, then $u \in N_G(H)$ and we take $P_u = (u)$.
- If $u \in H$, then let ℓ_u be an arbitrary leaf in the subtree of H rooted at u; possibly $u = \ell_u$. Since T is connected and $H \supseteq T$ is acyclic and rooted in $t \in T$, the subtree rooted at $u \in N_G(T) \cap H$ is disjoint from T. Hence $\ell_u \notin T$, so that $F \neq \{\ell_u\}$. As the instance is almost leafless we therefore have $\deg_G(\ell_u) > 1$. Because ℓ_u is a leaf of H this implies that $N_G(\ell_u)$ contains a vertex y other than the parent of ℓ_u in H, so that $y \in N_G(H)$. We let P_u be the path from u to ℓ_u through H, followed by the vertex $y \in N_G(H)$.

The paths we construct are distinct since their startpoints are. Two constructed paths cannot intersect in any vertex other than their endpoints, since they were extracted from different subtrees of H. Since we construct $|N_G(T)| > k(k+1)$ paths, each of which ends in $N_G(H)$ which has size at most k, some vertex $v \in N_G(H)$ is the endpoint of $k+2$ of the constructed paths. As shown in the beginning the proof, this establishes that v belongs to the neighborhood of any k-secluded supertree of F. Since $F \subseteq V(H)$ we have $v \notin F$.

All that is left to show is that we can find such a vertex v in the promised time bound. After contracting T into a source vertex s, for each $v \in V(G) \setminus F$, do $k+2$ iterations of the Ford-Fulkerson algorithm in order to check if there are $k+2$ internally vertex-disjoint sv-paths. If so, then return v. If for none of the choices of v this holds, then output that there is no k-secluded supertree of F in G. In order to see that this satisfies the claimed running time bound, note that there are $\mathcal{O}(n)$ choices for v, and $k+2$ iterations of Ford-Fulkerson runs can be implemented to run in $\mathcal{O}(k \cdot (n+m))$ time. □

6.3.2 The algorithm

Consider an input instance (G, k, F, T, w) of ELSS. If $G[F]$ contains a cycle, return $\emptyset$. Otherwise we remove all connected components of G that do not contain a vertex of F. If more than one connected component remains, return $\emptyset$. Then, while there is a degree-1 vertex v such that $F \neq \{v\}$, contract v into its neighbor as per Definition 6.3.1. While $N_G(T)$ contains a vertex $v \in F$, add v to T. Finally, if $N_G(F) = \emptyset$, return $\{(r, \emptyset)\}$ for some $r \in F$. Otherwise if $k = 0$, return $\emptyset$.

We proceed by considering the neighborhood of T as follows:

1. If any vertex $v \in N_G(T)$ has two neighbors in T, then recursively run this algorithm to obtain a set of descriptions $\mathfrak{X}'$ for $(G - v, k-1, F, T, w)$ and return $\{(r, \mathcal{X} \cup \{\{v\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}'\}$.
2. If $|N_G(T)| > k(k+1)$, apply Lemma 6.3.3. If it concludes that G does not contain a k-secluded supertree of F, return $\emptyset$. Otherwise let $v \in V(G) \setminus F$ be

the vertex it finds, obtain a set of descriptions $\mathfrak{X}'$ for $(G-v, k-1, F, T, w)$ and return $\{(r, \mathcal{X} \cup \{\{v\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}'\}$.

3. Pick some $v \in N_G(T)$ and let $P = (v = v_1, v_2, \ldots, v_\ell)$ be the unique[1] maximal path disjoint from T satisfying $\deg_G(v_i) = 2$ for each $1 \leq i < \ell$ and ($v_\ell \in N_G(T)$ or $\deg_G(v_\ell) > 2$).

   (a) If $v_\ell \notin F$, obtain a set of descriptions $\mathfrak{X}_1$ by recursively solving $(G - v_\ell, k-1, F, T, w)$. Otherwise take $\mathfrak{X}_1 = \emptyset$. (We find the k-secluded trees avoiding v_ℓ but containing $P - v_\ell$.)

   (b) If $P - F - v_\ell \neq \emptyset$, obtain a set of descriptions $\mathfrak{X}_2$ by recursively solving $(G - V(P - v_\ell), k-1, (F \setminus V(P)) \cup \{v_\ell\}, T, w)$. Otherwise take $\mathfrak{X}_2 = \emptyset$. (We find the k-secluded trees containing both endpoints of P which have one vertex in P as a neighbor.)

   (c) If $G[F \cup V(P)]$ is acyclic, obtain a set of descriptions $\mathfrak{X}_3$ by recursively solving $(G, k, F \cup V(P), T \cup V(P), w)$. Otherwise take $\mathfrak{X}_3 = \emptyset$. (We find the k-secluded trees containing the entire path P.)

   Let M be the set of minimum weight vertices in $P - F - v_\ell$ and define:

$$\begin{aligned}\mathfrak{X}'_1 &:= \{(r, \mathcal{X} \cup \{\{v_\ell\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}_1\}\\ \mathfrak{X}'_2 &:= \{(r, \mathcal{X} \cup \{M\}) \mid (r, \mathcal{X}) \in \mathfrak{X}_2\}\\ \mathfrak{X}'_3 &:= \mathfrak{X}_3.\end{aligned}$$

   For each $i \in [3]$ let w_i be the weight of an arbitrary $H \in \mathcal{T}_G(\mathfrak{X}'_i)$, or 0 if $\mathfrak{X}'_i = \emptyset$. Return the set $\mathfrak{X}'$ defined as $\bigcup_{\{i \in [3] \mid w_i = \max\{w_1, w_2, w_3\}\}} \mathfrak{X}'_i$.

6.3.3 Proof of correctness

In this section we argue that the algorithm described in Section 6.3.2 solves the ELSS problem. In various steps we identify a vertex v such that the neighborhood of any maximum-weight k-secluded supertree must include v. We argue that for these steps, the descriptions of the current instance can be found by adding $\{v\}$ to every description of the supertrees of T in $G - v$ if some preconditions are satisfied.

Lemma 6.3.4. *Let (G, k, F, T, w) be an* ELSS *instance and let $v \in V(G) \setminus F$. Let $\mathfrak{X}$ be a set of descriptions for $G - v$ such that $\mathcal{T}_{G-v}(\mathfrak{X}) = \mathrm{maxset}_w(\mathcal{S}^{k-1}_{G-v}(F))$ and $v \in N_G(H)$ for all $H \in \mathcal{T}_{G-v}(\mathfrak{X})$. Then we have:*

$$\mathcal{T}_G\left(\{(r, \mathcal{X} \cup \{\{v\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}\}\right) = \mathrm{maxset}_w\{H \in \mathcal{S}^k_G(F) \mid v \in N_G(H)\}.$$

[1] To construct P, initialize $P := (v = v_1)$; then while $\deg_G(v_{|V(P)|}) = 2$ and $N_G(v_{|V(P)|}) \setminus (V(P) \cup T)$ consists of a single vertex, append that vertex to P.

Proof. First observe that $\mathfrak{X}' = \{(r, \mathcal{X} \cup \{\{v\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}\}$ is a valid set of descriptions for G since $v \in N_G(H)$ for all $H \in \mathcal{T}_{G-v}(\mathfrak{X})$.

Consider a maximum-weight (with respect to w) k-secluded supertree $H \in \mathcal{S}_G^k(F)$ such that $v \in N_G(H)$. We show that it is contained in $\mathcal{T}_G(\mathfrak{X}')$. By Observation 6.2.7 we have that $H \in \mathcal{S}_{G-v}^{k-1}(F)$, that is, H is a $(k-1)$-secluded supertree of F in $G - v$. We argue that $H \in \text{maxset}_w(\mathcal{S}_{G-v}^{k-1}(F))$. For the sake of contradiction, suppose there is $H' \in \text{maxset}_w(\mathcal{S}_{G-v}^{k-1}(F))$ such that $w(H') > w(H)$. By Observation 6.2.6 it follows that H' is a k-secluded supertree of F in G. This contradicts the fact that H is of maximum weight among such supertrees. It follows that $H \in \text{maxset}_w(\mathcal{S}_{G-v}^{k-1}(F))$. By the assumption that $\mathcal{T}_{G-v}(\mathfrak{X}) = \text{maxset}_w(\mathcal{S}_{G-v}^{k-1}(F))$, we have that there is a description $(r, \mathcal{X}) \in \mathfrak{X}$ for $G - v$ that describes H. Since $(r, \mathcal{X} \cup \{\{v\}\}) \in \mathfrak{X}'$ is a description for H in G, we have that $H \in \mathcal{T}_G(\mathfrak{X}')$ as required.

In the other direction, consider some tree $J \in \mathcal{T}_G(\mathfrak{X}')$. We show that $J \in \text{maxset}_w\{H \in \mathcal{S}_G^k(F) \mid v \in N_G(H)\}$. Let $(r, \mathcal{X} \cup \{\{v\}\}) \in \mathfrak{X}'$ be a description that describes J. By Definition 6.2.1 we have $v \in N_G(J)$. Since $(r, \mathcal{X}) \in \mathfrak{X}$ describes J in $G - v$ and $\mathcal{T}_{G-v}(\mathfrak{X}) = \text{maxset}_w(\mathcal{S}_{G-v}^{k-1}(F))$, we have that $J \in \text{maxset}_w(\mathcal{S}_{G-v}^{k-1}(F))$. Since $v \in N_G(J)$, by Observation 6.2.6 we have that $J \in \{H \in \mathcal{S}_G^k(F) \mid v \in N_G(H)\}$. For the sake of contradiction, suppose that there is $J' \in \{H \in \mathcal{S}_G^k(F) \mid v \in N_G(H)\}$ such that $w(J') > w(J)$. Then we get that $J' \in \text{maxset}_w(\mathcal{S}_{G-v}^{k-1}(F))$, but this contradicts that $J \in \text{maxset}_w(\mathcal{S}_{G-v}^{k-1}(F))$. It follows that $J \in \text{maxset}_w\{H \in \mathcal{S}_G^k(F) \mid v \in N_G(H)\}$ as required. □

The next lemma will be used to argue correctness of Step 3(b) of the algorithm, in which we find k-secluded trees which avoid a single vertex from path P.

Lemma 6.3.5. *Let (G, k, F, T, w) be an* ELSS *instance and let P be a path in G with $\deg_G(v) = 2$ for all $v \in V(P)$ and $N_G(P) = \{a, b\}$ for some $a, b \in F$. Let $\mathfrak{X}$ be a set of descriptions for $G - V(P)$ such that $\mathcal{T}_{G-V(P)}(\mathfrak{X}) = \text{maxset}_w(\mathcal{S}_{G-V(P)}^{k-1}(F \setminus V(P)))$. Then for all $p \in V(P) \setminus F$ we have:*

$$\mathcal{T}_G(\{(r, \mathcal{X} \cup \{\{p\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}\}) = \text{maxset}_w\{H \in \mathcal{S}_G^k(F) \mid p \in N_G(H)\}.$$

Proof. For any $p \in V(P)$ we define $\mathfrak{X}^p = \{(r, \mathcal{X} \cup \{\{p\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}\}$. We show $\mathfrak{X}^p$ is a valid set of descriptions for G. Note that for any set S consisting of exactly one vertex from each set $X \in \mathcal{X} \cup \{\{p\}\}$ there is an $H \in T_{G-V(P)}(\mathfrak{X})$ with $N_G(H) = S \setminus \{p\}$ and $r \in V(H)$ since $\mathfrak{X}$ is a valid set of descriptions for $G - V(P)$. Since $\{a, b\} \subseteq V(H)$ and $N_G(P) = \{a, b\}$, we have that $H' = G[V(H) \cup V(P)]$ is the connected component of $G - (S \setminus p)$ containing r. Observe that $H' - p$ is acyclic and connected and since $p \in V(P) \subseteq V(H')$ we have that $p \in N_G(H' - p)$, hence $N_G(H' - p) = S$ and $\mathfrak{X}^p$ is a valid set of descriptions for G.

Next we show $\mathcal{T}_G(\mathfrak{X}^p) \supseteq \text{maxset}_w\{H \in \mathcal{S}_G^k(F) \mid p \in N_G(H)\}$ for any $p \in V(P) \setminus F$. Consider a k-secluded supertree $H \in \mathcal{S}_G^k(F)$ which has maximum weight (with respect to w) among those satisfying $p \in N_G(H)$. We show that $H \in \mathcal{T}_G(\mathfrak{X}^p)$.

By Note 6.2.7 we have that $H \in \mathcal{S}^{k-1}_{G-p}(F)$, that is, H is a $(k-1)$-secluded supertree of F in $G-p$. Observe that $H - V(P)$ remains connected so $H - V(P) \in \mathcal{S}^{k-1}_{G-V(P)}(F \setminus V(P))$. We argue that $H - V(P) \in \text{maxset}_w(\mathcal{S}^{k-1}_{G-V(P)}(F \setminus V(P)))$. For the sake of contradiction, suppose there is $H' \in \text{maxset}_w(\mathcal{S}^{k-1}_{G-V(P)}(F \setminus V(P)))$ such that $w(H') > w(H - V(P))$. Observe that $H'' := G[V(H') \cup V(P-p)]$ is a connected acyclic subgraph of G with $N_G(H'') = N_G(H') \cup \{p\}$, i.e., H'' is a k-secluded supertree of F in G. Since $w(H'') = w(H') + w(P-p) > w(H - V(P)) + w(P-p) = w(H)$ this contradicts that $H \in \text{maxset}_w(\mathcal{S}^k_G(F))$. It follows that $H - V(P) \in \text{maxset}_w(\mathcal{S}^{k-1}_{G-V(P)}(F \setminus V(P)))$. Since it is given that $\mathcal{T}_{G-V(P)}(\mathfrak{X}) = \text{maxset}_w(\mathcal{S}^{k-1}_{G-V(P)}(F \setminus V(P)))$, we have that there is a description $(r, \mathcal{X}) \in \mathfrak{X}$ for $G - V(P)$ that describes $H - V(P)$. Then $(r, \mathcal{X} \cup \{\{p\}\}) \in \mathfrak{X}^p$ is a description for H in G and we conclude that $H \in \mathcal{T}_G(\mathfrak{X}^p)$ as required.

Finally we show $\mathcal{T}_G(\mathfrak{X}^p) \subseteq \text{maxset}_w\{H \in \mathcal{S}^k_G(F) \mid p \in N_G(H)\}$ for any $p \in V(P) \setminus F$. Consider some tree $J \in \mathcal{T}_G(\mathfrak{X}^p)$. We show that $J \in \text{maxset}_w\{H \in \mathcal{S}^k_G(F) \mid p \in N_G(H)\}$. Clearly $J \in \{H \in \mathcal{S}^k_G(F) \mid p \in N_G(H)\}$, so it remains to show that $w(J) \geq w(J')$ for all $J' \in \{H \in \mathcal{S}^k_G(F) \mid p \in N_G(H)\}$. Suppose for contradiction that there exists such a J' for which $w(J) < w(J')$. Observe that $J - V(P)$ and $J' - V(P)$ are both $(k-1)$-secluded supertrees of $F \setminus V(P)$ in $G - V(P)$, i.e., they are contained in $\mathcal{S}^{k-1}_{G-V(P)}(F \setminus V(P))$. Since $V(P-p) \subseteq V(J)$ and $V(P-p) \subseteq V(J')$ we have that $w(J - V(P)) < w(J' - V(P))$, so $J - V(P) \notin \text{maxset}_w(\mathcal{S}^{k-1}_{G-V(P)}(F \setminus V(P)))$. Recall that $J \in T_G(\mathfrak{X}^p)$ and consider the description $(r, \mathcal{X} \cup \{\{p\}\}) \in \mathfrak{X}^p$ that describes J. Observe that $(r, \mathcal{X})$ describes $J - V(P)$ in $G - V(P)$, i.e., $J - V(P) \in \mathcal{T}_{G-V(P)}(\mathfrak{X})$. However it is given that $\mathcal{T}_{G-V(P)}(\mathfrak{X}) = \text{maxset}_w(\mathcal{S}^{k-1}_{G-V(P)}(F \setminus V(P)))$, a contradiction. Hence $J \in \text{maxset}_w\{H \in \mathcal{S}^k_G(F) \mid p \in N_G(H)\}$. □

The following lemma is used to argue that the problem to solve in Step 3 of the algorithm reduces to the three problems solved in the recursive calls.

Lemma 6.3.6. *Let (G, k, F, T, w) be an almost leafless* ELSS *instance such that G is connected and $N_G(F) \neq \emptyset$. Fix some $v \in N_G(T)$ and let $P = (v = v_1, v_2, \ldots, v_\ell)$ be the unique maximal path disjoint from T satisfying $\deg_G(v_i) = 2$ for each $1 \leq i < \ell$ and ($v_\ell \in N_G(T)$ or $\deg_G(v_\ell) > 2$). Then for any* maximum weight k*-secluded supertree H of F, exactly one of the following holds:*

1. *$v_\ell \in N(H)$ (so $v_\ell \notin F$),*
2. *$|N(H) \cap V(P - F - v_\ell)| = 1$ and $v_\ell \in V(H)$, or*
3. *$V(P) \subseteq V(H)$.*

Proof. First note that such a vertex v exists since $N_G(F) \neq \emptyset$ and G is connected, so $N_G(T) \neq \emptyset$. Furthermore since the instance is almost leafless, the path P is well defined. If there is no k-secluded supertree of F, then there is nothing to show. So

suppose H is a maximum-weight k-secluded supertree of F. We have $v \in V(P)$ is a neighbor of $T \subseteq F \subseteq V(H)$, so either $V(P) \subseteq V(H)$ or $V(P)$ contains a vertex from $N(H)$. In the first case Condition 3 holds, in the second case we have $|N(H) \cap V(P)| \geq 1$. First suppose that $|N(H) \cap V(P)| \geq 2$. Let $i \in [\ell]$ be the smallest index such that $v_i \in N(H) \cap V(P)$. Similarly let $j \in [\ell]$ be the largest such index. We show that in this case we can contradict the fact that H is a maximum-weight k-secluded supertree of F. Observe that $H' = V(H) \cup \{v_i, \ldots, v_{j-1}\}$ induces a tree since $(v_i, \ldots, v_{j-1})$ forms a path of degree-2 vertices and the neighbor v_j of v_{j-1} is not in H. Furthermore H' has a strictly smaller neighborhood than H and it has larger weight as vertices have positive weight. Since $F \subseteq V(H')$, this contradicts that H is a maximum-weight k-secluded supertree of F.

We conclude that $|N(H) \cap V(P)| = 1$. Let $i \in [\ell]$ be the unique index such that $N(H) \cap V(P) = \{v_i\}$. Clearly $v_i \notin F$. In the case that $i = \ell$, then Condition 1 holds. Otherwise if $i < \ell$, the first condition of Condition 2 holds. In order to argue that the second condition also holds, suppose that $v_\ell \notin V(H)$. Then $H \cup \{v_i, \ldots, v_{\ell-1}\}$ is a k-secluded supertree of F in G and it has larger weight than H as vertices have positive weight. This contradicts the fact that H has maximum weight, hence the second condition of Condition 2 holds as well. □

Armed with Lemmas 6.3.4 to 6.3.6 we are now ready to prove correctness of the algorithm.

Lemma 6.3.7. *The algorithm described in Section 6.3.2 is correct.*

Proof. Let $I = (G, k, F, T, w)$ be an ELSS instance. We prove correctness by induction on $|V(G) \setminus F|$. Assume the algorithm is correct for any input $(\hat{G}, \hat{k}, \hat{F}, \hat{T}, \hat{w})$ with $|V(\hat{G}) \setminus \hat{F}| < |V(G) \setminus F|$.

Before Step 1 We first prove correctness when the algorithm terminates before Step 1, which includes the base case of the induction. Note that if $G[F]$ contains a cycle, then no induced subgraph H of G with $F \subseteq V(H)$ can be acyclic. Therefore the set of maximum-weight k-secluded trees containing F is the empty set, so we correctly return $\emptyset$. Otherwise $G[F]$ is acyclic. Clearly any connected component of G that has no vertices of F can be removed. If there are two connected components of G containing vertices of F, then no induced subgraph of G containing all of F can be connected, again we correctly return the empty set. In the remainder we have that G is connected.

By iteratively applying Lemma 6.3.2 we conclude that a solution to the instance obtained after iteratively contracting (most) degree-1 vertices is also a solution to the original instance. Hence we can proceed to solve the new instance, which we know is almost leafless. In addition, observe that the contraction of degree-1 vertices maintains the property that G is connected and $G[F]$ is acyclic.

After exhaustively adding vertices $v \in N_G(T) \cap F$ to T we have that $G[T]$ is a connected component of $G[F]$. In the case that $N_G(F) = \emptyset$, then since G is

connected it follows that $F = T = V(G)$ and therefore T is the only maximum-weight k-secluded tree. For any $r \in V(G)$, the description $(r, \emptyset)$ describes this k-secluded tree, so we return $\{(r, \emptyset)\}$. In the remainder we have $N_G(F) \neq \emptyset$.

Since $N_G(F) \neq \emptyset$ and G is almost leafless, we argue that there is no 0-secluded supertree of F. Suppose G contains a 0-secluded supertree H of F, so $|N_G(H)| = 0$ and since $H \supseteq F$ is non-empty and G is connected we must have $H = G$, hence G is a tree with at least two vertices (since F and $N_G(F)$ are both non-empty) so G contains at least two vertices of degree-1, contradicting that G is almost leafless. So there is no k-secluded supertree of F in G and the algorithm correctly returns $\emptyset$ if $k = 0$.

Observe that the value $|V(G) \setminus F|$ cannot have increased since the start of the algorithm since we never add vertices to G and any time we remove a vertex from F it is also removed from G. Hence we can still assume in the remainder of the proof that the algorithm is correct for any input $(\hat{G}, \hat{k}, \hat{F}, \hat{T}, \hat{w})$ with $|V(\hat{G}) \setminus \hat{F}| < |V(G) \setminus F|$. To conclude this part of the proof, we have established that if the algorithm terminates before reaching Step 1, then its output is correct. On the other hand, if the algorithm continues we can make use of the following properties of the instance just before reaching Step 1:

Property 6.3.8. *If the algorithm does not terminate before reaching Step 1 then*

(i) the ELSS *instance* (G, k, F, T, w) *is almost leafless,*

(ii) $G[F]$ *is acyclic,*

(iii) $G[T]$ *is a connected component of* $G[F]$*,*

(iv) G *is connected,*

(v) $k > 0$*, and*

(vi) $N_G(F) \neq \emptyset$*.*

Step 1 Before arguing that the return value in Step 1 is correct, we observe the following.

Claim 6.3.9. If H is an induced subtree of G that contains T and $v \in N_G(T)$ has at least two neighbors in T, then $v \in N_G(H)$.

Proof. Suppose $v \notin N_G(H)$, then since $v \in N_G(T)$ and $T \subseteq V(H)$ we have that $v \in V(H)$. But then since T is connected, subgraph H contains a cycle. This contradicts that H is a tree and confirms that $v \in N_G(H)$. ⌟

Now consider the case that in Step 1 we find a vertex $v \in N_G(T)$ with two neighbors in T, and let $\mathfrak{X}'$ be the set of descriptions as obtained by the algorithm through recursively solving the instance $(G-v, k-1, F, T, w)$. Since $|V(G-v) \setminus F| < |V(G) \setminus F|$ (as $v \notin F$) we know by induction that $\mathcal{T}_{G-v}(\mathfrak{X}')$ is the set of all maximum-weight

$(k-1)$-secluded supertrees of F in $G-v$. Any $H \in \mathcal{T}_{G-v}(\mathfrak{X}')$ is an induced subtree of G with $T \subseteq V(H)$, so by Claim 6.3.9 we have $v \in N_G(H)$ for all $H \in \mathcal{T}_{G-v}(\mathfrak{X}')$. We can now apply Lemma 6.3.4 to conclude that $\mathcal{T}_G(\{(r, \mathcal{X} \cup \{\{v\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}'\})$ is the set of all maximum-weight k-secluded supertrees H of F in G for which $v \in N_G(H)$. Again by Claim 6.3.9 we have that $v \in N_G(H)$ for all such k-secluded supertrees of F, hence $\mathcal{T}_G(\{(r, \mathcal{X} \cup \{\{v\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}'\})$ is the set of all maximum-weight k-secluded supertrees of F in G. We argue non-redundancy of the output. Suppose that two descriptions $(r, \mathcal{X} \cup \{\{v\}\})$ and $(r', \mathcal{X}' \cup \{\{v\}\})$ describe the same supertree H of F in G. Note that then $(r, \mathcal{X})$ and $(r', \mathcal{X})$ describe the same supertree H of F in $G - v$, which contradicts the induction hypothesis that the output of the recursive call was correct and therefore non-redundant.

Concluding this part of the proof, we showed that if the algorithm terminates during Step 1, then its output is correct. On the other hand, if the algorithm continues after Step 1 we can make use of the following in addition to Property 6.3.8.

Property 6.3.10. *If the algorithm does not terminate before reaching Step 2 then no vertex $v \in N_G(T)$ has two neighbors in T.*

Step 2 In Step 2 we use Lemma 6.3.3 if $|N_G(T)| > k(k+1)$. The preconditions of the lemma are satisfied since $k > 0$ and the instance is almost leafless by Property 6.3.8. If it concludes that G does not contain a k-secluded supertree of F, then the algorithm correctly outputs $\emptyset$. Otherwise it finds a vertex $v \in V(G) \setminus F$ such that any k-secluded supertree H of F in G satisfies $v \in N_G(H)$. We argue that the algorithm's output is correct. Let $\mathfrak{X}'$ be the set of descriptions as obtained through recursively solving $(G - v, k - 1, F, T, w)$. Since $v \notin F$ we have $|(V(G - v) \setminus F| < |V(G) \setminus F|$, so by induction we have that $\mathcal{T}_{G-v}(\mathfrak{X}')$ is the set of all maximum-weight $(k-1)$-secluded supertrees of F in $G-v$. Furthermore by Observation 6.2.6 for any $H \in \mathcal{T}_{G-v}(\mathfrak{X}') = \mathcal{S}^{k-1}_{G-v}(F)$ we have $H \in \mathcal{S}^k_G(F)$, and therefore $v \in N_G(H)$. It follows that Lemma 6.3.4 applies to $\mathfrak{X}'$ so we can conclude that $\mathcal{T}_G(\{(r, \mathcal{X} \cup \{\{v\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}'\})$ is the set of maximum-weight k-secluded supertrees H of F in G for which $v \in N_G(H)$. Since we know there are no k-secluded supertrees H of F in G for which $v \notin N_G(H)$, it follows that $\mathcal{T}_G(\{(r, \mathcal{X} \cup \{\{v\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}\})$ is the set of maximum-weight k-secluded supertrees of F in G as required. Non-redundancy of the output follows as in Step 1.

To summarize the progress so far, we have shown that if the algorithm terminates before it reaches Step 3, then its output is correct. Alternatively, if we proceed to Step 3 we can make use of the following property, in addition to Properties 6.3.8 and 6.3.10, which we will use later in the running time analysis.

Property 6.3.11. *If the algorithm does not terminate before reaching Step 3 then $|N_G(T)| \leq k(k+1)$.*

Step 3 Fix some $v \in N_G(T)$, which exists as $N_G(T) \neq \emptyset$ by Property 6.3.8. Let $P = (v = v_1, \ldots, v_\ell)$ be a path as described in Lemma 6.3.6. By Lemma 6.3.6 we can partition the set $\text{maxset}_w(\mathcal{S}^k_G(F))$ of maximum-weight k-secluded supertrees of F in G into the following three sets:

- $\mathcal{T}_1 = \{H \in \text{maxset}_w(\mathcal{S}^k_G(F)) \mid v_\ell \in N_G(H)\}$,
- $\mathcal{T}_2 = \{H \in \text{maxset}_w(\mathcal{S}^k_G(F)) \mid |N(H) \cap V(P - F - v_\ell)| = 1 \text{ and } v_\ell \in V(H)\}$,
- $\mathcal{T}_3 = \{H \in \text{maxset}_w(\mathcal{S}^k_G(F)) \mid V(P) \subseteq V(H)\}$,

Consider the sets $\mathfrak{X}_1$, $\mathfrak{X}_2$, and $\mathfrak{X}_3$ of descriptions as obtained through recursion in Step 3 of the algorithm. By induction we have the following:

- $\mathcal{T}_{G-v_\ell}(\mathfrak{X}_1) = \text{maxset}_w(\mathcal{S}^{k-1}_{G-v_\ell}(F))$, since $|V(G - v_\ell) \setminus F| < |V(G) \setminus F|$,
- $\mathcal{T}_{G-V(P-v_\ell)}(\mathfrak{X}_2) = \text{maxset}_w(\mathcal{S}^{k-1}_{G-V(P-v_\ell)}((F \setminus V(P)) \cup \{v_\ell\}))$ since

$$|V(G - V(P - v_\ell)) \setminus ((F \setminus V(P)) \cup \{v_\ell\})| =$$
$$|V(G - V(P - v_\ell)) \setminus (F \cup \{v_\ell\})| < |V(G) \setminus F|, \text{ and}$$

- $\mathcal{T}_G(\mathfrak{X}_3) = \text{maxset}_w(\mathcal{S}^k_G(F \cup V(P)))$ since $|V(G) \setminus (F \cup V(P))| < |V(G) \setminus F|$.

Let $\mathfrak{X}'_1$, $\mathfrak{X}'_2$, and $\mathfrak{X}'_3$ be the sets of descriptions as computed in Step 3 of the algorithm.

Claim 6.3.12. The sets $\mathfrak{X}'_1$, $\mathfrak{X}'_2$, and $\mathfrak{X}'_3$ consist of valid descriptions for G.

Proof. To argue that $\mathfrak{X}'_1 = \{(r, \mathcal{X} \cup \{\{v_\ell\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}_1\}$ consists of valid descriptions for G we show for an arbitrary description $(r, \mathcal{X}) \in \mathfrak{X}_1$ for $G - v_\ell$ that $(r, \mathcal{X} \cup \{\{v_\ell\}\})$ is a valid description for G. Clearly $r \in V(G)$ and $\mathcal{X} \cup \{\{v_\ell\}\}$ consists of pairwise disjoint subsets of $V(G - r)$. Consider any set S' consisting of exactly one vertex from each set $X \in \mathcal{X} \cup \{\{v_\ell\}\}$. Clearly $v_\ell \in S'$. Let $S = S' \setminus \{v_\ell\}$. Since $(r, \mathcal{X})$ is a description for $G - v_\ell$, the connected component H of $G - v_\ell - S$ containing r is acyclic and satisfies $N_{G-v_\ell}(H) = S$. Note that the connected component of $G - S'$ containing r is identical to H and is therefore also acyclic. All that is left to argue is that $v_\ell \in N_G(H)$. If $\ell = 1$, then $v_\ell \in N_G(F)$ and the claim follows as H is a supertree of F. Otherwise note that since $(r, \mathcal{X}) \in \mathfrak{X}_1$ we have that $H \in \text{maxset}_w(\mathcal{S}^{k-1}_{G-v_\ell}(F))$, i.e., H is of maximum weight. Since all vertices of $V(P - v_\ell)$ have degree 2 in G, with $v_{\ell-1}$ adjacent to v_ℓ, the graph $G[V(H) \cup V(P - v_\ell)]$ is acyclic and $|N_G(H)| = |N_G(V(H) \cup V(P - v_\ell))|$. It follows that $V(P - v_\ell) \subseteq V(H)$ since otherwise the secluded tree $G[V(H) \cup V(P - v_\ell)]$ would have larger weight than H. Hence $v_{\ell-1} \in V(H)$ so $v_\ell \in N_G(H)$.

Next we argue $\mathfrak{X}'_2$ consists of valid descriptions for G. Recall that $\mathfrak{X}'_2 = \{(r, \mathcal{X} \cup \{M\}) \mid (r, \mathcal{X}) \in \mathfrak{X}_2\}$ where M is the set of minimum weight vertices in $P - F - v_\ell$, so it suffices to show for an arbitrary description $(r, \mathcal{X}) \in \mathfrak{X}_2$ for $G - V(P - v_\ell)$ that $(r, \mathcal{X} \cup \{M\}) \in \mathfrak{X}'_2$ is a valid description for G. Again it is easy to see

that $r \in V(G)$ and $\mathcal{X} \cup \{M\}$ consists of pairwise disjoint subsets of $V(G-r)$. Consider any set S' consisting of exactly one vertex from each set $X \in \mathcal{X} \cup \{M\}$. Let $S = S' \setminus M$ and $\{m\} = M \cap S'$. Since $(r, \mathcal{X})$ is a description for $G - V(P - v_\ell)$, the connected component H of $G - V(P - v_\ell) - S$ containing r is acyclic and satisfies $N_{G-V(P-v_\ell)}(H) = S$. Note that the connected component H' of $G - S'$ containing r is a supergraph of H and so $S \subseteq N_G(H')$. All that is left to argue is that H' is acyclic and $m \in N_G(H')$. Let u be the vertex in T that is adjacent to v_1. Note that this vertex is uniquely defined since no vertex in $N_G(T)$ has two neighbors in T. Since H is a supertree of $(F \setminus V(P)) \cup \{v_\ell\}$ and $u, v_\ell \in F$, it follows that $P - v_\ell$ is a path between two vertices in H, of which S' contains exactly one vertex chosen from M. Consequently, the component H' of $G - S$ satisfies $V(H') = V(H) \cup V(P - v_\ell - m)$, H' is acyclic, and $m \in N_G(H')$.

Finally since $\mathfrak{X}_3$ is a set of descriptions for G and $\mathfrak{X}'_3 = \mathfrak{X}_3$, the claim holds for $\mathfrak{X}'_3$. ⌟

Before we proceed to show that the output of the algorithm is correct, we prove two claims about intermediate results obtained by modifying the output of a recursive call.

Claim 6.3.13. $\mathcal{T}_G(\mathfrak{X}'_1) = \text{maxset}_w\{H \in \mathcal{S}^k_G(F) \mid v_\ell \in N_G(H)\}$

Proof. Recall $\mathfrak{X}'_1$ is defined as $\{(r, \mathcal{X} \cup \{\{v_\ell\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}_1\}$. We know $\mathcal{T}_{G-v_\ell}(\mathfrak{X}_1) = \text{maxset}_w(\mathcal{S}^{k-1}_{G-v_\ell}(F))$. In order to apply Lemma 6.3.4 we prove that $v_\ell \in N_G(H)$ for all $H \in \mathcal{T}_{G-v_\ell}(\mathfrak{X}_1)$. Let $H \in \mathcal{T}_{G-v_\ell}(\mathfrak{X}_1)$ be arbitrary. If $\ell = 1$, then as $v_\ell = v \in N_G(T)$ and $T \subseteq H$ we get $v_\ell \in N_G(H)$. Otherwise if $\ell > 1$, suppose for the sake of contradiction that $v_{\ell-1} \notin V(H)$. Then some vertex $u \in V(P - F - v_\ell)$ must be contained in $N_{G-v_\ell}(H)$. But then observe that $H \cup V(P - v_\ell)$ acyclic and has strictly larger weight than H, while $|N_{G-v_\ell}(H \cup V(P - v_\ell))| < |N_{G-v_\ell}(H)|$. This contradicts the choice of H. It follows that $v_{\ell-1} \in V(H)$ and therefore $v_\ell \in N_G(H)$. We can now apply Lemma 6.3.4 to obtain that $\mathcal{T}_G(\mathfrak{X}'_1) = \text{maxset}_w\{H \in \mathcal{S}^k_G(F) \mid v_\ell \in N_G(H)\}$. ⌟

Claim 6.3.14. If $m \in V(P - F - v_\ell)$ then $\mathcal{T}_G(\{(r, \mathcal{X} \cup \{\{m\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}_2\}) = \text{maxset}_w\{\hat{H} \in \mathcal{S}^k_G(F \cup \{v_\ell\}) \mid m \in N_G(\hat{H})\}$.

Proof. We show Lemma 6.3.5 applies to the instance $(G, k, F \cup \{v_\ell\}, T, w)$. Recall that by induction $\mathcal{T}_{G-V(P-v_\ell)}(\mathfrak{X}_2) = \text{maxset}_w(\mathcal{S}^{k-1}_{G-V(P-v_\ell)}((F \setminus V(P)) \cup \{v_\ell\})) = \text{maxset}_w(\mathcal{S}^{k-1}_{G-V(P-v_\ell)}((F \cup \{v_\ell\}) \setminus V(P - v_\ell)))$. Recall also that $v_1 = v \in N_G(T)$ has exactly one neighbor in $T \subseteq F$ by Property 6.3.10, let v' be this vertex. Observe that $P - v_\ell$ is a path in G with $\deg_G(p) = 2$ for all $p \in V(P - v_\ell)$ and $N_G(P - v_\ell) = \{v', v_\ell\}$ with $v', v_\ell \in F \cup \{v_\ell\}$. Then since $m \in V(P - F - v_\ell) = V(P - v_\ell) \setminus F$ we can apply Lemma 6.3.5 to obtain $\mathcal{T}_G(\{(r, \mathcal{X} \cup \{\{m\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}_2\}) = \text{maxset}_w\{\hat{H} \in \mathcal{S}^k_G(F \cup \{v_\ell\}) \mid m \in N_G(\hat{H})\}$. ⌟

We now show that all maximum-weight k-secluded supertrees of F in G are described by at least one description in our output. More formally, we show that $\text{maxset}_w(\mathcal{S}_G^k(F)) \subseteq \mathcal{T}_G(\mathfrak{X}'_1 \cup \mathfrak{X}'_2 \cup \mathfrak{X}'_3) \subseteq \mathcal{S}_G^k(F)$. To that end we first show that $\mathcal{T}_i \subseteq \mathcal{T}_G(\mathfrak{X}'_i) \subseteq \mathcal{S}_G^k(F)$ for each $i \in [3]$ in Claims 6.3.15 to 6.3.17.

Claim 6.3.15. $\mathcal{T}_1 \subseteq \mathcal{T}_G(\mathfrak{X}'_1) \subseteq \mathcal{S}_G^k(F)$

Proof. It follows from Claim 6.3.13 that $\mathcal{T}_G(\mathfrak{X}'_1) \subseteq \mathcal{S}_G^k(F)$ so it remains to show that $\mathcal{T}_1 \subseteq \mathcal{T}_G(\mathfrak{X}'_1)$. Let $H \in \mathcal{T}_1$ be arbitrary. By definition of $\mathcal{T}_1$ we have $H \in \text{maxset}_w(\mathcal{S}_G^k(F)) \subseteq \mathcal{S}_G^k(F)$ and $v_\ell \in N_G(H)$. So then clearly $H \in \text{maxset}_w\{H' \in \mathcal{S}_G^k(F) \mid v_\ell \in N_G(H')\} = \mathcal{T}_G(\mathfrak{X}'_1)$ (by Claim 6.3.13). Since $H \in \mathcal{T}_1$ was arbitrary we conclude $\mathcal{T}_1 \subseteq \mathcal{T}_G(\mathfrak{X}'_1)$ completing the proof. ⌟

Claim 6.3.16. $\mathcal{T}_2 \subseteq \mathcal{T}_G(\mathfrak{X}'_2) \subseteq \mathcal{S}_G^k(F)$

Proof. Recall $\mathfrak{X}'_2$ is defined as $\{(r, \mathcal{X} \cup \{M\}) \mid (r, \mathcal{X}) \in \mathfrak{X}_2\}$, where M is the set of minimum weight vertices in $P - F - v_\ell$. For any $u \in V(P - F - v_\ell)$ we define $\mathfrak{X}_2^u := \{(r, \mathcal{X} \cup \{\{u\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}_2\}$. By repeated application of Observation 6.2.5 we have that $\mathcal{T}_G(\bigcup_{u \in M} \mathfrak{X}_2^u) = \mathcal{T}_G(\{(r, \mathcal{X} \cup \{M\}) \mid (r, \mathcal{X}) \in \mathfrak{X}_2\})$. Observe that $\mathfrak{X}_2^u$ is a valid set of descriptions for G for each $u \in M$. By Observation 6.2.4 and definition of $\mathfrak{X}'_2$ we have $\bigcup_{u \in M} \mathcal{T}_G(\mathfrak{X}_2^u) = \mathcal{T}_G(\bigcup_{u \in M} \mathfrak{X}_2^u) = \mathcal{T}_G(\mathfrak{X}'_2)$. We will prove $\mathcal{T}_2 \subseteq \bigcup_{u \in M} \mathcal{T}_G(\mathfrak{X}_2^u) \subseteq \mathcal{S}_G^k(F)$.

We show $\mathcal{T}_2 \subseteq \bigcup_{u \in M} \mathcal{T}_G(\mathfrak{X}_2^u)$. Let $H \in \mathcal{T}_2$ be arbitrary. By definition of $\mathcal{T}_2$ we have $H \in \text{maxset}_w(\mathcal{S}_G^k(F))$, $|N_G(H) \cap V(P - F - v_\ell)| = 1$, and $v_\ell \in V(H)$. Then also $H \in \text{maxset}_w(\mathcal{S}_G^k(F \cup \{v_\ell\}))$. Let m be such that $N_G(H) \cap V(P - F - v_\ell) = \{m\}$. Since $m \in V(P - F - v_\ell)$, by Claim 6.3.14 we have that $\mathcal{T}_G(\mathfrak{X}_2^m) = \text{maxset}_w\{\hat{H} \in \mathcal{S}_G^k(F \cup \{v_\ell\}) \mid m \in N_G(\hat{H})\}$. Since $m \in N_G(H)$ and $H \in \text{maxset}_w(\mathcal{S}_G^k(F \cup \{v_\ell\}))$ clearly $H \in \text{maxset}_w\{\hat{H} \in \mathcal{S}_G^k(F \cup \{v_\ell\}) \mid m \in N_G(\hat{H})\} = \mathcal{T}_G(\mathfrak{X}_2^m)$. It follows that $H \in \bigcup_{u \in M} \mathcal{T}_G(\mathfrak{X}_2^u)$. Since H was arbitrary we conclude that $\mathcal{T}_2 \subseteq \bigcup_{u \in M} \mathcal{T}_G(\mathfrak{X}_2^u)$.

It remains to show that $\bigcup_{u \in M} \mathcal{T}_G(\mathfrak{X}_2^u) \subseteq \mathcal{S}_G^k(F)$. Let $u \in M$ be arbitrary. We show $\mathcal{T}_G(\mathfrak{X}_2^u) \subseteq \mathcal{S}_G^k(F)$. It suffices to show that when considering a set S consisting of one element from each set of a description $(r, \mathcal{X} \cup \{u\}) \in \mathfrak{X}_2^u$, the component H of $G - S$ containing r is a k-secluded supertree of F in G. This component H is a k-secluded tree in G since $\mathfrak{X}_2^u$ is a valid set of descriptions for G of order at most k. It remains to show that $F \subseteq V(H)$. By induction the component of $(G - V(P - v_\ell)) - (S \setminus \{u\})$ contains all of $(F \setminus V(P)) \cup \{v_\ell\}$. As the two neighbors of $V(P - v_\ell)$ both belong to $F \cup \{v_\ell\}$, the subpath of P before u and subpath after u are both reachable from r in $G - S$. Hence $V(P - u) \subseteq V(H)$, and since $u \in M \subseteq V(P - F - v_\ell)$ we know $u \notin F$, so $F \subseteq V(H)$. It follows that H is a k-secluded supertree of F in G so since $H \in \mathcal{T}_G(\mathfrak{X}_2^u)$ was arbitrary we have $\mathcal{T}_G(\mathfrak{X}_2^u) \subseteq \mathcal{S}_G^k(F)$. ⌟

Claim 6.3.17. $\mathcal{T}_3 \subseteq \mathcal{T}_G(\mathfrak{X}'_3) \subseteq \mathcal{S}^k_G(F)$

Proof. Recall $\mathfrak{X}'_3$ is defined to be equal to $\mathfrak{X}_3$, so we show $\mathcal{T}_3 \subseteq \mathcal{T}_G(\mathfrak{X}_3) \subseteq \mathcal{S}^k_G(F)$.

Let $H \in \mathcal{T}_3$ be arbitrary. By definition of $\mathcal{T}_3$ we have $H \in \text{maxset}_w(\mathcal{S}^k_G(F)) \subseteq \mathcal{S}^k_G(F)$ and $V(P) \subseteq V(H)$. So clearly $H \in \mathcal{S}^k_G(F \cup V(P))$. To show that $H \in \mathcal{T}_G(\mathfrak{X}_3) = \text{maxset}_w(\mathcal{S}^k_G(F \cup V(P)))$ we have to show for all $H' \in \mathcal{S}^k_G(F \cup V(P))$ that $w(H') \leq w(H)$. Suppose for contradiction there is an $H' \in \mathcal{S}^k_G(F \cup V(P))$ such that $w(H') > w(H)$. Clearly $H' \in \mathcal{S}^k_G(F)$ but then $w(H') > w(H)$ contradicts that $H \in \text{maxset}_w(\mathcal{S}^k_G(F)))$. So by contradiction it follows that $H \in \text{maxset}_w(\mathcal{S}^k_G(F \cup V(P))) = \mathcal{T}_G(\mathfrak{X}_3)$ and since $H \in \mathcal{T}_3$ was arbitrary we conclude $\mathcal{T}_3 \subseteq \mathcal{T}_G(\mathfrak{X}_3)$.

Finally observe that $\mathcal{T}_G(\mathfrak{X}_3) = \text{maxset}_w(\mathcal{S}^k_G(F \cup V(P))) \subseteq \mathcal{S}^k_G(F \cup V(P)) \subseteq \mathcal{S}^k_G(F)$, completing the proof. ⌟

It follows from Claims 6.3.15 to 6.3.17 that $\text{maxset}_w(\mathcal{S}^k_G(F)) = \mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3 \subseteq \mathcal{T}_G(\mathfrak{X}'_1) \cup \mathcal{T}_G(\mathfrak{X}'_2) \cup \mathcal{T}_G(\mathfrak{X}'_3) \subseteq \mathcal{S}^k_G(F)$. So then we have

$$\text{maxset}_w(\mathcal{S}^k_G(F)) = \text{maxset}_w(\mathcal{T}_G(\mathfrak{X}'_1) \cup \mathcal{T}_G(\mathfrak{X}'_2) \cup \mathcal{T}_G(\mathfrak{X}'_3)). \tag{6.1}$$

The algorithm proceeds to calculate values w_1, w_2, w_3 based on an arbitrary secluded tree in $\mathcal{T}_G(\mathfrak{X}'_1)$, $\mathcal{T}_G(\mathfrak{X}'_2)$, and $\mathcal{T}_G(\mathfrak{X}'_3)$ respectively. We show that, for any $i \in [3]$, all secluded trees in $\mathcal{T}_G(\mathfrak{X}'_i)$ have weight w_i.

- For $i = 1$, we know from Claim 6.3.13 that $\mathcal{T}_G(\mathfrak{X}'_1) = \text{maxset}_w\{H \in \mathcal{S}^k_G(F) \mid v_\ell \in N_G(H)\}$, and clearly all trees in $\text{maxset}_w\{H \in \mathcal{S}^k_G(F) \mid v_\ell \in N_G(H)\}$ have the same weight, which must be w_1.

- For $i = 2$, consider two arbitrary secluded trees H_1, H_2 in the set $\mathcal{T}_G(\mathfrak{X}'_2) = \mathcal{T}_G(\{(r, \mathcal{X} \cup \{M\}) \mid (r, \mathcal{X}) \in \mathfrak{X}_2\})$ where $M \subseteq V(P - F - v_\ell)$ is the set of minimum weight vertices in $P - F - v_\ell$. We show that $w(H_1) = w(H_2)$. Observe that $N_G(H_1) \cap M = \{m_1\}$ for some $m_1 \in V(P - v_\ell) \setminus F$, so $H_1 \in \mathcal{T}_G(\{(r, \mathcal{X} \cup \{\{m_1\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}_2\})$. Hence $H_1 \in \text{maxset}_w\{\hat{H} \in \mathcal{S}^k_G(F \cup \{v_\ell\}) \mid m_1 \in N_G(\hat{H})\}$ by Claim 6.3.14 since $m_1 \in V(P - v_\ell) \setminus F$. Similarly $H_2 \in \text{maxset}_w\{\hat{H} \in \mathcal{S}^k_G(F \cup \{v_\ell\}) \mid m_2 \in N_G(\hat{H})\}$ for some $m_2 \in V(P - v_\ell) \setminus F$. Consider the graph $H'_2 := G[V(H_2) \cup \{m_2\}] - m_1$ and observe that $H'_2 \in \{\hat{H} \in \mathcal{S}^k_G(F \cup \{v_\ell\}) \mid m_1 \in N_G(\hat{H})\}$. Additionally $w(H_2) = w(H'_2) + w(m_2) - w(m_1) = w(H'_2)$ since m_1, m_2 are of minimum weight in $V(P - F - v_\ell)$, so $H'_2 \in \text{maxset}_w\{\hat{H} \in \mathcal{S}^k_G(F \cup \{v_\ell\}) \mid m_1 \in N_G(\hat{H})\}$. It follows that $w(H_2) = w(H'_2) = w(H_1)$. Since $H_1, H_2 \in \mathcal{T}_G(\mathfrak{X}'_2)$ are arbitrary, we have that all secluded trees in $\mathcal{T}_G(\mathfrak{X}'_2)$ have the same weight, which must be w_2.

- For $i = 3$ we know that $\mathcal{T}_G(\mathfrak{X}'_3) = \mathcal{T}_G(\mathfrak{X}_3) = \text{maxset}_w(\mathcal{S}^k_G(F \cup V(P)))$, and clearly all trees in $\text{maxset}_w(\mathcal{S}^k_G(F \cup V(P)))$ have the same weight, which must be w_3.

Clearly it follows that

$$\begin{aligned}
\mathcal{T}_G(\mathfrak{X}') &= \mathcal{T}_G\left(\bigcup_{\{i\in[3]\mid w_i=\max\{w_1,w_2,w_3\}\}} \mathfrak{X}'_i\right) && \text{Definition of } \mathfrak{X}' \text{ in Step 3}\\
&= \text{maxset}_w(\mathcal{T}_G(\mathfrak{X}'_1 \cup \mathfrak{X}'_2 \cup \mathfrak{X}'_3)) && \text{Any tree in } \mathcal{T}_G(\mathfrak{X}'_i) \text{ has weight } w_i.\\
&= \text{maxset}_w(\mathcal{T}_G(\mathfrak{X}'_1) \cup \mathcal{T}_G(\mathfrak{X}'_2) \cup \mathcal{T}_G(\mathfrak{X}'_3)) && \text{Observation 6.2.4}\\
&= \text{maxset}_w(\mathcal{S}^k_G(F)). && \text{Equation (6.1)}
\end{aligned}$$

So the algorithm correctly returns a set of descriptions $\mathfrak{X}'$ for which $\mathcal{T}_G(\mathfrak{X}') = \text{maxset}_w(\mathcal{S}^k_G(F))$. To complete the proof of correctness, we show that $\mathfrak{X}'$ is non-redundant for G.

Claim 6.3.18. $\mathfrak{X}'$ is non-redundant for G.

Proof. Suppose for contradiction that $\mathfrak{X}'$ is redundant for G, i.e., there is a k-secluded tree H in G such that H is described by two distinct descriptions $(r_1, \mathcal{X}_1)$, $(r_2, \mathcal{X}_2) \in \mathfrak{X}'$. Since $\mathfrak{X}' \subseteq \mathfrak{X}'_1 \cup \mathfrak{X}'_2 \cup \mathfrak{X}'_3$ it suffices to consider the cases $(r_1, \mathcal{X}_1) \in \mathfrak{X}'_1$, $(r_1, \mathcal{X}_1) \in \mathfrak{X}'_2$, and $(r_1, \mathcal{X}_1) \in \mathfrak{X}'_3$.

- If $(r_1, \mathcal{X}_1) \in \mathfrak{X}'_1$, then $\{v_\ell\} \in \mathcal{X}_1$ by definition of $\mathfrak{X}'_1$. So then $v_\ell \in N_G(H)$. Since H is described by $(r_2, \mathcal{X}_2)$ there exists $X \in \mathcal{X}_2$ such that $v_\ell \in X$. If $(r_2, \mathcal{X}_2) \in \mathfrak{X}'_3 = \mathfrak{X}_3$ then X must be part of a description in $\mathfrak{X}_3$, so there must be a $H' \in \mathcal{T}_G(\mathfrak{X}_3)$ with $v_\ell \in N_G(H')$. However we know by induction that $\mathcal{T}_G(\mathfrak{X}_3) = \text{maxset}_w(\mathcal{S}^k_G(F \cup V(P)))$, so $v_\ell \in F \cup V(P) \subseteq V(H'')$ for all $H'' \in \mathcal{T}_{G-V(P-v_\ell)}(\mathfrak{X}_3)$. It follows that $(r_2, \mathcal{X}_2) \notin \mathfrak{X}'_3$. If $(r_2, \mathcal{X}_2) \in \mathfrak{X}'_2$ then we show this also leads to a contradiction. It follows from the definition of $\mathfrak{X}'_2$ that X must be part of a description in $\mathfrak{X}_2$, however we know by induction that $\mathcal{T}_{G-V(P-v_\ell)}(\mathfrak{X}_2) = \text{maxset}_w(\mathcal{S}^{k-1}_{G-V(P-v_\ell)}((F \setminus V(P)) \cup \{v_\ell\}))$, so $v_\ell \in (F \setminus V(P)) \cup \{v_\ell\} \subseteq V(H'')$ for all $H'' \in \mathcal{T}_{G-V(P-v_\ell)}(\mathfrak{X}_2)$. Hence $(r_2, \mathcal{X}_2) \notin \mathfrak{X}'_2$, and since also $(r_2, \mathcal{X}_2) \notin \mathfrak{X}'_3$ we have that $(r_2, \mathcal{X}_2) \in \mathfrak{X}'_1$, meaning $X = \{v_\ell\}$.

  Since $(r_1, \mathcal{X}_1)$ and $(r_2, \mathcal{X}_2)$ are distinct, $\{v_\ell\} \in \mathcal{X}_1$, and $\{v_\ell\} \in \mathcal{X}_2$ we have that $(r_1, \mathcal{X}_1 \setminus \{\{v_\ell\}\})$ and $(r_2, \mathcal{X}_2 \setminus \{\{v_\ell\}\})$ are distinct. Observe that $(r_1, \mathcal{X}_1 \setminus \{\{v_\ell\}\}) \in \mathfrak{X}_1$ and $(r_2, \mathcal{X}_2 \setminus \{\{v_\ell\}\}) \in \mathfrak{X}_1$. We know by induction that $\mathfrak{X}_1$ is non-redundant for $G - v$. However since H is a k-secluded tree in G with $v_\ell \in N_G(H)$ we have that H is a $(k-1)$-secluded tree in $G - v_\ell$, and clearly H is described by both $(r_1, \mathcal{X}_1 \setminus \{\{v_\ell\}\})$ and $(r_2, \mathcal{X}_2 \setminus \{\{v_\ell\}\})$, contradicting that $\mathfrak{X}_1$ is non-redundant for $G - v$.

- If $(r_1, \mathcal{X}_1) \in \mathfrak{X}'_2$, then without loss of generality we can assume that $(r_2, \mathcal{X}_2) \notin \mathfrak{X}'_1$ since otherwise we can swap the roles of $(r_1, \mathcal{X}_1)$ and $(r_2, \mathcal{X}_2)$ and the previous case would apply. Suppose that $(r_2, \mathcal{X}_2) \in \mathfrak{X}'_3 = \mathfrak{X}_3$, then $H \in \mathcal{T}_G(\mathfrak{X}_3)$ and we have by induction that $\mathcal{T}_G(\mathfrak{X}_3) = \text{maxset}_w(\mathcal{S}^k_G(F \cup V(P)))$

hence $v_\ell \in F \cup V(P) \subseteq V(H)$. This contradicts $v_\ell \in N_G(H)$ so $(r_2, \mathcal{X}_2) \notin \mathfrak{X}'_3$. This leaves as only option that $(r_2, \mathcal{X}_2) \in \mathfrak{X}'_2$.

Recall that $\mathfrak{X}'_2 = \{(r, \mathcal{X} \cup \{M\}) \mid (r, \mathcal{X}) \in \mathfrak{X}_2\}$ where $M \subseteq V(P - F - v_\ell)$, so $(r_1, \mathcal{X}_1 \setminus \{M\}) \in \mathfrak{X}_2$ and $(r_2, \mathcal{X}_2 \setminus \{M\}) \in \mathfrak{X}_2$. Since $\mathfrak{X}_2$ is a set of valid descriptions for $G - V(P - v_\ell)$ (by induction) we have that $\mathcal{X}_1 \setminus \{M\}$ and $\mathcal{X}_2 \setminus \{M\}$ contain only subsets of $V(G) \setminus V(P - v_\ell)$, so $N_G(H) \setminus M \subseteq V(G) \setminus V(P - v_\ell)$. Observe that since the path $P - v_\ell$ is connected to $H' := H - V(P - v_\ell)$ only via its endpoints, and H does not contain $m \in V(P)$ we have that H' remains connected so H' is a $(k-1)$-secluded tree in $G - V(P - v_\ell)$ described by $(r_1, \mathcal{X}_1 \setminus \{M\})$ as well as $(r_2, \mathcal{X}_2 \setminus \{M\})$. However this contradicts that $\mathfrak{X}_2$ is a non-redundant set of descriptions for $G - V(P - v_\ell)$ as given by induction.

- If $(r_1, \mathcal{X}_1) \in \mathfrak{X}'_3 = \mathfrak{X}_3$, then without loss of generality we can assume $(r_2, \mathcal{X}_2) \in \mathfrak{X}'_3 = \mathfrak{X}_3$ since otherwise we can swap the roles of $(r_1, \mathcal{X}_1)$ and $(r_2, \mathcal{X}_2)$ and one of the previous cases would apply. But then H is a k-secluded tree in G described by two distinct descriptions from $\mathfrak{X}_3$, i.e. $\mathfrak{X}_3$ is redundant for G contradicting the induction hypothesis. ⌟

This concludes the proof of Lemma 6.3.7 and establishes correctness. □

6.3.4 Runtime analysis

If all recursive calls in the algorithm would decrease k then, since for $k = 0$ it does not make any further recursive calls, the maximum recursion depth is k. However in Step 3(c) the recursive call does not decrease k. In order to bound the recursion depth, we show the algorithm cannot make more than $k(k+1)$ consecutive recursive calls in Step 3(c), that is, the recursion depth cannot increase by more than $k(k+1)$ since the last time k decreased. We do this by showing in the following three lemmas that $|N_G(T)|$ increases as consecutive recursive calls in Step 3(c) are made. Since the algorithm executes Step 2 if $|N_G(T)| > k(k+1)$, this limits the number of consecutive recursive calls in Step 3(c).

The following lemma states that under certain conditions, the neighborhood of T does not decrease during the execution of a single recursive call.

Lemma 6.3.19. *Let $(G_0, k_0, F_0, T_0, w_0)$ be an* ELSS *instance such that all leaves of G_0 are contained in T_0. If the algorithm does not terminate before Step 3, then the instance (G', k', F', T', w') when executing Step 3 satisfies $|N_{G'}(T')| \geq |N_{G_0}(T_0)|$.*

Proof. Since the algorithm does not terminate before Step 3 it follows that Steps 1 and 2 are not executed, so consider the part of the algorithm before Step 1. Throughout the proof we use (G, k, F, T, w) to refer to the instance at the time the algorithm evaluates it; initially $(G, k, F, T, w) = (G_0, k_0, F_0, T_0, w_0)$, but actions such as contracting leaves may change the instance during the execution. Suppose that all leaves of G are contained in T. We infer that $G[F]$ is acyclic, as

otherwise the algorithm would return $\emptyset$ before reaching Step 3. Removing the connected components of G that do not contain a vertex of F does not alter $|N_G(T)|$. Afterwards we know that G is connected, as otherwise the algorithm would return $\emptyset$. Consider a single degree-1 contraction step of a vertex v with $F \neq \{v\}$ that results in the instance $(G - v, k, F^*, T^*, w^*)$. Since we assume that all leaves are contained in T, we have that $v \in T$. Let u be the neighbor of v. By Definition 6.3.1 we have $F^* = (F \setminus \{v\}) \cup \{u\}$ and $T^* = (T \setminus \{v\}) \cup \{u\}$. If $u \in T$, then $N_{G-v}(T^*) = N_G(T)$ and therefore their size is equal. If $u \notin T$, then observe that u cannot be a leaf in G by assumption and therefore $N_G(u) \setminus \{v\} \neq \emptyset$. Since T is connected and v is a leaf in T we get $(N_G(u) \setminus \{v\}) \cap T = \emptyset$. It follows that $|N_{G-v}(T^*)| \geq |N_G(T)|$. These arguments can be applied for each consecutive contraction step to infer $|N_G(T)| \geq |N_{G_0}(T_0)|$ for the instance (G, k, F, T, w) after all contractions.

Next consider the step where if $N_G(T)$ contains a vertex $v \in F$, the vertex v is added to T. Since $G[F]$ is acyclic, $G[T]$ is connected, and $v \notin T$ is not a leaf, it follows that $N_G(v) \setminus T \neq \emptyset$ and $|N_G(v) \cap T| = 1$. It follows that $|N_G(T \cup \{v\})| \geq |N_G(T)|$. Again these arguments can be applied iteratively. For the instance (G, k, F, T, w) to which this step can no longer be applied, $G[T]$ is a connected component of $G[F]$.

Next we get that $N_G(F) \neq \emptyset$ as otherwise the algorithm would return $\{(r, \emptyset)\}$ for some $r \in F$. We also get $k > 0$, as otherwise $\emptyset$ would have been returned.

Since none of the steps decreased the size of the neighborhood of T, for the instance (G', k', F', T', w') at the time Step 3 is executed we conclude $|N_{G'}(T')| \geq |N_{G_0}(T_0)|$ as required. □

In the next lemma we show that the size of the neighborhood of T strictly increases as we make the recursive call in Step 3(c).

Lemma 6.3.20. *Let (G, k, F, T, w) be the instance considered in Step 3. If Step 3(c) branches on the instance $(G, k, F \cup V(P), T \cup V(P), w)$, then either $|N_G(T \cup V(P))| > |N_G(T)|$ or some vertex $u \in N_G(T \cup V(P))$ is adjacent to at least two vertices in $T \cup V(P)$.*

Proof. Consider the path $P = (v = v_1, \ldots, v_\ell)$ with $\deg_G(v_i) = 2$ for each $1 \leq i < \ell$ and ($v_\ell \in N_G(T)$ or $\deg_G(v_\ell) > 2$) as defined in Step 3. The precondition of Step 3(c) gives that $G[F \cup V(P)]$ is acyclic. Since $T \subseteq F$, this implies that $G[T \cup V(P)]$ is acyclic. It follows that $V(P) \cap N_G(T) = \{v\}$ and $\deg_G(v_\ell) > 2$. Hence $|N_G(T) \setminus \{v\}| = |N_G(T)| - 1$ and $|N_G(v_\ell) \setminus V(P)| \geq 2$. Observe that $N_G(T \cup V(P)) = (N_G(T) \setminus \{v\}) \cup (N_G(v_\ell) \setminus V(P))$ so if $(N_G(T) \setminus \{v\}) \cap (N_G(v_\ell) \setminus V(P)) = \emptyset$ we have $|N_G(T \cup V(P))| > |N_G(T)|$. Alternatively, suppose $u \in (N_G(T) \setminus \{v\}) \cap (N_G(v_\ell) \setminus V(P))$. Then the second condition holds; u has at least one neighbor in T as $u \in N_G(T) \setminus \{v\}$ and u is adjacent to $v_\ell \notin T \setminus \{v\}$. □

Finally we combine Lemmas 6.3.19 and 6.3.20 to show $|N_G(T)|$ is an upper bound to the number of consecutive recursive calls in Step 3(c).

Lemma 6.3.21. *If the recursion tree generated by the algorithm contains a path of $i \geq 1$ consecutive recursive calls in Step 3(c), and (G, k, F, T, w) is the instance considered in Step 3 where the i-th of these recursive calls is made, then $|N_G(T)| \geq i$.*

Proof. We use induction of i. First suppose $i = 1$ and let (G, k, F, T, w) be the instance considered in Step 3 where the first of these recursive calls is made. If $|N_G(T)| = 0$, then $G[T]$ is a component of G. However, since (G, k, F, T, w) is an instance considered in Step 3 we know that Properties 6.3.8, 6.3.10 and 6.3.11 apply. In particular $N_G(F) \neq \emptyset$, ruling out that $T = F$. However if $T \neq F$, then there are at least two connected components in G containing a vertex from F, contradicting that G is connected (Property 6.3.8). By contradiction we can conclude that $|N_G(T)| \geq 1 = i$.

Suppose $i \geq 2$ and let (G, k, F, T, w) be the instance considered in Step 3 where the i-th recursive call is made. Let (G', k', F', T', w') be the instance considered in Step 3 where the $(i-1)$-th recursive call is made. By induction we know $|N_{G'}(T')| \geq i-1$. Let P be as in Step 3 where the $(i-1)$-th recursive call is made, then by Lemma 6.3.20 we have that $|N_{G'}(T' \cup V(P))| > |N_{G'}(T')|$ or some vertex $u \in N_{G'}(T' \cup V(P))$ is adjacent to at least two vertices in $T' \cup V(P)$. Since we know that the recursive call on $(G', k', F' \cup V(P), T' \cup V(P), w')$ reaches Step 3 with the instance (G, k, F, T, w), we can rule out that some vertex $u \in N_{G'}(T' \cup V(P))$ is adjacent to at least two vertices in $T' \cup V(P)$ as this would mean the recursive call ends in Step 1. We can conclude instead that $|N_{G'}(T' \cup V(P))| > |N_{G'}(T')|$.

Note that since (G', k', F', T', w') is the instance in Step 3 we have that Properties 6.3.8, 6.3.10 and 6.3.11 apply. In particular, (G', k', F', T', w') is almost leafless, implying that all leaves in G' are contained in T'. It follows that all leaves in G' are also contained in $T' \cup V(P)$, so Lemma 6.3.19 applies to the input instance $(G', k', F' \cup V(P), T' \cup V(P), w')$ (as recursively solved in Step 3) and the instance (G, k, F, T, w) (as considered in Step 3 of that recursive call). So we obtain $|N_G(T)| \geq |N_{G'}(T' \cup V(P))| > |N_{G'}(T')| \geq i-1$, that is, $|N_G(T)| \geq i$. □

Since we know in Step 3 that $|N_G(T)| \leq k(k+1)$ (by Property 6.3.11) we can now claim that there are at most $k(k+1)$ consecutive recursive calls of Step 3(c), leading to a bound on the recursion depth of $\mathcal{O}(k^3)$. We argue that each recursive call takes $\mathcal{O}(kn^3)$ time and since we branch at most three ways, we obtain a running time of $3^{\mathcal{O}(k^3)} \cdot kn^3 = 3^{\mathcal{O}(k^3)} \cdot n^3$. However, with a more careful analysis we can give a better bound on the number of nodes in the recursion tree.

Lemma 6.3.22. *The algorithm described in Section 6.3.2 can be implemented to run in time $2^{\mathcal{O}(k \log k)} \cdot n^3$.*

Proof. Consider the recursion tree of the algorithm. We first prove that each recursive call takes $\mathcal{O}(kn^3)$ time (not including the time further recursive calls require). We then show that the recursion tree contains at most $2^{\mathcal{O}(k \log k)}$ nodes.

Runtime per node Consider the input instance (G, k, F, T, w) with $n = |V(G)|$ and $m = |E(G)|$. We can verify that $G[F]$ is acyclic in $\mathcal{O}(|F|)$ time using DFS. Again using DFS, in $\mathcal{O}(n+m)$ time identify all connected components of G and determine whether they contain a vertex of F. We can then in linear time remove all connected components that contain no vertex of F and return $\emptyset$ if more than one component remains. Finally exhaustively contracting degree-1 vertices into their neighbor is known to take $\mathcal{O}(n)$ time. Updating F and T only results in $\mathcal{O}(1)$ overhead for each contraction. Exhaustively adding vertices $v \in N_G(T) \cap F$ to T can be done in $\mathcal{O}(n)$ time since it corresponds to finding a connected component in $G[F]$ which is acyclic.

For Step 1 we can find a vertex $v \in N_G(T)$ with two neighbors in T in $\mathcal{O}(n^2)$ time by iterating over all neighbors of each vertex in T.

Determining the size of the neighborhood in Step 2 can be done in $\mathcal{O}(n^2)$ time. Applying Lemma 6.3.3 takes $\mathcal{O}(kn^3)$ time. So excluding the recursive call, Step 2 can be completed in $\mathcal{O}(kn^3)$ time.

For Step 3 an arbitrary $v \in N_G(T)$ can be selected in $\mathcal{O}(1)$ time, and the path P can be found in $\mathcal{O}(|P|) = \mathcal{O}(n)$ time as described in Footnote 1. Finally the results of the three recursive calls in Step 3 are combined. Selecting an arbitrary tree from $\mathcal{T}_G(\mathfrak{X}'_i)$ for any $i \in [3]$ involves selecting and arbitrary description $(r, \mathcal{X}) \in \mathfrak{X}'_i$ and then selecting, for each $X \in \mathcal{X}$ and arbitrary vertex $v \in X$. Now the tree can be found using DFS starting from r exploring an acyclic graph until it reaches the selected vertices from a set $X \in \mathcal{X}$. This all takes $\mathcal{O}(n)$ time. The weights of the selected secluded trees can be found in $\mathcal{O}(n)$ time as well. Finally we take the union of (a selection of) the three sets of descriptions. Since these sets are guaranteed to be disjoint, this can be done in constant time.

Number of nodes We now calculate the number of nodes in the recursion tree. To do this, label each edge in the recursion tree with a label from the set $\{1, 2, 3a, 3b, 3c\}$ indicating where in the algorithm the recursive call took place. Now observe that each node in the recursion tree can be uniquely identified by a sequence of edge-labels corresponding to the path from the root of the tree to the relevant node. We call such a sequence of labels a *trace*.

Note that for all recursive calls made in $1, 2, 3a$, and $3b$ the parameter (k) decreases, and for the call made in $3c$ the parameter remains the same. If $k \leq 0$ we do not make further recursive calls, so the trace contains at most k occurrences of $1, 2, 3a$, and $3b$. Next, we argue there are at most $k(k+1)$ consecutive occurrences of $3c$ in the trace.

Suppose for the sake of contradiction that the trace contains $k(k+1)+1$ consecutive occurrences of $3c$. Let (G, k, F, T, w) be the instance considered in Step 3 where the last of these recursive calls is made. By Lemma 6.3.21 we have $|N_G(T)| > k(k+1)$. This contradicts Property 6.3.11, so we can conclude the trace contains at most $k(k+1)$ consecutive occurrences of $3c$ and hence any valid trace has a total length of at most $k \cdot k(k+1) \in \mathcal{O}(k^3)$.

In order to count the number of nodes in the recursion tree, it suffices to count the number of different valid traces. Since a trace contains at most k occurrences that are not 3c we have that the total number of traces of length ℓ is $\binom{\ell}{k} \cdot 4^k \leq \ell^k \cdot 4^k = (4\ell)^k$. We derive the following bound on the total number of valid traces using the fact that $(k^c)^k = (2^{\log(k^c)})^k \in 2^{\mathcal{O}(k \log k)}$:

$$\sum_{1 \leq \ell \leq k^2(k+1)} (4\ell)^k \leq k^2(k+1) \cdot (4k^2(k+1))^k \in 2^{\mathcal{O}(k \log k)}.$$

We can conclude that the total number of nodes in the recursion tree is at most $2^{\mathcal{O}(k \log k)}$ so the overall running time is $2^{\mathcal{O}(k \log k)} \cdot kn^3 = 2^{\mathcal{O}(k \log k)} \cdot n^3$. □

6.3.5 Finding, enumerating, and counting large secluded trees

With the algorithm of Section 6.3.2 at hand we argue that we are able to enumerate k-secluded trees, count such trees containing a specified vertex, and solve LST.

Theorem 6.3.23. *There is an algorithm that, given a graph G, weight function w, and integer k, runs in time $2^{\mathcal{O}(k \log k)} n^4$ and outputs a set of descriptions $\mathfrak{X}$ such that $\mathcal{T}_G(\mathfrak{X})$ is exactly the set of maximum-weight k-secluded trees in G. Each such tree H is described by $|V(H)|$ distinct descriptions in $\mathfrak{X}$.*

Proof. Given the input (G, k, w), we proceed as follows. For each $v \in V(G)$, let $\mathfrak{X}_v$ be the output of the ELSS instance $(G, k, F = \{v\}, T = \{v\}, w)$ and let w_v be the weight of an arbitrary k-secluded supertree in $\mathcal{T}_G(\mathfrak{X}_v)$, or 0 if $\mathfrak{X}_v = \emptyset$. Note that all k-secluded trees described by $\mathfrak{X}_v$ have weight exactly w_v. Let $w^* := \max_{v \in V(G)} w_v$. If $w^* = 0$ then there are no k-secluded trees in G and we output $\mathfrak{X} = \emptyset$; otherwise we output $\mathfrak{X} := \bigcup \{\mathfrak{X}_v \mid v \in V(G) \wedge w_v = w^*\}$.

Clearly $\mathcal{T}_G(\mathfrak{X})$ is the set of all k-secluded trees in G of maximum weight. Since each $\mathfrak{X}_v$ is non-redundant, each maximum-weight k-secluded tree H is described by exactly $|V(H)|$ descriptions in $\mathfrak{X}$. □

By returning an arbitrary maximum-weight k-secluded tree described by any description in the output of Theorem 6.3.23, we have the following consequence.

Corollary 6.3.24. *There is an algorithm that, given a graph G, weight function w, and integer k, runs in time $2^{\mathcal{O}(k \log k)} n^4$ and outputs a maximum-weight k-secluded tree in G if one exists.*

The following theorem captures the consequences for counting.

Theorem 6.3.25. *There is an algorithm that, given a graph G, vertex $v \in V(G)$, weight function w, and integer k, runs in time $2^{\mathcal{O}(k \log k)} n^3$ and counts the number of k-secluded trees in G that contain v and have maximum weight out of all k-secluded trees containing v.*

Proof. Construct the ELSS instance $(G, k, F = \{v\}, T = \{v\}, w)$ and let $\mathfrak{X}$ be the output obtained by the algorithm described in Section 6.3.2. Note that this takes $2^{\mathcal{O}(k \log k)} n^3$ time by Lemma 6.3.22. Since the definition of ELSS guarantees that $\mathfrak{X}$ is non-redundant, each maximum-weight tree containing v is described by exactly one description in $\mathfrak{X}$. To solve the counting problem it therefore suffices to count how many distinct k-secluded trees are described by each description in $\mathfrak{X}$.

By Definition 6.2.1, for each description $(r, \mathcal{X}) \in \mathfrak{X}$, each way of choosing one vertex from each set $X \in \mathcal{X}$ yields a unique k-secluded tree. Hence the total number of maximum-weight k-secluded trees containing v is:

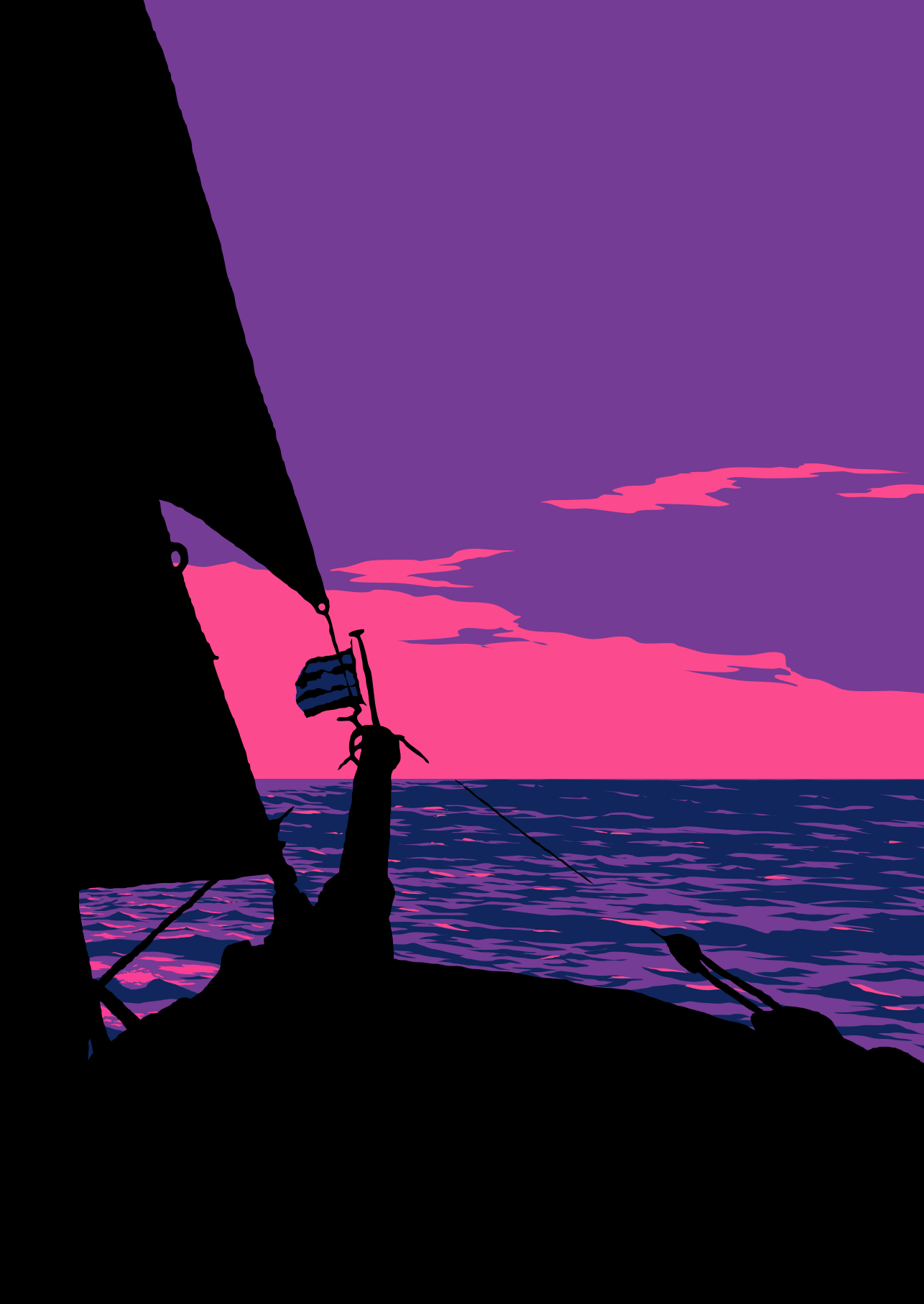
$$\sum_{(r,\mathcal{X})\in\mathfrak{X}} \prod_{X\in\mathcal{X}} |X|,$$

which can easily be computed in the stated time bound. □

6.4 Conclusion

We revisited the k-Secluded Tree problem first studied by Golovach et al. [67], leading to improved FPT algorithms with the additional ability to count and enumerate solutions. The non-trivial progress measure of our branching algorithm is based on a structural insight that allows a vertex that belongs to the *neighborhood* of every solution subtree to be identified, once the solution under construction has a sufficiently large open neighborhood. As stated, the correctness of this step crucially relies on the requirement that solution subgraphs are acyclic. It would be interesting to determine whether similar branching strategies can be developed to solve the more general k-Secluded Connected $\mathcal{F}$-Minor-Free Subgraph problem; the setting studied here corresponds to $\mathcal{F} = \{K_3\}$. While any $\mathcal{F}$-minor-free graph is known to be sparse, it may still contain large numbers of internally vertex-disjoint paths between specific pairs of vertices, which stands in the way of a direct extension of our techniques.

A second open problem concerns the optimal parameter dependence for k-Secluded Tree. The parameter dependence of our algorithm is $2^{\mathcal{O}(k \log k)}$. Can it be improved to single-exponential, or shown to be optimal under the Exponential Time Hypothesis?



7

Conclusion

7.1 Overview of results

We studied a number of graph problems revolving around finding large sparse subgraphs, in particular, $\mathcal{F}$-minor free subgraphs. As problems like these are often NP-hard, we made use of the tools and ideas from parameterized complexity, which gives a rigorous framework to show how the time required to solve a problem instance can be described more precisely by using one or more additional parameters rather than by using just the size of the problem instance. The notion of kernelization nicely illustrates this as it shows how, in polynomial time, the original problem instance can be encoded into a new instance whose size is bounded by a function of the parameter k, showing that the size n of the original instance only has a limited influence on the overall running time required to solve the instance. The techniques developed to attain these theoretical results sometimes lead to practical strategies to speed up solving NP-hard problems.

Much effort has gone into designing kernelization algorithms which produce instances that are as small as possible. In Chapter 3 we studied the Outerplanar Vertex Deletion problem parameterized by the solution size. We presented the first concrete kernelization algorithm for Outerplanar Vertex Deletion with a low degree polynomial bound on the size of the reduced instance. While the kernelization algorithm is not directly of practical interest, the formulation of concrete reduction rules gives a good insight in the types of structures found in large graphs with a small solution size.

In Chapter 4 we studied $\mathcal{F}$-Minor-Free Deletion problems (such as Outerplanar Vertex Deletion) and considered as parameter the vertex-deletion distance to a constant treewidth. Specifically we considered the smallest such constant c for which the deletion distance to a treewidth of c is at most as large as the solution size. We showed that most $\mathcal{F}$-Minor-Free Deletion problems do not admit a polynomial kernel under this parameterization, even if we consider Turing kernelization. For all remaining $\mathcal{F}$-Minor-Free Deletion problems we gave a Turing kernelization algorithm.

We argued for a new perspective on preprocessing in Chapter 5. The goal of kernelization to reduce the *size* of problem instances does not always explain or guarantee the desired speed up of the followup algorithm tasked with solving the problem. We argue that a reduction in the parameter may result in a more significant speed up of the followup algorithm. We introduced the antler decomposition to describe a condition under which one can efficiently find part of an optimal solution to Feedback Vertex Set, thereby reducing the natural parameter. The techniques exploit local properties of the input graph to argue membership in an optimal solution.

In Chapter 6 we investigated the problem of finding large k-secluded trees, which is closely related to finding antler structures. We presented a $2^{\mathcal{O}(k \log k)} \cdot n^4$-time algorithm to find, enumerate, and count maximum-weight k-secluded trees. This improves the previously best known running time which was doubly exponential in k.

7.2 Future work

Two important themes in this thesis are kernelization for $\mathcal{F}$-Minor-Free Deletion and the exploration of a new research direction in preprocessing. For both themes there is a variety of open questions and opportunities for future work.

Kernelization for $\mathcal{F}$-Minor-Free Deletion A longstanding open problem is the question whether all $\mathcal{F}$-Minor-Free Deletion problems parameterized by solution size admit a polynomial kernel. A polynomial kernel can be obtained if $\mathcal{F}$ contains a planar graph. The techniques used for this kernelization rely on the fact that an $\mathcal{F}$-minor free graph has bounded treewidth (in the case that $\mathcal{F}$ contains a planar graph). Another approach is needed to obtain a polynomial kernel for $\mathcal{F}$-Minor-Free Deletion for the cases where $\mathcal{F}$ does not contain a planar graph. One of the simplest of these cases is $\mathcal{F} = \{K_5, K_{3,3}\}$ corresponding to the Planar Deletion problem. In recent work Jansen and Włodarczyk [85] showed Planar Deletion admits a polynomial constant-factor approximate kernel. Their work shows that kernelizing Planar Deletion can be reduced to analyzing planar subgraphs connected to the rest of the graph only through vertices on their outer face. The key to use these insights to obtain a polynomial kernel is determining whether a planar graph H with a set B of t boundary

vertices all incident to the outer face, has a vertex set A of size at most $t^{\mathcal{O}(1)}$ such that for any graph G that contains H as a subgraph with $N_G(V(G) \setminus H) = B$, there is an optimal solution avoiding $V(H) \setminus A$.

For problems that admit a polynomial kernel, the natural question is whether we can obtain tight bounds on the size of the kernel. For Planar-$\mathcal{F}$ Deletion (parameterized by solution size) we do not have any concrete bounds on the size of the kernel in general. For some Planar-$\mathcal{F}$ Deletion problem (such as Vertex Cover, Feedback Vertex Set, and Outerplanar Vertex Deletion) explicit bounds are known. We asked in Chapter 3 whether the upper and lower bounds for Outerplanar Vertex Deletion can be brought closer together. In general not a lot is known about how the size of kernels for Planar-$\mathcal{F}$ Deletion depends on $\mathcal{F}$. Giannopoulou et al. [65] show that for each fixed η the Treedepth-η Deletion problem admits a kernel with $\mathcal{O}(k^6)$ vertices while the degree of the polynomial bounding the number of vertices in a kernel for Treedwidth-η Deletion increases with η (unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$). Can we identify all choices of $\mathcal{F}$ for which a uniformly polynomial kernel exists for Planar-$\mathcal{F}$ Deletion? How does the degree of the polynomial bounding the kernel size depend on $\mathcal{F}$ for the remaining cases? With regard to upper bounds, the techniques used in Chapter 3 inspired an explicit kernelization bound for Treewidth-2 Deletion [118] leading to the question whether these techniques can be further generalized to obtain explicit bounds for other Planar-$\mathcal{F}$ Deletion problems.

In addition to determining the smallest polynomial bounding the kernel size of different Planar-$\mathcal{F}$ Deletion problems parameterized by solution size, an interesting question is if smaller parameterizations can still lead to polynomial kernels. In Chapter 4 we show that the deletion distance to a constant treewidth is not one such parameter. Deletion distance to a graph with constant treedepth does admit a polynomial kernel [81], at least for the case where $\mathcal{F}$ contains only connected graphs. Can we determine for a wide variety of parameters whether or not they allow polynomial kernels for Planar-$\mathcal{F}$ Deletion, assuming $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$? For the parameters deletion distance to a (minor-closed) graph class $\mathcal{G}$, some results have been attained recently. Feedback Vertex Set admits a polynomial kernel under such a parameterization if and only if $\mathcal{G}$ has a constant elimination distance to a forest [39]. Elimination distance to a forest is the minimum number of "rounds" required to obtain forest, where a round consists of removing a single vertex from each connected component. This work followed a similar result for Vertex Cover which admits a polynomial kernel under the given parameterization if and only if $\mathcal{G}$ has constant bridge-depth [24]. The bridge-depth of a graph is the minimum number of "rounds" required to obtain the null graph, where each round consists of first contracting bridges (edges whose removal increases the number of connected components) and then deleting a vertex from each connected component. Can such results be obtained for a broader range of Planar-$\mathcal{F}$ Deletion problems?

Preprocessing to reduce the parameter As discussed in Chapter 5, efficient algorithms to reduce the size of a problem instance are not necessarily the only effective method to speed up the followup algorithm tasked with finding a solution. A reduction in the parameter may be more effective, but under which conditions can such a reduction be achieved? We propose to consider problems parameterized by solution size and explore under which conditions one can efficiently obtain part of an optimal solution, thereby reducing the parameter. We explored this question for Feedback Vertex Set and showed how a partial solution can be found efficiently under the condition that the graph contains sufficiently simple antlers. More precisely, if G has an antler of width k and order z then we can find at least k vertices of a minimum feedback vertex set of G in $2^{\mathcal{O}(k^5 z^2)} n^{\mathcal{O}(z)}$ time.

A natural question is whether this running time can be improved, in particular the dependency on z. One can show that for certain types of antlers with a "well connected" certificate[1], of arbitrary order $z \leq k$ and width k, a running time of $f(k) \cdot n^{\mathcal{O}(1)}$ can be achieved as the number of subsets C considered in Step 3 of the algorithm given in Lemma 5.5.15 can then be reduced to $n^{\mathcal{O}(1)}$. For other types of antlers with more loosely connected certificates it is unclear if such a running time can be achieved. Rather than measuring the complexity of the certificates based on their order (the highest feedback vertex number of its connected components), we wonder if algorithms exist with an efficient running time expressed using another measure of complexity, for example the treewidth of the certificate.

Another direction for future research is to explore if graph structures, such as the crown decomposition for Vertex Cover and the antler decomposition for Feedback Vertex Set, exist for other $\mathcal{F}$-Minor-Free Deletion problems. A good first candidate may be Outerplanar Vertex Deletion for which it may be possible to use the reduction rules presented in Chapter 3 in a similar vein as we used reduction rules inspired by the kernels for Feedback Vertex Set in Chapter 5.

Finally, we ask whether other types of guarantees on reducing the parameter can be given. When a reduction in the parameter can be achieved through finding part of an optimal solution, this could mean identifying vertices that are "obviously" required in any optimal solution in some way, for example because avoiding such a vertex is associated with a high cost. A reduction rule used in the simple kernel for Vertex Cover is to put any vertex with a degree higher than k into the solution. This is a safe reduction since the cost of avoiding v (i.e. including all neighbors of v in the solution) is higher than the cost of an acceptable solution (k). Can we formulate weaker conditions such that v is still guaranteed to be in an

[1]Consider for example antlers (C, F) with a C-certificate H where each connected component H' of H contains a $K_{\mathsf{fvs}(H')+2}$-minor, i.e., $C \cap V(H')$ is an optimal feedback vertex set in H' as H' contains a clique on $|C \cap V(H')| + 2$ vertices as a minor. Then $H' - C$ contains a tree whose neighborhood (in H') is exactly equal to $C \cap V(H')$. This allows us, in Step 3 of the algorithm given in Lemma 5.5.15, to consider only $\mathcal{O}(n)$ vertex sets (rather than $\mathcal{O}(n^z)$) by considering the neighborhoods of $\mathcal{O}(n)$ trees.

optimal solution while maintaining the property that such vertices can be found efficiently? Can these ideas be used to find part of an optimal solution for other $\mathcal{F}$-Minor-Free Deletion problems?

Index of Definitions

Bibliography

[1] Faisal N. Abu-Khzam, Rebecca L. Collins, Michael R. Fellows, Michael A. Langston, W. Henry Suters, and Christopher T. Symons. Kernelization algorithms for the vertex cover problem: Theory and experiments. In *Proc. 6th ALENEX/ANALC*, pages 62–69, 2004.

[2] Faisal N. Abu-Khzam, Michael R. Fellows, Michael A. Langston, and W. Henry Suters. Crown structures for vertex cover kernelization. *Theory Comput. Syst.*, 41(3):411–430, 2007. doi:10.1007/s00224-007-1328-0.

[3] Tobias Achterberg, Robert E. Bixby, Zonghao Gu, Edward Rothberg, and Dieter Weninger. Presolve reductions in mixed integer programming. Technical Report 16-44, ZIB, Takustr.7, 14195 Berlin, 2016. URL: http://nbn-resolving.de/urn:nbn:de:0297-zib-60370.

[4] Tobias Achterberg and Roland Wunderling. *Mixed Integer Programming: Analyzing 12 Years of Progress*, pages 449–481. Springer Berlin Heidelberg, 2013. doi:10.1007/978-3-642-38189-8_18.

[5] Akanksha Agrawal, Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Feedback vertex set inspired kernel for chordal vertex deletion. *ACM Trans. Algorithms*, 15(1):11:1–11:28, 2019. doi:10.1145/3284356.

[6] Takuya Akiba and Yoichi Iwata. Branch-and-reduce exponential/FPT algorithms in practice: A case study of vertex cover. *Theor. Comput. Sci.*, 609:211–225, 2016. doi:10.1016/j.tcs.2015.09.023.

[7] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[8] Haris Angelidakis, Pranjal Awasthi, Avrim Blum, Vaggos Chatziafratis, and Chen Dan. Bilu-Linial stability, certified algorithms and the independent set problem. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *Proc. 27th ESA*, volume 144 of *LIPIcs*, pages 7:1–7:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.7.

[9] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. URL: http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264.

[10] Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. *Inf. Process. Lett.*, 112(1-2):49–54, 2012. doi:10.1016/j.ipl.2011.10.006.

[11] Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999. doi:10.1137/S0895480196305124.

[12] Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. Optimal algorithms for hitting (topological) minors on graphs of bounded treewidth. In *Proc. 12th IPEC*, volume 89 of *LIPIcs*, pages 4:1–4:12, 2017. doi:10.4230/LIPIcs.IPEC.2017.4.

[13] C. Berge. Sur le couplage maximum d'un graphe. *Comptes rendus hebdomadaires des séances de l'Académie des sciences*, 247:258–259, 1958.

[14] René van Bevern, Till Fluschnik, George B. Mertzios, Hendrik Molter, Manuel Sorge, and Ondrej Suchý. The parameterized complexity of finding secluded solutions to some classical optimization problems on graphs. *Discret. Optim.*, 30:20–50, 2018. doi:10.1016/j.disopt.2018.05.002.

[15] René van Bevern, Till Fluschnik, and Oxana Yu. Tsidulko. Parameterized algorithms and data reduction for the short secluded s-t-path problem. *Networks*, 75(1):34–63, 2020. doi:10.1002/net.21904.

[16] Therese C. Biedl. Small drawings of outerplanar graphs, series-parallel graphs, and other planar graphs. *Discret. Comput. Geom.*, 45(1):141–160, 2011. doi:10.1007/s00454-010-9310-z.

[17] Daniel Binkele-Raible, Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. Kernel(s) for problems with no kernel: On out-trees with many leaves. *ACM Trans. Algorithms*, 8(4):38, 2012. doi:10.1145/2344422.2344428.

[18] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996. doi:10.1137/S0097539793251219.

[19] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.

[20] Hans L. Bodlaender. Kernelization: New upper and lower bound techniques. In *Proc. 4th IWPEC*, pages 17–37, 2009. doi:10.1007/978-3-642-11269-0_2.

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

[22] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM J. Discrete Math.*, 28(1):277–305, 2014. doi:10.1137/120880240.

[23] Hans L. Bodlaender and Thomas C. van Dijk. A cubic kernel for feedback vertex set and loop cutset. *Theory Comput. Syst.*, 46(3):566–597, 2010. doi:10.1007/s00224-009-9234-2.

[24] Marin Bougeret, Bart M. P. Jansen, and Ignasi Sau. Bridge-depth characterizes which structural parameterizations of vertex cover admit a polynomial kernel. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th ICALP 2020*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:19, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2020.16.

[25] Marin Bougeret and Ignasi Sau. How much does a treedepth modulator help to obtain polynomial kernels beyond sparse graphs? *Algorithmica*, 81(10):4043–4068, 2019. doi:10.1007/s00453-018-0468-8.

[26] Kevin Burrage, Vladimir Estivill-Castro, Michael R. Fellows, Michael A. Langston, Shev Mac, and Frances A. Rosamond. The undirected feedback vertex set problem has a poly(k) kernel. In Hans L. Bodlaender and Michael A. Langston, editors, *Proc. 2nd IWPEC*, volume 4169 of *Lecture Notes in Computer Science*, pages 192–202. Springer, 2006. doi:10.1007/11847250_18.

[27] Kevin Cattell, Michael J. Dinneen, Rodney G. Downey, Michael R. Fellows, and Michael A. Langston. On computing graph minor obstruction sets. *Theor. Comput. Sci.*, 233(1-2):107–127, 2000. doi:10.1016/S0304-3975(97)00300-9.

[28] Gary Chartrand and Frank Harary. Planar permutation graphs. *Annales de l'I.H.P. Probabilités et statistiques*, 3(4):433–438, 1967.

[29] Shiri Chechik, Matthew P. Johnson, Merav Parter, and David Peleg. Secluded connectivity problems. *Algorithmica*, 79(3):708–741, 2017. doi:10.1007/s00453-016-0222-z.

[30] Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex cover: Further observations and further improvements. *Journal of Algorithms*, 41(2):280–301, 2001. doi:10.1006/jagm.2001.1186.

[31] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010. doi:10.1016/j.tcs.2010.06.026.

[32] Benny Chor, Mike Fellows, and David W. Juedes. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In *Proc. 30th WG*, pages 257–269, 2004. doi:10.1007/978-3-540-30559-0_22.

[33] David Coudert, Florian Huc, and Jean-Sébastien Sereni. Pathwidth of outerplanar graphs. *J. Graph Theory*, 55(1):27–41, 2007. doi:10.1002/jgt.20218.

[34] Bruno Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

[35] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, London, 2015. doi:10.1007/978-3-319-21275-3.

[36] Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. On the hardness of losing width. *Theory Comput. Syst.*, 54(1):73–82, 2014. doi:10.1007/s00224-013-9480-1.

[37] Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. An improved FPT algorithm and a quadratic kernel for pathwidth one vertex deletion. *Algorithmica*, 64(1):170–188, sep 2012.

[38] Amit Daniely, Nati Linial, and Michael E. Saks. Clustering is difficult only when it does not matter. *CoRR*, abs/1205.4891, 2012. arXiv:1205.4891.

[39] David Dekker and Bart M.P. Jansen. Kernelization for feedback vertex set via elimination distance to a forest. In *WG 2022*. To appear.

[40] Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *J. ACM*, 61(4):23:1–23:27, jul 2014. doi:10.1145/2629620.

[41] Emilio Di Giacomo, Giuseppe Liotta, and Tamara Mchedlidze. Lower and upper bounds for long induced paths in 3-connected planar graphs. *Theor. Comput. Sci.*, 636:47–55, 2016. doi:10.1016/j.tcs.2016.04.034.

[42] Reinhard Diestel. *Graph Theory, 5th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2017. doi:10.1007/978-3-662-53622-3.

[43] Guoli Ding and Stan Dziobiak. Excluded-minor characterization of apex-outerplanar graphs. *Graphs Comb.*, 32(2):583–627, 2016. doi:10.1007/s00373-015-1611-9.

[44] Michael J. Dinneen. Too many minor order obstructions. *J. Univers. Comput. Sci.*, 3(11):1199–1206, 1997. doi:10.3217/jucs-003-11-1199.

[45] Michael J. Dinneen, Kevin Cattell, and Michael R. Fellows. Forbidden minors to graphs with small feedback sets. *Discrete Math.*, 230(1-3):215–252, 2001. doi:10.1016/S0012-365X(00)00083-2.

[46] Michael J. Dinneen and Liu Xiong. Minor-order obstructions for the graphs of vertex cover 6. *J. Graph Theory*, 41(3):163–178, 2002. doi:10.1002/jgt.10059.

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

[48] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, London, 2013. doi:10.1007/978-1-4471-5559-1.

[49] Andrew Drucker. New limits to classical and quantum instance compression. *SIAM J. Comput.*, 44(5):1443–1479, 2015. doi:10.1137/130927115.

[50] M. Ayaz Dzulfikar, Johannes Klaus Fichte, and Markus Hecher. The PACE 2019 parameterized algorithms and computational experiments challenge: The fourth iteration (invited paper). In Bart M. P. Jansen and Jan Arne Telle, editors, *Proc. 14th IPEC*, volume 148 of *LIPIcs*, pages 25:1–25:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.IPEC.2019.25.

[51] Michael R. Fellows. Blow-ups, win/win's, and crown rules: Some new directions in FPT. In *Proc. 29th WG*, pages 1–12, 2003.

[52] Michael R. Fellows. The lost continent of polynomial time: Preprocessing and kernelization. In *Proc. 2nd IWPEC*, pages 276–277, 2006. doi:10.1007/11847250_25.

[53] Michael R. Fellows, Bart M.P. Jansen, and Frances Rosamond. Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. *European Journal of Combinatorics*, 34(3):541–566, 2013. Combinatorial Algorithms and Complexity. doi:10.1016/j.ejc.2012.04.008.

[54] Henning Fernau. Kernelization, Turing kernels. In *Encyclopedia of Algorithms*, pages 1043–1045. Springer, 2016. doi:10.1007/978-1-4939-2864-4_528.

[55] Herbert J. Fleischner, Dennis P. Geller, and Frank Harary. Outerplanar graphs and weak duals. *Journal of the Indian Mathematical Society*, 38, 1974. URL: http://www.informaticsjournals.com/index.php/jims/article/view/16694.

[56] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006. doi:10.1007/3-540-29953-X.

[57] Till Fluschnik. *Elements of Efficient Data Reduction: Fractals, Diminishers, Weights and Neighborhoods*. PhD thesis, Technische Universität Berlin, 2020. doi:10.14279/depositonce-10134.

[58] Fedor V. Fomin, Petr A. Golovach, Nikolay Karpov, and Alexander S. Kulikov. Parameterized complexity of secluded connectivity problems. *Theory Comput. Syst.*, 61(3):795–819, 2017. doi:10.1007/s00224-016-9717-x.

[59] Fedor V. Fomin, Bart M.P. Jansen, and Michał Pilipczuk. Preprocessing subgraph and minor problems: When does a small vertex cover help? *Journal of Computer and System Sciences*, 80(2):468–495, 2014. doi:10.1016/j.jcss.2013.09.004.

[60] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, Geevarghese Philip, and Saket Saurabh. Hitting forbidden minors: Approximation and kernelization. *SIAM J. Discrete Math.*, 30(1):383–410, 2016. doi:10.1137/140997889.

[61] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar $\mathcal{F}$-deletion: Approximation, kernelization and optimal FPT algorithms. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 470–479, Washington, 2012. IEEE Computer Society. doi:10.1109/FOCS.2012.62.

[62] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, Cambridge, 2019. doi:10.1017/9781107415157.

[63] Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011. doi:10.1016/j.jcss.2010.06.007.

[64] Fabrizio Frati. Straight-line drawings of outerplanar graphs in $O(dn \log n)$ area. *Computational Geometry*, 45(9):524–533, 2012. doi:10.1016/j.comgeo.2010.03.007.

[65] Archontia C. Giannopoulou, Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. Uniform kernelization complexity of hitting forbidden minors. *ACM Trans. Algorithms*, 13(3):35:1–35:35, March 2017. doi:10.1145/3029051.

[66] Petr A. Golovach, Pinar Heggernes, Paloma T. Lima, and Pedro Montealegre. Finding connected secluded subgraphs. *CoRR*, abs/1710.10979, 2017. arXiv:1710.10979.

[67] Petr A. Golovach, Pinar Heggernes, Paloma T. Lima, and Pedro Montealegre. Finding connected secluded subgraphs. *J. Comput. Syst. Sci.*, 113:101–124, 2020. doi:10.1016/j.jcss.2020.05.006.

[68] Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *J. Comput. Syst. Sci.*, 72(8):1386–1396, 2006. doi:10.1016/j.jcss.2006.02.001.

[69] Jiong Guo, Falk Hüffner, and Rolf Niedermeier. A structural view on parameterizing problems: Distance from triviality. In Rodney G. Downey, Michael R. Fellows, and Frank K. H. A. Dehne, editors, *Parameterized and Exact Computation, First International Workshop, IWPEC 2004, Bergen, Norway, September 14-17, 2004, Proceedings*, volume 3162 of *Lecture Notes in Computer Science*, pages 162–173. Springer, 2004. doi:10.1007/978-3-540-28639-4_15.

[70] Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, 38(1):31–45, 2007. doi:10.1145/1233481.1233493.

[71] Anupam Gupta, Euiwoong Lee, Jason Li, Pasin Manurangsi, and Michał Włodarczyk. Losing treewidth by separating subsets. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1731–1749. SIAM, 2019. doi:10.1137/1.9781611975482.104.

[72] Danny Hermelin, Stefan Kratsch, Karolina Soltys, Magnus Wahlström, and Xi Wu. A completeness theory for polynomial (Turing) kernelization. *Algorithmica*, 71(3):702–730, 2015. doi:10.1007/s00453-014-9910-8.

[73] Demian Hespe, Sebastian Lamm, Christian Schulz, and Darren Strash. Wegotyoucovered: The winning solver from the PACE 2019 implementation challenge, vertex cover track. *CoRR*, abs/1908.06795, 2019. arXiv:1908.06795.

[74] Demian Hespe, Christian Schulz, and Darren Strash. Scalable kernelization for maximum independent sets. *ACM Journal of Experimental Algorithmics*, 24(1):1.16:1–1.16:22, 2019. doi:10.1145/3355502.

[75] John E. Hopcroft and Robert Endre Tarjan. Efficient algorithms for graph manipulation (algorithm 447). *Commun. ACM*, 16(6):372–378, 1973. doi:10.1145/362248.362272.

[76] Yoichi Iwata. Linear-time kernelization for feedback vertex set. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 68:1–68:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2017.68.

[77] Yoichi Iwata and Yusuke Kobayashi. Improved analysis of highest-degree branching for feedback vertex set. In Bart M. P. Jansen and Jan Arne Telle, editors, *Proc. 14th IPEC*, volume 148 of *LIPIcs*, pages 22:1–22:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.IPEC.2019.22.

[78] Bart M. P. Jansen. Turing kernelization for finding long paths and cycles in restricted graph classes. *J. Comput. Syst. Sci.*, 85:18–37, 2017. doi:10.1016/j.jcss.2016.10.008.

[79] Bart M. P. Jansen and Hans L. Bodlaender. Vertex cover kernelization revisited - upper and lower bounds for a refined parameter. *Theory Comput. Syst.*, 53(2):263–299, 2013. doi:10.1007/s00224-012-9393-4.

[80] Bart M. P. Jansen and Stefan Kratsch. Data reduction for graph coloring problems. *Inf. Comput.*, 231:70–88, 2013. doi:10.1016/j.ic.2013.08.005.

[81] Bart M. P. Jansen and Astrid Pieterse. Polynomial kernels for hitting forbidden minors under structural parameterizations. *Theor. Comput. Sci.*, 841:124–166, 2020. doi:10.1016/j.tcs.2020.07.009.

[82] Bart M. P. Jansen and Marcin Pilipczuk. Approximation and kernelization for chordal vertex deletion. *SIAM J. Discret. Math.*, 32(3):2258–2301, 2018. doi:10.1137/17M112035X.

[83] Bart M. P. Jansen, Marcin Pilipczuk, and Marcin Wrochna. Turing kernelization for finding long paths in graph classes excluding a topological minor. *Algorithmica*, 81(10):3936–3967, 2019. doi:10.1007/s00453-019-00614-4.

[84] Bart M. P. Jansen, Venkatesh Raman, and Martin Vatshelle. Parameter ecology for feedback vertex set. *Tsinghua Science and Technology*, 19(4):387–409, 2014. doi:10.1109/TST.2014.6867520.

[85] Bart M. P. Jansen and Michał Włodarczyk. Lossy planarization: A constant-factor approximate kernelization for planar vertex deletion. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC

2022, page 900–913, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3520021.

[86] Gwenaël Joret, Christophe Paul, Ignasi Sau, Saket Saurabh, and Stéphan Thomassé. Hitting and harvesting pumpkins. *SIAM J. Discret. Math.*, 28(3):1363–1390, 2014. doi:10.1137/120883736.

[87] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[88] Joachim Kneis, Alexander Langer, and Peter Rossmanith. A new algorithm for finding trees with many leaves. *Algorithmica*, 61(4):882–897, 2011. doi:10.1007/s00453-010-9454-5.

[89] Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. In *Proc. 53rd FOCS*, pages 450–459, 2012. doi:10.1109/FOCS.2012.46.

[90] Stefan Kratsch and Magnus Wahlström. Compression via matroids: A randomized polynomial kernel for odd cycle transversal. *ACM Trans. Algorithms*, 10(4):20:1–20:15, 2014. doi:10.1145/2635810.

[91] Jens Lagergren. Upper bounds on the size of obstructions and intertwines. *J. Comb. Theory, Ser. B*, 73(1):7–40, 1998. doi:10.1006/jctb.1997.1788.

[92] Euiwoong Lee. Partitioning a graph into small pieces with applications to path transversal. *Math. Program.*, 177(1–2):1–19, sep 2019. doi:10.1007/s10107-018-1255-7.

[93] John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *J. Comput. Syst. Sci.*, 20(2):219–230, 1980. doi:10.1016/0022-0000(80)90060-4.

[94] J. Leydold and P. Stadler. Minimal cycle bases of outerplanar graphs. *Electron. J. Comb.*, 5, 1998. URL: http://eudml.org/doc/119549.

[95] Daniel Lokshtanov. *New Methods in Parameterized Algorithms and Complexity*. PhD thesis, University of Bergen, Norway, 2009.

[96] Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Kernelization - Preprocessing with a guarantee. In *The Multivariate Algorithmic Revolution and Beyond*, pages 129–161, 2012. doi:10.1007/978-3-642-30891-8_10.

[97] Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Trans. Algorithms*, 11(2):15:1–15:31, 2014. doi:10.1145/2566616.

[98] Max-Jonathan Luckow and Till Fluschnik. On the computational complexity of length- and neighborhood-constrained path problems. *Inf. Process. Lett.*, 156:105913, 2020. doi:10.1016/j.ipl.2019.105913.

[99] Wolfgang Mader. Homomorphieeigenschaften und mittlere Kantendichte von Graphen. *Mathematische Annalen*, 174(4):265–268, 1967. doi:10.1007/BF01364272.

[100] Tamara Mchedlidze and Antonios Symvonis. Crossing-optimal acyclic hp-completion for outerplanar st-digraphs. *J. Graph Algorithms Appl.*, 15(3):373–415, 2011. doi:10.7155/jgaa.00231.

[101] Kerri Morgan and Graham Farr. Approximation algorithms for the maximum induced planar and outerplanar subgraph problems. *J. Graph Algorithms Appl.*, 11(1):165–193, 2007. doi:10.7155/jgaa.00141.

[102] Moni Naor, Leonard J. Schulman, and Srinivasan Aravind. Splitters and near-optimal derandomization. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 182–191. IEEE Computer Society, 1995. doi:10.1109/SFCS.1995.492475.

[103] G.L. Nemhauser and L.E.jun. Trotter. Vertex packings: structural properties and algorithms. *Math. Program.*, 8:232–248, 1975. doi:10.1007/BF01580444.

[104] Rolf Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In Jean-Yves Marion and Thomas Schwentick, editors, *27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010, March 4-6, 2010, Nancy, France*, volume 5 of *LIPIcs*, pages 17–32. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010. doi:10.4230/LIPIcs.STACS.2010.2495.

[105] Geevarghese Philip, Venkatesh Raman, and Yngve Villanger. A quartic kernel for pathwidth-one vertex deletion. In Dimitrios M. Thilikos, editor, *Graph Theoretic Concepts in Computer Science - 36th International Workshop, WG 2010, Zarós, Crete, Greece, June 28-30, 2010 Revised Papers*, volume 6410 of *Lecture Notes in Computer Science*, pages 196–207, 2010. doi:10.1007/978-3-642-16926-7_19.

[106] Marcin Pilipczuk, Michal Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Network sparsification for steiner problems on planar and bounded-genus graphs. *ACM Trans. Algorithms*, 14(4):53:1–53:73, 2018. doi:10.1145/3239560.

[107] Timo Poranen. Heuristics for the maximum outerplanar subgraph problem. *J. Heuristics*, 11(1):59–88, 2005. doi:10.1007/s10732-005-6999-6.

[108] W. V. Quine. The problem of simplifying truth functions. *The American Mathematical Monthly*, 59(8):521–531, 1952. doi:10.2307/2308219.

[109] Siddharthan Ramachandramurthi. The structure and number of obstructions to treewidth. *SIAM Journal on Discrete Mathematics*, 10(1):146–157, 1997. doi:10.1137/S0895480195280010.

[110] Jean-Florent Raymond and Dimitrios M. Thilikos. Recent techniques and results on the erdős-pósa property. *Discret. Appl. Math.*, 231:25–43, 2017. doi:10.1016/j.dam.2016.12.025.

[111] Neil Robertson and Paul D. Seymour. Graph minors. V. Excluding a planar graph. *J. Comb. Theory, Ser. B*, 41(1):92–114, 1986. doi:10.1016/0095-8956(86)90030-4.

[112] Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

[113] Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *J. Comb. Theory, Ser. B*, 92(2):325–357, 2004. doi:10.1016/j.jctb.2004.08.001.

[114] Juanjo Rué, Konstantinos S. Stavropoulos, and Dimitrios M. Thilikos. Outerplanar obstructions for a feedback vertex set. *Eur. J. Comb.*, 33(5):948–968, 2012. doi:10.1016/j.ejc.2011.09.018.

[115] Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. An FPT-algorithm for recognizing k-apices of minor-closed graph classes. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 95:1–95:20, Dagstuhl, 2020. Schloss Dagstuhl - Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2020.95.

[116] Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. k-apices of minor-closed graph classes. I. Bounding the obstructions. *CoRR*, 2021. arXiv:2103.00882.

[117] Saket Saurabh. Open problems from the workshop on kernelization (WorKer 2019), 2019. URL: https://www.youtube.com/watch?v=vCjG5zGjQr4.

[118] Jeroen L. G. Schols. Kernelization for treewidth-2 vertex deletion, 2022. arXiv:2203.10070.

[119] Alexander Schrijver. *Combinatorial Optimization. Polyhedra and Efficiency.* Springer, Berlin, 2003.

[120] Maciej M. Syslo. Characterizations of outerplanar graphs. *Discret. Math.*, 26(1):47–53, 1979. doi:10.1016/0012-365X(79)90060-8.

[121] Stéphan Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Trans. Algorithms*, 6(2):32:1–32:8, 2010. doi:10.1145/1721837.1721848.

[122] P. Toth. Dynamic programming algorithms for the zero-one knapsack problem. *Computing*, 25:29–45, 1980. doi:10.1007/BF02243880.

[123] Johannes Uhlmann and Mathias Weller. Two-layer planarization parameterized by feedback edge set. *Theor. Comput. Sci.*, 494:99–111, 2013. doi:10.1016/j.tcs.2013.01.029.

[124] René Van Bevern, Hannes Moser, and Rolf Niedermeier. Approximation and tidying—a problem kernel for s-plex cluster vertex deletion. *Algorithmica*, 62(3):930–950, 2012. doi:10.1007/s00453-011-9492-7.

[125] Klaus Wagner. Über eine eigenschaft der ebenen komplexe. *Mathematische Annalen*, 114(1):570–590, 1937. doi:10.1007/BF01594196.

[126] Mathias Weller. *Aspects of Preprocessing Applied to Combinatorial Graph Problems*. PhD thesis, Technische Universität Berlin, 2013.

[127] Chee-Keng Yap. Some consequences of non-uniform conditions on uniform classes. *Theor. Comput. Sci.*, 26:287–300, 1983. doi:10.1016/0304-3975(83)90020-8.

Summary

Parameterized Algorithms for Finding Large Sparse Subgraphs

Many of the problems we would like computers to solve are NP-hard, which means that the computing time and power required to solve the problem is expected to increase superpolynomially with the size of the input. Simply put, we cannot expect to solve large instances of NP-hard problems in general. Parameterized algorithms are able to solve a portion of these large instances. In parameterized complexity an instance of a (parameterized) problem is associated with a parameter k, which can be seen as a measure of the complexity of the problem instance. We say a problem is fixed parameter tractable if there is an algorithm that can solve the problem in time $f(k) \cdot n^{\mathcal{O}(1)}$ for some function f. Such algorithms can solve large problem instances with a small parameter.

An important subfield of parameterized complexity is that of kernelization. A kernelization algorithm takes as input a parameterized instance of size n and parameter k and it produces in polynomial time an equivalent problem instance whose size and parameter are bounded by a function of k. Rather than solving the problem, a kernelization algorithm preprocesses the problem such that, if k is not too large, even an inefficient algorithm can solve the preprocessed instance in reasonable time.

In this thesis we study parameterized algorithms for problems that revolve around finding large sparse subgraphs of an input graph. The first result we present is a kernelization algorithm for the problem of Outerplanar Vertex Deletion. This problem asks to determine whether a graph can be made outerplanar using a small number of vertex deletions, or equivalently, whether it contains a large outerplanar subgraph. We show that if we parameterize the problem by the number of allowed vertex deletions k then there is an elementary kernelization algorithm that produces an equivalent instance of size at most $\mathcal{O}(k^4)$.

The concept of kernelization can be extended to Turing kernelization. Where a kernelization algorithm preprocesses the input once and relies on an external algorithm to solve the final preprocessed instance, a polynomial Turing kernelization algorithm is allowed to rely (during its entire operation) on an external algorithm to solve any number of small (size at most $k^{\mathcal{O}(1)}$) problem instances. In this thesis

we study the existence of polynomial Turing kernelizations for a certain parameterization of the $\mathcal{F}$-Minor-Free Deletion problem. This problem asks whether a given graph can be made $\mathcal{F}$-minor-free by removing a small number of vertices, or equivalently, whether the graph contains a large $\mathcal{F}$-minor-free subgraph. This meta-problem encompasses many classical NP-hard problems including Feedback Vertex Set. We show that for most choices of $\mathcal{F}$ such a polynomial Turing kernelization algorithm is unlikely to exist for the studied parameterization. This rules out a polynomial Turing kernel for (among others) P_3-Minor-Free Deletion parameterized by the feedback vertex number. To complete the dichotomy we show that for all remaining choices of $\mathcal{F}$ there does exist a polynomial Turing kernelization algorithm for the studied parameterization.

Kernelization and Turing kernelization are based on the idea that the algorithms that solve the problem require the size of the problem instance to be small. While this is true for traditional algorithms for NP-hard problems, parameterized algorithms allow us to solve large problem instances with small parameters. For these algorithms the influence of the size of the input on the running time is overshadowed by the influence of the parameter. Existence of a (Turing) kernelization algorithm does therefore not guarantee a faster overall running time. We observe however that often existing proprocessing algorithms speed up the solving process significantly. This can only be explained by a decrease in the parameter. It turns out that often preprocessing algorithms succeed in reducing the parameter. The notion of (Turing) kernelization does not guarantee a reduction of the parameter. In this thesis we consider the classical problem of Feedback Vertex Set parameterized by solution size and study under what conditions this parameter can be efficiently reduced. We introduce a graph structure we dubbed "Antler decomposition" and a parameterized algorithm that is able to efficiently reduce the parameter under the condition that the input graph contains a sufficiently simple antler decomposition.

Closely related to the antler decomposition is the concept of secluded trees. A k-secluded tree is an acyclic connected subgraph with at most k neighbors. We give a parameterized algorithm that enumerates all maximum size k-secluded trees of a graph using a branching algorithm.

Curriculum Vitae

Huib Donkers was born on 17 February 1993 in Vlissingen. After completing his secondary education in 2011 at Nehalennia SSG in Middelburg, he studied Computer Science and Engineering at Eindhoven University of Technology. In 2015 he received his Bachelor's degree followed by his Master's degree in 2017. From 2017 he started his PhD at Eindhoven University of Technology under the supervision of dr. Bart M.P. Jansen of which the results are presented in this dissertation. During his PhD he was awarded the best student paper and best paper awards at the 47th International Workshop on Graph-Theoretic Concepts in Computer Science in 2021.