Parameterized Graph Modification Beyond the Natural Parameter

Parameterized Graph Modification Beyond the Natural Parameter

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de rector magnificus prof.dr. S.K. Lenaerts, voor een commissie aangewezen door het College voor Promoties, in het openbaar te verdedigen op dinsdag 4 juli 2023 om 16:00 uur

door

Jari Johannes Henricus de Kroon

geboren te Tilburg

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

Voorzitter:	prof.dr. E.R. van den Heuvel
Promotoren:	dr. B.M.P. Jansen
	prof.dr. M.T. de Berg
Leden:	prof.dr. H.L. Bodlaender (Universiteit Utrecht)
	prof.dr. F. Fomin (University of Bergen)
	prof.dr. F.C.R. Spieksma
	dr. J. Nederlof (Universiteit Utrecht)
	prof.dr. D. Paulusma (Durham University)

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

The work in this thesis has been supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 803421, ReduceSearch).

Cover design by Jari de Kroon. Cover design based on an adaptation of a Delaunay triangulation of a random point set.

Printed by Ipskamp Printing, Enschede.

A catalogue record is available from the Eindhoven University of Technology Library.

ISBN: 978-90-386-5774-5

Acknowledgements

A lot has happened in the last four years, and there are a couple of people I want to thank for supporting me along the way. First and foremost I want to thank the people at TU/e who made my time as a PhD student a very pleasant experience. I have a lot to thank Bart for. I followed your bachelor and master course during my studies and I was pretty much hooked on algorithms. During the end of my master you allowed me to visit Fedor in Norway, where besides meeting many nice people I got to expand my love for hiking and running. During this time in Norway you asked me to do a PhD with you, and I am still very happy to have said yes. I really enjoyed our weekly meetings, where often we would find ourselves sitting silently lost in our thoughts. I also want to thank my other co-authors Ashwin, Ben, Diptapriyo, Huib, Michał, and Venkatesh. Your enthusiasm, attention to detail, and work ethic motivated me a lot. I want to thank my pre-pandemic office mates Shivesh, Henk, and Martijn and, after moving to the seventh floor, my new office mates Huib, Ruben, and Céline. Between work, you helped me gain a lot of geography knowledge that may or may not be useless and we spent quite some time overthinking solutions to puzzle games. I also want to thank my other past and current ALGA colleagues for fun discussions during lunch, the board games, and other social activities such as paddleboarding and (online) cooking sessions. Thank you Agnes, Aleksandr, Andrés, Arpan, Arthur, Bettina, Bram, Dániel, Irene, Irina, Jules, Kevin[2], Leonidas, Leonie, Leyla, Marcel, Mark, Max[2], Meivan, Morteza, Nathan, Pantea, Thijs[2], Tim, Tom, Willem, and Wouter.

During my PhD I had the privilege to travel to many interesting places for various workshops and conferences, places such as Bergen in Norway and Koper in Slovenia. I am thankful to the many people that I got to meet during these trips that shared my interest in algorithmic graph theory. I also want to thank the anonymous reviewers of these conferences that provided invaluable feedback to my submissions as well as the committee members for their time and effort spent during their evaluation of this thesis.

I would also like to thank some people outside of academia. I want to thank the people who kept me distracted during numerous (sport) activities: my teammates and coaches at the Spoordonkse Boys (r.i.p. Eric), my friends for our weekly futsal sessions and various (road)trips, my sisters for introducing me to bouldering (to which I in turn dragged along some of my friends), and finally the friends with whom I play the occasional online game. Also a thank you to my family, especially my parents. You motivated me to pursue the things in life that bring me joy, something I am very grateful for.[1] Despite having an empty nest you keep a bedroom ready at all times, making it feel like I have two places to call home. Finally a thank you to Anouk. Most of the last four years was spent in our lovely small apartment in Eindhoven and we both grew a lot during this time. Thank you for your loving support and listening to me talk about my (graph) problems.

[1] Happiness is een theesmaakje.

Contents

Part I

Introduction

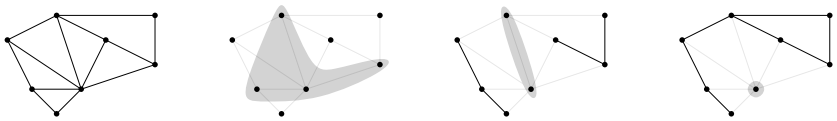
Chapter 1

Prelude

An algorithm is a generic set of instructions that can be used (by a computer) to perform some task for any given input. Common tasks that are solved by a computer are for instance sorting a list of numbers, or computing a shortest path between two points in a road network. There can be many algorithms for the same task. The study of algorithms aims to design new algorithms, proving their correctness, and analyzing the number of steps needed to complete the task in terms of the size of the input. Other aspects such as computer memory requirements may be considered too.

This thesis is devoted to the study of algorithms whose input is a graph. A graph G consists of a set of vertices $V(G)$ and a set of edges $E(G)$ that connect some pairs of vertices. Figure 1.1a depicts a graph where vertices are represented by points and edges by lines. Graphs can have many properties, some of which are based on the absence of certain structures contained in them such as cycles. A cycle in a graph consists of a list of vertices where the first and last vertex are the same, such that each consecutive pair of vertices is connected by an edge. A graph is acyclic if it contains no cycles and a graph is bipartite if it has no cycles of odd length. Besides these examples, many more graph properties exist [30].

The tasks we consider are so-called *vertex-deletion problems*: find the smallest vertex set whose removal results in a graph with some desirable property. This graph modification problem is also known as $\mathcal{H}$-deletion, where $\mathcal{H}$ is the class of graphs with the desired property. A deletion set whose removal results in a graph in $\mathcal{H}$ is often called an $\mathcal{H}$-modulator or $\mathcal{H}$-deletion set. Figure 1.1 shows vertex-deletion sets to an edgeless, an acyclic, and a bipartite graph. The corresponding vertex-deletion problems are known in the



(a) (b) (c) (d)

Figure 1.1: A graph (a) with vertex-deletion sets to an edgeless (b), an acyclic (c), and a bipartite (d) graph. Deleted vertices (and their incident edges) are displayed in grey.

literature as Vertex Cover (VC), Feedback Vertex Set (FVS), and Odd Cycle Transversal (OCT) respectively.

Many problems can be stated as graph modification problems, with applications in for instance data clustering [103] and deadlock prevention [156]. Furthermore, some have been the subject of theoretical study for over half a century [117], adding to the development of the theory of NP-completeness [88] and leading to the discovery of new algorithmic design techniques [159]. As an example, the following murder mystery hides a graph modification problem.[1]

Who killed the Duke of Densmore?

A while ago the Duke of Densmore was killed by a bomb. The bomb was placed in the bedroom of the castle to which he moved after divorcing from his eighth wife. The will of the Duke was lost during the explosion, which supposedly was very unfavorable for one of his ex-wives. Before the explosion, the Duke had invited all of his eight ex-wives to his castle. The police suspect that one of the ex-wives was the culprit, but were unable to prove it. Sherlock Holmes and Watson are asked to solve this unsolved case. Sherlock remembers hearing about the case and recalls that the bomb was manufactured to fit perfectly inside one of the armor statues in the Duke's bedroom, implying that the murderer must have visited the castle multiple times. Holmes and Watson interview all of the eight ex-wives. They cannot remember exactly during what time they were at the castle, but they do remember which of the other ex-wives they met at the castle:

- Ann met Betty, Charlotte, Edith, Felicia, and Georgia.
- Betty met Ann, Charlotte, and Helen.

[1] Adapted from 'Qui a tué le duc de Densmore?' by Claude Berge.

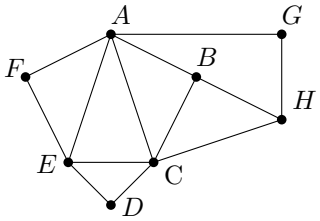


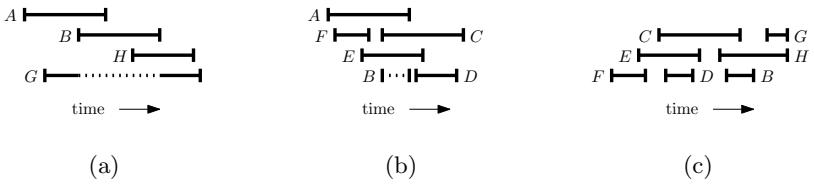
Figure 1.2: Meeting graph of the ex-wives.

- Charlotte met Ann, Betty, Diana, Edith, and Helen.
- Diana met Charlotte and Edith.
- Edith met Ann, Charlotte, Diana, and Felicia.
- Felicia met Ann and Edith.
- Georgia met Ann and Helen.
- Helen met Betty, Charlotte, and Georgia.

All of them swear that they only visited the castle once (they lived at other estates during their relationship with the Duke). Holmes notices their answers match, for instance Ann said she met Betty and Betty said she met Ann. The entrance to the castle is located near the great hall where the wives were sitting, so it is safe to assume that if two wives were at the castle at the same time then at least one would have seen the other. Sherlock draws the graph in Figure 1.2 corresponding to their answers and abbreviates the ex-wives by their first letter. He exclaims: this graph uniquely determines a single murderer!

In order to see see how an understanding of graph modification problems allowed Sherlock to solve the mystery, notice that each ex-wife (except the lying murderer) was at the castle once, and their stay can be represented by a time interval $[t_a, t_d]$ with arrival time t_a and departure time t_d. Two innocent ex-wives met at the castle if and only if their time intervals overlap.

First consider the part of the graph with vertices A, B, G, and H. Suppose none of them is the murderer, then since there is no edge between A and H, their time intervals should not overlap. The time interval of B should overlap with both intervals of A and H. This situation is shown in Figure 1.3a. But then the time interval of G should overlap with both intervals of A and H, while not overlapping with the interval of B. This is impossible, so we can conclude that either Ann, Betty, Georgia, or Helen is the murderer. The same reasoning also works for the part of graph with vertices A, C, H, and G. Assuming there is only one murderer, this rules out Betty as a suspect.



(a) (b) (c)

Figure 1.3: Two partial incorrect interval models in (a) and (b). In the latter it is impossible to assign an interval for B that overlaps A and C but not E. A complete interval model of the innocent ex-wives is given in (c).

Finally consider the part of the graph with vertices A, B, C, D, E, and F, and suppose none of them is the murderer. The intervals of A, C, and E should be pairwise overlapping, while each remaining interval should only overlap with two of them. This situation is shown in Figure 1.3b. This time it is less obvious, but no matter which way you draw the intervals of A, C, and E, at least one of the remaining intervals cannot be drawn while respecting the correct adjacencies of the graph. Since neither Georgia nor Helen is part of this set of vertices, we have found our murderer: it was Ann! Figure 1.3c shows a valid drawing of the intervals of the seven innocent ex-wives.

A set of time intervals as described above is known as an interval model. The intersection graph of such a model has a vertex for each interval and an edge whenever two intervals overlap. Such a graph is known as an interval graph. Note that after deleting one vertex from the graph in Figure 1.2, namely the vertex of the murderer, the remaining graph consists of the innocent ex-wives and therefore there should be an interval model of their visits to the castle. In other words, after deleting one vertex from the graph, the remaining graph should be an interval graph. The murder mystery is a graph modification problem.

Deleting more vertices

In the example above we only required to delete a single vertex from the graph to obtain a solution. A computer can quickly check for each vertex if deleting it results in an interval graph (this can be computed efficiently). If however the deletion set becomes larger, it becomes less clear how a computer can efficiently find an optimal solution. Suppose the murderer had help and we want to find the smallest group of liars whose omission results in consistent data.

Now you might think: computers are fast, could we try all options? The number of vertex subsets of an n-vertex graph is 2^n. Naively, for each vertex

Table 1.1: Indication of the time it takes to enumerate all 2^n vertex-subsets of an n-vertex graph if we could enumerate them at a speed of 5 gigahertz ($5 \cdot 10^9$ per second). At the time of writing this is the upper end of the spectrum for a CPU in a personal computer. As we need at least one clock cycle for each enumerated item, these times are a very conservative estimate.

n (size)	2^n (subsets)	$2^n/(5 \cdot 10^9)$ (time)
25	33554432	7 milliseconds
50	1.1 quadrillion	63 days
75	$3.8 \cdot 10^{22}$	239434 years
100	$1.27 \cdot 10^{30}$	8 trillion years

subset, you could try if deleting the vertices results in an interval graph. The solution would then be the smallest subset whose deletion resulted in an interval graph. While checking if a graph is interval can be done efficiently, Table 1.1 shows that even for a 100 vertex graph enumerating all vertex subsets would take longer than the age of planet earth.

Since the naive approach was not feasible, one might hope that there exists a different algorithm that can solve the problem efficiently. There can be many algorithms for the same task after all. However, assuming the widely accepted complexity assumption $\mathsf{P} \neq \mathsf{NP}$, it turns out that no algorithm whose time scales with a polynomial function of n can solve the problem. Interval Deletion is not the only hard vertex-deletion problem. Lewis and Yannakakis [127] showed that vertex-deletion is NP-hard for all non-trivial graph properties that are closed under deleting vertices. This includes problems like Vertex Cover, Feedback Vertex Set, and Odd Cycle Transversal that were mentioned before. There are various ways of dealing with NP-hard problems, one of which being parameterized complexity.

Parameterized complexity

The complexity assumption $\mathsf{P} \neq \mathsf{NP}$ states that for so-called NP-hard problems, no algorithm is able to efficiently produce an optimal solution for all instances. Here efficiently means that the running time of the algorithm is bounded by some polynomial of the input size. In order to deal with these problems, several approaches exist. A first option is to design exponential-time algorithms (see textbook [77]). In practice such algorithms are only able to solve very small instances. An example of such an algorithm is the naive interval deletion algorithm discussed before that just loops over all possible options. A second

option is to consider dropping the optimality requirement and to settle for solutions which may not be optimal, but which are provably not much worse than an optimal solution. The field of approximation algorithms deals with these types of algorithms (see textbooks [169, 173]). In this thesis we take the viewpoint of *parameterized complexity*, where the aim is to discover which aspects (besides the total input size) cause an input to be hard to solve. The goal is to give algorithms that scale well with the total input size, but may scale exponentially in these other aspects.

In the case of vertex-deletion problems, one could think about input instances where only a small number of vertices (say k) need to be deleted to obtain the desired property. Can we efficiently find such solutions? For some graph properties the answer turns out to be yes. For instance, there is an algorithm for Odd Cycle Transversal whose running time is proportional to $3^k \cdot n(n+m)$ (here n and m are the number of vertices and edges of the input graph respectively).[2] Note that the exponential part of the running time only scales with the size k of the solution we are looking for, not with the total size of the graph. This indicates that, if we are looking for 'small' solutions, we can still find them efficiently even though the problem is NP-hard. The input to a parameterized problem consists of a problem instance of size n and a parameter (often denoted by k). One would like to find an algorithm for the problem that solves it in time $f(k) \cdot n^c$ for some constant c. The running time scales polynomially with the input size n, but may scale arbitrarily badly with the parameter k depending on the function $f(\cdot)$. If such an algorithm exists, the parameterized problem is said to be *fixed-parameter tractable* (FPT). Over the last few decades the field of parameterized complexity has grown into a large and active research area. There is a big toolkit of techniques that can be used to come up with FPT algorithms, overviews of which can be found in several textbooks [48, 59, 60, 82].

Choice of parameter

Recall that for vertex-deletion problems, the task is to delete a minimum number of vertices such that the remaining graph has a certain desired property. For these kinds of problems, the size of a minimum solution is often used as the parameter and is referred to as the *natural* parameter. An FPT algorithm with this parameter restricts the exponential blowup of the running time to the size of the solution. The problems VC, FVS, OCT, and Interval Deletion that have been mentioned so far are known to be FPT with respect to this natural

[2] This is not the best known dependence on k.

parameter. This is not the only possible choice of a parameter, for instance for weighted problems we could consider the target weight as a parameter. Besides parameters that are related to the problem, like the solution size, other commonly used parameters are related to some graph structure. There is a plethora of such parameters, like treewidth, treedepth, genus, degeneracy, pathwidth, and many others [70]. Many NP-hard graph problems become efficiently solvable on graphs where these structural parameters have small values [98]. There are a lot of possibilities when it comes to choosing a parameter. The main question that this thesis is concerned with is the following: can we restrict the exponential blowup of the running time for vertex-deletion problems to even smaller parameters? We explore parameterized graph modification beyond the natural parameter in two directions.

Hybrid parameterization

The first direction consists of so-called hybrid parameters, which we deal with in Part II. Here the aim is to use parameters that are a hybrid between the natural parameter (solution size) and some structural graph parameter. We focus our attention to *treedepth* and *treewidth*, the latter of which arguably being the most well-known structural graph parameter. These hybrid parameters were recently introduced and Chapter 3 is dedicated to giving relevant background and formal definitions. Intuitively, but slightly incorrectly, one could think of these hybrid parameters as being the minimum treedepth or treewidth of any $\mathcal{H}$-modulator. The aim is then to obtain algorithms that use the structural measure of a solution as the parameter rather than its size.

The treedepth of a graph is equivalent to the minimum number of rounds needed to delete all vertices, where in each round you get to delete one vertex from each connected component of the graph. Instead of stopping at the empty graph, we can stop this procedure when the graph belongs to some graph class $\mathcal{H}$. This alteration of treedepth is known as $\mathcal{H}$-elimination distance and is never larger than either the treedepth or the minimum size of an $\mathcal{H}$-modulator of the graph. In Chapter 4 we show that computing $\mathcal{H}$-elimination distance is (non-uniform) FPT for various graph classes $\mathcal{H}$, most notably for the class of bipartite graphs.

The treewidth of a graph says something about how close the graph is to being a tree. Similarly to treedepth, this notion can be generalized to $\mathcal{H}$-treewidth, where 'leaves of the tree' can be arbitrarily large parts of the graph that belong to the graph class $\mathcal{H}$ while having only a small neighborhood. This is an even stronger parameter, meaning that whenever $\mathcal{H}$-elimination distance is bounded, so is $\mathcal{H}$-treewidth. In Chapter 5 we give FPT algorithms

for computing decompositions of bounded $\mathcal{H}$-treewidth. We show that whenever $\mathcal{H}$-Deletion is FPT parameterized by solution size, then we can compute a decomposition whose width is at most eight times the $\mathcal{H}$-treewidth of the graph. In Chapter 6 we give FPT algorithms for vertex-deletion problems that use these decompositions. In particular, for classes of bipartite and planar graphs, we obtain algorithms whose dependency on the hybrid parameter is as good as the best-known dependency on the natural parameter, which are known to be tight under the Exponential Time Hypothesis.

Essential vertices

In the second direction we take a different approach. Existing FPT results often show that solutions can be found efficiently when they are small. We wonder: can we still do something for instances whose optimal solution is large? The hybrid parameterization route from Part II answers the question positively whenever solutions are large but structured. In Part III instead we try to figure out under which conditions we can find a part of an optimal solution. If we succeed in finding part of an optimal solution, then the FPT algorithm that uses the natural solution size parameterization is sped up as it is looking for a smaller solution. The complexity assumption $\mathsf{P} \neq \mathsf{NP}$ implies that we cannot always succeed in finding part of an optimal solution in polynomial time. If we could, then by repeatedly executing this preprocessing step we would find an optimal solution in polynomial time for NP-hard problems. Therefore our task description should be slightly more novel. We define the notion of c-essential vertices, which are vertices that belong to all solutions whose size is at most c times the optimal solution size. In Chapter 7 we give algorithms for finding c-essential vertices for various vertex-deletion problems. Our positive results are mostly based on a special type of packing/covering duality in graphs which after one vertex deletion belong to the target graph class $\mathcal{H}$. For instance, we show how to find all 2-essential vertices for the Odd Cycle Transversal problem. Furthermore we show how to use this subroutine to give an FPT algorithm for vertex-deletion problems whose parameter is not the total solution size, but the number of non-essential vertices of a solution.

List of publications

Chapter 4 is based on the following publication.

> Bart M. P. Jansen and Jari J. H. de Kroon. FPT algorithms to compute the elimination distance to bipartite graphs and more. In

Proceedings of the 47th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2021, volume 12911 of *Lecture Notes in Computer Science*, pages 80–93. Springer, 2021. doi:10.1007/978-3-030-86838-3_6

Chapter 5 is based on the following manuscript that is under submission.

Bart M. P. Jansen, Jari J. H. de Kroon, and Michał Włodarczyk. Beyond important separators: Single-exponential FPT algorithms for decomposing into secluded $\mathcal{H}$-subgraphs.

Chapter 6 is based on the following publication.

Bart M. P. Jansen, Jari J. H. de Kroon, and Michał Włodarczyk. Vertex deletion parameterized by elimination distance and even less. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2021*, pages 1757–1769. ACM, 2021. doi:10.1145/3406325.3451068

Chapter 7 is based on the following publication.

Benjamin Merlin Bumpus, Bart M. P. Jansen, and Jari J. H. de Kroon. Search-space reduction via essential vertices. In *Proceedings of the 30th Annual European Symposium on Algorithms, ESA 2022*, volume 244 of *LIPIcs*, pages 30:1–30:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.30

The following articles were contributed to during the PhD project, but are not covered in the thesis.

- Bart M. P. Jansen and Jari J. H. de Kroon. Preprocessing vertex-deletion problems: Characterizing graph properties by low-rank adjacencies. *J. Comput. Syst. Sci.*, 126:59–79, 2022. doi:10.1016/j.jcss.2021.12.003
- Huib Donkers, Bart M. P. Jansen, and Jari J. H. de Kroon. Finding k-secluded trees faster. In *Proceedings of the 48th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2022*, volume 13453 of *Lecture Notes in Computer Science*, pages 173–186. Springer, 2022. doi:10.1007/978-3-031-15914-5_13
- Ashwin Jacob, Jari J. H. de Kroon, Diptapriyo Majumdar, and Venkatesh Raman. Parameterized complexity of deletion to scattered graph classes. *CoRR*, abs/2105.04660, 2021. arXiv:2105.04660

Chapter 2

Preliminaries

This chapter introduces several notions and definitions that form the starting point of this thesis. Section 2.1 introduces some basic notation regarding sets and functions. Section 2.2 covers relevant background material about graphs, classes of graphs, and several graph parameters. Section 2.3 deals with notions from computational complexity, mainly covering parameterized complexity. Finally Section 2.4 introduces graph problems that are considered in this thesis.

2.1 Sets and functions

We assume basic familiarity with set notation, we mention some important notions and conventions here. Sets that contain elementary elements (like vertices or edges) are often denoted by upper case letters (e.g. S, T), whereas sets containing more involved objects (like sets of vertices or graphs) are denoted by calligraphic letters (e.g. $\mathcal{Y}$, $\mathcal{H}$). The set $\{1, \ldots, p\}$ is denoted by $[p]$. For a finite set S, we denote by 2^S the powerset of S consisting of all its subsets. A set S with property P is said to be inclusion-minimal if no strict subset of S satisfies P. A partition of a set S is a collection $S_1, \ldots, S_\ell$ of subsets of S such that $S = \bigcup_{i \in [\ell]} S_i$ and $S_i \cap S_j = \emptyset$ for each $i, j \in [\ell]$. Note that this definition allows $S_i = \emptyset$ for any $i \in [\ell]$, this is relevant when quantifying over all possible partitions for a fixed ℓ.

Consider the function $f \colon A \to B$. For $A' \subseteq A$, by $f_{|A'}$ we denote the function f whose domain is restricted to A'. For $b \in B$, we denote the set of elements $a \in A$ such that $f(a) = b$ by $f^{-1}(b)$.

2.2 Graphs

A graph G consists of a set of vertices $V(G)$ (sometimes referred to as nodes) and a set of edges $E(G)$ between pairs of vertices. Unless specified otherwise, we consider graphs that are undirected, finite, and simple (no duplicate edges or self-loops). In this setting, each edge is a set consisting of exactly two distinct vertices. An edge $\{u, v\} \in E(G)$ may be written as uv for short. We refer to $|V(G)|$ by n and $|E(G)|$ by m if there can be no confusion about which graph they are referring to.

For a set of vertices $S \subseteq V(G)$, by $G[S]$ we denote the graph induced by S, which is the graph on vertex set S and edge set $\{uv \in E(G) \mid u, v \in S\}$. Any such graph is an induced subgraph of G. We use shorthand $G - v$ and $G - S$ for $G[V(G) \setminus \{v\}]$ and $G[V(G) \setminus S]$, respectively. The open neighborhood $N_G(v)$ of $v \in V(G)$ is defined as $\{u \in V(G) \mid \{u, v\} \in E(G)\}$. The closed neighborhood of v is $N_G[v] = N_G(v) \cup \{v\}$. For $S \subseteq V(G)$, we have $N_G[S] = \bigcup_{v \in S} N_G[v]$ and $N_G(S) = N_G[S] \setminus S$. We may drop subscripts referring to a graph G whenever it is clear from context. Two non-adjacent vertices u and v are said to be false twins if $N_G(u) = N_G(v)$. True twins are adjacent and satisfy $N_G[u] = N_G[v]$ instead.

A walk of length $\ell \geq 0$ in a graph G is a sequence $(v_1, \ldots, v_{\ell+1})$ of $\ell + 1$ vertices such that $v_i v_{i+1} \in E(G)$ for each $i \in [\ell]$. A walk is closed if $v_1 = v_{\ell+1}$, that is, the first and last vertex of the sequence are the same. A path of length $\ell \geq 0$ in G is a walk of length ℓ where the sequence consists of distinct vertices. Observe that a single vertex is a walk/path of length zero. A cycle of length $\ell \geq 3$ is a sequence $(v_1, \ldots, v_\ell)$ of ℓ distinct vertices with edges between consecutive pairs and an edge between v_1 and v_ℓ. Note that the length of a path or cycle is determined by the number of edges. A cycle or path is induced if no edges other than those defined above are present in the graph. A chordless cycle (also known as a hole) is an induced cycle of length at least four. Induced paths are also sometimes referred to as being chordless. A cycle or path is odd if its length (number of edges) is odd. A uv-path is a path whose first vertex is u and last vertex is v, the vertices u and v are said to be the endpoints of the path. Two uv-paths are internally vertex disjoint if their vertex sets only intersect at u and v. A graph G is connected if there is a uv-path for all $u, v \in V(G)$. An (S, T)-path is a uv-path with $u \in S$ and $v \in T$. A set $C \subseteq V(G)$ is called connected in G if the graph $G[C]$ is connected. A connected component of a graph is an inclusion maximal induced subgraph that is connected. We may sometimes refer to the vertex set of a connected component rather than the subgraph. The set of connected components of G is denoted $\mathrm{cc}(G)$.

For a graph G and vertex sets $S, P \subseteq V(G)$, we denote by $R_G(S, P)$ the set of vertices which can be reached (by a path) in $G - P$ from at least one vertex in the set $S \setminus P$.

Separators. For two sets $S, T \subseteq V(G)$ in a graph G, a set $P \subseteq V(G)$ is an unrestricted (S, T)-separator if no connected component of $G - P$ contains a vertex from both $S \setminus P$ and $T \setminus P$. Note that such a separator may intersect $S \cup T$. Equivalently, P is an (S, T)-separator if each (S, T)-path contains a vertex of P. The minimum cardinality of such a separator is denoted $\lambda_G(S, T)$.

A restricted (S, T)-separator is an unrestricted (S, T)-separator P which satisfies $P \cap (S \cup T) = \emptyset$. A left-restricted (S, T)-separator is an unrestricted (S, T)-separator P which satisfies $P \cap S = \emptyset$. Let $\lambda^{\mathrm{L}}_G(S, T)$ denote the minimum size of a left-restricted (S, T)-separator, or $+\infty$ if no such separator exists (which happens when $S \cap T \neq \emptyset$). Menger's theorem gives the following relation between path packings and separators, which can be computed for instance via the Ford-Fulkerson method.

Theorem 2.1 ([166, §9][44, §26])**.** *For any graph G and $S, T \subseteq V(G)$, the minimum cardinality of an unrestricted (S, T)-separator is equal to the maximum cardinality of a packing of vertex disjoint (S, T)-paths. Similarly, for any graph G and non-adjacent $s, t \in V(G)$, the minimum cardinality of a restricted $(\{s\}, \{t\})$-separator is equal to the maximum cardinality of a packing of internally vertex disjoint st-paths. Moreover, these packings and separators can be computed in polynomial time.*

In the field of parameterized complexity a special type of separator, known as an important separator, is often used. These were pioneered by Marx [140, 143] and refined by follow-up work by several authors [40, 131]. A (restricted) (S, T)-separator P is important if it is inclusion-minimal and there is no (restricted) (S, T)-separator P' such that $|P'| \leq |P|$ and $R_G(S, P) \subsetneq R_G(S, P')$. In words, a minimal separator is important if no separator of equal size can leave more vertices reachable from S. With this definition, they show that the number of important (restricted, but any set of forbidden vertices can be picked [48, Definition 8.49]) (S, T)-separators in an n-vertex graph G with vertex sets S and T can be bounded.

Theorem 2.2 ([48, Theorem 8.51])**.** *Let $S, T \subseteq V(G)$ and $k \geq 0$ be an integer. The set $\mathcal{S}_k$ of all important (restricted) (S, T)-separators has size at most 4^k and can be constructed in time $\mathcal{O}(|\mathcal{S}_k| \cdot k^2 \cdot (n + m))$.*

While we do not directly use important separators in our results, we do discuss them in order to compare to our enumeration algorithm in Chapter 5.

Contractions and minors. A contraction of an edge $uv \in E(G)$ is an operation that turns a graph G into another graph G'. Given a graph G, it introduces a new vertex adjacent to all of $N_G(\{u, v\})$, after which both u and v are deleted. By this definition, doing a contraction in a simple graph results in a simple graph as no parallel edges are created. The graph obtained by contracting $uv \in E(G)$ is denoted G/uv.

We say that a graph H is a *contraction* of a graph G, if we can turn G into H by a series of edge contractions. Furthermore, H is a *minor* of G, if we can turn G into H by a series of edge contractions, edge deletions, and vertex deletions. We can represent the result of such a process with a mapping $\phi\colon V(H) \to 2^{V(G)}$, such that subgraphs $(G[\phi(h)])_{h\in V(H)}$ are connected and vertex-disjoint, with an edge of G between a vertex in $\phi(u)$ and a vertex in $\phi(v)$ for all $uv \in E(H)$. The sets $\phi(h)$ are called branch sets and the family $(\phi(h))_{h\in V(H)}$ is called a minor-model of H in G.

2.2.1 Graph classes and special graphs

So far we have seen several properties and notions pertaining to individual graphs. This subsection introduces some relevant definitions regarding classes of graphs.

Two graphs G and H are isomorphic if there is a bijection $f\colon V(G) \to V(H)$ such that $uv \in E(G)$ if and only if $f(u)f(v) \in E(H)$. We say that a graph G contains a graph H if there is an induced subgraph of G isomorphic to H. If G does not contain H, it is said to be H-free. A class of graphs, or graph class, is a (possibly infinite) set of graphs. For a graph class $\mathcal{H}$, G is said to be $\mathcal{H}$-free if it is H-free for each $H \in \mathcal{H}$. An overview of various graph classes and their properties is given in [30]. A graph class $\mathcal{H}$ is called *hereditary* if for any $G \in \mathcal{H}$, every induced subgraph of G also belongs to $\mathcal{H}$. For consistency reasons, we say that each hereditary class contains the null graph (the graph on zero vertices). A graph class $\mathcal{H}$ is hereditary if and only if there exists a (possibly infinite) collection of graphs $\mathcal{F}_\mathcal{H}$ such that $G \in \mathcal{H}$ if and only if G is $\mathcal{F}_\mathcal{H}$-free. Such a collection $\mathcal{F}_\mathcal{H}$ is known as a forbidden induced subgraph characterization for $\mathcal{H}$. When referring to such a characterization, we usually mean the set of vertex-minimal graphs that do not belong to $\mathcal{H}$, that is, the set of graphs $F \notin \mathcal{H}$ such that $F - v \in \mathcal{H}$ for each $v \in V(F)$. Each hereditary graph class is uniquely characterized by such a vertex-minimal collection. The disjoint union of two graphs G_1 and G_2 is the graph on vertex set $V(G_1) \cup V(G_2)$ and edge set $E(G_1) \cup E(G_2)$, where vertices of G_2 are annotated to enforce $V(G_1) \cap V(G_2) = \emptyset$. A graph class $\mathcal{H}$ is *union-closed* if for any $G_1, G_2 \in \mathcal{H}$, the disjoint union of G_1 and G_2 also belongs to $\mathcal{H}$.

A graph is complete if each pair of distinct vertices is adjacent, the complete graph with q vertices is denoted K_q. A set of pairwise adjacent vertices is referred to as a clique. A graph is empty if it has no edges and a set of pairwise non-adjacent vertices is called an independent set (in the literature also commonly referred to as a stable set).

Trees. A tree is a connected graph that is acyclic, that is, it contains no cycle. Its vertices are typically referred to as nodes, especially when there is another graph in the current context. A forest is a disjoint union of trees. A rooted tree is a tuple (T, r), where T is a tree and $r \in V(T)$ is a chosen root vertex. In a rooted tree T with root r, we say that $t \in V(T)$ is an ancestor of $t' \in V(T)$ (equivalently t' is a descendant of t) if t lies on the (unique) path from r to t'. We denote the subtree of T rooted at $t \in V(T)$ by T_t. A rooted tree is binary if each node has at most two children. The depth of a rooted tree is measured by the number of edges on a longest root-to-leaf path.

Perfect graphs. A graph G admits a proper q-coloring, if there exists a function $c\colon V(G) \to [q]$ such that $c(u) \neq c(v)$ for all $uv \in E(G)$. The chromatic number of a graph denotes the smallest q for which it admits a proper q-coloring. A graph G is perfect if for every induced subgraph H of G, the chromatic number of H is equal to the size of a largest clique in H. Equivalently, a graph is perfect if it has no induced cycle of odd length at least five or its edge complement [43]. Various NP-hard problems, such as computing a largest clique or largest independent set, turn out to be polynomial time solvable in perfect graphs [94]. Bipartite, chordal, and interval graphs are subclasses of perfect graphs. A graph is bipartite if and only if it admits a proper 2-coloring, or equivalently, a partition into two independent sets. Its forbidden induced subgraph characterization consists of cycles of odd length. A bipartite graph is complete if the vertices can be partitioned into two non-empty independent sets V_1 and V_2 such that $uv \in E(G)$ for every $u \in V_1$ and $v \in V_2$. The graph $K_{p,q}$ denotes the complete bipartite graph that admits a vertex partition that satisfies $|V_1| = p$ and $|V_2| = q$. The graph $K_{1,3}$ is known as the claw and more generally, $K_{1,q}$ is known as a star graph. A graph is chordal if it does not contain any induced cycle of length at least four, that is, it contains no chordless cycles. Interval graphs form a subclass of chordal graphs and are the intersection graph of a set of closed intervals on the real line: a graph G with $V(G) = [n]$ is interval if and only if there exists a collection of closed intervals $I_1, \ldots, I_n$ such that $ij \in E(G)$ if and only if I_i and I_j have a non-empty intersection.

Minor-closed classes. Minor-closed classes of graphs form a special subset of hereditary classes. A graph class $\mathcal{H}$ is minor-closed if for every $G \in \mathcal{H}$, every minor of G is also contained in $\mathcal{H}$. For a minor-closed class of graphs $\mathcal{H}$, there exists a finite set of graphs $\mathcal{F}_{\mathcal{H}}$ (see [48, Corollary 6.11]), such that $G \in \mathcal{H}$ if and only if G has no minor isomorphic to a graph in $\mathcal{F}_{\mathcal{H}}$. Arguably the most well-known minor-closed class of graphs are the planar graphs, which are the graphs that can be embedded in the plane without edge intersections (except at vertices). A graph is planar if and only if it has no K_5 or $K_{3,3}$ as a minor. While many NP-hard problems such as Independent Set remain NP-hard in these graph classes, often faster algorithms [58] or better approximation algorithms [93] are attainable.

2.2.2 Graph decompositions and parameters

This subsection introduces two well-known graph parameters. Our interest in these parameters stems from the fact that in graphs where these parameters are bounded, many computationally hard problems become tractable (see Section 2.3). Roughly speaking the first one, treedepth, measures how far the graph is from being a star. It corresponds to the minimum number of rounds needed to delete the entire graph, where in each round a vertex can be deleted from each connected component. Different definitions exist in the literature that are equivalent to treedepth, for instance the vertex ranking number [22].

Definition 2.3 ([148])**.** The treedepth of a graph G, denoted $\mathbf{td}(G)$, is defined recursively as follows.

$$\mathbf{td}(G) = \begin{cases} 0 & \text{if } |V(G)| = 0 \\ 1 + \min_{v \in V(G)}(\mathbf{td}(G - v)) & \text{if } G \text{ is connected and } |V(G)| > 1 \\ \max_{i=1}^{d}(\mathbf{td}(G_i)) & \text{if } G \text{ is disconnected and} \\ & G_1, \ldots, G_d \text{ are its components} \end{cases}$$

Where treedepth measured how far the graph is from being a star, the next parameter treewidth measures how far the graph is from being a tree. As was the case with treedepth, the concept of treewidth was rediscovered several times under different names (see the bibliographic notes at [48, §7]).

Definition 2.4 ([161], cf. [18])**.** A tree decomposition of a graph G is a pair (T, χ) where T is a tree, and $\chi \colon V(T) \to 2^{V(G)}$, such that:

1. For each $v \in V(G)$ the nodes $\{t \in V(T) \mid v \in \chi(t)\}$ form a non-empty connected subtree of T.

2. For each edge $uv \in E(G)$ there is a node $t \in V(T)$ with $\{u, v\} \subseteq \chi(t)$.

The *width* of (T, χ) is defined as $\max_{t \in V(T)} |\chi(t)| - 1$. The *treewidth* of a graph G, denoted by $\mathbf{tw}(G)$, is the minimum possible width of a tree decomposition of G.

The treewidth of the null graph is -1. The sets $\chi(t)$ are sometimes referred to as *bags* of the decomposition. The treewidth of a graph is at most its treedepth, that is, $\mathbf{tw}(G) \leq \mathbf{td}(G)$ for any graph G (see [48, Exercise 7.54]). One important observation about tree decompositions is that for every clique there always is a bag that contains it completely.

Observation 2.5 (see [48, Exercise 7.6])**.** *Let (T, χ) be a tree decomposition of G and suppose $A \subseteq V(G)$ forms a clique. Then there exists a node $t \in V(T)$ such that $A \subseteq \chi(t)$.*

Recall that for a rooted tree T, the subtree of T rooted at $t \in V(T)$ is denoted T_t. The vertices represented in this subtree are denoted by $\chi(T_t) = \bigcup_{t \in V(T_t)} \chi(t)$. The bags of a tree decomposition correspond to separators in the graph. Even more, the intersection of two bags of adjacent nodes also form a separator. This is formalized by the following lemma.

Lemma 2.6. *[48, Lemma 7.3] Let (T, χ) be a tree decomposition of a graph G. For an edge $e = \{t_1, t_2\} \in E(T)$, let T^{t_1}, T^{t_2} denote the subtrees of $T - e$ containing t_1, t_2 respectively. Then $\chi(t_1) \cap \chi(t_2)$ is an unrestricted $(\chi(T^{t_1}), \chi(T^{t_2}))$-separator.*

A lot of the material of Part II is about extensions to treedepth and treewidth. Since these extensions have only recently been introduced, their definitions and background are postponed to Chapter 3.

2.3 Computational complexity

We assume the reader is familiar with classical complexity theory, in particular with asymptotic notation ($\mathcal{O}$, Ω) and the complexity classes P and NP. We introduce the most important notions from (parameterized) complexity that we need. More background on parameterized complexity can be found in various textbooks [48, 59, 74].

Parameterized Complexity. A parameterized problem L is a subset $\Sigma^* \times \mathbb{N}$ for some finite alphabet Σ. Such a problem is called *fixed-parameter tractable*

(FPT) if there exists an algorithm $\mathcal{A}$ that given $(x, k) \in \Sigma^* \times \mathbb{N}$ decides whether $(x, k) \in L$ in time $f(k) \cdot |x|^{\mathcal{O}(1)}$ for some computable function $f\colon \mathbb{N} \to \mathbb{N}$. The complexity class containing all fixed-parameter tractable problems is called FPT. As is often done in parameterized complexity literature, we may sometimes refer to (the running time of) an algorithm that runs in time $f(k) \cdot |x|^{\mathcal{O}(1)}$ as being FPT. Instances (x, k) of L with $(x, k) \in L$ are often said to be yes-instances, while no-instances satisfy $(x, k) \notin L$.

There exist a hierarchy of complexity classes 'above' FPT, known as the W-hierarchy, that consists of classes W[t] for $t \geq 1$. For our purposes, the classes W[1] and W[2] are relevant as well as the containment relation FPT $\subseteq$ W[1] $\subseteq$ W[2]. As with $\mathsf{P} \subseteq \mathsf{NP}$, the subset relations are believed to be strict. Parameterized problems that are hard for W[1] (or W[2]) are believed not to admit FPT algorithms. This hardness notion is based on parameterized reductions that transform one parameterized problem into another in FPT time (as opposed to polynomial time reductions in classical complexity). More formally (see [48, §13.1]) for two parameterized problems A and B, a parameterized reduction from A to B takes an instance (x, k) of A, runs in time $f(k) \cdot |x|^{\mathcal{O}(1)}$ for some computable function f, outputs an instance (x', k') of B with $k' \leq g(k)$ for some computable function g, such that $(x, k) \in A$ if and only if $(x', k') \in B$. If such a reduction exists and B is FPT, then A is also FPT. The class W[1] corresponds to all parameterized problems for which there is a parameterized reduction from the k-Clique problem parameterized by k, which asks whether the graph has a clique of size k. Similarly, the class W[2] corresponds to all parameterized problems that reduce from k-Dominating Set parameterized by k, which asks whether the graph has k vertices whose closed neighborhood is the entire vertex set.

Uniformity. The notion of FPT as mentioned above is more precisely known as strongly uniform FPT [59, Definition 2.4]. We will not be referring to it as such, but the distinction is relevant when we talk about non-uniform algorithms. A parameterized problem L is called *non-uniform fixed-parameter tractable* if there exists a constant c, a function $f\colon \mathbb{N} \to \mathbb{N}$, and a collection of algorithms $\{\mathcal{A}_k \mid k \in \mathbb{N}\}$ such that for each $k \in \mathbb{N}$, $\mathcal{A}_k$ correctly decides for a given instance (x, k) whether $(x, k) \in L$ in time $f(k) \cdot |x|^c$. This non-uniform notion can be seen as a first step towards classifying whether a problem is FPT or not. Instead of having to design a single algorithm, there is more freedom by allowing a different algorithm for every value of k. One way to obtain non-uniform algorithms is through the work on graph minors of Robertson and Seymour (see [48, §6.3] for an overview). Testing whether H is a minor

of G can be done in time $f(H) \cdot |V(G)|^3$, while every minor-closed class of graphs has a finite set of forbidden minors as mentioned before. For instance graphs of treewidth at most k are minor-closed and therefore for each k there is an unknown finite number of forbidden minors $\mathcal{F}_k$. This gives a non-uniform algorithm to determine the treewidth of a graph: a different algorithm for each k is obtained by hard coding $\mathcal{F}_k$ in its description and simply testing whether the graph has one of the forbidden minors or not. Follow up work can then focus on obtaining uniform algorithms by computing the forbidden minors, or as is often the case, by coming up with completely different algorithms altogether. In Chapter 4 we give non-uniform algorithms.

Exponential Time Hypothesis. For parameterized problems that are FPT, one may wish to provide lower bounds to show that the algorithm is as fast as it can be in some sense. For this purpose, the Exponential Time Hypothesis [104] (ETH) can be used. A nice overview of their definition and consequences is given in [48, §14.1]. We briefly repeat some parts here. Let δ_q be the infimum of the set of constants c for which there exists an algorithm solving q-SAT (deciding if there is a satisfying assignment for a q-CNF formula with n variables) in time $\mathcal{O}(2^{cn})$. Then the ETH states that $\delta_3 > 0$. The ETH implies that 3-SAT cannot be solved in $2^{o(n)}$ time, that is, in time subexponential in the number of variables. Furthermore it implies FPT $\neq$ W[1] [48, Thm. 14.21]. For certain parameterized problems with parameter k, ETH can be used to rule out algorithms with running time $2^{o(f(k))} \cdot n^{\mathcal{O}(1)}$. An algorithm for such a parameterized problem with running time $2^{\mathcal{O}(f(k))} \cdot n^{\mathcal{O}(1)}$ is said to be tight for ETH, if assuming ETH no algorithm can exist with running time of the form $2^{o(f(k))} \cdot n^{\mathcal{O}(1)}$.

Memory requirements. The space usage of an (FPT) algorithm refers to the amount of memory that is needed during its execution. While we mostly avoid implementation details, for us the following distinction is relevant. An FPT algorithm for a parameterized problem with parameter k may use space exponential in k, or use space proportional to a polynomial of the input size. In other words, the space usage is either bounded by $f(k) \cdot n^{\mathcal{O}(1)}$ for some (computable) function f or by $n^{\mathcal{O}(1)}$. An algorithm of the latter type is said to be polynomial-space. Typically, dynamic programming algorithms over a tree decomposition of width k use space exponential in k. For enumeration problems, whose task it is to output a number of say vertex sets satisfying certain properties, it may be the case that the total output size is not bounded by $n^{\mathcal{O}(1)}$. For such problems it may still be possible to obtain polynomial-space

algorithms that do not construct the complete output at once, for instance by outputting vertex sets one by one and removing them from storage.

Monadic Second-Order Logic. Monadic second-order logic (MSO) is a logic of graphs that can be used to express graph properties. We follow notation as in [48, Section 7.4.1]. Variables can be used for vertices, edges, sets of vertices, and sets of edges.[1] The syntax consists of Boolean operators such as $\neg, \vee, \wedge, \Rightarrow, \Leftrightarrow$ together with universal ($\forall$) and existential ($\exists$) quantifiers. Finally there are a couple of binary operators for set containment ($\in$), vertex-edge incidence ($\mathbf{inc}(v, e)$), equality ($=$). Using these operators we can express other useful constructs like subsets ($\subseteq$) and adjacency between vertices ($\mathbf{adj}(u, v)$).

The main reason for using this logic is for the use of Courcelle's theorem, which roughly says that for graphs of bounded treewidth we can evaluate MSO formulas of constant length in linear time. For an MSO formula ϕ, we denote the length of the encoding of ϕ as a string by $||\phi||$.

Theorem 2.7 (Courcelle's theorem, see [48, Theorem 7.11])**.** *Let ϕ be an MSO formula and G be a graph. Suppose a tree decomposition of G of width t is provided. Then there exists an algorithm that verifies whether ϕ is satisfied in G in time $f(||\phi||, t) \cdot n$, for some computable function f.*

Since an optimal tree decomposition for graphs of treewidth t can be computed in time $2^{\mathcal{O}(t^3)} \cdot n$ [17], for the purpose of classification we can even drop the assumption of a given tree decomposition from the theorem.

Counting monadic second-order logic (CMSO) is the extension of MSO where for every two constants q and r we can test if the size of a set S is equal to r modulo q. Courcelle's theorem also extends to this version of the logic (see for instance the discussion towards the end of [82, §14.5.2]). While we use black-box theorems in Chapter 4 that are stated in terms of this extended logic, we do not strictly need this extension. The formulas we construct are MSO formulas.

2.4 Graph problems

We mostly focus on the $\mathcal{H}$-Deletion problem, which is defined for each fixed graph class $\mathcal{H}$. Given a graph G, it asks to find a minimum cardinality vertex set $X \subseteq V(G)$ such that $G - X \in \mathcal{H}$. Such a vertex set X is called an

[1] In this thesis we use MSO_2, the MSO_1 variant does not allow for quantification over sets of edges.

$\mathcal{H}$-modulator or $\mathcal{H}$-deletion set. The cardinality of such a minimum set is sometimes referred to as the distance to $\mathcal{H}$. Note that if $\mathcal{H}$ is hereditary a solution always exists since $V(G)$ is a solution. The classical result of Lewis and Yannakakis [127] shows that $\mathcal{H}$-Deletion is NP-hard for any nontrivial hereditary graph class $\mathcal{H}$. Here a graph class is nontrivial, if it contains infinitely many graphs and furthermore excludes infinitely many graphs. For instantiations of $\mathcal{H}$ we drop the hyphen, for instance Chordal Deletion corresponds to $\mathcal{H}$-Deletion where $\mathcal{H}$ is the class of chordal graphs. We focus our attention to the graph classes introduced in Section 2.2, most notably bipartite, chordal, and interval graphs as well as graphs with a finite set of forbidden induced subgraphs or forbidden minors. Certain variants of $\mathcal{H}$-Deletion are better known under different names. To mention a few, deleting to the class of edgeless graphs is known as Vertex Cover (or Node Cover), to bipartite graphs as Odd Cycle Transversal, and to a forest as Feedback Vertex Set.

When parameterized by the solution size k, a few different formulations exist in the literature. We consider the one where we are given a graph G and an integer k, and the task is to either find a minimum-size $\mathcal{H}$-deletion set or report that no such set of size at most k exists. As already alluded to in Chapter 1, the $\mathcal{H}$-Deletion problem is FPT for many hereditary graph classes $\mathcal{H}$ [35, 37, 79, 142, 159].

In the literature two other formulations of the parameterized version of $\mathcal{H}$-Deletion are common. The *decision* variant asks whether a solution of size at most k exists. Another variant, the *non-minimum* one, asks for a solution of size at most k or to report that no such solution exists, which is equivalent to ours apart from the requirement of returning a minimum solution. An FPT algorithm for $\mathcal{H}$-Deletion that uses one of these two formulation implies an FPT algorithm for our formulation by self-reduction.

Lemma 2.8. *Given an algorithm for the non-minimum variant that runs in $f(k,n)$ time, we can solve $\mathcal{H}$-Deletion in $\mathcal{O}(k \cdot f(k,n))$ time. Similarly, given an algorithm for the decision variant that runs in $f(k,n)$ time, we can solve $\mathcal{H}$-Deletion in $(k + n^{\mathcal{O}(1)}) \cdot f(k,n)$ time.*

Proof. By running the algorithm for the non-minimum variant for each value k' from 0 up to k, we can find the smallest value for which there is a solution and return it. If for all these values no solution is found, then clearly there is no solution of size at most k which can be reported. For the case we are given an algorithm for the decision variant, we can do the same procedure to determine the size k' of a minimum solution. Then, a vertex v belongs to an optimal solution precisely when the instance obtained when removing v has a solution

of size $k' - 1$. If so, we continue with the smaller graph and recursively look for a solution of size $k' - 1$ in the smaller graph. If not, then we continue with the next vertex. This way we can build a minimum solution by calling the algorithm $\mathcal{O}(n)$ times. The time needed to update the graph before each call depends on how it is stored in memory. As the polynomial factor of the running time of our results is typically not optimized, we bound it simply by $n^{\mathcal{O}(1)}$. □

Most of our hardness reductions start from either Dominating Set (which was mentioned already in Section 2.3) or Hitting Set. In the decision variant of Dominating Set, we are given a graph G and an integer k, and the task is to decide if there is a subset $S \subseteq V(G)$ of size at most k such that $N_G[S] = V(G)$, that is, each vertex not in S has a neighbor in S. In the decision variant of the Hitting Set problem, we are given a universe U, a collection $\mathcal{F}$ of subsets of U, and an integer k. The task is to decide if there is a subset $S \subseteq U$ of size at most k such that $S \cap F \neq \emptyset$ for each $F \in \mathcal{F}$. Like Dominating Set, Hitting Set parameterized by k is W[2]-hard [48, Thm. 13.25]. While it is not really a graph problem, one can interpret the problem as finding a vertex set that intersects all hyperedges in the hypergraph with vertex set U and hyperedges $\mathcal{F}$.

Part II

Hybrid Parameters

Chapter 3

Background on Hybrid Parameters

This chapter contains a concise overview of some background, definitions, and preliminaries that forms the common thread of Part II regarding hybrid parameters. See [6] for a recent survey.

3.1 Introduction

The field of parameterized algorithmics [48, 60] develops fixed-parameter tractable (FPT) algorithms to solve NP-hard problems exactly, which are provably efficient on inputs whose parameter value is small. A recent line of work aims to unify two lines of research in parameterized algorithms for vertex-deletion problems that were previously mostly disjoint. On the one hand, there are algorithms that work on a structural decomposition of the graph, whose running time scales exponentially with a graph-complexity measure but polynomially with the size of the graph. Examples of such algorithms include dynamic programming over a tree decomposition [21, 25] (which forms a recursive decomposition by small separators), dynamic-programming algorithms based on cliquewidth [46], rankwidth [102, 151, 152], and Booleanwidth [31] (which are recursive decompositions of a graph by simply structured although not necessarily small separations). The second line of algorithms are those that work with the "solution size" as the parameter, whose running time scales exponentially with the solution size. Such algorithms take advantage of the properties of inputs that admit small solutions. Examples of the latter

include the celebrated *iterative compression* algorithm to find a minimum odd cycle transversal [159] and algorithms to compute a minimum vertex-deletion set whose removal makes the graph chordal [38, 142], interval [37], or planar [112, 119, 144].

In our study of hybrid parameterizations in Part II, our aim is to combine the best of both these lines of research in order to obtain fixed-parameter tractable algorithms for parameterizations which can simultaneously be smaller than natural parameterizations by solution size and width measures like treewidth. To achieve this, we use parameters that are a hybrid of these two that have been recently introduced in the literature. They correspond to relaxed versions of treedepth (Definition 2.3) and treewidth (Definition 2.4). We give a high-level description of the hybrid parameters we employ, formal definitions are postponed to Section 3.2. Each hybrid parameter is targeted at a specific graph class $\mathcal{H}$. For the purpose of this overview, one should think of $\mathcal{H}$ as a hereditary and union-closed graph class.

The first type of parameter we employ is the $\mathcal{H}$-elimination distance introduced by Bulian and Dawar [32, 33], which admits a recursive definition similar to treedepth. It corresponds to the minimum number of rounds needed to get to a graph in $\mathcal{H}$, where in each round a vertex of each connected component can be deleted. The fact that it is upper bounded by treedepth follows from the fact that treedepth is equal to the number of rounds to delete *all* vertices this way. The process of eliminating vertices explains the name *elimination distance*. The elimination process can be represented by a tree structure called an $\mathcal{H}$-elimination forest, whose depth corresponds to $\mathbf{ed}_{\mathcal{H}}(G)$. Intuitively, as in standard elimination forests for treedepth (see [148]), each layer of the tree corresponds to the vertices that are deleted in that round and any two endpoints of an edge are have an ancestor-descendant relationship. The leaves of the tree however can be arbitrarily large parts of the graph that induce a graph belonging to $\mathcal{H}$. These large parts are called *base components*. These decompositions can be used to obtain polynomial-space algorithms, similarly as for treedepth [13, 85, 155].

The second type of parameter we employ is called $\mathcal{H}$-treewidth, which denotes the minimum width of a *tree $\mathcal{H}$-decomposition* of the graph. These decompositions are obtained by relaxing the definition of treewidth and were recently introduced by Eiben et al. [62], building on similar hybrid parameterizations used in the context of solving SAT [86] and CSPs [87]. The definition was originally stated in terms of torsos. The torso of a vertex set X is the graph obtained by turning the neighborhood of every connected component of $G - X$ into a clique, followed by deleting all of $V(G) \setminus X$. The $\mathcal{H}$-treewidth then is

the minimum treewidth of the torso of an $\mathcal{H}$-modulator X of G.[1] This also gives another way to look at $\mathcal{H}$-elimination distance, namely as the minimum treedepth of the torso of an $\mathcal{H}$-modulator of G (see Lemma 3.5). We can extend the definition of a tree decomposition to incorporate the components outside the torso that belong to $\mathcal{H}$. A tree $\mathcal{H}$-decomposition of a graph G is a tree decomposition of G, together with a set $L \subseteq V(G)$ of base vertices (a formal definition is postponed to Definition 3.6). Base vertices are not allowed to occur in more than one bag, and the base vertices in a bag must induce a subgraph belonging to $\mathcal{H}$. The connected components of $G[L]$ are called the base components of the decomposition. The width of such a decomposition is defined as the maximum number of non-base vertices in any bag, minus one. A tree $\mathcal{H}$-decomposition therefore represents a decomposition of a graph by small separators, into subgraphs which are either small or belong to $\mathcal{H}$. The minimum width of such a decomposition for G is the $\mathcal{H}$-treewidth of G, denoted $\mathbf{tw}_{\mathcal{H}}(G)$. We remark that this parameter is mainly interesting for the case where graphs from $\mathcal{H}$ can have unbounded treewidth, for instance $\mathcal{H} \in \{\mathsf{bipartite}, \mathsf{chordal}, \mathsf{planar}\}$, as otherwise $\mathbf{tw}_{\mathcal{H}}(G)$ is comparable with $\mathbf{tw}(G)$ (see Lemma 3.11). Therefore we do not study classes such as trees or outerplanar graphs.

We illustrate the key ideas of the graph decomposition for the case of Odd Cycle Transversal (OCT), which is the vertex-deletion problem which aims to obtain a bipartite graph. Here we care about instantiations of the decompositions defined above for $\mathcal{H}$ being the class bip of bipartite graphs. Observe that if G has an odd cycle transversal of k vertices, then $\mathbf{ed}_{\mathsf{bip}}(G) \leq k$. To see this, observe that whenever a graph G has an odd cycle transversal X of size k, then in each round we can eliminate (at least) one vertex from X and reach the bipartite graph $G - X$ after at most $|X| = k$ rounds. In the other direction, as illustrated in Figure 3.1, the value $\mathbf{ed}_{\mathsf{bip}}(G)$ may be arbitrarily much smaller than the size of a minimum OCT. At the same time, the value of $\mathbf{ed}_{\mathsf{bip}}(G)$ may be arbitrarily much smaller than the rankwidth (and hence treewidth, cliquewidth, and treedepth) of G, since the $n \times n$ grid graph is bipartite but has rankwidth $n-1$ [116]. Hence a time bound of $f(\mathbf{ed}_{\mathsf{bip}}(G)) \cdot n^{\mathcal{O}(1)}$ can be arbitrarily much better than bounds which are fixed-parameter tractable with respect to the size of an optimal odd cycle transversal or with respect to pure width measures of G. Working with $\mathbf{ed}_{\mathsf{bip}}$ as the parameter for Odd Cycle Transversal therefore facilitates algorithms which simultaneously improve on the solution-size [159] and width-based [132] algorithms for OCT.

[1] A small technicality, each connected component of $G - X$ should belong to $\mathcal{H}$. If $\mathcal{H}$ is union-closed this is equivalent.

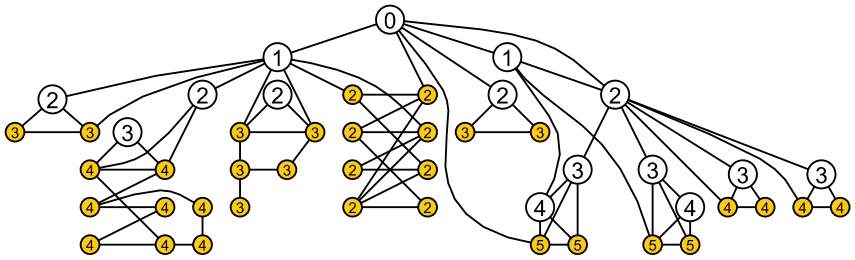


Figure 3.1: The vertex labels correspond to depth values in a bip-elimination forest of depth 5. By attaching triangles to vertices of depth at most three, the minimum size of an odd cycle transversal can increase boundlessly without increasing $\mathbf{ed}_{\mathsf{bip}}$. The shaded vertices are the base vertices of the decomposition and induce a bipartite graph.

Early work in this area focused on whether computing these parameter values exactly is FPT. For minor-closed graph classes $\mathcal{H}$ it can be shown that graphs of $\mathcal{H}$-elimination distance at most k are minor-closed and therefore characterized by a finite set of forbidden minors. This leads to non-uniform algorithms to recognize graphs of $\mathcal{H}$-elimination distance at most k for minor-closed $\mathcal{H}$ using the Graph Minor algorithm [162]. Bulian and Dawar [33] show that these forbidden minors can be constructed, resulting in a uniform algorithm with unknown parameter dependence. Morelle et al. [145] give an algorithm with explicit parameter dependence. Apart from minor-closed families $\mathcal{H}$, some isolated results are known about FPT algorithms to compute $\mathbf{ed}_{\mathcal{H}}$ and $\mathbf{tw}_{\mathcal{H}}$ exactly, parameterized by the parameter value. In recent work, Agrawal and Ramanujan [5] give an FPT algorithm to compute the elimination distance to a cluster graph, as part of a kernelization result using the corresponding structural parameterization. Eiben et al. [62] show that when $\mathcal{H}$ is the class of graphs of rankwidth at most c for some constant c, then computing $\mathbf{tw}_{\mathcal{H}}$ is FPT. Bulian and Dawar [32] considered the elimination distance to graphs of bounded degree d and gave an FPT approximation algorithm. Lindermayr et al. [129] showed that the elimination distance of a *planar* graph to a bounded-degree graph can be computed in FPT time. Agrawal et al. [4] obtained non-uniform FPT algorithms for computing the elimination distance to any family $\mathcal{H}$ defined by a finite number of forbidden induced subgraphs. Fomin et al. [76] extend this to properties expressible by a first order-logic formula, which includes graph

classes with a finite forbidden subgraph characterization. In Chapter 4 we show that computing the $\mathcal{H}$-elimination distance and $\mathcal{H}$-treewidth for $\mathcal{H}$ being the class of bipartite graphs is non-uniformly FPT. By now the complexity is mostly settled, at least in the non-uniform setting. Agrawal et al. [3] showed that whenever $\mathcal{H}$ is definable in counting monadic second order logic (CMSO) and union-closed, then computing $\mathcal{H}$-treewidth and $\mathcal{H}$-elimination distance is non-uniform FPT whenever the solution size parameterization of $\mathcal{H}$-Deletion is non-uniform FPT.

The significance of hybrid graph measures stems from their use as parameters, in our case for the $\mathcal{H}$-Deletion problem. It turns out that in many cases, FPT algorithms for $\mathcal{H}$-Deletion parameterized by solution size and treewidth/treedepth can be extended to obtain FPT algorithms parameterized by $\mathcal{H}$-treewidth or $\mathcal{H}$-elimination distance. This generalization significantly extends the tractability horizon for $\mathcal{H}$-Deletion in terms of which types of instances can be guaranteed to be solved efficiently, since for classes $\mathcal{H}$ of unbounded treewidth these measures can be much lower than both the solution size and the treewidth of an input graph. The work of Agrawal et al. [3] mentioned above also has far-reaching consequences in this paradigm. They show that when $\mathcal{H}$ is a graph class which is (1) hereditary and union-closed, (2) expressible in CMSO, and (3) admits a (non-uniform) FPT algorithm for $\mathcal{H}$-Deletion parameterized by the solution size, then there is a non-uniform FPT algorithm to find a minimum $\mathcal{H}$-modulator parameterized by $\mathbf{tw}_{\mathcal{H}}$ and $\mathbf{ed}_{\mathcal{H}}$. The non-uniformity of this method has roots in the technique of recursive understanding, which relies on non-constructive arguments. This meta-theorem shows that at least with respect to the existence of non-uniform FPT algorithms, for the broad class of CSMO-expressible classes $\mathcal{H}$ (for which $\mathcal{H}$-Deletion parameterized by treewidth is always FPT via Courcelle's theorem), being able to solve $\mathcal{H}$-Deletion parameterized by solution size automatically implies tractability of the problem parameterized by $\mathbf{tw}_{\mathcal{H}}$. While this meta-theorem settles the complexity almost completely in the non-uniform setting, obtaining uniform algorithms with decent parameter dependencies remains of interest. We obtain such algorithms through the results of Chapters 5 and 6. In Chapter 5 we give an algorithm that, for hereditary and union-closed graph classes $\mathcal{H}$ for which $\mathcal{H}$-Deletion parameterized by solution size is FPT, outputs a tree $\mathcal{H}$-decomposition of width at most $8k + 8$ where $k = \mathbf{tw}_{\mathcal{H}}(G)$ in time that is FPT in k. In Chapter 6 we show that, if additionally a version of $\mathcal{H}$-Deletion with undeletable vertices is solvable, then we can use this tree $\mathcal{H}$-decomposition to solve $\mathcal{H}$-Deletion. For instance, we obtain algorithms for Planar Deletion and OCT parameterized by $\mathbf{tw}_{\mathcal{H}}$ that are tight for ETH. These algorithms show that for certain problems, the dependence on hybrid

parameters can be as good as their natural parameterization.

Related work. While the two hybrid parameters above have gained in popularity in the last few years, they are not the only ones that have a hybrid flavor. One parameter worth mentioning that is perhaps closest in spirit is one introduced by Eiben et al [64] called the *well-structure number*. For a specific graph class $\mathcal{H}$, it denotes the smallest number k of vertex sets in an $\mathcal{H}$-modulator $\mathcal{X} = \bigcup_{i \in [k]} X_i$ such that (1) each vertex set X_i induces a graph with rank-width at most k and (2) each vertex set X_i is a so-called split-module, roughly speaking a subset of X_i forms a complete bipartite graph with a subset of $V(G) \setminus \mathcal{X}$. They showed that computing a large clique or small vertex cover is FPT with this parameter if and only if these problems can be solved in polynomial time in $\mathcal{H}$. This parameter is both upper bounded by the minimum size of an $\mathcal{H}$-modulator of G and by the rank-width of G [64, Proposition 1]. However, bounded $\mathcal{H}$-elimination distance does not directly imply bounded well-structure number. Consider a graph with an $\mathcal{H}$-modulator X such that X induces an edgeless graph and each connected component of $G - X$ is adjacent to a single vertex of X. In this case, the $\mathcal{H}$-elimination distance is 1, while the well-structure number scales with $|X|$. A variant of the parameter, where each of the k sets X_i induces a graph of constant rank-width, has also been used to study kernelization [63].

Organization. In the remainder of this chapter we introduce some formal definitions and preliminaries that are relevant throughout Part II. The remainder of this part is organized as follows. Chapter 4 contains classification results regarding the computation of hybrid graph parameters, both positive results as well as hardness results. In Chapter 5 we show how to compute tree $\mathcal{H}$-decompositions whose width is a constant factor times $\mathbf{tw}_{\mathcal{H}}$. In Chapter 6 we show how to solve $\mathcal{H}$-Deletion on these decompositions, extending standard techniques regarding dynamic programming over tree decompositions.

3.2 Preliminaries for hybrid parameters

Definition 3.1 ([33])**.** For a hereditary graph class $\mathcal{H}$ and a graph G, the $\mathcal{H}$-elimination distance of G, denoted $\mathbf{ed}_{\mathcal{H}}(G)$, is defined recursively as follows.

$$\mathbf{ed}_{\mathcal{H}}(G) = \begin{cases} 0 & \text{if } G \text{ is connected and } G \in \mathcal{H} \\ 1 + \min_{v \in V(G)}(\mathbf{ed}_{\mathcal{H}}(G - v)) & \text{if } G \text{ is connected and } G \notin \mathcal{H} \\ \max_{i=1}^{d}(\mathbf{ed}_{\mathcal{H}}(G_i)) & \text{if } G \text{ is disconnected and} \\ & G_1, \ldots, G_d \text{ are its components} \end{cases}$$

Note 3.2. *Note that the hereditary requirement makes sure that it is well-defined, as any hereditary class of graphs contains the null graph. Strictly speaking we could replace the hereditary condition by containment of the null graph, but since all our applications consider hereditary graph classes we do not take this route. Whenever we talk about $\mathcal{H}$-elimination distance we simply assume it to be well-defined.*

A tree structure which encodes this recursion (with non-necessarily optimal depth) is called an $\mathcal{H}$-elimination forest of G, see Figure 3.1 for an example for the class of bipartite graphs.

Definition 3.3. For a graph class $\mathcal{H}$, an $\mathcal{H}$-elimination forest of graph G is a pair (T, χ) where T is a rooted forest and $\chi\colon V(T) \to 2^{V(G)}$, such that:

1. For each internal node t of T we have $|\chi(t)| = 1$.
2. The sets $(\chi(t))_{t \in V(T)}$ form a partition of $V(G)$.
3. For each edge $uv \in E(G)$, if $u \in \chi(t_1)$ and $v \in \chi(t_2)$ then t_1, t_2 are in ancestor-descendant relation in T.
4. For each leaf t of T, the graph $G[\chi(t)]$, called a base component, belongs to $\mathcal{H}$.

The *depth* of T is the maximum number of edges on a root-to-leaf path. We refer to the union of base components as the set of base vertices.

A pair (T, χ) is a (standard) elimination forest if it satisfies the above for $\mathcal{H}$ consisting only of the 0-vertex graph, that is, the base components are empty.

It is straight-forward to verify that for any G and hereditary $\mathcal{H}$, the minimum depth of an $\mathcal{H}$-elimination forest of G is equal to the $\mathcal{H}$-elimination distance as defined recursively above. This is the reason the depth of an $\mathcal{H}$-elimination forest is defined in terms of the number of edges, while the traditional definition

of treedepth counts vertices on root-to-leaf paths. Similarly, the treedepth of G is the minimum depth of a standard elimination forest. While the definition above is convenient for coming up with algorithms parameterized by $\mathcal{H}$-elimination distance and is therefore worth mentioning, in this thesis we focus on algorithms parameterized by $\mathcal{H}$-treewidth and only cover some classification results for $\mathcal{H}$-elimination distance. For these classification results it is more convenient to work with the torso-based definition that is already mentioned in Section 3.1.

Definition 3.4. [87, Definition 4] Let G be a graph and $X \subseteq V(G)$. The *torso of* X, denoted by $\mathbf{T}_G(X)$, is the graph obtained by turning the neighborhood of every connected component of $G - X$ into a clique, followed by deleting all of $V(G) \setminus X$.

We may drop the subscript G if the graph is clear from context. We show the equivalence of $\mathbf{ed}_{\mathcal{H}}$ and the minimum treedepth of the torso of a vertex set whose removal results in connected components belonging to $\mathcal{H}$.

Lemma 3.5. *Let $\mathcal{H}$ be a hereditary graph class and k a non-negative integer. The $\mathcal{H}$-elimination distance of G is at most k if and only if there exists $X \subseteq V(G)$ with $\boldsymbol{td}(\boldsymbol{T}_G(X)) \leq k$ such that each connected component of $G - X$ belongs to $\mathcal{H}$.*

Proof. For the first direction, we prove by induction on k that if G has a set $X \subseteq V(G)$ such that $C \in \mathcal{H}$ for each connected component C of $G - X$ and $\mathbf{td}(\mathbf{T}_G(X)) \leq k$, then $\mathbf{ed}_{\mathcal{H}}(G) \leq k$.

For the base case $k = 0$, note that $\mathbf{td}(\mathbf{T}_G(X)) = 0$ implies that $X = \emptyset$, so that each connected component of $G - X$ belongs to $\mathcal{H}$. By Definition 3.1 we have $\mathbf{ed}_{\mathcal{H}}(C) = 0$ for each connected component C of G and therefore $\mathbf{ed}_{\mathcal{H}}(G) = 0 \leq k$.

For the induction step we have $k > 0$. To show that $\mathbf{ed}_{\mathcal{H}}(G) \leq k$, by Definition 3.1 it suffices to prove that each $C \in \mathrm{cc}(G)$ satisfies $\mathbf{ed}_{\mathcal{H}}(C) \leq k$. Let $X_C := C \cap X$. If $X_C = \emptyset$ then $C \in \mathcal{H}$ (since C is a connected component of $G - X$) so $\mathbf{ed}_{\mathcal{H}}(C) = 0 \leq k$. In the remainder assume that $X_C \neq \emptyset$. Observe that $\mathbf{T}_G(X)$ has $\mathbf{T}_C(X_C)$ as a connected component, and that $\mathbf{T}_C(X_C)$ is connected since C is connected and $X_C \subseteq C$. By definition of $\mathbf{td}$ there exists a vertex $x \in X_C$ such that $\mathbf{td}(\mathbf{T}_C(X_C) - x) = \mathbf{td}(\mathbf{T}_C(X_C)) - 1$. Let $X'_C := X_C \setminus \{x\}$ and let $C' := C - x$. Note that $\mathbf{T}_{C'}(X'_C) = \mathbf{T}_C(X_C) - x$: it makes no difference whether we first turn the neighborhood of each component of $C - X_C$ into a clique, remove $C \setminus X_C$, and then remove x, or whether we start from $C' = C - x$, turn the neighborhood of each component of $C' - X'_C$ into a clique, and

then remove $C' \setminus X'_C$. By induction on C' and X'_C with $k' := \mathbf{td}(\mathbf{T}_{C'}(X'_C)) < k$, it follows that $\mathbf{ed}_{\mathcal{H}}(C') \leq \mathbf{td}(\mathbf{T}_{C'}(X'_C)) = \mathbf{td}(\mathbf{T}_C(X_C)) - 1 \leq \mathbf{td}(\mathbf{T}_G(X)) - 1$, where the last inequality follows since $\mathbf{T}_C(X_C)$ is a connected component of $\mathbf{T}_G(X)$. By Definition 3.1, since C is connected we have $\mathbf{ed}_{\mathcal{H}}(C) \leq 1 + \min_{v \in C} \mathbf{ed}_{\mathcal{H}}(C-v) \leq 1 + \mathbf{ed}_{\mathcal{H}}(C-x) \leq 1 + (\mathbf{td}(\mathbf{T}_G(X)) - 1) = \mathbf{td}(\mathbf{T}_G(X)) \leq k$, which completes this direction of the proof.

For the converse direction, we prove that if $\mathbf{ed}_{\mathcal{H}}(G) \leq k$ then G has a vertex set X such that $C \in \mathcal{H}$ for each $C \in \mathrm{cc}(G - X)$ and $\mathbf{td}(\mathbf{T}_G(X)) \leq k$. We use an induction on $k + |V(G)|$. If $\mathbf{ed}_{\mathcal{H}}(G) = 0$ then by Definition 3.1 we have $C \in \mathcal{H}$ for each $C \in \mathrm{cc}(G)$ so that $X = \emptyset$ suffices. For the induction step we have $\mathbf{ed}_{\mathcal{H}}(G) > 0$. We distinguish two cases, depending on the connectivity of G.

If G is connected, then since $\mathbf{ed}_{\mathcal{H}}(G) > 0$ we have $G \notin \mathcal{H}$. Hence by Definition 3.1 we have $\mathbf{ed}_{\mathcal{H}}(G) = 1 + \min_{v \in V(G)} \mathbf{ed}_{\mathcal{H}}(G - v)$. Let x be a vertex for which equality is attained. Since $\mathbf{ed}_{\mathcal{H}}(G-x) = \mathbf{ed}_{\mathcal{H}}(G) - 1 < k$, by induction on $G' := G - x$ there exists a set $X' \subseteq V(G')$ such that $\mathbf{td}(\mathbf{T}_{G'}(X')) \leq k - 1$ and each connected component of $G' - X'$ belongs to $\mathcal{H}$. Define $X := X' \cup \{x\}$ and note that the connected components of $G' - X'$ are the same as those of $G - X$ and therefore belong to $\mathcal{H}$. Furthermore note that $\mathbf{td}(\mathbf{T}_G(X)) \leq 1 + \mathbf{td}(\mathbf{T}_{G'}(X')) \leq 1 + (k-1)$ since the graph $\mathbf{T}_{G'}(X')$ can be obtained from $\mathbf{T}_G(X)$ by removing the vertex x. Hence $\mathbf{td}(\mathbf{T}_G(X)) \leq k$, proving the claim.

Now suppose that G is disconnected, so that $\mathbf{ed}_{\mathcal{H}}(G) = \max_{C \in \mathrm{cc}(G)} \mathbf{ed}_{\mathcal{H}}(C)$. For each $C \in \mathrm{cc}(G)$ we have that $|V(C)| < |V(G)|$ and $\mathbf{ed}_{\mathcal{H}}(C) \leq \mathbf{ed}_{\mathcal{H}}(G) = k$, so we may apply the induction hypothesis to C to obtain a set $X_C \subseteq V(C)$ such that $\mathbf{td}(\mathbf{T}_C(X_C)) \leq \mathbf{ed}_{\mathcal{H}}(C) \leq k$ and each connected component of $C - X_C$ belongs to $\mathcal{H}$. Let $X := \bigcup_{C \in \mathrm{cc}(G)} X_C$. Clearly each connected component of $G - X$ belongs to $\mathcal{H}$. Observe that each connected component of the graph $\mathbf{T}_G(X)$ is equal to $\mathbf{T}_C(X_C)$ for some $C \in \mathrm{cc}(G)$, so that each connected component H of $\mathbf{T}_G(X)$ satisfies $\mathbf{td}(H) \leq \mathbf{td}(\mathbf{T}_C(X_C)) \leq k$ for some $C \in \mathrm{cc}(G)$. By Definition 3.1, the fact that each component of $\mathbf{T}_G(X)$ has treedepth at most k ensures $\mathbf{td}(\mathbf{T}_G(X)) \leq k$, which concludes the proof. □

We switch our attention to $\mathcal{H}$-treewidth and tree $\mathcal{H}$-decompositions. The following definition captures the relaxed notion of a tree decomposition.

Definition 3.6. For a hereditary graph class $\mathcal{H}$, a tree $\mathcal{H}$-decomposition of graph G is a triple (T, χ, L) where $L \subseteq V(G)$, T is a rooted tree, and $\chi \colon V(T) \to 2^{V(G)}$, such that:

1. For each $v \in V(G)$ the nodes $\{t \in V(T) \mid v \in \chi(t)\}$ form a non-empty connected subtree of T.
2. For each edge $uv \in E(G)$ there is a node $t \in V(T)$ with $\{u, v\} \subseteq \chi(t)$.
3. For each vertex $v \in L$, there is a unique $t \in V(T)$ for which $v \in \chi(t)$, with t being a leaf of T.
4. For each node $t \in V(T)$, the graph $G[\chi(t) \cap L]$ belongs to $\mathcal{H}$.

The *width* of a tree $\mathcal{H}$-decomposition is defined as $\max_{t \in V(T)} |\chi(t) \setminus L| - 1$. The $\mathcal{H}$-treewidth of a graph G, denoted $\mathbf{tw}_{\mathcal{H}}(G)$, is the minimum width of a tree $\mathcal{H}$-decomposition of G. The connected components of $G[L]$ are called base components and the vertices in L are called base vertices.

In the definition of width, we subtract one from the size of a largest bag to mimic treewidth. Note 3.2 also applies here, but the hereditary requirement also allows Item 4 to be strengthened as follows.

Observation 3.7. *For a hereditary graph class $\mathcal{H}$ and a graph G, there exists a minimum-width tree $\mathcal{H}$-decomposition (T, χ, L) such that $G[\chi(t) \cap L]$ is connected for every $t \in V(T)$.*

The following observation allows us to translate properties of tree decompositions to tree $\mathcal{H}$-decompositions.

Observation 3.8. *If (T, χ, L) is a tree $\mathcal{H}$-decomposition of G, then (T, χ) is a regular tree decomposition of G, albeit of possibly larger width.*

Next we show that the definition is equivalent to the torso-based definition, which more closely represents the original definition of Eiben et al. [62]. We remark that in our conference version [110], the width was defined to be at least 0 so that graphs $G \in \mathcal{H}$ get $\mathbf{tw}_{\mathcal{H}}(G) = 0$. In order to be consistent with the original definition based on torso, the above definition gives $\mathbf{tw}_{\mathcal{H}}(G) = -1$.

Lemma 3.9. *Let $\mathcal{H}$ be a hereditary class of graphs and $k \geq -1$ an integer. The $\mathcal{H}$-treewidth of G is at most k if and only if there exists $X \subseteq V(G)$ with $\boldsymbol{tw}(\boldsymbol{T}_G(X)) \leq k$ such that each connected component of $G - X$ belongs to $\mathcal{H}$.*

Proof. In the forward direction let (T, χ, L) be an optimal tree $\mathcal{H}$-decomposition of width at most k. Let $X = V(G) \setminus L$. Since each vertex of L appears in a single leaf t of T, and because t must cover all outgoing edges of said vertex, it follows that each connected component of $G - X = G[L]$ appears in a single leaf bag $\chi(t)$ for some $t \in V(T)$. Since $G[\chi(t) \cap L]$ belongs to $\mathcal{H}$, and because

$\mathcal{H}$ is hereditary, it follows that each connected component of $G[\chi(t) \cap L]$ is also contained in $\mathcal{H}$. This shows that each connected component of $G - X$ is in $\mathcal{H}$. Observe that (T, χ'), where $\chi'(t) = \chi(t) \setminus L$ for each $t \in V(T)$ is a valid tree decomposition of $G[X] = G - L$ of width at most k. Let $t_C \in V(T)$ be the vertex such that $V(C) \subseteq \chi(t_C)$ for each connected component C of $G[L]$. As mentioned before, also all the neighbors of C are contained in $\chi(t_C)$ since this is the only node that can cover these edges. It follows that (T, χ') is also a valid tree decomposition of $\mathbf{T}_G(X)$ of width at most k, completing this direction of the proof.

In the other direction, let $X \subseteq V(G)$ be a vertex set such that each connected component of $G - X$ belongs to $\mathcal{H}$ and $\mathbf{tw}(\mathbf{T}_G(X)) \leq k$. We argue that $\mathbf{tw}_{\mathcal{H}}(G) \leq k$. Let (T, χ) be a tree decomposition of $\mathbf{T}_G(X)$ (in the case that $\mathbf{T}_G(X)$ is the null graph, let T be a single node r with $\chi(r) = \emptyset$). We can create a tree $\mathcal{H}$-decomposition (T', χ', L) of G as follows. Let $\chi'(t) = \chi(t)$ for each $t \in V(T)$ and let $L = V(G) \setminus X$. For each connected component C of $G[L]$, the neighborhood of C is a clique in $\mathbf{T}_G(X)$. By Observation 2.5 it follows that exists a node $t \in V(T)$ such that this neighborhood is contained in $\chi(t)$. Assign an arbitrary such node t_C to each connected component C of $G[L]$. Attach a new leaf t'_C to t_C and let $\chi'(t'_C) = N_G[V(C)]$. By construction we have that (T', χ', L) is a valid tree $\mathcal{H}$-decomposition of G, completing the proof. □

The following observation easily follows from the torso-based definitions and the fact that $\mathbf{tw}(G) \leq \mathbf{td}(G)$ (see [48, Exercise 7.54]) and the fact that at least one vertex of an $\mathcal{H}$-modulator can be eliminated per round in Definition 3.1.

Observation 3.10. *For any hereditary class of graphs $\mathcal{H}$ and graph G, we have $\boldsymbol{tw}_{\mathcal{H}}(G) \leq \boldsymbol{ed}_{\mathcal{H}}(G) \leq \boldsymbol{mod}_{\mathcal{H}}(G)$, where $\boldsymbol{mod}_{\mathcal{H}}(G)$ denotes the minimum cardinality of an $\mathcal{H}$-modulator.*

We conclude with a lemma that shows that $\mathbf{tw}(G)$ and $\mathbf{tw}_{\mathcal{H}}(G)$ are comparable if graphs in $\mathcal{H}$ have bounded treewidth, which shows that the parameter is mainly interesting for classes that can have unbounded treewidth.

Lemma 3.11. *Suppose (T, χ, L) is a tree $\mathcal{H}$-decomposition of G of width k and the maximal treewidth in $\mathcal{H}$ is d. Then the treewidth of G is at most $d + k + 1$. Moreover, if the corresponding decompositions are given, then the requested tree decomposition of G can be constructed in polynomial time.*

Proof. For a node $t \in V(T)$, the graph $G[\chi(t) \cap L]$ belongs to $\mathcal{H}$, so it admits a tree decomposition (T_t, χ_t) of width d. Consider a tree T_1 given as a disjoint union of T and $\bigcup_{t \in V(T)} T_t$ with additional edges between each t and any

node from $V(T_t)$. We define $\chi_1(t) = \chi(t) \setminus L$ for $t \in V(T)$ and $\chi_1(x) = \chi_t(x) \cup (\chi(t) \setminus L)$ for $x \in V(T_t)$. The maximum size of a bag in T_1 is at most $\max_{t \in V(T)} |\chi(t) \setminus L| + \max_{t \in V(T),\, x \in V(T_t)} |\chi_t(x)| \leq d + k + 2$.

Let us check that (T_1, χ_1) is a tree decomposition of G, starting from condition (1). If $v \in L$ then it belongs to exactly one set $\chi(t) \cap L$ and $\chi_1^{-1}(v) = \chi_t^{-1}(v)$. If $v \notin L$, then $\chi_1^{-1}(v) = \chi^{-1}(v) \cup \bigcup_{t \in \chi^{-1}(v)} V(T_t)$. In both cases these are connected subtrees of T_1.

Now we check condition (2) for $uv \in E(G)$. If $u, v \in L$, then both u, v belong to a single set $\chi(t) \cap L$ and there is a bag $\chi_t(x)$ containing u, v. If $u, v \notin L$, then both u, v appear in some bag of T and also in its counterpart in T_1. If $u \in L$, $v \notin L$, then for some $t \in V(T)$ we have $u \in \chi(t) \cap L$, $v \in \chi(t) \setminus L$ and $u \in \chi_t(x)$ for some $x \in V(T_t)$. Hence, $u, v \in \chi_1(x)$. The conditions (3,4) are not applicable to a standard tree decomposition. □

Chapter 4

FPT Classification of Hybrid Parameters

4.1 Introduction[1]

In this chapter we study the computation of parameters the $\mathcal{H}$-elimination distance and $\mathcal{H}$-treewidth for hereditary graph classes $\mathcal{H}$, which are hybrids between the structural parameters treewidth and treedepth, and the size of an $\mathcal{H}$-deletion set. Relevant background information can be found in Chapter 3.

Our main results show that for both parameters $\mathbf{tw}_{\mathcal{H}}$ and $\mathbf{ed}_{\mathcal{H}}$, deciding if their value is at most k in an input graph is non-uniform fixed-parameter tractable parameterized by k when $\mathcal{H}$ is the class of bipartite graphs. Uniform versus non-uniform algorithms were discussed in Chapter 2. As a side-product of our proof, we show that both are non-uniformly FPT when $\mathcal{H}$ is characterized by a finite number of forbidden induced subgraphs, generalizing the results of Agrawal et al. [4] who showed it for $\mathbf{ed}_{\mathcal{H}}$. The non-uniformity of our algorithms stems from the use of a meta-theorem by Lokshtanov et al. [135, Theorem 23] which encapsulates the technique of *recursive understanding*. This theorem essentially states that for any problem expressible in *Counting Monadic Second Order* (CMSO) logic, the effort of classifying whether the problem is in FPT is reduced to inputs that are (s, c)-unbreakable (formally defined later). The theorem allows us to use the technique of recursive understanding in a black box manner, leading to a streamlined proof at the expense of obtaining non-uniform algorithms. We believe that uniform algorithms can be obtained using the

[1] This chapter is mostly based on [108]. The hardness proof in Section 4.4 stems from [110].

same approach by implementing the recursive understanding step from scratch and deriving an explicit bound on the sizes of representatives for the canonical congruence for $\mathbf{ed}_{\mathcal{H}}$ and $\mathbf{tw}_{\mathcal{H}}$ on t-boundaried graphs. As the running times would not be practical in any case, we did not pursue this route.

Our proof is independent of that of Agrawal et al. [4], but is based on an older approach inspired by the earlier work of Ganian et al. [87] that contains similar ideas. The key ingredient for our work is the insight that the approach based on recursive understanding used by Ganian et al. [87] to compute a hybrid parameterization for instances of constraint satisfaction problems, can be applied more generally to aid in the computation of $\mathbf{ed}_{\mathcal{H}}$ and $\mathbf{tw}_{\mathcal{H}}$. We can lift one of their main lemmas to a more general setting, where it roughly shows that given an $(s(k), 2k)$-unbreakable graph G and an $\mathcal{H}$-deletion set X in G that is a *subset* of some (unknown) structure that witnesses the value of $\mathbf{tw}_{\mathcal{H}}$ or $\mathbf{ed}_{\mathcal{H}}$, we can determine in FPT time whether such a witness exists. This allows $\mathbf{ed}_{\mathcal{H}}$ and $\mathbf{tw}_{\mathcal{H}}$ to be computed in FPT time if we can efficiently find a deletion set with the stated property. For families $\mathcal{H}$ defined by finitely many forbidden induced subgraphs, a simple bounded-depth branching algorithm suffices. Our main contribution is for bipartite graphs, where we show that the relation between odd cycle transversals and graph separators that lies at the heart of the iterative compression algorithm for OCT [159], can be combined with the fact that there are only few minimal (u, v)-separators of size at most $2k$ in $(s(k), 2k)$-unbreakable graphs, to obtain an $\mathcal{H}$-deletion set with the crucial property described above.

It is worth mentioning that the results of this chapter are also mostly covered by more recent work. Agrawal et al. [3, Thm 1.1], as mentioned in Chapter 3, show that for any hereditary union-closed graph class $\mathcal{H}$ that is CMSO expressible, computing $\mathcal{H}$-elimination distance and $\mathcal{H}$-treewidth is non-uniform FPT if and only if computing a minimum $\mathcal{H}$-modulator is non-uniform FPT parameterized by solution size. This is the case for both bipartite graphs and graphs with a finite set of forbidden *connected* induced subgraphs. The union-closed condition stems from the fact that their theorem also implies an algorithm for $\mathcal{H}$-Deletion parameterized by hybrid parameters. This requirement is indeed necessary to get this implication, in Chapter 6 we give an example of a class $\mathcal{H}$ that is not union-closed for which $\mathcal{H}$-Deletion is NP-hard in graphs with $\mathcal{H}$-elimination distance zero. The case where the finite set of forbidden induced subgraphs is not necessarily connected is also covered by the work of Fomin et al. [76], who show that computing elimination distance is non-uniform FPT for graph properties expressible by a first order-logic formula.

We complement our non-uniform algorithms with a hardness result for computing hybrid parameters for $\mathcal{H}$ being the class of perfect graphs, using

an existing reduction from Hitting Set to the Perfect Deletion problem of Heggernes et al. [101]. Assuming FPT $\neq$ W[1], we show that no FPT-approximation algorithms are possible for $\mathbf{ed}_{\mathsf{perfect}}$ or $\mathbf{tw}_{\mathsf{perfect}}$. More formally, for any computable functions f and g, we show the following. There can be no algorithm running in time $f(k) \cdot n^{\mathcal{O}(1)}$ that, given a graph G and integer k, either outputs a certificate that shows that these parameters have value at most $g(k) \cdot k$, or correctly decides that these parameters are strictly larger than k.

Organization. The remainder of this chapter is organized as follows. In Section 4.2 we introduce some preliminaries. In Section 4.3 we show that deciding if $\mathbf{tw}_{\mathcal{H}}(G) \leq k$ or $\mathbf{ed}_{\mathcal{H}}(G) \leq k$ is non-uniform FPT for graph classes $\mathcal{H}$ with a finite set of forbidden induced subgraphs and for bipartite graphs. Finally, in Section 4.4 we show that unless W[1] = FPT, we cannot expect the same for the class of perfect graphs.

4.2 Preliminaries for classification

A parameter is a function that assigns an integer to each graph. A parameter f is minor-closed if $f(H) \leq f(G)$ for each minor H of G.

4.2.1 $\mathcal{H}$-treewidth and $\mathcal{H}$-elimination distance

As shown in Lemma 3.5, the $\mathcal{H}$-elimination distance is equivalent to the minimum treedepth of the torso of a set $X \subseteq V(G)$ such that each component of $G-X$ belongs to $\mathcal{H}$. For such a set X with $\mathbf{td}(\mathbf{T}_G(X)) = k$, we say that X is an $\mathbf{ed}_{\mathcal{H}}$ witness of depth k. Similarly, as shown in Lemma 3.9, the $\mathcal{H}$-treewidth of a graph is equivalent to the minimum treewidth of the torso of such a set X. If $\mathbf{tw}(\mathbf{T}_G(X)) = k$, we say that X is an $\mathbf{tw}_{\mathcal{H}}$ witness of width k. Since the torso operation on X turns the neighborhood of each connected component of $G - X$ into a clique, by Observation 2.5 we get the following.

Observation 4.1. *If X is a $\boldsymbol{tw}_{\mathcal{H}}$ witness of width $k-1$ (respectively $\boldsymbol{ed}_{\mathcal{H}}$ witness of depth k) in a graph G, then $|N_G(C)| \leq k$ for every $C \in \mathrm{cc}(G - X)$.*

The main problem we analyze is the following decision problem.

$\mathcal{H}$-treewidth / $\mathcal{H}$-elimination distance	**Parameter:** k
Input: A graph G, an integer k.	
Question: Decide whether $\mathbf{tw}_{\mathcal{H}}(G) \leq k-1$ / $\mathbf{ed}_{\mathcal{H}}(G) \leq k$.	

Definition 4.2. [135] Let G be a graph and $s, c \in \mathbb{N}$. A partition (X, C, Y) of $V(G)$ is an (s, c)-separation in G if:

- C is a separator, that is, no edge has one endpoint in X and one in Y,
- $|C| \leq c$, $|X| \geq s$, and $|Y| \geq s$.

A graph G is (s, c)-unbreakable if there is no (s, c)-separation in G.

The following is similar to Lemma 21 of Ganian et al. [87]. Essentially the lemma shows that for unbreakable graphs with bounded hybrid parameter, either the non-hybrid parameter is also bounded, or there is a unique large connected component in $G - X$ for any witness X.

Lemma 4.3. *Let G be an (s, c)-unbreakable graph for $s, c \in \mathbb{N}$ and $\mathcal{H}$ be a hereditary graph class such that $\boldsymbol{tw}_{\mathcal{H}}(G) \leq k-1$ (resp. $\boldsymbol{ed}_{\mathcal{H}}(G) \leq k$) and $c \geq k$. Then at least one of the following holds:*

1. *$\boldsymbol{tw}(G) \leq s + k - 1$ (resp. $\boldsymbol{td}(G) \leq s + k - 1$),*
2. *each $\boldsymbol{tw}_{\mathcal{H}}$ (resp. $\boldsymbol{ed}_{\mathcal{H}}$) witness X of G satisfies the following:*
   - *$G - X$ has exactly one connected component C of size at least s, and*
   - *$|V(G) \setminus N[C]| < s$ and $|X| \leq s + k - 1$.*

Proof. Consider an arbitrary witness X. If all connected components of $G - X$ have size at most $s - 1$, then we argue that (1) holds. In the $\mathbf{tw}_{\mathcal{H}}(G) \leq k - 1$ case, by Observation 3.7 there is a minimum-width tree $\mathcal{H}$-decomposition (T, χ, L) where each base component is connected. It follows that each leaf bag (the only bags with a non-empty intersection with L) has size at most $k + s - 1$ and so that (T, χ) is a tree decomposition of width at most $s + k - 2$, which is even stronger than (1). In the $\mathbf{ed}_{\mathcal{H}}(G) \leq k$ case, the components of $G - X$ can be eliminated in at most $s - 1$ rounds so that $\mathbf{td}(G) \leq s + k - 1$.

In the remaining case let C be some component of $G - X$ of size at least s. First observe that $N(C) \subseteq X$ and $|N(C)| \leq k$ by Observation 4.1. If $|V(G) \setminus N[C]| \geq s$, then $(C, N(C), V(G) \setminus N[C])$ is an (s, c)-separation as $k \leq c$. Since G is (s, c)-unbreakable we must have that $|V(G) \setminus N[C]| < s$. Since for any connected component C' of $G - X$ besides C it holds that $V(C') \subseteq V(G) \setminus N[C]$, we get $|V(C')| < s$ too. Finally note that $X \subseteq V(G) \setminus C$, $|V(G) \setminus N[C]| < s$, and $|N(C)| \leq k$ imply that $|X| \leq s + k - 1$ for any witness X and hence (2) holds. □

The following lemma bounds the number of small connected vertex sets with a small neighborhood and the time needed to enumerate all of them. It has been referred to as the firefighters lemma [83]. It was originally stated for connected sets of exactly b vertices with an open neighborhood of exactly f vertices.

Lemma 4.4. *[84, Lemma 3.2] Let G be a graph. For every $v \in V(G)$ and $b, f \geq 0$, the number of connected vertex sets $B \subseteq V(G)$ such that:*

(a) $v \in B$,

(b) $|B| \leq b + 1$, *and*

(c) $|N(B)| \leq f$

is at most $b \cdot f \cdot \binom{b+f}{b}$. Furthermore they can be enumerated in $\mathcal{O}(n^2 \cdot b^2 \cdot f \cdot (b+f) \cdot \binom{b+f}{b})$ time using polynomial space.

4.2.2 Capturing hybrid parameterizations with CMSO

Counting monadic second-order logic (CMSO) was introduced in Chapter 2. For a more complete introduction we refer to the book of Courcelle and Engelfriet [45].

Let $\mathcal{H}$ be a graph class. We say that containment in $\mathcal{H}$ is expressible in CMSO if there exists a CMSO formula $\varphi_{\mathcal{H}}$ such that for any graph G it holds that $G \models \varphi_{\mathcal{H}}$ if and only if $G \in \mathcal{H}$.

Lemma 4.5. *There exist CMSO-formulas with the following properties:*

1. *For any graph H, there exists a formula $\varphi_{\mathrm{H-MINOR}}(X)$ such that for any graph G and any $X \subseteq V(G)$ it holds that $(G, X) \models \varphi_{\mathrm{H-MINOR}}(X)$ if and only if H is a minor of $G[X]$.*
2. *For any graph class $\mathcal{H}$ characterized by a finite set of forbidden induced subgraphs, there exists a formula $\varphi_{\mathcal{H}}$ such that for any graph G it holds that $G \models \varphi_{\mathcal{H}}$ if and only if graph $G \in \mathcal{H}$.*
3. *There exists a formula φ_{BIP} such that for any graph G it holds that $G \models \varphi_{BIP}$ if and only if graph G is bipartite.*
4. *For each $k \in \mathbb{N}$, for each graph class $\mathcal{H}$ such that containment in $\mathcal{H}$ is CMSO expressible, and for each minor-closed parameter f, there exists a formula $\varphi_{(k,\mathcal{H},f)}(X)$ such that for any graph G and any $X \subseteq V(G)$ we have $(G, X) \models \varphi_{(k,\mathcal{H},f)}(X)$ if and only if the following holds: $f(\boldsymbol{T}_G(X)) \leq k$ and $C \in \mathcal{H}$ for each $C \in \mathrm{cc}(G - X)$.*

Proof. For (1) see for instance Corollary 1.14 [45], we repeat it here as we adapt it for (4). The formula $\textsc{conn}(X)$ certifies that the graph $G[X]$ is connected (see [48, page 178]). Using this formula, minor containment can then be expressed by the existence of the appropriate branch sets (see for instance [48, Section 6.3]). In the second formula, h denotes $|V(H)|$.

$$\begin{aligned}\textsc{conn}(X) = \forall_{Y\subseteq V(G)} \colon ((\exists_{u,v\in X} \colon u \in Y \wedge v \notin Y) \Rightarrow \\ (\exists_{u,v\in X} \colon u \in Y \wedge v \notin Y \wedge \mathbf{adj}(u,v)))\end{aligned}$$

$$\varphi_{\text{H}-\textsc{minor}}(X) = \exists_{Y_1,\ldots,Y_h\subseteq X} \colon (\bigwedge_{1\leq i\leq h} ((\exists_y \colon y \in Y_i) \wedge \textsc{conn}(Y_i))$$

$$\wedge \bigwedge_{1\leq i<j\leq h} \neg\exists_y (y \in Y_i \wedge y \in Y_j) \wedge \bigwedge_{(i,j)\in E(H)} \exists_{u,v} (u \in Y_i \wedge v \in Y_j \wedge \mathbf{adj}(u,v)))$$

For (2), let $\mathcal{F}_{\mathcal{H}}$ be the forbidden induced subgraph characterization of $\mathcal{H}$, where $H \in \mathcal{F}_{\mathcal{H}}$ is a graph on vertex set $[h]$. A formula $\varphi_{\mathcal{H}}$ is given below and is similar to that of checking for a minor.

$$\begin{aligned}\varphi_{\mathcal{H}} = \bigwedge_{H\in\mathcal{F}_{\mathcal{H}}} \neg\exists_{v_1,\ldots,v_h\in V(G)} \colon (\bigwedge_{1\leq i<j\leq h} v_i \neq v_j \\ \wedge \bigwedge_{(i,j)\in E(H)} \mathbf{adj}(v_i,v_j) \wedge \bigwedge_{(i,j)\notin E(H)} \neg\mathbf{adj}(v_i,v_j))\end{aligned}$$

Since a graph is bipartite if and only if it has a proper 2-coloring (which effectively is a partition into two independent sets), the following formula shows (3).

$$\begin{aligned}\textsc{part}(X_1,X_2) &= \forall_{v\in V(G)} \colon (v \in X_1 \wedge v \notin X_2) \vee (v \notin X_1 \wedge v \in X_2) \\ \textsc{indp}(X) &= \forall_{u,v\in X} \colon \neg\mathbf{adj}(u,v) \\ \varphi_{BIP} &= \exists_{X_1,X_2\subseteq V(G)} \colon \textsc{part}(X_1,X_2) \wedge \textsc{indp}(X_1) \wedge \textsc{indp}(X_2)\end{aligned}$$

Finally for (4) note that since f is minor-closed, the set of graphs F with $f(F) \leq k$ has a finite set of forbidden minors by the Graph Minor Theorem of Robertson and Seymour. Using formula (1), we can check whether $\mathbf{T}_G(X)$ contains a forbidden minor. The only thing we need to change is the use of **adj**: an edge uv is in $\mathbf{T}_G(X)$ if either u and v are adjacent, or if there is a path whose internal vertices are not in X.

$$\begin{aligned}\textsc{Tadj}(u,v,X) = \mathbf{adj}(u,v) \vee \exists_{P\subseteq V(G)} \colon (u,v \in P \wedge \textsc{conn}(P) \\ \wedge \forall_{w\in P} \colon (w = u \vee w = v \vee w \notin X))\end{aligned}$$

Finally we can check if each connected component C of $\mathrm{cc}(G - X)$ is in $\mathcal{H}$ by going over every vertex subset and verifying that if it is connected, disjoint from X, and maximal, then it induces a graph in $\mathcal{H}$. The latter is CMSO expressible by the precondition of (4). □

Since both treewidth and treedepth are minor-closed parameters, we note the following from the lemma above.

Note 4.6. *For each $k \in \mathbb{N}$ and graph class $\mathcal{H}$ such that containment in $\mathcal{H}$ is CMSO-expressible, there exists a formula $\varphi_{(k,\mathcal{H},\boldsymbol{tw})}$ (respectively $\varphi_{(k,\mathcal{H},\boldsymbol{td})}$) such that (G, k) is a* YES*-instance of $\mathcal{H}$-*TREEWIDTH *(respectively $\mathcal{H}$-*ELIMINATION DISTANCE*) if and only if $G \models \varphi_{(k,\mathcal{H},\boldsymbol{tw})}$ (respectively $G \models \varphi_{(k,\mathcal{H},\boldsymbol{td})}$).*

CMSO formulas can have free variables. A graph together with an evaluation of free variables is called a *structure*. We denote the problem of evaluating a CMSO formula φ on a structure by CMSO[φ]. The following theorem is the main tool used to achieve our algorithms, we apply it only to formulas without free variables.

Theorem 4.7. *[135, Theorem 22] Let φ be a CMSO formula. For all $c \in \mathbb{N}$, there exists $s \in \mathbb{N}$ such that if there exists an algorithm that solves* CMSO[φ] *on (s, c)-unbreakable structures in time $\mathcal{O}(n^d)$ for some $d > 4$, then there exists an algorithm that solves* CMSO[φ] *on general structures in time $\mathcal{O}(n^d)$.*

Using the theorem above one can also derive a meta-algorithmic result for obtaining non-uniform FPT algorithms. As our formulation differs slightly from its original form, we provide a short proof which roughly follows the proof from the arXiv version [136, Theorem 3].

Theorem 4.8. *[135, Theorem 23] Let $\hat{\varphi}$ be a CMSO formula. For all $\hat{c} \colon \mathbb{N}_0 \to \mathbb{N}_0$, there exists $\hat{s} \colon \mathbb{N}_0 \to \mathbb{N}_0$ such that if* CMSO[$\hat{\varphi}$] *parameterized by k is non-uniform FPT on $(\hat{s}(k), \hat{c}(k))$-unbreakable structures, then* CMSO[$\hat{\varphi}$] *parameterized by k is non-uniform FPT on general structures.*

Proof. Let $\hat{c} \colon \mathbb{N}_0 \to \mathbb{N}_0$ and define $\hat{s} \colon \mathbb{N}_0 \to \mathbb{N}_0$ as follows. For all $k \in \mathbb{N}_0$, let $\hat{s}(k)$ be the constant s in Theorem 4.7 and $c = \hat{c}(k)$. Suppose that CMSO[$\hat{\varphi}$] is FPT on $(\hat{s}(k), \hat{c}(k))$-unbreakable structures. Then for every fixed k we can solve it in $\mathcal{O}(n^d)$ time for some fixed $d > 4$. By Theorem 4.7 it follows that we can solve CMSO[$\hat{\varphi}$] in $\mathcal{O}(n^d)$ time for every fixed k on general structures. Therefore the problem is non-uniform FPT on general structures. □

4.3 Computing hybrid parameters

In this section we present our algorithms. In Section 4.3.1 we present a key lemma. In Section 4.3.2 we use it to deal with graph classes that are characterized by a finite number of forbidden induced subgraphs, and in Section 4.3.3 we deal with bipartite graphs.

4.3.1 Extracting witnesses from deletion sets contained in them

Our strategy for solving $\mathcal{H}$-TREEWIDTH and $\mathcal{H}$-ELIMINATION DISTANCE is similar to that of Lemmas 9 and 10 of Ganian et al. [87] and is based on Lemma 4.3. Given an $(s(k), c(k))$-unbreakable graph, either the treewidth of the graph is bounded (1) and we can solve the problem directly using Courcelle's Theorem, or each witness is of bounded size and introduces some structure (2).

In the following lemma we assume we are in the latter case (hence the $\mathbf{tw}(G) > s(k) + k$ condition) and are given some $\mathcal{H}$-deletion set Y. We show that given an $(s(k), c(k))$-unbreakable graph, in FPT time we can find a witness X such that $Y \subseteq X$ if such a witness exists. The algorithm is non-uniform, since in general it may not be known how to obtain $s(k)$.

Lemma 4.9. *Consider functions $c, s\colon \mathbb{N} \to \mathbb{N}$ such that $c(x) \geq x$ for each $x \in \mathbb{N}$. Let $\mathcal{H}$ be a hereditary graph class such that containment in $\mathcal{H}$ is solvable in polynomial time. There is a function $f\colon \mathbb{N} \to \mathbb{N}$ and a constant d such that the following holds: for each k there is an algorithm that runs in time $f(k) \cdot n^d$ that, given an $(s(k), c(k))$-unbreakable graph with $\boldsymbol{tw}(G) > s(k) + k$ and an $\mathcal{H}$-deletion set Y of size at most $s(k) + k$, decides whether there is an $\boldsymbol{tw}_{\mathcal{H}}(G)$ witness X of width at most $k - 1$ (respectively $\boldsymbol{ed}_{\mathcal{H}}(G)$ witness X of depth at most k) such that $Y \subseteq X$.*

Proof. We refer to a witness as either being an $\mathbf{tw}_{\mathcal{H}}$ witness of width at most $k - 1$ or an $\mathbf{ed}_{\mathcal{H}}$ witness of depth at most k. Given a set $X \subseteq V(G)$, we can verify that it is a witness by testing whether $\mathbf{tw}(\mathbf{T}_G(X)) \leq k - 1$ (respectively $\mathbf{td}(\mathbf{T}_G(X)) \leq k$) in FPT time [17, 160] and verifying that each connected component $C \in \mathrm{cc}(G - X)$ is contained in $\mathcal{H}$, which can be done in polynomial time by assumption.

We show that we can find a witness if it exists, by doing the above verification for FPT many vertex subsets $D \subseteq V(G)$, as follows.

1. For each $y \in Y$, let $\mathcal{C}_y$ be the set of connected vertex sets S with $y \in S$, $|S| \leq s(k)$ and $|N(S)| \leq k$. For each $B \subseteq Y$ with $|B| \leq k$, a choice tuple

t_B contains an entry for each $y \in Y \setminus B$, where entry $t_B[y]$ is some set $C_y \in \mathcal{C}_y$.

2. For each $B \subseteq Y$ with $|B| \leq k$ and each choice tuple t_B, if $G - (Y \cup \bigcup_{y \in Y \setminus B} N(t_B[y]))$ has exactly one connected component C of size at least $s(k)$ and $|V(G) \setminus N[C]| < s(k)$, apply the witness verification test to $D = Y \cup \bigcup_{y \in Y \setminus B} N(t_B[y]) \cup Q$ for each $Q \subseteq V(G) \setminus N[C]$.

3. Return the logical or of all witness verification tests.

We argue that the algorithm runs in FPT time. Note that as $|Y| \leq s(k) + k$, there are at most $\binom{s(k)+k}{k}$ choices for B. Furthermore $\mathcal{C}_y$ can be computed in FPT time using Lemma 4.4, hence the number of choice tuples is also FPT many. For each choice for B and each choice tuple t_B, there are at most $2^{s(k)}$ choices for Q. Since each vertex set can be verified to be a witness in FPT time, the running time claim follows.

Finally we argue correctness of the algorithm. Since $\mathbf{tw}(G) > s(k) + k$ (and also $\mathbf{td}(G) > s(k) + k$ as $\mathbf{tw}(G) \leq \mathbf{td}(G) - 1$), by Lemma 4.3 any witness X is of size at most $s(k) + k - 1$, the graph $G - X$ has exactly one large connected component C of size at least $s(k)$, and $|V(G) \setminus N[C]| < s(k)$.

Suppose G has a witness that is a superset of Y. Fix some witness X of minimal cardinality with $Y \subseteq X$ and let C be the unique component of size at least $s(k)$ of $G - X$. Note that since $C \cap X = \emptyset$, we have $C \cap Y = \emptyset$. The situation is sketched in Figure 4.1.

Let $B = N(C) \cap Y$. By Observation 4.1 we have $|N(C)| \leq k$, hence the branching algorithm makes this choice for B at some point. For each $y \in Y \setminus B$, let C_y be the connected component of $G - N[C]$ containing y. Since $|V(G) \setminus N[C]| < s(k)$ and $|N(C)| \leq k$, we have that $|V(C_y)| < s(k)$ and $|N(C_y)| \leq k$. Note that $N(C_y) \subseteq N(C) \subseteq X$. The branching algorithm at some point tries the choice tuple t_B where $t_B[y] = C_y$ for each $y \in Y \setminus B$. Consider the set $A = Y \cup \bigcup_{y \in Y \setminus B} N(t_B[y])$. Note that $A \subseteq X$ by construction.

If $N(C) \subseteq A$, then the single large component of $G - A$ of size at least $s(k)$ is exactly C. Since $|V(G) \setminus N[C]| < s(k)$, it follows that $X = A \cup Q$ for some $Q \subseteq V(G) \setminus N[C]$. It follows that the algorithm correctly identifies X in this case.

The only remaining case is $N(C) \not\subseteq A$. We argue that this cannot happen when witness X is of minimal cardinality. Suppose $N(C) \not\subseteq A$ and let $v \in N(C) \setminus A$. Let $Z = Y \cup \bigcup_{y \in Y \setminus B} N[C_y]$ and note that we take the *closed* neighborhoods of the components, instead of the *open* neighborhoods as in the definition of A. Let C_v^* be the connected component of $G - (C \cup Z)$ that contains v. We argue that $X \setminus C_v^*$ is a witness. Note that $C_v^* \cap Y = \emptyset$ by construction

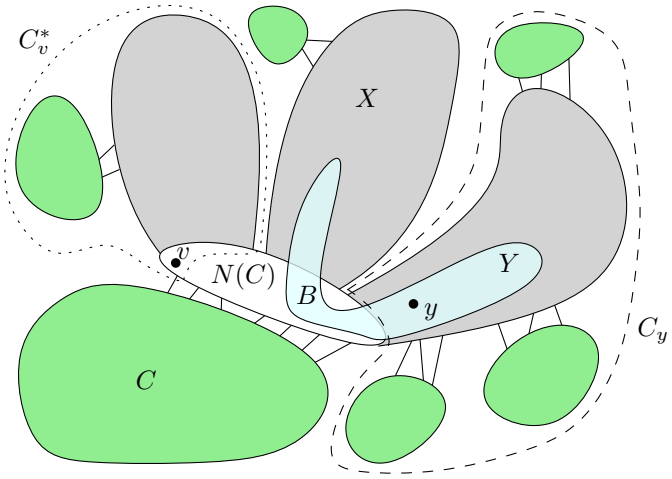


Figure 4.1: Situation sketch of Lemma 4.9. The set X in grey denotes a witness and the set C is the single large component of $G - X$.

as $Y \subseteq Z$. Because Y is an $\mathcal{H}$-deletion set and $\mathcal{H}$ is hereditary, it follows that for each connected component C' in $G - (X \setminus C_v^*)$ we have $C' \in \mathcal{H}$. We argue that $N(C_v^*) \subseteq N[C]$. Since C_v^* is a connected component of $G - (C \cup Z)$ we have $N(C_v^*) \subseteq C \cup Z$, so it suffices to show that $N(C_v^*) \cap Z \subseteq N(C)$. Assume for a contradiction that C_v^* contains a vertex v' adjacent to some $z \in Z \setminus N(C)$; note that $v' \notin Z$. If $z \in Y$, then $z \in Y \setminus N(C) = Y \setminus B$ and the connected component C_z of $G - N[C]$ is adjacent to v', implying $v' \in N[C_z]$ and therefore $v' \in Z$; a contradiction. If $z \notin Y$, then by definition of Z we have $z \in N[C_y]$ for some $y \in Y \setminus B$. Since $N(C_y) \subseteq N(C)$ this implies $z \in C_y$. But then $v' \notin C \cup Z$ is adjacent to a vertex of the component C_y of $G - N[C]$, so $v' \in Z$ by definition of Z; a contradiction.

Since $N(C_v^*) \subseteq N[C]$ and v is adjacent to at least one vertex in C as $v \in N(C)$, it follows that $C \cup C_v^*$ is a connected component of $G - (X \setminus C_v^*)$ with $N(C \cup C_v^*) \subseteq N(C)$. Therefore $\mathbf{T}_G(X \setminus C_v^*)$ is an induced subgraph of $\mathbf{T}_G(X)$. We conclude that $X \setminus C_v^* \supseteq Y$ is a witness. Since X was assumed to be of minimal cardinality, we arrive at a contradiction and hence $A \supseteq N(C)$. $\square$

4.3.2 Classes $\mathcal{H}$ with finitely many forbidden induced subgraphs

We give a first simple application of Lemma 4.9 for classes $\mathcal{H}$ with finitely many forbidden induced subgraphs, that results from the fact that all $\mathcal{H}$-modulators of bounded size can be enumerated [35].

Theorem 4.10. *Let $\mathcal{H}$ be a graph class characterized by a finite set of forbidden induced subgraphs. Then $\mathcal{H}$-*TREEWIDTH *and $\mathcal{H}$-*ELIMINATION DISTANCE *are non-uniformly fixed-parameter tractable.*

Proof. By Lemma 4.5 containment in $\mathcal{H}$ is CMSO expressible, therefore by Note 4.6 there exists a formula $\varphi_{(k,\mathcal{H},f)}$ for each $f \in \{\mathbf{tw}, \mathbf{td}\}$ such that an instance (G, k) of $\mathcal{H}$-TREEWIDTH (respectively $\mathcal{H}$-ELIMINATION DISTANCE) is a YES-instance if and only if $G \models \varphi_{(k,\mathcal{H},f)}$. Furthermore, containment in $\mathcal{H}$ is polynomial time solvable, as we can verify that a graph does not contain any of the finitely many forbidden induced subgraphs.

We argue that both problems are in FPT when the input graph G is $(s(k), k)$-unbreakable for any $s\colon \mathbb{N} \to \mathbb{N}$. If $\mathbf{tw}(G) \leq s(k) + k$, we solve the problems directly using Courcelle's Theorem (Theorem 2.7) using $\varphi_{(k,\mathcal{H},f)}$. Otherwise by Lemma 4.3 each witness X is of size at most $s(k) + k - 1$. We can enumerate all minimal $\mathcal{H}$-deletion sets $\mathcal{Y}$ of size at most $s(k) + k - 1$ in FPT time by finding a forbidden induced subgraph and branching in all finitely many ways of deleting its vertices [35]. Since any witness X is an $\mathcal{H}$-deletion set, for some $Y \in \mathcal{Y}$ we have $Y \subseteq X$. Hence we solve the problem by calling Lemma 4.9 for each $Y \in \mathcal{Y}$ with $c(k) = k$. Applying Theorem 4.8 concludes the proof. □

Using known characterizations by a finite number of forbidden induced subgraphs (cf. [30]) we obtain the following corollary to Theorem 4.10.

Corollary 4.11. *Let $\mathcal{H}$ be set of graphs that are either (1) cliques, (2) claw-free, (3) of degree at most d for fixed d, (4) cographs, or (5) split graphs. $\mathcal{H}$-*TREEWIDTH *and $\mathcal{H}$-*ELIMINATION DISTANCE *are non-uniformly fixed-parameter tractable.*

4.3.3 Bipartite graphs

We use shorthand bip to denote the class of bipartite graphs. The problem of deleting k vertices to obtain a bipartite graph is better known as the ODD CYCLE TRANSVERSAL (OCT) problem. The following lemma encapsulates the connection between odd cycle transversals and separators that forms the

key of the iterative-compression algorithm for OCT due to Reed, Smith, and Vetta [159]. A partitioned odd cycle transversal (OCT) W is simply an odd cycle transversal with a partition (W_L, W_I) of W.

Lemma 4.12. *For each partitioned OCT $W = (W_L, W_I)$ of G, for each partition $(W_{L,1}, W_{L,2})$ of W_L into two independent sets, for each proper 2-coloring $c\colon V(G) \setminus W \to [2]$ of $G - W$, we have the following equivalence for each $X \subseteq V(G) \setminus W$: the graph $(G - W_I) - X$ has a proper 2-coloring with $W_{L,1}$ color 1 and $W_{L,2}$ color 2 if and only if the set X separates A from R in the graph $G - W$, with:*

$$A = (N_{G-W_I}(W_{L,1}) \cap c^{-1}(1)) \cup (N_{G-W_I}(W_{L,2}) \cap c^{-1}(2))$$
$$R = (N_{G-W_I}(W_{L,1}) \cap c^{-1}(2)) \cup (N_{G-W_I}(W_{L,2}) \cap c^{-1}(1)).$$

Observe that $c^{-1}(i) \subseteq V(G - W)$ for each $i \in [2]$, so that $A \cup R \subseteq V(G - W)$, and that the separator X is unrestricted, it is allowed to intersect $A \cup R$.

Proof. ($\Rightarrow$) Suppose that $(G - W_I) - X$ has a proper 2-coloring with $W_{L,1}$ color 1 and $W_{L,2}$ color 2. Suppose for a contradiction that X is not an (A, R)-separator in $G - W$, that is, in $(G - W) - X$ there is a connected component H simultaneously containing a vertex $a \in A$ and a vertex $r \in R$. Note that H is also a connected subgraph of $G - W$ and therefore bipartite, which means that if $|V(H)| \geq 2$ there is a unique partition of H into two independent sets, so that H has exactly two proper 2-colorings depending on which independent set is called color 1 and which is called color 2. Note that if $|V(H)| = 1$, the fact that H has exactly two proper 2-colorings is trivial. It follows that any proper 2-coloring of H either coincides with the 2-coloring c of $G - W$, or is such that every vertex gets the opposite of its current color under c.

The fact that $a \in A$ means by definition that either we have $c(a) = 1$ and a is adjacent to a vertex of $W_{L,1}$, or $c(a) = 2$ and a is adjacent to a vertex of $W_{L,2}$. In either case, it shows that in any proper 2-coloring of $(G - W_I) - X$ in which $W_{L,1}$ gets color 1 and $W_{L,2}$ gets color 2, the color of a must be different from its color under c. By an analogous argument, the fact that $r \in R$ means that in any proper 2-coloring of $(G - W_I) - X$ in which $W_{L,1}$ gets color 1 and $W_{L,2}$ gets color 2, the color of r must be identical to its color under c.

Since a and r belong to the same connected subgraph H of $(G - W_I) - X$, in any proper 2-coloring they either both change their color compared to c, or both keep their color compared to c. This is a contradiction to the fact that a changed color and r remained of the same color.

($\Leftarrow$) For the converse, consider a set X that separates A from R in $G - W$. We construct a proper 2-coloring c' of $(G - W_I) - X$ in which $W_{L,1}$ gets color 1

and $W_{L,2}$ gets color 2, as follows. Let $c'(v \in W_{L,1}) = 1$ and $c'(v \in W_{L,2}) = 2$. For each connected component of $(G - W) - X$ that contains a vertex from R, let its coloring under c' be identical to its coloring under c. For each connected component of $(G - W) - X$ that contains no vertex from R, let its coloring under c' be the opposite of its coloring under c. Since c was a proper coloring, there are no color conflicts among vertices of $(G - W) - X$. Since both $W_{L,1}$ and $W_{L,2}$ are independent sets, there are no color conflicts among $W_{L,1}$ or among $W_{L,2}$. It remains to verify that each edge connecting W_L to a vertex of $(G - W) - X$ is properly colored. But this follows from our construction: all neighbors of $W_{L,1}$ with color 1 under c belong to A and therefore have their coloring swapped to 2 in c'. Similarly all neighbors of $W_{L,2}$ with color 2 under c belong to A and have their coloring swapped to 1 in c'. Finally, neighbors of $W_{L,1}$ with color 2 in c belong to R and therefore have the same color 2 in c', and neighbors of $W_{L,2}$ with color 1 in c belong to R and have the same color 1 in c', ensuring these edges are properly colored as well. □

With the above lemma at hand we are now ready to show the main result of this section, namely non-uniform FPT algorithms for computing bip-TREEWIDTH and bip-ELIMINATION DISTANCE. These are obtained by enumerating a list of odd cycle transversals with the guarantee that at least one of them is contained in a witness. The result then follows by Lemma 4.9.

Lemma 4.13. *The* bip-TREEWIDTH *and* bip-ELIMINATION DISTANCE *problems are non-uniformly fixed-parameter tractable.*

Proof. By Lemma 4.5 containment in the class of bipartite graphs is CMSO expressible, therefore by Note 4.6 there exists a formula $\varphi_{(k,\mathsf{bip},f)}$ for each $f \in \{\mathbf{tw}, \mathbf{td}\}$ such that an instance (G, k) of bip-TREEWIDTH (respectively bip-ELIMINATION DISTANCE) is a YES-instance if and only if $G \models \varphi_{(k,\mathsf{bip},f)}$. We argue that both problems are FPT in $(s(k), 2k)$-unbreakable graphs for any $s \colon \mathbb{N} \to \mathbb{N}$. Note that the theorem then follows by Theorem 4.8.

Let G be an $(s(k), 2k)$-unbreakable graph. As before, we use the term witness to either refer to an $\mathbf{tw}_{\mathcal{H}}$ witness of width at most $k - 1$ or an $\mathbf{ed}_{\mathcal{H}}$ witness of depth at most k, depending on the problem being solved. We first test whether $\mathbf{tw}(G) \leq s(k) + k$, in FPT time [17]. If so, then we can solve the problems directly using Courcelle's Theorem (Theorem 2.7) using $\varphi_{(k,\mathsf{bip},f)}$. Otherwise by Lemma 4.3 the size of each witness in G is at most $s(k) + k - 1$, and for each witness X there is a unique connected component of $G - X$ of at least $s(k)$ vertices, henceforth called the *large component*. We use a two-step process to find an odd cycle transversal that is a subset of some witness (if a witness exists), so that we may invoke Lemma 4.9 to find a witness.

For a witness X^* in G and an odd cycle transversal W of G, we say that a partition (W_L, W_I) of W is *weakly consistent* with X^* if for the unique large component C of $G - X^*$ we have that $W \cap C = W_L$, $|W_L| \leq k$, and $W \subseteq C \cup X^*$. An odd cycle transversal W is *strongly consistent* with X^* if $W \subseteq X^*$.

The next two claims show that these types of OCTs can be computed efficiently in the $(s(k), 2k)$-unbreakable input graph G. We assume we are in the $\mathbf{tw}(G) > s(k) + k$ case so that there is a unique large component by Lemma 4.3 and so that a weakly consistent partition is well defined.

Claim 4.14. There is an FPT algorithm that outputs a list of partitioned OCTs in G with the guarantee that for each witness X, there is a partitioned OCT on the list that is weakly consistent with X.

Proof. The algorithm proceeds as follows.

1. Initialize an empty list $\mathcal{W}$. Compute a minimum cardinality odd cycle transversal $W \subseteq V(G)$ of size at most $s(k) + k - 1$. If no such OCT exists, return the empty list.

2. For each $y \in V(G)$, let $\mathcal{C}_y$ be the set of connected vertex sets S with $y \in S$, $|S| \leq s(k)$ and $|N(S)| \leq k$. For each partition $P = (W_L, W_I, W_R)$ of W, a choice tuple t_P contains an entry for each $y \in W_R$, where entry $t_P[y]$ is some set $C_y \in \mathcal{C}_y$. (Each partition corresponds to a guess of how W intersects a witness: vertices in W_L correspond to those in the *large* component, W_I corresponds to vertices *inside* the witness, and W_R corresponds to vertices in the *remaining* small components.)

3. For each partition $P = (W_L, W_I, W_R)$ of W and each choice tuple t_P, if $(W \setminus W_R) \cup \bigcup_{y \in W_R} N(t_P[y])$ is an OCT, then add $(W_L, W_I \cup \bigcup_{y \in W_R} N(t_P[y]))$ to $\mathcal{W}$.

4. Return the list $\mathcal{W}$.

We argue the running time of the steps described above. The first step can be done in time $3^{s(k)+k} \cdot n^{\mathcal{O}(1)}$ [159, 48]. For each $y \in V(G)$, computing $\mathcal{C}_y$ is in FPT by Lemma 4.4. Since there are $3^{s(k)+k-1}$ possible partitions P and FPT many choice tuples t_P, the running time follows. To see the correctness of the algorithm, first note that each partition in the output is an OCT by construction. All that is left to show is the output guarantee. Consider some witness X and C be the unique large component of $G - X$. Let $P = (W_L, W_I, W_R)$ be the partition such that $W \cap C = W_L$, $W \cap X = W_I$, and $W_R \subseteq V(G) \setminus (X \cup C)$. To see that $|W_L| \leq k$, observe that if this was not the

case we would obtain a smaller OCT by taking $(W \setminus W_L) \cup N(C)$, contradicting W has minimum cardinality. For each $y \in W_R$, let C_y be the connected component of $G - X$ containing y. Note that $|C_y| \leq s(k)$ by Lemma 4.3 and $|N(C_y)| \leq k$ by Observation 4.1. Therefore for some choice tuple t_P we have $t_P[y] = C_y$ for each $y \in W_R$. It follows that $(W_L, W_I \cup \bigcup_{y \in W_R} N(t_P[y]))$ that is contained in the list satisfies the output requirement for witness X. ■

Claim 4.15. There is an FPT algorithm that, given a partitioned OCT that is weakly consistent with some (unknown) witness X in G, outputs a list of OCTs in G such that at least one is strongly consistent with X.

Proof. Let $W = (W_L, W_I)$ be the given partitioned OCT. If $|W| > s(k)+k-1$, then no witness is strongly consistent with W by Lemma 4.3, hence we may assume $|W| \leq s(k) + k - 1$.

1. Initialize an empty list $\mathcal{W}$. For each $y \in V(G)$, let $\mathcal{C}_y$ be the set of connected vertex sets S with $y \in S$, $|S| \leq s(k)$ and $|N(S)| \leq 2k$. Let c^* be an arbitrary proper 2-coloring of $G - W$ and let $B_i^* = (c^*)^{-1}(i)$ for each $i \in [2]$.

2. For each partition (W_1, W_2) of W_L, let $B_1 = N(W_2) \setminus W$ and $B_2 = N(W_1) \setminus W$. Let $A = (B_1 \cap B_2^*) \cup (B_2 \cap B_1^*)$ and $R = (B_1 \cap B_1^*) \cup (B_2 \cap B_2^*)$.

   (a) For each choice $Q \in \{A, R\}$ with $|Q| \leq s(k) + k$, for each $D \subseteq Q$ with $|D| \leq k$, choice tuple $t_{Q,D}$ has an entry for each $y \in Q \setminus D$, where entry $t_{Q,D}[y]$ is some vertex set $C_y \in \mathcal{C}_y$.

   (b) For each choice $Q \in \{A, R\}$ with $|Q| \leq s(k) + k$, for each $D \subseteq Q$ with $|D| \leq k$, and for each choice tuple $t_{Q,D}$, add $(W \cup D \cup \bigcup_{y \in Q \setminus D} N(t_{Q,D}[y])) \setminus W_L$ to $\mathcal{W}$ in case it is an OCT.

The resulting list $\mathcal{W}$ is given as the output of the algorithm. The running time follows from Lemma 4.4 and the fact that there are FPT many choices for (W_1, W_2), D, and tuple $t_{Q,D}$.

We argue the correctness of the algorithm. Note that each set in the output list is an OCT by construction. Consider some witness X with (W_L, W_I) weakly consistent with X and let C be the unique large component of $G - X$, which is bipartite by definition of witness. Let $Y = (W \setminus W_L) \cup N(C) \subseteq X$, note that Y is an OCT of G. Let $c\colon V(G) \setminus Y \to [2]$ be a proper 2-coloring of $G - Y$. For some partition (W_1, W_2) of W_L we have $W_i \subseteq c^{-1}(i)$ for each $i \in [2]$. Note that since $Y \setminus W_I \subseteq N(C)$, we have that $|Y \setminus W_I| \leq k$.

By Lemma 4.12, it follows that $Y \setminus W_I \subseteq N(C)$ separates A and R in $G - W$. Note that $B_i \subseteq N[C]$ for each $i \in [2]$ since $W_L \subseteq C$, therefore $A \subseteq N[C]$ and

$R \subseteq N[C]$. Observe that $W_L \cup N(C)$ is an (A, R)-separator of size at most $2k$ in G. Therefore, since G is $(s(k), 2k)$-unbreakable, for at least one $Q \in \{A, R\}$ the vertex set reachable from $Q \setminus (W_L \cup N(C))$ in $G - (W_L \cup N(C))$ has size at most $s(k)$. Since A and R are disjoint from $W \supseteq W_L$ by definition, this implies $|Q| \leq s(k) + k$. Hence the algorithm tries this choice as $|Q| \leq s(k) + k$ is satisfied. Let $D = N(C) \cap Q$. For each $y \in Q \setminus D$, let C_y be the connected component of $G - (N(C) \cup W_L)$ containing y. Note that $|C_y| \leq s(k)$ and $|N(C_y)| \leq 2k$. Let the choice tuple $t_{Q,D}$ be such that $t_{Q,D}[y] = C_y$ for each $y \in Q \setminus D$. Observe that $(D \cup \bigcup_{y \in Q \setminus D} N(t_{Q,D}[y])) \setminus W_L \subseteq N(C)$ is an (A, R)-separator in $G - W$. Therefore $(W_I \cup D \cup \bigcup_{y \in Q \setminus D} N(t_{Q,D}[y])) \setminus W_L$ is an OCT by Lemma 4.12 contained in X, concluding the proof. ■

With the two claims above, we can solve the problem as follows. Compute a list of partitions $\mathcal{W}$ using Claim 4.14 and use each $W \in \mathcal{W}$ as input to Claim 4.15. Using the output $\mathcal{U}$ of Claim 4.15, call Lemma 4.9 for each $U \in \mathcal{U}$. By the output guarantee of the claims, for each witness X we call the lemma with $U \subseteq X$ at some point, thus solving the problem. □

4.4 No FPT-approximation for finding perfect witnesses

In this section we show that for $\mathcal{H}$ being the class of perfect graphs, no FPT-approximation algorithms exist for computing $\mathbf{tw}_{\mathcal{H}}$ or $\mathbf{ed}_{\mathcal{H}}$ unless W[1] = FPT.

For a function $g \colon \mathbb{N} \to \mathbb{N}$, a fixed-parameter tractable $g(k)$-approximation algorithm for a parameterized minimization problem is an algorithm that, given an instance of size n with parameter value k, runs in time $f(k) \cdot n^{\mathcal{O}(1)}$ for some computable function f, and has the guarantee that whenever the instance has a solution of size at most k it outputs a solution of size at most $g(k) \cdot k$. Equivalently, given an instance (G, k), there is an algorithm that either returns a solution of size at most $g(k) \cdot k$ or reports there is no solution of size at most k.

In the k-Hitting Set problem, we are given a universe U, a set system $\mathcal{F}$ over U, and a parameter k, and the task is to find a smallest cardinality set $S \subseteq U$ such that $F \cap S \neq \emptyset$ for all $F \in \mathcal{F}$. As shown by Karthik et al. [118], we have the following inapproximability result for k-Hitting Set.[2]

[2]The result is originally stated for Dominating Set, but directly carries over to Hitting Set as stated in the paper [118].

Theorem 4.16 ([118]). *There is no FPT $g(k)$-approximation for k-Hitting Set for any computable function g assuming W[1] $\neq$ FPT.*

The decision variant of the k-Hitting Set problem, which asks if a solution of size at most k exists, is known to be W[2]-hard. In the k-Perfect Deletion problem, we are given a graph G and an integer k, and ask for a smallest cardinality set $S \subseteq V(G)$ such that $G - S$ is a perfect graph. By the Strong Perfect Graph Theorem [43], this amounts to deleting odd induced cycles of length at least 5 (odd holes) and their edge complements (odd anti-holes) from the graph. Heggernes et al. [101] show that the decision variant of k-Perfect Deletion is W[2]-hard, reducing from k-Hitting Set.

We show that their reduction also rules out good approximations for $\mathbf{ed}_{\mathsf{perfect}}$ and $\mathbf{tw}_{\mathsf{perfect}}$, where perfect denotes the class of perfect graphs. Since $\mathcal{H}$-treewidth and $\mathcal{H}$-elimination distance are decision problems rather than minimization problem, we say that a fixed-parameter tractable $g(k)$-approximation for these problems either certifies that these parameters have value at most $g(k) \cdot k$ through returning an appropriate witness X, or correctly reports that the respective parameter is strictly greater than k. We show that we cannot $g(k)$-approximate perfect-treewidth and perfect-elimination distance in FPT time. Towards that goal, we first show the following intermediate lemma. For convenience we use a tree perfect-decomposition rather than a witness here, however one can easily be transformed into the other.

Lemma 4.17. *Let $g\colon \mathbb{N} \to \mathbb{N}$ be a computable function. Assuming W[1] $\neq$ FPT, there is no algorithm that, given a graph G and integer k, runs in time $f(k) \cdot n^{\mathcal{O}(1)}$ for some computable function f and either determines that the minimum size of a perfect deletion set in G is larger than k, or outputs a tree* perfect*-decomposition of G of width $g(k) \cdot k$.*

Proof. Suppose that an algorithm $\mathcal{A}$ as described in the lemma statement does exist. We will use it to build an FPT-time $2 \cdot g(k)$-approximation for k-Hitting Set, thereby showing W[1] = FPT by Theorem 4.16. By contraposition, this will prove the lemma.

We use the construction of [101] to reduce an instance of k-Hitting Set to k-Perfect Deletion. Let $(U, \mathcal{F}, k)$ be an instance of k-Hitting set. Assume $|F| \geq 2$ for all $F \in \mathcal{F}$. Build an instance (G, k) for k-Perfect Deletion as follows.

- Create an independent set X on $|U|$ vertices, let $X = \{v_u \mid u \in U\}$.
- For each set $F = \{u_1, \ldots, u_t\} \in \mathcal{F}$, add $|F| + 1$ new vertices $h_1, \ldots, h_{t+1}$. The set $\mathcal{G}_F = \{h_1, \ldots, h_{t+1}\}$ is the *set gadget* for F. Furthermore let

$X_F = \{v_{u_1}, \ldots, v_{u_t}\}$ be the universe vertices corresponding to F. Add edges $h_1 v_{u_1}$, $v_{u_1} h_2$, $h_2 v_{u_2}, \ldots, v_{u_t} h_{t+1}$, $h_{t+1} h_1$. Note that $G[\mathcal{G}_F \cup X_F]$ induces an odd hole of length at least 5.

- Take the pairwise join of the set gadgets, that is, make all vertices of $\mathcal{G}_F$ adjacent to all vertices of $\mathcal{G}_{F'}$ for distinct $F, F' \in \mathcal{F}$.

The following series of claims are proven by Heggernes et al. [101, Claim 1–4, Thm. 1].

1. The graph $G - X$ is a cograph and therefore perfect.
2. Any hole in G intersects X and exactly one set gadget $\mathcal{G}_F$.
3. Any anti-hole in G has length 5 and is therefore a hole of length 5.
4. If $S \subseteq V(G)$ such that $G - S$ is perfect, then there is $S' \subseteq X$ with $|S'| \leq |S|$ such that $G - S'$ is perfect.
5. For each $\ell \geq 0$, the instance $(U, \mathcal{F})$ has a solution of size at most ℓ if and only if G has a perfect-deletion set of size at most ℓ.

We note that the construction above is independent of k and approximation preserving due to Property 5. The reduction can easily be computed in polynomial time.

The FPT approximation algorithm for k-Hitting Set constructs the instance (G, k) from $(U, \mathcal{F}, k)$, and then applies $\mathcal{A}$ to (G, k). If $\mathcal{A}$ concludes that the minimum size of a perfect deletion set in G is larger than k, then by Property 5 we can conclude that a minimum hitting set is larger than k, and reject the instance.

Otherwise, let (T, χ, L) be the resulting tree perfect-decomposition of width $d \leq g(k) \cdot k$ output by the algorithm. To transform the decomposition of G into a hitting set, we distinguish two cases.

First, suppose not a single gadget vertex h is contained in a base component, that is, $\mathcal{G}_F \cap L = \emptyset$ for each set gadget $\mathcal{G}_F$. Let $S \subseteq V(G)$ be a set that contains a single arbitrary vertex from each set gadget $\mathcal{G}_F$. Then $G[S]$ induces a clique by construction, and therefore there is a single bag $\chi(t)$ for some $t \in V(T)$ which contains all vertices of S. Since no gadget vertex is in a base component and so $S \cap L = \emptyset$, we have $|\chi(t) \setminus L| \geq |S|$. It follows that $d \geq |S| - 1 \geq |\mathcal{F}| - 1$. Hence we can simply take an arbitrary universe element from every set, and get a hitting set of size at most $d + 1 \leq 2 \cdot g(k) \cdot k$.

If the previous case does not apply, then there exists $F^* \in \mathcal{F}$ such that some $h \in \mathcal{G}_{F^*}$ is contained in $\chi(t) \cap L$ for a leaf $t \in V(T)$. Fix such a leaf t

and $h \in \mathcal{G}_{F^*} \cap (\chi(t) \setminus L)$, we construct a hitting set S for $(U, \mathcal{F})$ based on t as follows. For each $v \in \chi(t) \setminus L$, if $v = v_u \in X$ then add u to S. Otherwise $v \in \mathcal{G}_F$ is a gadget vertex and we add u to S for an arbitrary $u \in F$. Clearly $|S| \leq |\chi(t) \setminus L| \leq d+1$. To see that S is a valid hitting set, consider an arbitrary $F' \in \mathcal{F}$. Then either $h \in \mathcal{G}_{F'}$ or h is adjacent to every vertex of $\mathcal{G}_{F'}$ by the join step in the construction of G. Since $G[\mathcal{G}_{F'} \cup X_{F'}]$ induces an odd hole, at least one of its vertices u is outside the base component as the base component is perfect. Consider a shortest path P from h to u in $G[\mathcal{G}_{F'} \cup X_{F'} \cup \{h\}]$. Note that such a path must exist by the previous observation. Let w be the first vertex of P such that $w \notin L$. Note that $w \in \chi(t) \setminus L$ by Observation 6.3. Now since w is either a universe vertex or a gadget vertex, it follows that S contains an element that hits F'.

In both cases we have constructed a hitting set of size at most $d + 1 \leq 2 \cdot g(k) \cdot k$. But now we have an algorithm that either concludes that a minimum hitting set has size larger than k, or gives a hitting set of size at most $2 \cdot g(k) \cdot k$ in FPT time. Using Theorem 4.16, this implies W[1] = FPT. □

Lemma 4.17 leads to FPT-inapproximability results for perfect-TREEWIDTH and perfect-ELIMINATION DISTANCE, using the fact that a small perfect deletion set trivially results in witnesses of small treewidth and treedepth.

Theorem 4.18. *Assuming W[1] ≠ FPT, there is no FPT $g(k)$-approximation for* perfect-TREEWIDTH *or* perfect-ELIMINATION DISTANCE *for any computable function g.*

Proof. We show that an FPT-approximation for either problem would give an algorithm satisfying the conditions of Lemma 4.17 and therefore imply W[1] = FPT.

Suppose there is an FPT $g(k)$-approximation for perfect-TREEWIDTH for some computable function g. If, on input (G, k), the algorithm decides that $\mathbf{tw}_{\mathsf{perfect}}(G) > k$, then we can also conclude that the minimum size of a perfect deletion set of G is larger than k by Observation 3.10. Otherwise, it provides a witness X so that each connected component of $G - X$ is perfect and that $\mathbf{tw}(\mathbf{T}_G(X)) \leq g(k) \cdot k$. By constructing a tree decomposition of $\mathbf{T}_G(X)$ (following the proof of Lemma 3.9), we can construct a tree perfect-decomposition in FPT time.

Now suppose there is an FPT $g(k)$-approximation for perfect-ELIMINATION DISTANCE for some computable function g. If, on input (G, k), the algorithm decides that $\mathbf{ed}_{\mathsf{perfect}}(G) > k$, then we can also conclude that the minimum size of a perfect deletion set of the input graph is larger than k. Otherwise, it provides a witness X so that each connected component of $G - X$ is perfect

and that $\mathbf{td}(\mathbf{T}_G(X)) \leq g(k) \cdot k$. By Observation 3.10, X also witnesses that $\mathbf{tw}(\mathbf{T}_G(X)) \leq g(k) \cdot k$ and so we can proceed as in the previous case. □

In the above FPT-inapproximability results for perfect-TREEWIDTH and perfect-ELIMINATION DISTANCE, we strongly rely on the fact that instances constructed in the W[2]-hardness reduction from k-HITTING SET to k-PERFECT DELETION are very dense. While k-WHEEL-FREE DELETION is also W[2]-hard [130], which is shown via an approximation-preserving reduction, this does not directly lead to FPT-inapproximability of wheel-free-TREEWIDTH and wheel-free-ELIMINATION DISTANCE, as the instances produced by that reduction are much less dense. In particular, in such instances there is no direct way to translate a graph decomposition into a wheel-free deletion set. We therefore do not know whether wheel-free-TREEWIDTH and wheel-free-ELIMINATION DISTANCE admit FPT-approximations.

4.5 Conclusion

We have shown that $\mathcal{H}$-elimination distance and $\mathcal{H}$-treewidth are non-uniformly fixed-parameter tractable for $\mathcal{H}$ being the class of bipartite graphs, and whenever $\mathcal{H}$ is defined by a finite set of forbidden induced subgraphs. While the algorithms presented here solve the decision variant of the problem, by self-reduction they can be used to identify a witness if one exists [71]. The main observation driving such a self-reduction is the following: if $\mathbf{tw}_{\mathcal{H}}(G) \leq k$, then for an arbitrary $v \in V(G)$ there exists a $\mathbf{tw}_{\mathcal{H}}(G)$-witness that contains v if and only the graph G' obtained from G by inserting a minimal forbidden induced subgraph into $\mathcal{H}$ and identifying one of its vertices with v, still satisfies $\mathbf{tw}_{\mathcal{H}}(G') \leq k$. An iterative process, where we continue with G' in case the answer remains yes and continue with G otherwise, can identify all vertices of a witness in this way.

While we have focused on the established notions of $\mathbf{tw}_{\mathcal{H}}$ and $\mathbf{ed}_{\mathcal{H}}$, the ideas presented here can be generalized using minor-closed graph parameters f other than treewidth and treedepth. As long as f can attain arbitrarily large values, implying its value on a clique grows with the size of the clique, and $\mathcal{H}$ is characterized by a finite set of forbidden induced subgraphs, we believe our approach can be generalized to answer questions of the form: does G have an $\mathcal{H}$-deletion set X for which $f(\mathbf{T}_G(X)) \leq k$?

Finally we have seen hardness results regarding the computation of hybrid parameters for the class of perfect graphs. Unless FPT = W[1], not only is there no FPT algorithm that computes them exactly, there is even no FPT-

approximation for it. It would be interesting to see if there is a dichotomy to be found here. Are there classes for which computing these hybrid parameters exactly is hard, but approximations can be achieved?

Chapter 5

Constructing Tree $\mathcal{H}$-Decompositions

5.1 Introduction

As mentioned in Chapter 3, if $\mathcal{H}$-Deletion is FPT parameterized by solution size, then in many cases the same is true for the parameterization by $\mathbf{tw}_{\mathcal{H}}$ (at least in the non-uniform setting). We aim to understand this connection on a fine-grained level. If the parameterizations by solution size and treewidth can be solved in single-exponential time, does it imply the same for the hybrid parameterization? The main difficulty in resolving this question for concrete cases of $\mathcal{H}$-Deletion such as Odd Cycle Transversal lies in understanding the time needed to compute a suitable tree $\mathcal{H}$-decomposition. The mentioned meta-theorem [3] shows that this can be done in time $f(k) \cdot n^{\mathcal{O}(1)}$ for some unspecified function f. In earlier work, FPT approximations were developed [110] for computing the hybrid measures for graph classes satisfying certain conditions. More precisely, when the $\mathcal{H}$-treewidth/$\mathcal{H}$-elimination distance of the input graph G is at most k then the algorithms run in time $f(k) \cdot n^{\mathcal{O}(1)}$ and output a corresponding decomposition of width/depth $k^{\mathcal{O}(1)}$. In most cases, the dependence on the parameter can be described as $f(k) = 2^{k^{\mathcal{O}(1)}}$, which is a significant improvement compared to the non-uniform bounds of the meta-theorem, but is in many cases worse than the parameter dependence of the best-known algorithm parameterized by solution size. In this chapter we give FPT algorithms to compute tree $\mathcal{H}$-decompositions of approximately optimal width. We state our main result, where we assume oracle access to an algorithm

for $\mathcal{H}$-Deletion which, given a graph G and parameter k, either outputs a minimum $\mathcal{H}$-modulator or concludes that no such set of size at most k exists. In the complexity analysis we do not count the resources spent within the oracle calls. Instead, we specify the number of calls to the oracle to simplify applications of Theorem 5.1 for concrete graph classes $\mathcal{H}$.

Theorem 5.1. *Let $\mathcal{H}$ be a hereditary and union-closed class of graphs. There exists an algorithm that, using oracle-access to an algorithm $\mathcal{A}$ for $\mathcal{H}$-Deletion, takes as input an n-vertex m-edge graph G and integer k, runs in time $\mathcal{O}(3^{3k} \cdot kn^2(n+m))$ and polynomial space, makes at most $3^{3k+1} \cdot n^2$ calls to $\mathcal{A}$ on induced subgraphs of G and parameter $k+1$, and either concludes that $\boldsymbol{tw}_{\mathcal{H}}(G) > k$ or outputs a tree $\mathcal{H}$-decomposition of width at most $8k+8$ with $\mathcal{O}(n)$ nodes.*

This significantly improves upon both the approximation guarantees and running times of the best known uniform algorithms for the problem of approximating $\mathcal{H}$-treewidth [110]. In particular, we obtain a single-exponential $\mathcal{O}(1)$-approximation for computing $\mathcal{H}$-treewidth for every class $\mathcal{H}$ for which $\mathcal{H}$-Deletion admits a single-exponential algorithm parameterized by the solution size. This applies to $\mathcal{H} \in \{\mathsf{bipartite}, \mathsf{interval}\}$, or any graph class $\mathcal{H}$ specified by a finite family of forbidden induced subgraphs. For $\mathcal{H} \in \{\mathsf{planar}, \mathsf{chordal}\}$ the running time becomes $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$.

The presented improvement is of special importance for the classes of bipartite and planar graphs. In both cases the parameter dependence for solving $\mathcal{H}$-Deletion when given a tree $\mathcal{H}$-decomposition of width t, as we will see in Chapter 6, is essentially the same as for parameterization by treewidth [111]: respectively $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$ and $2^{\mathcal{O}(t \log t)} \cdot n^{\mathcal{O}(1)}$. The latter running times are tight under ETH [48, 153]. Due to Theorem 5.1, these running times can also be achieved under parameterization by $\mathcal{H}$-treewidth even when no decomposition is provided in the input. These constitute the first natural problems with tight running times under a hybrid parameterization, and show that parameterized algorithms for hybrid parameterizations can be as fast as for their solution-size counterparts. The techniques we use to obtain Theorem 5.1 mainly consists of two parts, namely secluded samples and locally optimal solutions, which we introduce next.

Secluded sample. To discuss our work, the following terminology will be useful. A vertex set $S \subseteq V(G)$ or induced subgraph $G[S]$ of a graph G is said to be k-secluded if $|N_G(S)| \leq k$, that is, the number of vertices outside S which are adjacent to a vertex of S is bounded by k. The connection between $\mathcal{H}$-treewidth and k-secluded connected subgraphs lies in the following fact:

for a tree $\mathcal{H}$-decomposition (T, χ, L) of width $k-1$ of a graph G, each base component C (connected component of $G[L]$) is k-secluded.

In order to obtain our result, we introduce a notion which generalizes the algorithmic enumeration primitive that has been called the *firefighters lemma* [83]. In its original formulation [84, Lemma 3.1] it states that in an undirected graph G, the number of size-$(p+1)$ connected vertex sets C that contain a prescribed vertex r and for which $|N_G(C)| = q$ is bounded by $\binom{p+q}{q}$ and is therefore independent of the total size of the graph (recall Lemma 4.4). For us and many applications, the setting that $p \approx q$ will be the main interest. In that regime, the lemma has a simple proof by bounded-depth branching and says that the number of connected k-secluded subgraphs on k vertices containing a prescribed vertex is $\mathcal{O}(4^k)$. The firefighters lemma has been frequently applied in parameterized algorithms, often combined with the technique of recursive understanding to analyze so-called *unbreakable* graphs [3, 4, 78, 87, 91, 108, 135], but also in other settings [16, 83, 92] such as the study of Secluded Path problems [75].

Our generalization of the firefighters lemma deals with graphs in which a subset $X \subseteq V(G)$ of *terminal vertices* is chosen. Rather than enumerating k-secluded connected subgraphs containing r which consist of (at most) k vertices, we will be interested in k-secluded connected subgraphs of arbitrary size, but which contain at most k terminals. For k-secluded connected subgraphs containing r and more than k terminals, it will be useful to have a sample of k terminal vertices contained in them. We introduce a notion called *k-secluded sample* to formalize these ideas (see Figure 5.1 for an illustration) and prove that the size of such a sample can be bounded by $2^{\mathcal{O}(k)}$.

Definition 5.2. For a graph G, sets $X, S \subseteq V(G)$, and integers k, s, we say that a family $\mathcal{Y} \subseteq 2^X$ is an s-secluded k-sample of (X, S) if the following conditions hold.

1. Each set $Y \in \mathcal{Y}$ has at most k elements.
2. For every non-empty connected s-secluded set $C \subseteq V(G)$ satisfying $S \subseteq C$:
   (a) if $|C \cap X| \leq k$, then $C \cap X \in \mathcal{Y}$,
   (b) if $|C \cap X| > k$, then there exists $Y \in \mathcal{Y}$ such that $Y \subseteq C \cap X$ and $|Y| = k$.

An s-secluded k-sample of $(X, \emptyset)$ is abbreviated as an s-secluded k-sample of X. A related algorithm of Telle and Villanger [167] enumerates all minimal

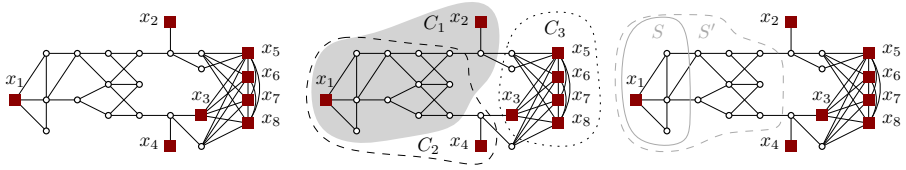


Figure 5.1: Illustration of the notion of s-secluded k-sample for $s = 3$ and $k = 2$. Left: a graph with vertex subset X drawn as squares. Middle: some 3-secluded connected subgraphs C_1, C_2, C_3, intersecting X in different subsets. To cover these subgraphs, it suffices for a sample to contain (respectively) sets $\{x_1, x_2\}$, $\{x_1, x_4\}$, and $\{x_3, x_5\}$. Right: For any s-secluded connected subgraph C containing S, the set $C \cup S'$ is also connected and s-secluded while having the same intersection with X. Hence s-secluded k-samples of (X, S) and (X, S') are identical notions.

subsets connecting a set of terminals, however their bound crucially is exponential in the number of vertices. We present an algorithm for enumerating a secluded sample which is single-exponential in $s + k$. Its analysis entails an upper bound on the size of such a family. The definition of s-secluded k-sample is chosen to facilitate this single-exponential bound, and this is the reason we work with a sample of up to k terminals contained in s-secluded subgraphs, rather than asking to enumerate all possible intersections of the terminal set with a connected s-secluded subgraph. If G is a clique and all its vertices are terminals, each way of omitting s vertices yields a maximal connected s-secluded subgraph spanning a distinct subset of terminals, of which there are $\binom{n}{s} \approx n^s$. The current formulation allows for an FPT-type bound, while still being sufficient for a range of applications. The following theorem captures our result for computing an s-secluded k-sample.

Theorem 5.3. *For an n-vertex m-edge graph G, set $X \subseteq V(G)$, and integers k, s, there exists an s-secluded k-sample $\mathcal{Y}$ of X of size at most $3^{k+2s} \cdot n$. Furthermore, there is an algorithm that, given G, X, S, k, and s, runs in time $\mathcal{O}(3^{k+2s} \cdot s \cdot n(n+m))$ and enumerates such a sample $\mathcal{Y}$ in polynomial space. For a non-empty set $S \subseteq V(G)$, there exists an s-secluded k-sample $\mathcal{Y}$ of (X, S) of size at most 3^{k+2s}; in this case the running time becomes $\mathcal{O}(3^{k+2s} \cdot s(n+m))$.*

This theorem can be seen as a far-reaching generalization of the $\mathcal{O}(2^{k+s})$ upper bound on the number s-secluded subgraphs of size at most k which contain a fixed vertex given by the firefighters lemma. While this is a special case of Theorem 5.3 for $X = V(G)$, we show that secluded samples for arbitrary X have

to be larger, by providing an $\Omega(5.03^k)$ lower bound for $k = s$ (Theorem 5.24). This also gives a separation between secluded samples and important separators.

The opening step of our algorithm for Theorem 5.3 is to consider a more general problem in which we search for secluded sets containing all of S and none of T. We can initialize T to be the empty set, which means it does not restrict the choice of secluded set. By adding vertices to S or T in branching steps of the enumeration algorithm, the sets grow and the size of a minimum (S, T)-separator increases accordingly. The size of a minimum (S, T)-separator disjoint from S is an important progress measure for the algorithm: if it ever exceeds s, there can be no s-secluded set containing all of S and none of T and therefore the enumeration is finished.

The branching steps are informed by the farthest minimum (S, T)-separator, similarly as the enumeration algorithm for important separators [140, 40], but are significantly more involved because we have to keep track of terminals. A further distinctive feature of our algorithm is that the decision made by branching can be to add certain vertices to the set T, while the important-separator enumeration only branches by enriching S. A key step is to use submodularity to infer that a certain vertex set is contained in *all* (inclusion-maximal) secluded subgraphs under consideration when other branching steps are inapplicable. This branching scheme results in a single-exponential, polynomial-space algorithm. Using secluded samples we can obtain the main object we need to attain Theorem 5.1, namely k-locally irredundant solutions.

Locally optimal solutions. The parameterized complexity of *local search* has been investigated by several authors [14, 28, 49, 69, 96, 97, 141]. Unfortunately, this line of research has shown that parameterized local search is typically W[1]-hard in general graphs. For example, given a graph G, a vertex cover $X \subseteq V(G)$, and a parameter k that governs the size of the local search neighborhood, it is W[1]-hard [69, Theorem 8] to determine whether there is a vertex cover which is smaller than X and has Hamming distance at most k from X. Similarly, it is known that checking whether a TSP tour in a weighted undirected graph G can be improved by replacing at most k edges is W[1]-hard [141]. The complexity decreases on graphs of bounded local treewidth, where local search for Vertex Cover and several other problems becomes fixed-parameter tractable [69, §5]. However, using our generalization of the firefighters lemma, we obtain FPT algorithms for a type of local search in *general graphs* for $\mathcal{H}$-Deletion problems. In the end we use such a locally optimal $\mathcal{H}$-deletion set to guide the approximation of $\mathcal{H}$-treewidth.

Suppose that $\mathcal{H}$ is hereditary and closed under taking disjoint unions, and

suppose that G has a connected induced subgraph C such that $C \in \mathcal{H}$. In this case, an optimal $\mathcal{H}$-modulator contains at most $|N_G(C)|$ vertices from $N_G[C]$, as otherwise it could be improved by replacing $X \cap N_G[C]$ by $N_G(C)$ to obtain another $\mathcal{H}$-modulator X': the resulting component C of $G - X'$ belongs to $\mathcal{H}$ by assumption, while the remaining components of $G - X'$ are induced subgraphs of those in $G - X$ and therefore belong to $\mathcal{H}$ since it is hereditary. Hence any $\mathcal{H}$-modulator X that contains more than $|N_G(C)|$ vertices from the set $N_G[C]$ is redundant. Conversely, we say that an $\mathcal{H}$-modulator X is k-locally irredundant if for every k-secluded connected induced subgraph C that belongs to $\mathcal{H}$ we have $|N_G[C] \cap X| \leq |N_G(C)|$.

Using our extension of the firefighters lemma we can show that, whenever $\mathcal{H}$-Deletion is FPT parameterized by solution size, one can compute a k-locally irredundant $\mathcal{H}$-modulator in FPT time.

Theorem 5.4. *Let $\mathcal{H}$ be a hereditary and union-closed class of graphs. There exists an algorithm that, using oracle-access to an algorithm $\mathcal{A}$ for $\mathcal{H}$-Deletion, takes as input an n-vertex m-edge graph G, integer k, and computes a k-locally irredundant $\mathcal{H}$-modulator in G in time $\mathcal{O}(3^{3k} \cdot kn^2(n+m))$ and polynomial space, making at most $3^{3k+1} \cdot n^2$ calls to $\mathcal{A}$ on induced subgraphs of G and parameter k.*

Previously computing $\mathcal{H}$-treewidth was considered to be conceptually difficult because the particular structure of the graph class has to be taken into account when deciding which parts of the graph are allowed as base components. Once you have a k-locally irredundant solution X for $\mathcal{H}$-Deletion however, a related subproblem can be solved that finds a tree decomposition that only 'pays' for the vertices in X. Any set of vertices disjoint from X can be part of a base component, meaning the resulting decomposition can be turned into a tree $\mathcal{H}$-decomposition. An optimal solution for this subproblem is a 2-approximation for $\mathcal{H}$-treewidth, which together with the adapted 4-approximation for treewidth ([157] cf. [48, Thm 7.18]) gives the factor 8-approximation of Theorem 5.1.

Organization. The remainder of this chapter is organized as follows. We present formal preliminaries in Section 5.2. In Section 5.3 we present the algorithm for computing a secluded sample and the simple application to k-locally irredundant modulators. The main application to hybrid graph decompositions is given in Section 5.4. We conclude in Section 5.5.

5.2 Preliminaries for constructing decompositions

In a graph G, a vertex set $C \subseteq V(G)$ is called k-secluded if $|N_G(C)| \leq k$. For future reference, we record the formal definition of a k-locally irredundant solution to $\mathcal{H}$-Deletion.

Definition 5.5. For a hereditary and union-closed graph class $\mathcal{H}$, an $\mathcal{H}$-modulator $X \subseteq V(G)$ is called k-locally irredundant if for every k-secluded connected set $C \subseteq V(G)$ for which $G[C] \in \mathcal{H}$ it holds that $|N_G[C] \cap X| \leq |N_G(C)|$.

The next observation, which follows from a simple exchange argument as described in Section 5.1, will be useful when arguing about k-locally irredundant solutions to $\mathcal{H}$-Deletion.

Observation 5.6. *Consider a hereditary and union-closed graph class $\mathcal{H}$. Let $X \subseteq V(G)$ be an $\mathcal{H}$-modulator in G and $C \subseteq V(G)$ satisfy $G[C] \in \mathcal{H}$. Then $(X \setminus C) \cup N_G(C)$ is also an $\mathcal{H}$-modulator in G.*

Observation 5.6 implies that any minimum-size $\mathcal{H}$-modulator must be k-locally irredundant for every value of k.

5.2.1 Graph decompositions

Instead of working with tree $\mathcal{H}$-decompositions directly, we introduce a slightly more general concept of a *based tree decomposition*. It is effectively equivalent to a tree $\mathcal{H}$-decomposition where $\mathcal{H}$ is the class of all graphs, so that one of the conditions is vacuously true. This definition will be convenient for presenting our proofs.

Definition 5.7. A based tree decomposition of graph G is a triple (T, χ, L) where $L \subseteq V(G)$, T is a rooted tree, and $\chi \colon V(T) \to 2^{V(G)}$, such that:

1. For each $v \in V(G)$ the nodes $\{t \in V(T) \mid v \in \chi(t)\}$ form a non-empty connected subtree of T.
2. For each edge $uv \in E(G)$ there is a node $t \in V(T)$ with $\{u, v\} \subseteq \chi(t)$.
3. For each vertex $v \in L$, there is a unique $t \in V(T)$ for which $v \in \chi(t)$, with t being a leaf of T.

The *based width* of (T, χ, L) is defined as $\max_{t \in V(T)} |\chi(t) \setminus L| - 1$. The connected components of $G[L]$ are called *base components* and the vertices in L are called *base vertices*.

When working with based tree decompositions, it will often be convenient to assume that the subgraphs $G[\chi(t) \cap L]$ are connected. A based tree decomposition which does not satisfy this condition can easily be transformed into one that does: create multiple copies of t, one for every connected component of $G[\chi(t) \cap L]$, and make them children to a common parent with a bag $\chi(t) \setminus L$.

Observation 5.8. *Let (T, χ, L) be a based tree decomposition of a graph G. There exists a based tree decomposition (T', χ', L) of the same based width such that for every node $t \in V(T')$ the graph $G[\chi(t) \cap L]$ is connected.*

Since the vertices in L are not counted towards the based width, the following notion allows is to identify vertices which must be accounted for in the width.

Definition 5.9. For a graph G and a set $X \subseteq V(G)$, the based treewidth of (G, X), denoted $\mathbf{btw}(G, X)$, is the minimum based width of a based tree decomposition (T, χ, L) of G for which $L \cap X = \emptyset$. The treewidth of G equals the based treewidth of $(G, V(G))$.

Similarly as treewidth, based treewidth is a monotone parameter with respect to taking induced subgraphs.

Observation 5.10. *Let G be a graph and $X, W \subseteq V(G)$. Then $\boldsymbol{btw}(G[W], X \cap W) \leq \boldsymbol{btw}(G, X)$.*

We can define $\mathcal{H}$-treewidth in the language of based tree decompositions, obtaining a definition equivalent to the one given in Definition 3.6.

Definition 5.11. For a hereditary graph class $\mathcal{H}$, a tree $\mathcal{H}$-decomposition of a graph G is a based tree decomposition (T, χ, L), for which every node $t \in V(T)$ satisfies $G[\chi(t) \cap L] \in \mathcal{H}$. The $\mathcal{H}$-treewidth of G, denoted $\mathbf{tw}_{\mathcal{H}}(G)$, is the minimum based width of a tree $\mathcal{H}$-decomposition of G.

For nodes t_1, t_2 of based tree decomposition, when t_1 is a leaf and t_2 is its parent, then the set $\chi(t_1) \cap \chi(t_2)$ is contained in the set of non-base vertices of t_1, that is, $\chi(t_1) \setminus L$. We make note of several simple, yet useful, corollaries that follow from Lemma 2.6 and Definition 5.7.

Observation 5.12. *Let (T, χ, L) be a based tree decomposition of a graph G.*

1. *For each node $t \in V(T)$ the set $\chi(t) \setminus L$ is an unrestricted $(\chi(t), V(G) \setminus \chi(t))$-separator.*
2. *For each node $t \in V(T)$ we have $N_G(\chi(t) \cap L) \subseteq \chi(t) \setminus L$. Consequently, the set $\chi(t) \cap L$ is $(k+1)$-secluded, where k denotes the based width of (T, χ, L).*
3. *For each set $L^* \subseteq L$ for which $G[L^*]$ is connected there is a unique node $t \in V(T)$ such that $L^* \subseteq \chi(t)$ while no vertex of L^* occurs in $\chi(t')$ for $t' \neq t$.*

5.2.2 Extremal separators and submodularity

Unrestricted, restricted, and left-restricted vertex separators were introduced in Chapter 2. Recall that the minimum size of a left-restricted (S,T)-separator is denoted by $\lambda^{\mathrm{L}}_G(S,T)$, which equals $+\infty$ if no such separator exists (which happens when $S \cap T \neq \emptyset$). Vertex separators can be computed by applying the Ford-Fulkerson method on vertex-capacitated flow networks. An introduction to the method is given in [44, Section 26.2].

Theorem 5.13 (cf. [48, Theorem 8.2])**.** *There is an algorithm that, given an n-vertex m-edge graph $G = (V, E)$, disjoint sets $S, T \subseteq V(G)$, and an integer k, runs in time $\mathcal{O}(k(n+m))$ and determines whether there exists a restricted (S,T)-separator of size at most k. If so, then the algorithm returns a separator of minimum size.*

By the following observation we can translate properties of restricted separators into properties of unrestricted or left-restricted separators.

Observation 5.14. *Let G be a graph and $S, T \subseteq V(G)$. Consider the graph G' obtained from G by adding a new vertex t adjacent to each $v \in T$. Then $P \subseteq V(G)$ is a left-restricted (S,T)-separator in G if and only if P is a restricted (S,t)-separator in G'.*

Furthermore, consider the graph G'' obtained from G' by adding a new vertex s adjacent to each $v \in S$. Then $P \subseteq V(G)$ is an unrestricted (S,T)-separator in G if and only if P is a restricted (s,t)-separator in G''.

The following submodularity property of the cardinality of the open neighborhood is well-known, see [166, §44.12] and [125, Footnote 3].

Lemma 5.15 (Submodularity)**.** *Let G be a graph and $A, B \subseteq V(G)$. Then the following holds:*

$$|N_G(A)| + |N_G(B)| \geq |N_G(A \cap B)| + |N_G(A \cup B)|.$$

Recall that $R_G(S, P)$ denotes the set of vertices which can be reached in $G - P$ from at least one vertex in the set $S \setminus P$.

Lemma 5.16. *Let G be a graph and $S, T \subseteq V(G)$ be two disjoint non-adjacent vertex sets. There exist minimum restricted (S,T)-separators P^- (closest) and P^+ (farthest), such that for each minimum restricted (S,T)-separator P, it holds that $R_G(S, P^-) \subseteq R_G(S, P) \subseteq R_G(S, P^+)$. Moreover, if a minimum restricted (S,T)-separator has size k, then P^- and P^+ can be identified in $\mathcal{O}(k(n+m))$ time.*

Proof. It is well-known (cf. [48, Thm. 8.5] for the edge-based variant of this statement, or [125, §3.2] for the same concept with slightly different terminology) that the existence of these separators follows from submodularity (Lemma 5.15), while they can be computed by analyzing the residual network when applying the Ford-Fulkerson algorithm to compute a minimum separator. We sketch the main ideas for completeness.

By merging S into a single vertex s^+ and merging T into a single vertex t^-, which is harmless because a restricted separator is disjoint from $S \cup T$, we may assume that S and T are singletons. Transform G into an edge-capacitated directed flow network D in which s^+ is the source and t^- is the sink. All remaining vertices $v \in V(G) \setminus (S \cup T)$ are split into two representatives v^-, v^+ connected by an arc (v^-, v^+) of capacity 1. For each edge $uv \in E(G)$ with $u, v \in V(G) \setminus \{s^+, t^-\}$ we add arcs $(u^+, v^-), (u^-, v^+)$ of capacity 2. For edges of the form s^+v we add an arc (s^+, v^-) of capacity 2 to D. Similarly, for edges of the form t^-v we add an arc (v^+, t^-) of capacity 2. Then the minimum size k of a restricted (S,T)-separator in G equals the maximum flow value in the constructed network, which can be computed by k rounds of the Ford-Fulkerson algorithm. Each round can be implemented to run in time $\mathcal{O}(n+m)$. From the state of the residual network when Ford-Fulkerson terminates we can extract P^- and P^+ as follows: the set P^- contains all vertices $v \in V(G) \setminus (S \cup T)$ for which the source can reach v^- but not v^+ in the final residual network. Similarly, P^+ contains all vertices $v \in V(G) \setminus (S \cup T)$ for which v^+ can reach the sink but v^- cannot. □

By Observation 5.14, we can apply the lemma above for left-restricted separators too: when the sets S, T are disjoint, then S is non-adjacent to t in the graph obtained by adding a vertex t adjacent to every vertex in T.

We use the extremal separators identified in Lemma 5.16 to argue when adding a vertex to S or T increases the separator size. The following statement is not symmetric because we work with the non-symmetric notion of a left-restricted separator.

Lemma 5.17. *Let G be a graph, let S, T be disjoint vertex sets, and let P^- and P^+ be the closest and farthest minimum* left-restricted (S,T)*-separators. Then for any vertex $v \in V(G)$, the following holds:*

1. $\lambda^{\mathrm{L}}_G(S \cup \{v\}, T) > \lambda^{\mathrm{L}}_G(S,T)$ *if and only if* $v \in R_G(T, P^+) \cup P^+$.
2. $\lambda^{\mathrm{L}}_G(S, T \cup \{v\}) > \lambda^{\mathrm{L}}_G(S,T)$ *if and only if* $v \in R_G(S, P^-)$.

Proof. Adding a vertex to S or T can never decrease the separator size, so for both cases, the left-hand side is either equal to or strictly greater than the right-hand side.

(1). Observe that if $v \notin R_G(T, P^+) \cup P^+$, then P^+ is also a left-restricted $(S \cup \{v\}, T)$-separator which implies $\lambda^{\mathrm{L}}_G(S \cup \{v\}, T) = \lambda^{\mathrm{L}}_G(S,T)$. If $v \in T$, then (1) holds as $\lambda^{\mathrm{L}}_G(S \cup \{v\}, T) = +\infty$. Consider now $v \in (R_G(T, P^+) \cup P^+) \setminus T$. We argue that adding it to S increases the separator size. Assume for a contradiction that there exists a minimum left-restricted $(S \cup \{v\}, T)$-separator P of size at most $\lambda^{\mathrm{L}}_G(S,T) = |P^+|$. Note that since P is left-restricted, we have $v \notin P$. Observe that P is also a left-restricted (S,T)-separator. By Lemma 5.16 we have $R_G(S,P) \subseteq R_G(S, P^+)$. Since $v \in (R_G(T, P^+) \cup P^+) \setminus T$, it follows that $v \notin R_G(S,P)$. We do a case distinction on v to construct a path Q from v to T.

- In the case that $v \in P^+ \setminus T$, then since P^+ is a minimum separator it must be inclusion-minimal. Therefore, since $P^+ \setminus \{v\}$ is not an (S,T)-separator, it follows that v has a neighbor in $R_G(T, P^+)$ and so there is a path Q from v to T in the graph induced by $R_G(T, P^+) \cup \{v\}$ such that $V(Q) \cap P^+ = \{v\}$.
- In the case that $v \in R_G(T, P^+) \setminus T$, then by definition there is a path from v to T in the graph induced by $R_G(T, P^+)$.

Since P is a left-restricted $(S \cup \{v\}, T)$-separator and therefore $v \notin P$, it follows that P contains at least one vertex $u \in V(Q)$ that is not in $R_G(S, P^+) \cup P^+$. Let P' be the set of vertices adjacent to $R_G(S,P)$. Since all vertices of P' belong to P while $u \notin P'$, it follows that P' is a left-restricted (S,T)-separator that is strictly smaller than P, a contradiction to $|P| \leq \lambda^{\mathrm{L}}_G(S,T)$.

(2). If $v \notin R_G(S, P^-)$, then P^- is a left-restricted $(S, T \cup \{v\})$-separator as well which implies $\lambda^{\mathrm{L}}_G(S, T \cup \{v\}) = \lambda^{\mathrm{L}}_G(S,T)$. If $v \in R_G(S, P^-)$, suppose that there exists a minimum left-restricted $(S, T \cup \{v\})$-separator P of size $|P^-|$. Note

that $v \notin S$, as otherwise no such separator exists. Furthermore P is also a left-restricted (S,T)-separator. By Lemma 5.16 we have $R_G(S,P^-) \subseteq R_G(S,P)$. But since $v \notin R_G(S,P)$ we reach a contradiction as $R_G(S,P) \not\supseteq R_G(S,P^-)$. □

The following lemma captures the idea that if $\lambda^{\mathrm{L}}_G(S,T \cup Z) > \lambda^{\mathrm{L}}_G(S,T)$, then there is a single vertex from Z whose addition to T already increases the size of a minimum left-restricted (S,T)-separator.

Lemma 5.18. *Let G be a graph, $S \subseteq V(G)$, and $T, Z \subseteq V(G) \setminus S$. If there is no vertex $v \in Z$ such that $\lambda^{\mathrm{L}}_G(S,T \cup \{v\}) > \lambda^{\mathrm{L}}_G(S,T)$, then $\lambda^{\mathrm{L}}_G(S,T) = \lambda^{\mathrm{L}}_G(S,T \cup Z)$. Furthermore if $\lambda^{\mathrm{L}}_G(S,T) \leq s$, then in $\mathcal{O}(s(n+m))$ time we can either find such a vertex v or determine that $\lambda^{\mathrm{L}}_G(S,T) = \lambda^{\mathrm{L}}_G(S,T \cup Z)$.*

Proof. Let P^- be the minimum left-restricted (S,T)-separator which is closest to S. If for every $v \in Z$ the value of $\lambda^{\mathrm{L}}_G(S,T \cup \{v\})$ equals $\lambda^{\mathrm{L}}_G(S,T)$ then Lemma 5.17 implies that each $v \in Z$ lies outside $R_G(S,P^-)$ so $Z \cap R_G(S,P^-) = \emptyset$. Then P^- is a left-restricted $(S,T \cup Z)$-separator of size $\lambda^{\mathrm{L}}_G(S,T)$.

On the other hand, if there is a vertex $v \in Z$ for which $\lambda^{\mathrm{L}}_G(S,T \cup \{v\}) > \lambda^{\mathrm{L}}_G(S,T)$ then $v \in R_G(S,P^-)$. Hence, in order to detect such a vertex it suffices to compute the closest minimum left-restricted (S,T)-separator P^-, which can be done in time $\mathcal{O}(s(n+m))$ via Lemma 5.16. □

Finally, the last lemma of this section uses submodularity to argue that the neighborhood size of a vertex set C with $S \subseteq C \subseteq V(G) \setminus T$ does not increase when taking its union with the reachable set $R_G(S,P)$ with respect to a minimum left-restricted (S,T)-separator P.

Lemma 5.19. *If $P \subseteq V(G)$ is a minimum left-restricted (S,T)-separator in a graph G and $S' = R_G(S,P)$, then for any set C with $S \subseteq C \subseteq V(G) \setminus T$ we have $|N_G(C \cup S')| \leq |N_G(C)|$.*

Proof. Observe that since P is a minimum left-restricted (S,T)-separator, we have $|P| = \lambda^{\mathrm{L}}_G(S,T)$ and $P = N_G(S')$. We apply the submodular inequality to the sets C and S'.

$$|N_G(C)| + |N_G(S')| \geq |N_G(C \cup S')| + |N_G(C \cap S')| \geq |N_G(C \cup S')| + \lambda^{\mathrm{L}}_G(S,T).$$

Here the last step comes from the fact that $S \subseteq S' \subseteq V(G) \setminus T$ since it is the set reachable from S with respect to a left-restricted (S,T)-separator, so that $C \cap S'$ contains all of S and is disjoint from T. This implies that $N_G(C \cap S')$ is a left-restricted (S,T)-separator, so that $|N_G(C \cap S')| \geq \lambda^{\mathrm{L}}_G(S,T)$.

As $|N_G(S')| = |P| = \lambda^{\mathrm{L}}_G(S,T)$, cancelling these terms from both sides proves $|N_G(C)| \geq |N_G(C \cup S')|$ which completes the proof. □

5.3 Secluded samples

In this section we present our generalization of the firefighters lemma, together with a simple application to k-locally irredundant solutions. Before presenting our enumeration algorithm, we briefly explain how one can obtain a simpler version of Theorem 5.3 with worse guarantees on the size of a secluded sample and the running time.[1] A related algorithm has been given for the purpose of designing an FPT approximation for k-Subset Edge Separator [100, Lemma 4.3]. It can be adapted to compute an s-secluded k-sample of (X, S) for non-empty S.

First, let us focus only on Condition (2a) in Definition 5.2. To obtain a graph G' from G, add a new vertex x. For each vertex $v \in X$, insert a copy u_v in G' adjacent to v and x. Then for any s-secluded connected set C in G containing S and at most k vertices from X, observe that $N_G(C)$ together with $\{u_v \mid v \in C \cap X\}$ forms a restricted (S, x)-separator of size at most $k + s$ in G'. We can detect $C \cap X$ via important (restricted) separators (see Theorem 2.2): there is an important (S, x) separator P with $C' = R_G(S, P)$ of size at most $k+s$ such that $C \subseteq C'$. Note that by construction $|C' \cap X| \leq k+s$. We can add each subset of $C' \cap X$ of size at most k to the secluded sample, which ensures that $C \cap X$ is contained in it and therefore satisfies (2a). To also satisfy (2b) one can do the following: let $X = \{v_1, \ldots, v_{|X|}\}$ and $X_i = \{v_1, \ldots v_i\}$. Iterate the above procedure for each X_i ($i \in [|X|]$) instead of for X only. This construction yields an s-secluded k-sample of (X, S) of size $2^{\mathcal{O}(k+s)} \cdot n$ for non-empty S.

While the algorithm described above would be sufficient for our purpose, we give an algorithm whose running time has a better dependency on s, k, and n. Furthermore our algorithm results in a better size bound on the secluded sample, which is independent of n for the case of non-empty S.

5.3.1 Enumeration algorithm

We formulate a lemma which is slightly stronger than the statement of Theorem 5.3, with an additional set $T \subseteq X \setminus S$ to be provided in the input. Initially T will be set to $\emptyset$. The task is to compute an s-secluded k-sample of (X, S) with the relaxation that we only require Property 2 to hold for connected s-secluded sets $C \supseteq S$ satisfying $C \cap T = \emptyset$. This allows us to consider a recursive algorithm which might repeatedly add vertices from X to T, as long as this increases the size of a minimum left-restricted (S, T)-separator. This provides us with a convenient progress measure.

[1] We thank an anonymous reviewer for pointing out an improvement here.

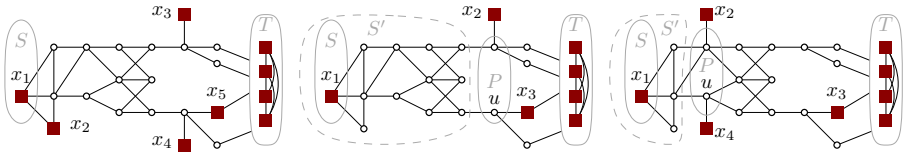


Figure 5.2: Illustration of the branching steps in Enum for $k = 2$ and $s = 3$ on three graphs with $\lambda^{\mathrm{L}}(S, T) = 2$. Vertices in X are drawn as squares. The set T is a clique of four vertices with identical closed neighborhoods. Left: Step 3 applies for $v = x_2$, since $\lambda^{\mathrm{L}}(S, T) = 2$ while $\lambda^{\mathrm{L}}(S, T \cup \{x_2\}) = 3$. Middle: Case 4(c)i applies for the chosen $u \in P$, since $\lambda^{\mathrm{L}}(S' \cup \{u\}, T) = \lambda^{\mathrm{L}}(S' \cup \{u\}, X \setminus S') = 3$. Right: Case 4(c)ii applies since $\lambda^{\mathrm{L}}(S' \cup \{u\}, T \cup \{x_4\}) = 3 > \lambda^{\mathrm{L}}(S' \cup \{u\}, T) = 2$; the former holds since a left-restricted separator is allowed to contain x_4 but not u.

Lemma 5.20. *Let G be an n-vertex m-edge graph and sets $X, S, T \subseteq V(G)$ satisfy $S \neq \emptyset$ and $T \subseteq X \setminus S$. For any integers k, s, there exists a family $\mathcal{Y} \subseteq 2^X$ of size at most 3^{k+2s} with the following properties.*

1. *Each set $Y \in \mathcal{Y}$ has at most k elements.*
2. *For every connected s-secluded set $C \subseteq V(G)$ with $S \subseteq C \subseteq V(G) \setminus T$:*
   - *(a) if $|C \cap X| \leq k$, then $C \cap X \in \mathcal{Y}$,*
   - *(b) if $|C \cap X| > k$, then there exists $Y \in \mathcal{Y}$ such that $Y \subseteq C \cap X$ and $|Y| = k$.*

Furthermore, there is an algorithm that, given G, X, S, T, k, and s, runs in time $\mathcal{O}(3^{k+2s} \cdot s(n+m))$ and enumerates such a family $\mathcal{Y}$ in polynomial space.

Proof. We begin by describing the algorithm. Algorithm $\mathsf{Enum}(G, X, S, T, k, s)$ solves the enumeration task as follows.

1. Stop the algorithm if one of the following holds:
   - (a) $\lambda^{\mathrm{L}}_G(S, T) > s$, or
   - (b) the vertices of S are not contained in a single connected component of G,

   No connected s-secluded set C satisfying the conditions imposed by S and T exists.

2. If one of the following holds:
   (a) the connected component of G containing S contains no vertex of $X \setminus (S \cup T)$, or
   (b) $|S \cap X| \geq k$,

   then output the set $S \cap X$ (or an arbitrary size-k subset of it, if $|S \cap X| > k$) and stop the algorithm.

   The given output always suffices. Note that if the algorithm continues, then $\lambda^{\mathrm{L}}_G(S, X \setminus S) > 0$ since S is non-empty.

3. If there is a vertex $v \in X \setminus (S \cup T)$ so that $\lambda^{\mathrm{L}}_G(S, T \cup \{v\}) > \lambda^{\mathrm{L}}_G(S, T)$:
   - Call $\mathsf{Enum}(G, X, S, T \cup \{v\}, k, s)$.
   - Call $\mathsf{Enum}(G, X, S \cup \{v\}, T, k, s)$.

   We exhaustively try both options: v either belongs to C or does not. Both recursive calls make progress since λ^{L} or $|S \cap X|$ increases.

4. Otherwise ($\lambda^{\mathrm{L}}_G(S, T) = \lambda^{\mathrm{L}}_G(S, X \setminus S) > 0$ by Lemma 5.18 for $Z = X \setminus (S \cup T)$) let P be the farthest minimum left-restricted $(S, X \setminus S)$-separator in G, pick an arbitrary $u \in P$ and let $S' = R_G(S, P)$:
   (a) Call $\mathsf{Enum}(G - u, X \setminus \{u\}, S', T \setminus \{u\}, k, s - 1)$.
   (b) If $u \in X \setminus T$, then call $\mathsf{Enum}(G, X, S' \cup \{u\}, T, k, s)$.
   (c) If $u \notin X$ and $\lambda^{\mathrm{L}}_G(S' \cup \{u\}, T) \leq s$:
      i. If $\lambda^{\mathrm{L}}_G(S' \cup \{u\}, T) = \lambda^{\mathrm{L}}_G(S' \cup \{u\}, X \setminus S')$:
         - Call $\mathsf{Enum}(G, X, S' \cup \{u\}, T, k, s)$.
      ii. Otherwise, let $v \in X \setminus (S \cup T)$ be such that $\lambda^{\mathrm{L}}_G(S' \cup \{u\}, T \cup \{v\}) > \lambda^{\mathrm{L}}_G(S' \cup \{u\}, T)$:
         - Call $\mathsf{Enum}(G, X, S' \cup \{u\}, T \cup \{v\}, k, s)$.
         - Call $\mathsf{Enum}(G, X, S' \cup \{u, v\}, T, k, s)$.

   Below we show that we can consider all of S' to be part of C, implying that either $u \in C$ or $u \in N_G(C)$ and so we do not branch on adding u to T. In the case that $u \notin X$, we need to do more careful branching to make sure the progress measure improves. In Steps 4a and 4b we make progress because s decreases or $S \cap X$ grows. Observe that $\lambda^{\mathrm{L}}(S, T) = \lambda^{\mathrm{L}}(S', T)$ and $X \cap S = X \cap S'$. In Step 4c, adding u to S' increases $\lambda^{\mathrm{L}}_G(S', X \setminus S)$ as $P \ni u$ is the farthest minimum separator. If $\lambda^{\mathrm{L}}_G(S' \cup \{u\}, T)$ grows with it, we make progress in Step 4(c)i. Otherwise, there is a vertex to make progress on in Step 4(c)ii identically to Step 3.

This concludes the description of the algorithm. The remainder of the proof is devoted to its analysis. We first establish some claims to justify that the sets output by the algorithm indeed satisfy the stated conditions. Afterwards we analyze the running time of the algorithm, which will also lead to an upper-bound on the number of sets given in the output.

The following claim effectively shows that when the algorithm reaches Step 4, replacing S by S' makes no difference for the sets to be enumerated.

Claim 5.21. *Let (G, X, S, T, k, s) be an instance such that the algorithm reaches Step 4, with corresponding definitions for P, u, and S'. Let C be an s-secluded set with $S \subseteq C \subseteq V(G) \setminus T$. Then $(C \cup S') \cap X = C \cap X$ and $|N_G(C \cup S')| \leq |N_G(C)|$.*

Proof. Observe that since P is a left-restricted $(S, X \setminus S)$-separator, we have that $P \cap S = \emptyset$ and so $S \subseteq S' = R_G(S, P)$. Furthermore $S' \cap X = S \cap X$ and so $(C \cup S') \cap X = C \cap X$, giving the first point. For the second point, we argue that P is a minimum left-restricted (S, T) separator. Since Step 3 is not applicable, this follows from Lemma 5.18 with $Z = X \setminus (S \cup T)$. Hence we have that $\lambda^{\mathrm{L}}_G(S, T) = \lambda^{\mathrm{L}}_G(S, X \setminus S) = |P|$. Consequently, Lemma 5.19 implies $|N_G(C \cup S')| \leq |N_G(C)|$ since $S' = R_G(S, P)$. ■

From the claim above, in Step 4 of the algorithm when trying to find the intersection $C \cap X$ to add on to the family, we may as well look for the intersection $(C \cup S') \cap X$. Since $P \subseteq N_G[C \cup S']$, this allows us to branch in only two directions for a vertex $u \in P$: one where we consider u to be in the neighborhood of the secluded component, and one where u is part of the secluded component. We move on to the full correctness proof.

Claim 5.22. *Let C^* be a connected s-secluded set in G with $S \subseteq C^* \subseteq V(G) \setminus T$. The output of* Enum *includes $C^* \cap X$ if its size is at most k. Otherwise, the output includes a size-k subset of $C^* \cap X$.*

Proof. Let C with $S \subseteq C \subseteq V(G) \setminus T$ be a set which is inclusion-maximal with respect to being connected, s-secluded, and having $C \cap X = C^* \cap X$. To prove the claim, it suffices to argue that the algorithm outputs a sufficiently large subset of $C \cap X$.

We prove the claim by induction on $(k - |S \cap X|) + s + (s - \lambda^{\mathrm{L}}_G(S, T))$. In the induction step we will argue that at least one of the three summands drops. Consider an instance (G, X, S, T, k, s).

Observe that since C is s-secluded, we must have $\lambda^{\mathrm{L}}_G(S, T) \leq s$ and so the algorithm can safely stop in Step 1a if $\lambda^{\mathrm{L}}_G(S, T) > s$. Since C is connected, we have that S (and C) are part of the same connected component in G. The

algorithm therefore correctly stops in Step 1b. In the case that the connected component of G containing S has no vertex of $X \setminus (S \cup T)$, then $C \cap X = S \cap X$ and the algorithm gives the correct output in Step 2 and can safely stop. Finally in the case that $|S \cap X| \geq k$, then the output in Step 2 already suffices for the desired output for $C \cap X$, again it can safely stop.

This covers the correctness of the base case, that is, all cases where either $|S \cap X| \geq k$ or $\lambda_G^{\mathrm{L}}(S,T) > s$. Observe that $s < 0$ implies the latter. Therefore, the base case includes all instances with induction measure up to and including zero. In the remainder, suppose that $|S \cap X| < k$ and $\lambda_G^{\mathrm{L}}(S,T) \leq s$. Note that this implies that $s \geq 0$ and therefore the induction measure is positive.

Suppose that the algorithm is correct for smaller values of the induction variable and continues past Step 2. Suppose Step 3 applies and consider the chosen vertex $v \in X \setminus (S \cup T)$ for which $\lambda_G^{\mathrm{L}}(S, T \cup \{v\}) > \lambda_G^{\mathrm{L}}(S,T)$. Observe that either $v \in C$ or $v \notin C$. In Step 3 we branch on these two cases by adding v to either S or to T. We argue that in both cases, the induction measure drops. In the case that we add v to S, then $|(S \cup \{v\}) \cap X| > |S \cap X|$, while $\lambda_G^{\mathrm{L}}(S \cup \{v\}, T) \geq \lambda_G^{\mathrm{L}}(S,T)$ and so the measure drops. In the case that we add v to T, then the measure drops since the precondition of Step 3 states that $\lambda_G^{\mathrm{L}}(S, T \cup \{v\}) > \lambda_G^{\mathrm{L}}(S,T)$. The correctness then follows by induction.

Now suppose that no such vertex v exists and the algorithm reaches Step 4. Due to Claim 5.21, we have $(C \cup S') \cap X = C \cap X$ and $|N_G(C \cup S')| \leq |N_G(C)|$, where $S' = R_G(S,P)$. Since C was picked inclusion-maximal with respect to being s-secluded and having a fixed intersection with X, it holds that $S' \subseteq C$. Observe that P is non-empty since the algorithm did not terminate in Step 2a. Since $P = N_G(S')$ and therefore $P \subseteq N_G[C]$, for any vertex $u \in P$, we must have that either u belongs to the secluded component C that we try to sample from, or to the at most s vertices in $N_G(C)$. If $u \in N_G(C)$, then C is $(s-1)$-secluded in $G - u$, which we find in Step 4a. We argue that the measure drops. Clearly $\lambda_G^{\mathrm{L}}(S,T)$ drops by at most one, while s decreases by one and S stays the same, which implies that the measure drops by at least one.

If $u \in C$, then we have that $u \notin T$. We do a case distinction on whether $u \in X \setminus T$ or $u \notin X$. First suppose that $u \in X \setminus T$. Then we sample from $C \cap X$ in Step 4b since $S' \cup \{u\} \subseteq C$, and the measure drops since $|(S' \cup \{u\}) \cap X| > |S' \cap X| \geq |S \cap X|$, while any left-restricted $(S' \cup \{u\}, T)$-separator is also a left-restricted (S,T)-separator.

Next suppose that $u \notin X$. We essentially do the same, we add u to S'. Since $S' \cup \{u\} \subseteq C$ we get that $\lambda_G^{\mathrm{L}}(S' \cup \{u\}, T) \leq s$ and so the precondition of Step 4c is satisfied (this condition is there to keep the running time in check). Note that since P was a farthest minimum $(S, X \setminus S)$-separator and $S \cap X = S' \cap X$, we get that $\lambda_G^{\mathrm{L}}(S' \cup \{u\}, X \setminus S') > \lambda_G^{\mathrm{L}}(S, X \setminus S)$ by Lemma 5.17.

If $\lambda^{\mathrm{L}}_G(S' \cup \{u\}, X \setminus S') = \lambda^{\mathrm{L}}_G(S' \cup \{u\}, T)$, then the algorithm in Step 4c solves the instance $(G, X, S' \cup \{u\}, T, k, s)$ as in Step 4b, whose measure drops as $\lambda^{\mathrm{L}}_G(S' \cup \{u\}, T) > \lambda^{\mathrm{L}}_G(S, T)$. This last inequality follows as $\lambda^{\mathrm{L}}_G(S, T) = \lambda^{\mathrm{L}}_G(S, X \setminus S) > 0$ by Lemma 5.18 for $Z = X \setminus (S \cup T)$. Otherwise, the contrapositive of the first point of Lemma 5.18 with $Z = X \setminus (S \cup T)$ implies there is a vertex $v \in X \setminus (S' \cup \{u\} \cup T) = X \setminus (S \cup T)$ such that $\lambda^{\mathrm{L}}_G(S' \cup \{u\}, T \cup \{v\}) > \lambda^{\mathrm{L}}_G(S' \cup \{u\}, T)$ (essentially showing that Step 3 would have implied in the next iteration, we unfold it for a tighter analysis). We find a valid output for $C \cap X$ by branching on both options for v: either $v \in C$ or not. The measure drops in both calls compared to the original instance. ■

Claim 5.23. *Algorithm* Enum *runs in time* $\mathcal{O}(3^{k+2s} \cdot s(n+m))$ *using polynomial space. It outputs at most* 3^{k+2s} *sets.*

Proof. First we analyze the running time without considering the time spent on recursive calls. Deciding if $\lambda^{\mathrm{L}}_G(S, T) > s$, as required in Step 1a, can be done in $\mathcal{O}(s(n + m))$ time by Theorem 5.13 and Observation 5.14.

Assuming we have direct access to vertices, with an overhead of $\mathcal{O}(n)$ we can assume each vertex has an attribute regarding its containment in X, S, and T. For Steps 1b and 2a, using BFS or DFS, in $\mathcal{O}(n + m)$ time we can find the connected components of G and check if only one component contains vertices from S and if so, if it contains vertices from $X \setminus (S \cup T)$. Finally for Step 2b and the output of Step 2, the set intersection can be computed and outputted in $\mathcal{O}(n)$ time.

If we reach Step 3, then we have $\lambda^{\mathrm{L}}_G(S, T) \leq s$. Therefore, according to Lemma 5.18 with $Z = X \setminus (S \cup T)$ in $\mathcal{O}(s(n + m))$ time we can decide if there exists a vertex $v \in X \setminus (S \cup T)$ such that $\lambda^{\mathrm{L}}_G(S, T \cup \{v\}) > \lambda^{\mathrm{L}}_G(S, T)$. For Step 4 we can compute the farthest left-restricted $(S, X \setminus S)$-separator in time $\mathcal{O}(s(n+m))$ by Lemma 5.16. Deciding if $\lambda^{\mathrm{L}}_G(S' \cup \{u\}, T) \leq s$ and subsequently if $\lambda^{\mathrm{L}}_G(S' \cup \{u\}, T) = \lambda^{\mathrm{L}}_G(S' \cup \{u\}, X \setminus S')$ can be done in $\mathcal{O}(s(n+m))$ time. Finally finding a vertex $v \in X \setminus (S \cup T)$ such that $\lambda^{\mathrm{L}}_G(S' \cup \{u\}, T \cup \{v\}) > \lambda^{\mathrm{L}}_G(S' \cup \{u\}, T)$ can be done in $\mathcal{O}(s(n + m))$ time by Lemma 5.18 with $Z = X \setminus (S \cup T)$ as before. In total this gives a running time of $\mathcal{O}(s(n + m))$ per node in the recursion tree.

For every recursive call, the measure $(k - |S \cap X|) + s + (s - \lambda^{\mathrm{L}}_G(S, T))$ drops by at least one as shown in the correctness proof in Claim 5.22. Since there are no more recursive calls when the measure hits zero (see Claim 5.22), and for each instance at most three recursive calls are made, it follows that the recursion tree is a ternary tree of depth at most $k+2s$. The number of leaves in such a tree is at most 3^{k+2s}, which yields an upper bound on the size of family

$\mathcal{Y}$ as the output is only generated in the leaf nodes. The total number of nodes in the recursion tree is at most the number of nodes in a perfect ternary tree of depth $k + 2s$, that is, $\frac{3}{2} \cdot 3^{k+2s}$. The claimed running time follows. As the recursion depth is polynomial and the work in one iteration takes polynomial space, the bound on space usage also follows. ■

This concludes the proof of Lemma 5.20. □

The lemma above gives the second claim of Theorem 5.3 by setting $T = \emptyset$. In order to obtain the first claim, it suffices to apply Lemma 5.20 multiple times, with $T = \emptyset$ and $S = \{v\}$ for each vertex $v \in V(G)$.

5.3.2 Lower bound

We complement Theorem 5.3 with a lower bound showing that for $k = s$ the number of elements in a k-secluded k-sample can be $\Omega(5.03^k)$. A motivation for this is to provide a separation between the necessary size of a k-secluded k-sample and an upper bound of 4^k feasible for (a) the number of important (s,t)-separators of size $\leq k$ and (b) the number of k-secluded subgraphs on at most k vertices in the firefighters lemma. We therefore do not optimize this construction further. The lower bound applies to any family that just meets Condition (2a) in Definition 5.2: such a family must contain all subsets $X' \subseteq X$ of size $\leq k$ for which there exists a k-secluded connected set C containing a specified vertex r and satisfying $C \cap X = X'$.

Theorem 5.24. *For each sufficiently large k divisible by 3, there exists a graph G with a vertex r and a vertex set $X \subseteq V(G)$ such that any k-secluded k-sample of $(X, \{r\})$ contains at $\Omega(5.03^k)$ elements.*

Proof. Recall that in a binary tree each node has at most two children. A binary tree is said to be full if each node has zero or two children and perfect if, additionally, it has a maximum number of nodes in all layers. Let $k = 3\ell$ and T be a perfect binary tree of depth $2\ell - 1$ rooted at vertex r (recall that the depth of a tree is measured in the number of edges in a longest root-to-leaf path). We obtain G from T by attaching to every vertex $v \in V(T)$ two degree-1 vertices $\bar{v}_0, \bar{v}_1$. Let $X = V(G) \setminus V(T)$. We construct a family $\mathcal{C} \subseteq 2^{V(G)}$ as follows. An example of the construction is given in Figure 5.3.

1. Choose a full binary subtree T' of T rooted at r with exactly 2ℓ leaves.
2. Choose a size-ℓ subset F of internal vertices of T'.
3. Choose a function $\phi\colon F \to \{0,1\}$.

4. Let S consist of the leaves of T' and all vertices of the form $\bar{v}_{\phi(v)}$ for $v \in F$.

5. Add $R_G(\{r\}, S)$ to $\mathcal{C}$.

We shall prove that every $C \in \mathcal{C}$ is k-secluded, the set $|C \cap X|$ has at most k elements, and the sets $C \cap X$ differ for different C. This implies that any k-secluded k-sample of $(X, \{r\})$ must contain at least $|\mathcal{C}|$ elements.

Let $C = R_G(\{r\}, S)$ for some S constructed as above. The set S contains 2ℓ elements which are the leaves of T' and ℓ elements of the form $\bar{v}_{\phi(v)}$ for $v \in F$. Hence, C is (3ℓ)-secluded. A full binary tree with 2ℓ leaves has exactly $2\ell - 1$ internal nodes. This gives $4\ell - 2$ vertices from X adjacent in G to the internal vertices of T'. Since $|F| = \ell$, we obtain exactly $3\ell - 2 < k$ elements from X in C.

Now we show that for different sets $C_1, C_2 \in \mathcal{C}$ it holds that $C_1 \cap X \neq C_2 \cap X$. Suppose that C_1, C_2 originate from different subtrees T_1', T_2' chosen at step (1). These subtrees have the same size so there exists a vertex $v \in V(T)$ which is an internal vertex in T_1' but not in T_2'. At least one of the vertices $\bar{v}_0, \bar{v}_1$ is present in C_1 while both are missing in C_2, hence $C_1 \cap X \neq C_2 \cap X$. If both C_1, C_2 originate from the same subtree T' then they differ in the choice of F or ϕ. This means that different size-ℓ sets of the form $\{\bar{v}_{\phi(v)}\}_{v \in F}$ are missing in C_1, C_2.

Finally, we estimate the size of $\mathcal{C}$. It is known that the number of (rooted and ordered) full binary trees with 2ℓ leaves equals the $(2\ell - 1)$-th Catalan number, that is, $\frac{1}{2\ell}\binom{4\ell-2}{2\ell-1}$.

In order to argue that this equals the number of choices for T', observe that every such tree has depth at most $2\ell - 1$ and so it appears as a rooted subtree of T. To see this, note that for any root-to-leaf path with h edges, each of the h internal vertices must have a second child and the subtree rooted at this child has at least one leaf. This implies that a full binary tree of depth h has at least $h + 1$ leaves. The number of choices for F is $\binom{2\ell-1}{\ell}$ and the number of choices for ϕ is 2^ℓ.

Using the binomial theorem, for any integer $x > 0$ we have the following.

$$2^{2x} = (1+1)^{2x} = \sum_{i=0}^{2x} \binom{2x}{i} \leq (2x+1) \cdot \binom{2x}{x}$$

Here the inequality is due to the term $\binom{2x}{i}$ being maximized for $i = x$. This shows that $\binom{2x}{x} \geq \frac{2^{2x}}{2x+1}$. Similarly, $\binom{2x-1}{x} \geq \frac{2^{2x-1}}{2x}$. Using these bounds we can give a lower bound on the size of $\mathcal{C}$, which follows by multiplying the number of different choices for T', F, and ϕ.

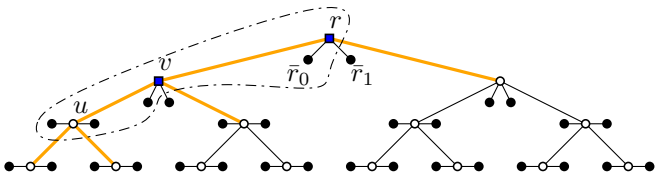


Figure 5.3: Illustration of the construction of Theorem 5.24 for $\ell = 2$. The black vertices denote the terminal set X. A choice for a full binary tree T' indicated in orange, and a choice for $F \subseteq \{u, v, r\}$ is indicated with blue squares. The dashed line indicates a set added to $\mathcal{C}$ from some choice of ϕ.

$$\begin{aligned}
\frac{1}{2\ell}\binom{4\ell-2}{2\ell-1} \cdot \binom{2\ell-1}{\ell} \cdot 2^\ell &\geq \frac{1}{2\ell} \cdot \frac{2^{4\ell-2}}{4\ell-1} \cdot \frac{2^{2\ell-1}}{2\ell} \cdot 2^\ell \\
&= \frac{2^{7\ell-3}}{16\ell^3 - 4\ell^2} \\
&= \frac{2^{7(k/3)}}{8(16(k/3)^3 - 4(k/3)^2)} && (\ell = k/3) \\
&\geq \frac{5.039^k}{128(k/3)^3 - 32(k/3)^2} && (2^{7/3} \geq 5.039)
\end{aligned}$$

For sufficiently large k, the denominator is clearly seen to be positive and upper bounded by c^k for any $c > 1$. It follows that $\mathcal{C}$ contains $\Omega(5.03^k)$ elements for sufficiently large k. □

5.3.3 Locally irredundant solutions

We apply the machinery of secluded samples to prove Theorem 5.4. To this end, we give an algorithm that performs a local improvement as long as the given solution is not k-locally irredundant.

In the following statement we assume oracle access to an algorithm $\mathcal{A}$ which outputs a minimum solution to $\mathcal{H}$-Deletion, if there is such a solution of size at most k. In the complexity analysis we do not count the resources spent within the oracle calls and only count time and space linear to the input and output of the oracle. As we are interested in a proper complexity analysis for concrete graph classes $\mathcal{H}$, we specify the number of calls to the oracle.

Lemma 5.25. *Let $\mathcal{H}$ be a hereditary and union-closed class of graphs. There exists an algorithm that, using oracle-access to an algorithm $\mathcal{A}$ for $\mathcal{H}$-Deletion, takes as input an n-vertex m-edge graph G, an $\mathcal{H}$-modulator X, and integer k, runs in time $\mathcal{O}(3^{3k} \cdot kn(n+m))$ and polynomial space, makes at most $3^{3k+1} \cdot n$ calls to $\mathcal{A}$ on induced subgraphs of G and parameter k, and either correctly concludes that X is k-locally irredundant or outputs an $\mathcal{H}$-modulator strictly smaller than X.*

Proof. We invoke the algorithm from Theorem 5.3 to enumerate a k-secluded $(k+1)$-sample $\mathcal{Y}$ of X of size at most $3^{3k+1} \cdot n$. For each $Y \in \mathcal{Y}$ we execute the algorithm $\mathcal{A}$ on the graph $G - (X \setminus Y)$ with parameter $|Y| - 1 \leq k$. If it returns some $\mathcal{H}$-modulator Y' in $G - (X \setminus Y)$ of size less than $|Y|$, then $X' = (X \setminus Y) \cup Y'$ is an $\mathcal{H}$-modulator in G smaller than X and we can then return X'. If we do not detect any such improvement while checking all $Y \in \mathcal{Y}$, we report that X is k-locally irredundant.

Suppose that X is not k-locally irredundant. We claim that then the algorithm above must return some smaller $\mathcal{H}$-modulator. By definition, there exists a k-secluded connected set $C \subseteq V(G)$ such that $G[C] \in \mathcal{H}$ and $|N_G[C] \cap X| > |N_G(C)|$. The last inequality is equivalent to $|C \cap X| > |N_G(C) \setminus X|$. A k-secluded $(k+1)$-sample of X must contain some set $Y \subseteq C \cap X$ of size $\min(|C \cap X|, k+1)$. From Observation 5.6 we know that $(X \setminus Y) \cup N_G(C)$ is also an $\mathcal{H}$-modulator. We argue that $|N_G(C) \setminus X| < |Y|$. In the case that $Y = C \cap Y$, then this follows from the inequality above. Otherwise, if Y is a size-$k+1$ subset of $C \cap X$, then this follows from the fact that C is k-secluded and so $|N_G(C)| \leq k$. From $|N_G(C) \setminus X| < |Y|$ we infer that the graph $G - (X \setminus Y)$ has an $\mathcal{H}$-modulator of size at most $|Y| - 1$ and the algorithm $\mathcal{A}$ invoked for $G - (X \setminus Y)$ returns some solution Y' smaller than Y. This concludes the proof of correctness.

The algorithm from Theorem 5.3 runs in time $3^{3k} \cdot kn(n+m)$ and polynomial space. We do not store the family $\mathcal{Y}$ but rather process each element of $\mathcal{Y}$ once it is revealed, leading to only polynomial space usage. For each of the $3^{3k+1} \cdot n$ sets in the family $\mathcal{Y}$ we invoke the algorithm $\mathcal{A}$ on an induced subgraph of G with parameter at most k. □

Starting from the trivial solution $X = V(G)$, this process converges to a k-locally irredundant $\mathcal{H}$-modulator after at most n reduction steps. This proves Theorem 5.4.

5.4 Approximating $\mathcal{H}$-treewidth

This section is devoted to proving Theorem 5.1. First we show that once one is equipped with a locally irredundant solution X for $\mathcal{H}$-Deletion, the task of approximating $\mathcal{H}$-treewidth of G reduces to approximating the based treewidth of (G, X).

Lemma 5.26. *Consider a graph class $\mathcal{H}$ which is hereditary and union-closed, a graph G of $\mathcal{H}$-treewidth at most k, and a $(k+1)$-locally irredundant $\mathcal{H}$-modulator $X \subseteq V(G)$. Then $\boldsymbol{btw}(G, X) \leq 2k + 1$.*

Proof. Let (T, χ, L) be a tree $\mathcal{H}$-decomposition of G of based width at most k. Hence for every node $t \in V(T)$ we have $G[\chi(t) \cap L] \in \mathcal{H}$ and $|\chi(t) \setminus L| \leq k + 1$. By Observation 5.8, we can assume that for every node $t \in V(T)$ the graph $G[\chi(t) \cap L]$ is connected. Let $L_t = \chi(t) \cap L$. By Observation 5.12, we have $N_G(L_t) \subseteq \chi(t) \setminus L$, and so $|N_G(L_t)| \leq k + 1$. Since X is $(k + 1)$-locally irredundant, it must hold that $|L_t \cap X| \leq k + 1$. Consider a new based tree decomposition: $(T, \chi, L \setminus X)$. Changing the last parameter might only affect Condition (3) in Definition 5.7 but since $L \setminus X \subseteq L$ this condition is preserved and $(T, \chi, L \setminus X)$ is indeed a valid based tree decomposition. For every node t we can bound the size of $\chi(t) \setminus (L \setminus X)$ by $|\chi(t) \setminus L| \leq k + 1$ plus $|(\chi(t) \cap L) \cap X| \leq k + 1$, which gives $2k + 2$ in total. The claim follows. □

Next, we are going to adapt the single-exponential approximation algorithm for treewidth [157] [48, Theorem 7.18] to work with a based tree decomposition.

Theorem 5.27. *There is an algorithm that, given an n-vertex m-edge graph G, set $X \subseteq V(G)$, and integer k, runs in time $\mathcal{O}(8^k \cdot kn(n + m))$ and polynomial space, and either outputs a based tree decomposition (T, χ, L) of G of width at most $4k + 4$, with $\mathcal{O}(n)$ nodes, and such that $L \cap X = \emptyset$ or correctly reports that $\boldsymbol{btw}(G, X) > k$.*

Before proving the theorem, we collect several properties of graph separations. A pair (A, B) of subsets from $V(G)$ is called a *separation* in G if $A \cup B = V(G)$ and G has no edges between $A \setminus B$ and $B \setminus A$. Recall that $\lambda_G(X, Y)$ stands for the size of a minimum unrestricted (X, Y)-separator in G.

Observation 5.28. *For two sets $X, Y \subseteq V(G)$, it holds that $\lambda_G(X, Y) \leq k$ if and only if there exists a corresponding separation (A, B) in $V(G)$ such that $X \subseteq A$, $Y \subseteq B$, and $|A \cap B| \leq k$.*

The following fact follows from Theorem 5.13 and Observation 5.14.

Corollary 5.29. *There is an algorithm that, given an n-vertex m-edge graph G, sets $X, Y \subseteq V(G)$, and integer k, runs in time $\mathcal{O}(k(n+m))$ and determines whether $\lambda_G(X,Y) \leq k$. If so, the algorithm also returns a corresponding separation in G.*

In a graph G of treewidth $\leq k$, for any set S of $3k+4$ vertices we can find a separation (A,B) of G with $|A \cap B| \leq k+1$ such that the sets $A \setminus B$ and $B \setminus A$ both contain at most $2k+2$ elements of S (see [48, Corollary 7.21]). This is not necessarily true if we only have a bound on based treewidth $\mathbf{btw}(G,X) \leq k$ because a large fraction of the set S might lie in a single base component in any optimal-width decomposition. But then the large fraction of S can be separated from X by the neighborhood of such a base component. We show that this is in fact the only scenario in which we cannot split S in a balanced way.

Lemma 5.30. *Let G be a graph, $X \subseteq V(G)$, and $k \geq 0$ be an integer. Suppose that $\boldsymbol{btw}(G,X) \leq k$. Let $S \subseteq V(G)$ be of size $3k+4$. Then one of the following holds:*

1. *there is a partition (S_A, S_B) of S such that $k+2 \leq |S_A| \leq |S_B| \leq 2k+2$ and $\lambda_G(S_A, S_B) \leq k+1$, or*
2. *there is a set $S' \subseteq S$ such that $k+2 \leq |S'|$ and $\lambda_G(S', X) \leq k+1$.*

Proof. Let (T, χ, L) be a based tree decomposition of G with $L \cap X = \emptyset$ of based width at most k. We consider two cases. First suppose that there exists a node $t \in V(T)$ for which $|\chi(t) \cap S| \geq k+2$. By Definition 5.7 each non-leaf bag of (T, χ, L) contains at most $k+1$ vertices, so t is a leaf in T and $|\chi(t) \setminus L| \leq k+1$. Note that $\chi(t) \cap X \subseteq \chi(t) \setminus L$. Let $S' = \chi(t) \cap S$. Then by Observation 5.12, $\chi(t) \setminus L$ is an unrestricted (S', X)-separator of size at most $k+1$ and so we have outcome (2).

In the rest of the proof we assume that for each $t \in V(T)$ we have $|S \cap \chi(t)| \leq k+1$, we show that we get outcome (1). Let $y \in V(T)$ be a node satisfying the following two conditions: (a) $|S \cap \chi(T_y)| \geq k+2$, (b) no child of y satisfies the first condition. Note that such a node exists because the root satisfies the condition (a). Next observe that, since the first case does not hold, y is not a leaf and therefore $|\chi(y)| \leq k+1$. As $|S| = 3k+4$, we have $|S \setminus \chi(T_y)| \leq 2k+2$.

If $|S \setminus \chi(T_y)| \geq k+2$, then by Lemma 2.6, the set $\chi(y)$ is an unrestricted $(S \cap \chi(T_y), S \setminus \chi(T_y))$-separator and we obtain outcome (1) by setting $S_A = S \cap \chi(T_y)$ and $S_B = S \setminus \chi(T_y)$. In the remainder, we have $|S \setminus \chi(T_y)| < k+2$. Let $D_1, \ldots, D_p$ denote the vertex sets of the connected components of $G - \chi(y)$ and let $a_i = |S \cap D_i|$ for $i \in [p]$. Since $|S \setminus \chi(T_y)| \leq k+1$ and $|S \cap \chi(T_t)| \leq k+1$

for every child t of y by (b), we have $a_i \leq k+1$ for each $i \in [p]$. Let $j \in [p]$ be the smallest index for which $\sum_{i=1}^{j} a_i \geq k+2$. Such an index exists because $\sum_{i=1}^{p} a_i \geq 3k+4-|\chi(y)| \geq 2k+3$. The upper bound on a_j implies that $\sum_{i=1}^{j} a_i \leq 2k+2$. We define a separation (A,B) of G as follows: $A = \bigcup_{i=1}^{j} D_i \cup \chi(y)$, $B = \bigcup_{i=j+1}^{p} D_i \cup \chi(y)$. It is clearly a valid separation and $A \cap B = \chi(y)$. We have $|S \setminus A|, |S \setminus B| \leq 2k+2$. We turn the separation into the desired partition of S, as follows. Start from the sets $S_A = (S \cap A) \setminus \chi(y)$ and $S_B = (S \cap B) \setminus \chi(y)$. Since $3k+4 \leq 2 \cdot (2k+2)$, we can greedily allocate vertices from $S \cap \chi(y)$ to the two sides of the partition so that each contains at most $2k+2$ vertices from S. Since the partition (S_A, S_B) of S is consistent with the separation (A,B) of order $k+1$, we have $\lambda_G(S_A, S_B) \leq k+1$ by Observation 5.28. □

We follow the proof of [48, Theorem 7.18] which gives a classic algorithm for approximating (standard) treewidth. The main difference is that due to scenario (2) in Lemma 5.30 we need to handle the basic instances differently whereas the recursive scheme stays the same. In this scenario we are able to separate at least $k+2$ vertices in S from the set X with a small separator. This allows us to create a base component which contains at least $k+2$ vertices from S and is disjoint from X, and then focus on the subproblem where S is significantly smaller.

Proof of Theorem 5.27. We shall provide an algorithm that recursively solves the following more general task in the time and space specified by the theorem. The theorem then follows by calling Decompose$(G, k, \emptyset, X)$.

Decompose(G, k, S, X)
Input: Graph G, integer k, sets $S, X \subseteq V(G)$ such that $|S| \leq 3k+3$.
Task: Output a set $L \subseteq V(G) \setminus (S \cup X)$ and a based tree decomposition (T, χ, L) of G of based width at most $4k+4$ such that S is contained in the root bag of the decomposition, or correctly report that $\mathbf{btw}(G, X) > k$.

First, if $|S \cup X| \leq 4k+5$ we simply set $L = V(G) \setminus (S \cup X)$ and return a tree decomposition consisting of a single node with a bag $V(G)$. We call such instances *basic*. Assume from now on that $|S \cup X| > 4k+5$, so in particular $|V(G)| > 4k+5 > 3k+3$. This allows us to choose a set $\widehat{S} \supsetneq S$ of size exactly $3k+4$ (the choice of $\widehat{S} \setminus S$ is arbitrary, this step is important only for the running time analysis). Now we can apply Lemma 5.30 with respect to graph G and sets $X, \widehat{S}$. We enumerate all subsets of $\widehat{S}$ to either find a partition (S_A, S_B) of $\widehat{S}$ satisfying condition (1) or a subset $S' \subseteq \widehat{S}$ satisfying

condition (2). In order to certify these conditions, for each such partition (S_A, S_B) and for each such S' we run Ford-Fulkerson (Corollary 5.29) to find an unrestricted (S_A, S_B)-separator of size at most $k+1$ if it exists, or an unrestricted (S', X)-separator if it exists. In both cases Corollary 5.29 also gives a corresponding separation in G. If we cannot find any such structure, we report that $\mathbf{btw}(G, X) > k$. We handle the two possible outcomes separately.

Balanced separation of $\widehat{S}$. Suppose that we have found a partition (S_A, S_B) of $\widehat{S}$ such that $k+2 \leq |S_A|, |S_B| \leq 2k+2$ and $\lambda_G(S_A, S_B) \leq k+1$. Let (A, B) be a corresponding separation of G with $S_A \subseteq A$, $S_B \subseteq B$, and $|A \cap B| \leq k+1$. Let $\widehat{S}_A = S_A \cup (A \cap B)$ and $\widehat{S}_B = S_B \cup (A \cap B)$. We have $|\widehat{S}_A|, |\widehat{S}_B| \leq 3k+3$. We create instances $(G[A], k, \widehat{S}_A, X \cap A)$ and $(G[B], k, \widehat{S}_B, X \cap B)$. Note that if the based treewidth of (G, X) is bounded by k then by Observation 5.10 so are $\mathbf{btw}(G[A], X \cap A)$ and $\mathbf{btw}(G[B], X \cap B)$. If any recursive call fails to build a decomposition we can report that $\mathbf{btw}(G, X) > k$. Suppose now that we have obtained the corresponding based tree decompositions (T_A, χ_A, L_A), (T_B, χ_B, L_B) of based width at most $4k+4$. Let us refer to their root nodes as respectively r_A, r_B. We have $\widehat{S}_i \subseteq \chi_i(r_i)$ for $i \in \{A, B\}$. Note that both L_A, L_B are disjoint from $\widehat{S} \cup X$. Let T be obtained by taking a disjoint union of T_A and T_B followed by creating a new root r with children r_A, r_B and χ be defined as follows: $\chi_{|T_A} = \chi_A$, $\chi_{|T_B} = \chi_B$ and $\chi(r) = \widehat{S} \cup (A \cap B) = \widehat{S}_A \cup \widehat{S}_B$. We have $|\chi(r)| \leq 4k+5$ so the width upper bound is preserved.

Claim 5.31. *The triple $(T, \chi, L_A \cup L_B)$ forms a valid based tree decomposition of G.*

Proof. We need to check the conditions of Definition 5.7. Let $v \in V(G)$: we show that $Y_v = \{t \in V(T) \mid t \in \chi(t)\}$ is non-empty and connected. If $v \in A \setminus \widehat{S}_A$, then $v \notin B$ and Y_v is a non-empty connected subtree of T_A. If $v \in \widehat{S}_A \setminus B$, then $Y_v \cap V(T_A)$ is connected and contains r_A, so adding r does not affect connectivity. After considering the symmetric cases, it remains to check $v \in \widehat{S}_A \cap \widehat{S}_B = A \cap B$. The set Y_v is then a union of $\{r\}$ and two connected subtrees, each containing a child of r, so it is connected in T.

Now consider an edge $uv \in E(G)$. Since (A, B) is a separation of G, it must be either $\{u, v\} \subseteq A$ or $\{u, v\} \subseteq B$ (or both). Hence, there exists a node $t \in V(T_A) \cup V(T_B)$ so that $\{u, v\} \subseteq \chi(t)$.

Finally, let $v \in L_A$. There is a unique leaf node $t_v \in V(T_A)$ for which $v \in \chi(t_v)$. By the specification of the subproblem, $L_A \subseteq A \setminus (\widehat{S}_A \cup X)$. Hence, $v \notin \chi(r)$ and $v \notin \chi(t)$ for any $t \in V(T_B)$, so t_v is the unique node in T whose

bag contains v; it remains a leaf after insertion of r. The case $v \in L_B$ is analogous. ■

Small (S', X)-separator. Now suppose that we have found a set $S' \subseteq \widehat{S}$ such that $k+2 \leq |S'|$ and $\lambda_G(S', X) \leq k+1$. Let (A, B) be a corresponding separation of G satisfying $S' \subseteq A$, $X \subseteq B$, $|A \cap B| \leq k+1$. Let $\widehat{S}_A = (\widehat{S} \cap A) \cup (A \cap B)$ and $\widehat{S}_B = (\widehat{S} \setminus A) \cup (A \cap B)$. We have $|\widehat{S}_A| \leq 4k+5$ and $|\widehat{S}_B| \leq 3k+3$. We create instances $(G[A], k, \widehat{S}_A, X \cap A)$ and $(G[B], k, \widehat{S}_B, X \cap B)$. The first instance does not obey the upper bound on $\widehat{S}_A$ but since $X \cap A \subseteq \widehat{S}_A$, the instance $(G[A], k, \widehat{S}_A, X \cap A)$ is basic and therefore can be solved trivially. The second instance obeys the specification and can be solved recursively. The merging of the computed tree decompositions and its analysis is the same as in the previous case.

A single recursive call takes time $\mathcal{O}(8^k \cdot k(n+m))$ due to 2^{3k+4} calls to the Ford-Fulkerson algorithm (Corollary 5.29). This requires only polynomial space. It remains to show that the number of recursive calls is $\mathcal{O}(n)$. For an instance (G, k, S, X) we argue that the total number of non-basic nodes in its recursion tree is at most $|V(G) \setminus S|$ by induction on the number of vertices in the graph. This holds trivially if the instance is basic. Otherwise we recurse into two instances $(G[A], k, \widehat{S}_A, X \cap A)$, $(G[B], k, B, \widehat{S}_B, X \cap B)$ for some separation (A, B) of G so that $\widehat{S} \cap A \subseteq \widehat{S}_A$ and $\widehat{S} \cap B \subseteq \widehat{S}_B$.

We first argue that both instances have strictly fewer vertices. In the case that (A, B) was obtained through a balanced separation of $\hat{S}$, then since $k+2 \leq |S_A|, |S_B| \leq 2k+2$ while $|A \cap B| \leq k+1$, we get that both A and B contain a vertex that is not in $A \cap B$. It follows that $G[A]$ and $G[B]$ contain strictly fewer vertices. In the case that (A, B) was obtained through a small (S', X)-separator, we have $|S'| \geq k+2$, $S' \subseteq A$, and $X \subseteq B$ with $|A \cap B| \leq k+1$. It follows that A contains at least one vertex not in $A \cap B$. Since $|X \cup S| > 4k+5$ while $|S| \leq 3k+3$, we get $|X| > k+2$ and so B contains at least one vertex not in $A \cap B$. Again it follows that $G[A]$ and $G[B]$ contain strictly fewer vertices. By induction on the number of vertices in the graph, the recursion trees of the two created instances have at most $|A \setminus \widehat{S}_A|$ and $|B \setminus \widehat{S}_B|$ non-basic nodes respectively.

Note that $(A \setminus \widehat{S}_A, B \setminus \widehat{S}_B)$ is a partition of a proper subset of $V(G) \setminus S$ because $\widehat{S} \supsetneq S$. It follows that the number of non-basic nodes in the recursion tree is at most $|A \setminus \widehat{S}_A| + |B \setminus \widehat{S}_B| + 1 \leq |V(G) \setminus S|$. Finally, either the root instance is basic or the number of basic nodes is at most twice the number of non-basic ones. The claim follows. □

Putting together the 2-approximation argument from Lemma 5.26 and the 4-approximation algorithm for based treewidth yields the main result of this section.

Theorem 5.1. *Let $\mathcal{H}$ be a hereditary and union-closed class of graphs. There exists an algorithm that, using oracle-access to an algorithm $\mathcal{A}$ for $\mathcal{H}$-Deletion, takes as input an n-vertex m-edge graph G and integer k, runs in time $\mathcal{O}(3^{3k} \cdot kn^2(n+m))$ and polynomial space, makes at most $3^{3k+1} \cdot n^2$ calls to $\mathcal{A}$ on induced subgraphs of G and parameter $k+1$, and either concludes that $\boldsymbol{tw}_{\mathcal{H}}(G) > k$ or outputs a tree $\mathcal{H}$-decomposition of width at most $8k+8$ with $\mathcal{O}(n)$ nodes.*

Proof. We begin with Theorem 5.4 to construct some $(k+1)$-locally irredundant $\mathcal{H}$-modulator $X \subseteq V(G)$ within the claimed complexity bounds. By Lemma 5.26 we know that if $\mathbf{tw}_{\mathcal{H}}(G) \leq k$ then $\mathbf{btw}(G, X) \leq 2k+1$. We execute the algorithm from Theorem 5.27 to find a based tree decomposition (T, χ, L) of G of based width at most $4 \cdot (2k+1) + 4 = 8k+8$, such that $L \cap X = \emptyset$. This step takes time $8^k \cdot kn(n+m)$. If the algorithm returns no decomposition and reports that $\mathbf{btw}(G, X) > 2k+1$ then we can also report that $\mathbf{tw}_{\mathcal{H}}(G) > k$. Otherwise, for every node $t \in V(T)$ it holds that $G[\chi(t) \cap L]$ is an induced subgraph of $G - X$ and, since X is an $\mathcal{H}$-modulator and $\mathcal{H}$ is hereditary, $G[\chi(t) \cap L]$ belongs to $\mathcal{H}$. Hence, (T, χ, L) is indeed a tree $\mathcal{H}$-decomposition. □

5.5 Conclusion

We have introduced a new algorithmic enumeration primitive based on secluded connected subgraphs generalizing the firefighters lemma. The high-level idea behind the algorithm is *enumeration via separation*: by introducing an artificial set T and considering the more general problem of enumerating secluded subgraphs containing S but disjoint from T, we can analyze the progress of the recursion in terms of the size of a minimum (left-restricted) (S, T)-separator. We expect this idea to be useful in scenarios beyond the ones studied here.

Our main application of the notion of secluded sample was the efficient computation of an 8-approximation to $\mathcal{H}$-treewidth, using an algorithm for the solution-size parameterization of $\mathcal{H}$-Deletion as a black box. To understand the relative power of the parameterizations by solution size, treewidth, and $\mathcal{H}$-treewidth, the remaining bottleneck consists of using the tree $\mathcal{H}$-decomposition to compute a minimum $\mathcal{H}$-modulator. Can the latter be done as efficiently when using a tree $\mathcal{H}$-decomposition as when using a standard tree decomposition? For some problems like Odd Cycle Transversal and Vertex Planarization,

this is indeed the case as we will see in Chapter 6. But when the current-best dynamic-programming algorithm over a tree decomposition uses advanced techniques such as *cut and count* [52] or the *rank-based approach* [21, 50, 146], it is currently not clear how to lift such an algorithm to work on a tree $\mathcal{H}$-decomposition.

We conclude with some directions for future research. The first concerns an extension of our techniques to *directed* graphs. We expect the branching step of our enumeration algorithms in terms of left-restricted minimum separators to be applicable in directed graphs. However, there are multiple ways to generalize the notion of a connected secluded induced subgraph to the directed setting: one can consider weak connectivity, strong connectivity, or a rooted variant where we consider all vertices reachable from a source vertex s. Similarly, one can define seclusion in terms of the number of in-neighbors, out-neighbors, or both. A thorough exploration of the design space and its algorithmic consequences is left for future work.

One aspect we have not considered here is that of above-guarantee parameterizations. For Multiway Cut, the problem remains single-exponential FPT [53, 95, 105, 106] when parameterized above the value of a fractional solution to the linear program (which forms a lower bound on solution size). Similarly, the bound on the number of important (S, T)-separators of size at most k can be refined from 4^k to $2^{2k-\lambda(S,T)}$, where $\lambda(S, T)$ is the size of a minimum (unrestricted) (S, T)-separator (cf. [48, Exercise 8.9], [131, Lemma 6]). Using similar ideas, we expect that the output guarantees of our enumeration algorithm concerning s-secluded subgraphs containing S but disjoint from T can be refined in terms of the difference $k + 2s - \lambda^{\mathrm{L}}_G(S, T)$. Since T starts as empty in our applications, we did not pursue this direction here.

A final direction concerns the optimization of the polynomial part of the running time. For standard treewidth, a 2-approximation can be computed in time $2^{\mathcal{O}(k)} \cdot n$ [123], which was obtained after a long series of improvements (cf. [23, Table 1]) on both the approximation factor and dependence on n. Can a constant-factor approximation to $\mathcal{H}$-treewidth be computed in time $2^{\mathcal{O}(k)} \cdot (n + m)$ for graph classes $\mathcal{H}$ like bipartite graphs?

Chapter 6

Solving Vertex-Deletion Problems with Tree $\mathcal{H}$-Decompositions

6.1 Introduction

The field of parameterized algorithmics [48, 60] develops fixed-parameter tractable (FPT) algorithms to solve NP-hard problems exactly, which are provably efficient on inputs whose parameter value is small. In order to extend the class of instances that remain tractable, it is worthwhile to find the smallest parameter for which these problems remain FPT. This chapter adds to this direction by giving algorithms for solving $\mathcal{H}$-Deletion problems under hybrid parameterizations, which are never larger than either the natural parameter (the smallest size of an $\mathcal{H}$-modulator) or structural graph measures like treewidth or treedepth. These vertex-deletion problems are among the most prominent problems studied in parameterized algorithmics [41, 81, 128, 137]. We extend or develop new algorithms for Odd Cycle Transversal, Vertex Planarization, Chordal Deletion, and H-free Deletion for fixed connected H, among others. The hybrid parameter we consider is $\mathcal{H}$-treewidth, background to which is given in Chapter 3. We obtain our algorithms by doing dynamic programming over tree $\mathcal{H}$-decompositions.

The content of this chapter is based on [110], which originally contained approximation algorithms for computing tree $\mathcal{H}$-decompositions and $\mathcal{H}$-elimination

$\mathcal{H}$	$f_{\mathcal{H}}(k)$	reference
bipartite	$2^{\mathcal{O}(k)}$	Corollary 6.14
minors (1)	$2^{k^{\mathcal{O}(1)}}$	Corollary 6.32
planar	$2^{\mathcal{O}(k \log k)}$	Corollary 6.32
subgraphs (2)	$2^{\mathcal{O}(k^{2c})}$	Corollary 6.39
chordal	$2^{\mathcal{O}(k^2)}$	Corollary 6.53
interval	$2^{\mathcal{O}(k^{96})}$	Corollary 6.72

Table 6.1: Parameter dependence on $k = \mathbf{tw}_{\mathcal{H}}$ for solving $\mathcal{H}$-Deletion for various classes $\mathcal{H}$. For (1), $\mathcal{H}$ is characterized by a finite set of connected forbidden minors (constants depending on these minors are hidden). For (2), $\mathcal{H}$ is characterized by a finite set of connected forbidden induced subgraphs on at most c vertices each.

forests, as well as algorithms using both of these decompositions to solve $\mathcal{H}$-Deletion problems. This chapter focuses on the latter, and uses the (better) approximation algorithms from Chapter 5 for computing tree $\mathcal{H}$-decompositions instead. Since $\mathbf{tw}_{\mathcal{H}}(G) \leq \mathbf{ed}_{\mathcal{H}}(G)$ (Observation 3.10), we restrict our attention to using tree $\mathcal{H}$-decompositions for solving $\mathcal{H}$-Deletion problems. This allows for a more concise and coherent overview of the techniques and results that are part of this line of work.

We present the main result of this chapter. By applying problem-specific insights to various problem-specific graph decompositions, we obtain FPT algorithms for vertex-deletion to $\mathcal{H}$ parameterized by $\mathbf{tw}_{\mathcal{H}}$.

Theorem 6.1. *Given a graph G and a tree $\mathcal{H}$-decomposition of G of width k, we can solve $\mathcal{H}$-Deletion in time $f_{\mathcal{H}}(k) \cdot n^{\mathcal{O}(1)}$ as specified in Table 6.1.*

While Theorem 5.1 yields decompositions for any hereditary and union-closed $\mathcal{H}$ for which $\mathcal{H}$-Deletion is FPT parameterized by solution size, the problem-specific adaptations needed to exploit the decompositions require significant technical work. We chose to focus on a number of key applications. We purposely restrict to the case where the forbidden induced subgraphs or minors are connected. With good reason, in Section 6.5 we show that $\mathcal{H}$-Deletion is NP-hard for some $\mathcal{H}$ that is not union-closed already in graphs with $\mathcal{H}$-elimination distance zero.

The requirement of Theorem 6.1 that a tree $\mathcal{H}$-decomposition needs to be part of the input can be overcome through the results of Chapter 5. By Theorem 5.1, it turns out that we can construct a tree $\mathcal{H}$-decomposition of width

$\mathcal{O}(k)$ with the same parameter dependence as in Theorem 6.1 (or even faster), resulting in the following corollary.

Corollary 6.2. *For all graph classes mentioned in Table 6.1, the* $\mathcal{H}$-Deletion *problem can be solved in time* $f_{\mathcal{H}}(k) \cdot n^{\mathcal{O}(1)}$ *when parameterized by* $k = \boldsymbol{tw}_{\mathcal{H}}$.

All our algorithms are uniform: a single algorithm works for all values of the parameter. As already mentioned in Chapter 3, due to the results of Agrawal et al. [3], for any hereditary union-closed graph class that is expressible in CMSO, $\mathcal{H}$-Deletion is non-uniform FPT parameterized by $\mathcal{H}$-treewidth whenever it is non-uniform FPT parameterized by solution-size. These conditions hold for all the classes mentioned in Table 6.1. Our contribution in Theorem 6.1 therefore lies in the development of *uniform* algorithms with concrete bounds on the running times. The parameter dependency for the classes of bipartite and planar graphs for solving $\mathcal{H}$-Deletion is even tight under SETH [48, Thm. 14.41] and ETH [153] respectively.

We consider the results an important step in the quest to identify the smallest problem parameters that can explain the tractability of inputs to NP-hard problems [70, 98, 150] (cf. [149, §6]). The corollary explains how, for example, classes of inputs for Odd Cycle Transversal whose solution size and rankwidth are both large, can nevertheless be solved efficiently and exactly.

Techniques. Intuitively, the use of decompositions in our algorithms is a strong generalization of the ubiquitous idea of splitting a computation on a graph into independent computations on its connected components. Even if the components are not fully independent but admit limited interaction between them, via small vertex sets, Theorem 6.1 exploits this algorithmically. We borrow from known techniques regarding dynamic programming over regular tree decompositions (see [48, §7.3] for an introduction). One common twist to our algorithms is that while base components may be arbitrarily large, we know that an optimal solution contains at most $k+1$ vertices from such components (Observation 6.5), so a subroutine based on the solution-size parameterization can be used to populate the entries of the dynamic programming table for the leaf bags. For the case of Odd Cycle Transversal, a straightforward extension of the known treewidth algorithm is sufficient. For the other graph classes, more elaborate ideas are needed.

Deferring technical details, these ideas are based on analyzing equivalence relations among t-boundaried graphs which can be glued along their boundary [10, 24, 26, 67]. For such graphs a subset of t vertices is labeled 1 through t and designated as the boundary of the graph. Given two t-boundaried graphs,

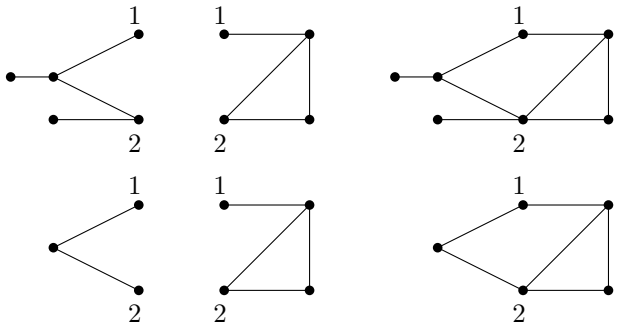


Figure 6.1: Top row shows two 2-boundaried graphs G_1 and G_3 and the graph $G_1 \oplus G_3$ obtained by gluing G_1 with G_3. The bottom row shows two 2-boundaried graphs G_2 and G_3 and the graph $G_2 \oplus G_3$ obtained from gluing G_2 with G_3. As Lemma 6.47 will show, the graphs G_1 and G_2 are equivalent with respect to membership to the class of chordal graphs.

a gluing operation identifies vertices with the same label of the boundaries to create a single graph. Two boundaried graphs G_1 and G_2 are equivalent with respect to membership in $\mathcal{H}$, if for each boundaried graph H the graph obtained from gluing G_1 and H is contained in $\mathcal{H}$ if and only if the same holds for G_2 glued with H. These notions are illustrated in Figure 6.1.

By bounding the number of equivalence classes resulting from this equivalence relation, and bounding the size of the smallest graph from each class, we can greatly reduce the number of states in the dynamic programming table. Most of the technical work lies in obtaining bounds on the size of a smallest representative of each equivalence class. Especially for chordal and interval graphs this requires new ideas. One important insight worth mentioning is the fact that our equivalence relation is based on 'membership in $\mathcal{H}$' rather than 'vertex deletion to $\mathcal{H}$'. While a definition of the latter type has finite integer index [24], which says something about the boundedness of a more involved equivalence relation that takes solution sizes into account, these equivalence classes would be more difficult to analyze [115]. Moreover, our bounds may be of interest beyond the analysis of vertex-deletion problems.

Organization. In Section 6.2 we give some additional preliminaries that are needed for this chapter. In Section 6.3 we show how an existing algorithm can be adapted to work with the parameterizations $\mathbf{tw}_{\mathcal{H}}$, leading to an algorithm solving Odd Cycle Transversal. In Section 6.4 we present a meta-algorithm

and apply it to several problems using new or known bounds on the sizes of minimal representatives of suitable equivalence classes. In Section 6.5 we briefly argue why the restriction to union-closed graph classes is necessary in our results. We conclude with a range of open problems and research directions in Section 6.6.

6.2 Preliminaries for solving vertex-deletion

When working with tree $\mathcal{H}$-decompositions, we will often exploit the following structure of base components that follows straight-forwardly from the definition. It is essentially equivalent to Observation 5.12.

Observation 6.3. *Let $\mathcal{H}$ be a graph class and let (T, χ, L) be a tree $\mathcal{H}$-decomposition of a graph G.*

- *For each set $L^* \subseteq L$ for which $G[L^*]$ is connected there is a unique node $t \in V(T)$ such that $L^* \subseteq \chi(t)$ while no vertex of L^* occurs in $\chi(t')$ for $t' \neq t$, and such that $N_G(L^*) \setminus L \subseteq \chi(t) \setminus L$.*
- *For each node $t \in V(T)$ we have $N_G(\chi(t) \cap L) \subseteq \chi(t) \setminus L$.*

The following easily follows from Observation 5.6.

Observation 6.4. *Suppose $\mathcal{H}$ is a hereditary and union-closed class of graphs, $X \subseteq V(G)$ is a minimum $\mathcal{H}$-deletion set in a graph G, and $Z \subseteq V(G)$ satisfies $G[Z] \in \mathcal{H}$. Then $|X \cap Z| \leq |N_G(Z)|$.*

It is particularly useful when we are guaranteed that Z has a small neighborhood. This is exactly the case for the base components of tree $\mathcal{H}$-decompositions: the base components of a tree $\mathcal{H}$-decomposition of width $k-1$ have a neighborhood of size at most k.

Observation 6.5. *Suppose $\mathcal{H}$ is a hereditary and union-closed class of graphs, $X \subseteq V(G)$ is a minimum $\mathcal{H}$-deletion set in G, and $Z \subseteq V(G)$ is a base component of a tree $\mathcal{H}$-decomposition of width $k-1$. Then $|X \cap Z| \leq k$.*

Nice decompositions. We now introduce a standardized form of tree $\mathcal{H}$-decompositions which is useful to streamline dynamic-programming algorithms. It generalizes the nice (standard) tree decompositions which are often used in the literature and were introduced by Kloks [122, Definition 13.1.4]. A similar construction has been introduced by Eiben et al. [62, Definition 8], however we treat the leaves in a slightly different way for our convenience.

Definition 6.6. Let $\mathcal{H}$ be a graph class. A tree $\mathcal{H}$-decomposition (T, χ, L) is *nice* if the tree T is rooted at a vertex r such that $\chi(r) \cap L = \emptyset$ and the following properties hold in addition to those of Definition 3.6:

1. Each node of T has at most two children.
2. If $t \in V(T)$ has two distinct children c_1, c_2, then $\chi(t) = \chi(c_1) = \chi(c_2)$. (This implies that $\chi(t) \cap L = \chi(c_1) \cap L = \chi(c_2) \cap L = \emptyset$.)
3. If node t has a single child c, then one of the following holds:
   (a) There is a vertex $v \in V(G) \setminus L$ such that $\chi(t) = \chi(c) \cup \{v\}$.
   (b) There is a vertex $v \in V(G) \setminus L$ such that $\chi(t) = \chi(c) \setminus \{v\}$.
   (c) The node c is a leaf and $\chi(t) = \chi(c) \setminus L$.

   We say that a standard tree decomposition (T, χ) is nice if $(T, \chi, \emptyset)$ satisfies all conditions above, and furthermore, for each node t with a single child we either have 3a or 3b.

As our approach to dynamic programming over a nice tree $\mathcal{H}$-decomposition handles all nodes with a single child in the same way, regardless of whether a vertex is being introduced or forgotten compared to its child, we do not use the commonly used terminology of *introduce* or *forget* nodes in this work. We also do not require the leaf nodes (and root) to be empty, which is typically the case for nice tree decompositions.

We are going to show that any tree $\mathcal{H}$-decomposition can efficiently be transformed into a nice one without increasing its width. The analogous construction for standard tree decompositions is well-known. The original algorithm of Kloks [122, Lemma 13.1.3] is an iterative algorithm based on a perfect elimination ordering of an appropriate chordal completion of the graph. Instead we use the algorithm as presented by Fomin et al. [82]. Their first step (Lemma 14.18) is to make the given tree decomposition simple, that is, for any two distinct nodes t_1 and t_2, neither $\chi(t_1) \subseteq \chi(t_2)$ nor $\chi(t_2) \subseteq \chi(t_1)$. This is achieved by iteratively contracting edges of the tree for which such a subset relation holds. Their next step (Lemma 14.23) is to make the simple tree decomposition nice by adding and reordering certain nodes, which is also hinted towards in the book of Cygan et al. [48, Hint 7.2].

Lemma 6.7. *Let $\mathcal{H}$ be a graph class. There is an algorithm that, given an n-vertex graph G and a tree $\mathcal{H}$-decomposition (T, χ, L) of G of width k, runs in time $(n + |V(T)|) \cdot k^{\mathcal{O}(1)}$ and outputs a* nice *tree $\mathcal{H}$-decomposition (T', χ', L) of G of width at most k satisfying $|V(T')| = \mathcal{O}(kn)$.*

Proof. Let $G_0 = G - L$ be the graph G without the base components. The tree $\mathcal{H}$-decomposition (T, χ, L) induces a (standard) tree decomposition (T_0, χ_0) of G_0 of the same width (take $T_0 = T$ and set $\chi_0(t) = \chi(t) \setminus L$ for each $t \in V(T_0)$). By going over the vertices of L we can assume each vertex has an attribute regarding its containment in L. Then obtaining (T_0, χ_0) takes $\mathcal{O}(n + k|V(T)|)$ time. We would like to turn (T_0, χ_0) into a nice decomposition and then append the base components back. However, we would like to remember the bags to which the base components are connected in order to reattach them later. For each node $t \in V(T_0)$ we keep a linked list of vertex sets corresponding to leaves of the original decomposition. Initially, for each leaf $t \in V(T)$ with $\chi(t) \cap L \neq \emptyset$ this linked list contains a copy of $\chi(t)$ and each internal node gets an empty linked list. The total volume of these copies is $\mathcal{O}(n + k|V(T)|)$ and the linked lists can be initialized in time bounded by this volume.

Apply [82, Lemma 14.18] to make (T_0, χ_0) simple in $\mathcal{O}(n + |V(T_0)|k^2)$ time, which keeps contracting an edge t_1t_2 for which either $\chi_0(t_1) \subseteq \chi_0(t_2)$ or $\chi_0(t_2) \subseteq \chi_0(t_1)$. The only additional step we need to take is that for every contraction, we merge the corresponding linked lists. Note that for each contraction of t_1t_2 into t_{12}, for every set $\chi(t')$ (which corresponds to a leaf bag in the original tree $\mathcal{H}$-decomposition) in the merged linked list of t_{12}, it holds that $\chi(t') \setminus L \subseteq \chi(t_{12})$. Let (T_0', χ_0') be the resulting simple tree decomposition. By [82, Lemma 14.19], we have that $|V(T_0')| \leq |V(G_0)| \leq n$.

Next we add back the base components to create a tree $\mathcal{H}$-decomposition (T_1', χ_1', L). Initially, let $T_1' = T_0'$ and $\chi_1'(t) = \chi_0'(t)$ for each $t \in V(T_0')$. For each node $t \in V(T_0')$, for each set $\chi(t')$ in its linked list, add two nodes t_a and t_b. Make t_a a child of t and set $\chi_1'(t_a) = \chi(t') \setminus L$. Make t_b a child of t_a and set $\chi_1'(t_b) = \chi(t')$. Note that since each node in L is in a single bag of the original tree, we add $\mathcal{O}(n)$ nodes this way.

Finally, apply the procedure of [82, Lemma 14.23] to make T' nice, first making the tree binary and adding duplicate bags to ensure that (1) and (2) hold. For nodes with a single child, note that we do not need to do anything for the edge between the newly added t_a and t_b, since these already satisfy (3c). For each remaining edge t_1t_2 with t_2 the single child of t_1, add nodes that forget one by one the vertices in $\chi_1'(t_1) \setminus \chi_1'(t_2)$, and then add nodes that one by one introduce nodes in $\chi_1'(t_2) \setminus \chi_1'(t_1)$. For each of these nodes, either (3a) or (3b) holds. This takes $n \cdot k^{\mathcal{O}(1)}$ time and results in a tree T' with $\mathcal{O}(kn)$ nodes. □

To exploit the separators which are encoded in a tree $\mathcal{H}$-decomposition, the following definition is useful.

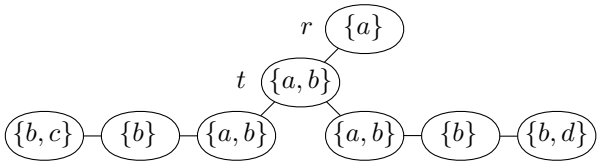


Figure 6.2: Sketch of a (nice) tree $\mathcal{H}$-decomposition $(T, \chi, \emptyset)$ of a graph with T rooted at r and the sets $\chi(v)$ displayed inside the nodes. We have $\pi(t) = \{a\}$ and $\kappa(t) = \{b, c, d\}$.

Definition 6.8. A *tri-separation* in a graph G is a partition (A, X, B) of $V(G)$ such that no vertex in A is adjacent to any vertex of B. The set X is the *separator* corresponding to the tri-separation. The *order* of the tri-separation is defined as $|X|$.

The following notions will be used to extract tri-separations from rooted tree decompositions, they are sketched in Figure 6.2. Recall that T_t denotes the subtree of T rooted at t.

Definition 6.9. Let $\mathcal{H}$ be a graph class and let (T, χ, L) be a (nice) tree $\mathcal{H}$-decomposition of a graph G, rooted at some node r. For each node $t \in V(T)$ of the decomposition, we define the functions $\pi_{T,\chi}, \kappa_{T,\chi} \colon V(T) \to 2^{V(G)}$:

1. For a non-root node t with parent s, we define $\pi_{T,\chi}(t) := \chi(s)$. For r we set $\pi_{T,\chi}(r) = \emptyset$.
2. For an arbitrary node t, we define $\kappa_{T,\chi}(t) := (\bigcup_{t' \in T_t} \chi(t')) \setminus \pi(t)$.

We omit the subscripts T, χ when they are clear from context.

Intuitively, $\pi(t)$ is the bag of the parent of t (if there is one) and $\kappa(t)$ consists of those vertices that occur in bags in the subtree T_t but not in the parent bag of t. In other words, the subtree $\chi^{-1}(v)$ of T associated with $v \in \kappa(t)$ is fully contained in T_t. The following observations about κ will be useful later on.

Observation 6.10. *If (T, χ, L) is a nice tree $\mathcal{H}$-decomposition of a graph G, then the following holds.*

1. *For the root r of the decomposition tree, $\kappa(r) = V(G)$.*
2. *If node t' is a child of node t, then $\kappa(t') \subseteq \kappa(t)$.*
3. *If c_1, c_2 are distinct children of a node t, then $\kappa(c_1) \cap \kappa(c_2) = \emptyset$.*

Tri-separations can be deduced from tree decompositions using κ and π, the following observation follows from Lemma 2.6.

Observation 6.11. *Let $\mathcal{H}$ be a graph class and let (T, χ, L) be a nice tree $\mathcal{H}$-decomposition of graph G, where T is rooted at some node r. Let $t \in V(T)$, let $A := \kappa(t)$ and let $X := \chi(t) \cap \pi(t)$. Then $(A, X, V(G) \setminus (A \cup X))$ is a tri-separation in G.*

6.3 Extending existing algorithms

In some cases one can extend existing algorithms for $\mathcal{H}$-Deletion that do dynamic programming on tree decompositions to use tree $\mathcal{H}$-decompositions instead. In most classical dynamic programming algorithms on tree decompositions, some trivial constant time computation is done in the leaves. In our setting we do more work in the leaves, without changing the dependency on the parameter. As an example, in this section we show how to obtain an FPT algorithm for the Bipartite Deletion problem when given a tree bip-decomposition. Here bip is the class of bipartite graphs. This problem is better known as the Odd Cycle Transversal (OCT) problem. Here we are given a graph G, and ask for the (size of a) minimum vertex set $S \subseteq V(G)$ such that $G - S$ is bipartite. Such a set S is called an odd cycle transversal. We introduce some more terminology. Recall that a graph is bipartite if and only if it admits a proper 2-coloring, this 2-coloring is referred to as a bipartition. For a bipartite graph G we say that $A, B \subseteq V(G)$ occur in opposite sides of a bipartition of G, if there is a proper 2-coloring $c\colon V(G) \to \{1, 2\}$ of G such that $A \subseteq c^{-1}(1)$ and $B \subseteq c^{-1}(2)$. Another characterization is that a graph is bipartite if and only if it does not contain an odd length cycle, explaining the name Odd Cycle Transversal. The following problem will be useful, it occurs as part of the iterative compression algorithm for OCT parameterized by solution size [48, Section 4.4].

Annotated Bipartite Coloring (ABC)
Input: A bipartite graph G, two sets $B_1, B_2 \subseteq V(G)$, and an integer k.
Task: Return a minimum-cardinality set $X \subseteq V(G)$ such that $G - X$ has a bipartition with $B_1 \setminus X$ and $B_2 \setminus X$ on opposite sides, or return $\perp$ if no such X exists of size at most k.

The sets B_1 and B_2 can be seen as precolored vertices that any coloring after deletion of some vertices should respect. The problem is solvable in polynomial time.

Lemma 6.12 ([48, Lemma 4.15])**.** Annotated Bipartite Coloring *can be solved in* $\mathcal{O}(k(n+m))$ *time.*

Fiorini et al. [72] give an explicit algorithm for computing the size of a minimum odd cycle transversal in a tree decomposition of width **tw** in time $\mathcal{O}(3^{3\mathbf{tw}} \cdot \mathbf{tw} \cdot n)$. We follow their algorithm to show that a similar strategy works when given a tree bip-decomposition. While the algorithm can be improved by making use of more properties of nice tree decompositions (see the discussion after the proof), we chose to only adjust the computation in the leaf bags for ease of presentation. For this reason most of the reasoning overlaps with [72], we include some of it for completeness.

Theorem 6.13. Odd Cycle Transversal *can be solved in time* $3^{3k} \cdot n^{\mathcal{O}(1)}$ *when given a tree* bip*-decomposition of width* $k-1$ *consisting of* $n^{\mathcal{O}(1)}$ *nodes.*

Proof. First, we apply Lemma 6.7 to turn the given tree bip-decomposition (T, χ, L) into a nice one. This takes time $\mathcal{O}(n + |V(T)| \cdot k^2)$ and generates a nice tree bip-decomposition of the same width with $\mathcal{O}(kn)$ nodes. In particular, we can assume that T is a binary tree.

Let $t^* \in V(T)$ be the root of T. Recall that T_t denotes the subtree of T rooted at t. For each $e = uv \in E(G)$, assign a specific node $t(e) \in V(T)$, such that $u, v \in \chi(t)$. Such a node exists by Definition 2.4. Let E_t be the set of edges assigned to $t \in V(T)$. Let us define graph $G(t)$ with vertex set $\chi(t)$ and edge set E_t, and the graph $G(T_t)$ with vertex set $\bigcup_{t' \in T_t} \chi(t')$ and edge set $\bigcup_{t' \in T_t} E_{t'}$. We associate with $t \in V(T)$ a set $\mathcal{A}_t$ of ordered triples $\Pi_t = (L_t, R_t, W_t)$, each of which forming a partition of $\chi(t) \setminus L$ into exactly three parts (some of which may be empty). The set L_t should not be confused with L (it is in fact disjoint from L), the notation is chosen to be consistent with previous work. The sets L_t and R_t should be seen as parts of the bag that end up in the 'left' and 'right' side of the bipartition. Note that $|\mathcal{A}_t|$ is at most 3^k.

The algorithm works from the leaves up, and for each partition Π_t stores the size of a minimum odd cycle transversal $\hat{W}_t$ in $G(T_t)$ such that $W_t \subseteq \hat{W}_t$ and L_t and R_t are in opposite color classes of $G(T_t) - \hat{W}_t$. This value is stored in $f(\Pi)$, and is infinity if no such transversal exists. If L_t or R_t is not an independent set in $G(t)$, set $f(\Pi_t) = \infty$. Now suppose that both L_t and R_t are independent sets in $G(t)$. For a non-leaf t with a child r, a partition of Π_r is said to be consistent with Π_t, denoted $\Pi_r \sim \Pi_t$, if $W_t \cap V(T_r) \subseteq W_r$, $L_t \cap V(T_r) \subseteq L_r$, and $R_t \cap V(T_r) \subseteq R_r$. Let r, s be children of t; if there is only one child, we omit the terms for s. We have

$$f(\Pi_t) = \min_{\Pi_r \sim \Pi_t, \Pi_s \sim \Pi_t} f(\Pi_r) + f(\Pi_s) + |W_t - (W_r \cup W_s)| - |W_r \cap W_s|$$

Since $\chi(t) \cap L = \emptyset$ for non-leaf nodes $t \in V(T)$, the correctness follows as in [72]. For a leaf $t \in V(T)$, we set

$$f(\Pi_t) = |W_t| + \textsc{abc}(G[\chi(t) \cap L], N(L_t) \cap \chi(t) \cap L, N(R_t) \cap \chi(t) \cap L, n)$$

Here ABC() denotes the result of an Annotated Bipartite Coloring instance, which can be solved in polynomial time by Lemma 6.12. The correctness in the leaves follows as in the iterative compression based algorithm for Odd Cycle Transversal (see [48, Section 4.4]).

The total work in the internal nodes remains $\mathcal{O}(k \cdot 3^{3k} \cdot n)$ as in [72]. The total work in the leaves takes $3^k \cdot n^{\mathcal{O}(1)}$ time. □

We note that the algorithm of Fiorini et al. [72] we adapted above does not have the best possible dependency on treewidth. The dependency on $\mathbf{tw}_{\mathsf{bip}}$ of our algorithm can be improved by exploiting more properties of nice tree $\mathcal{H}$-decompositions, to improve the exponential factor to 3^k. Similarly, the polynomial in the running time can be improved to $k^{\mathcal{O}(1)} \cdot (n + m)$. Two additional ideas are needed to achieve this speed-up. First of all, the computations for Annotated Bipartite Coloring in the leaf nodes can be cut off. By Observation 6.5, an optimal solution contains at most $k + 1$ vertices from any base component of a bip-decomposition of width k. Hence it suffices to set the budget for the ABC instance to $k+1$, and assign a table cell the value ∞ if no solution was found. While this causes some entries in the table to become ∞, the optimal solution and final answer are preserved. The second additional idea that is needed is an efficient data structure for testing adjacencies in graphs of bounded bip-treewidth, similarly as explained in [20]. We chose to present a simpler yet slower algorithm for ease of readability.

Since we can compute tree bip-decompositions approximately, we conclude with the following.

Corollary 6.14. *The* Odd Cycle Transversal *problem can be solved in time* $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ *when parameterized by* $k = \boldsymbol{tw}_{\mathsf{bip}}(G)$.

Proof. Since OCT can be solved in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ [159], by Theorem 5.1 it follows that we can compute a tree bip-decomposition of width $\mathcal{O}(k)$ in $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time. The result follows by supplying this decomposition to Theorem 6.13. The algorithm that computes the minimum size of an odd cycle transversal can be turned into an algorithm that constructs a minimum solution by standard techniques or self-reduction (see Lemma 2.8), in the same asymptotic time bounds. □

The parameter dependence is the same as the treewidth dependence in the result of Fiorini et al. [72]. Furthermore it is ETH-tight due to [48, Thm. 14.6] by observing that there is a reduction from Vertex Cover to Odd Cycle Transversal that starting from a graph G creates a graph G', with $\mathcal{O}(|V(G)|)$ vertices and $\mathcal{O}(|E(G)|)$ edges, by adding for each $uv \in E(G)$ a vertex x and edges ux and vx to G'.

6.4 Generic algorithms via A-exhaustive families

6.4.1 A-exhaustive families and boundaried graphs

In this section we introduce the main tools needed to present our most general framework for dynamic programming on tree $\mathcal{H}$-decompositions. We follow the ideas of gluing graphs and finite state property (also known as finite index) dating back to the results of Fellows and Langston [67] (cf. [10, 26]).

The following definition formalizes a key idea for the dynamic programming routine: to compute a restricted set of partial solutions out of which an optimal solution can always be built.

Definition 6.15. Let $\mathcal{H}$ be a graph class, let G be a graph, and let $A \subseteq V(G)$. Then we say that a family $\mathcal{S} \subseteq 2^A$ of subsets of A is A-exhaustive for $\mathcal{H}$-Deletion on G if for every minimum-size set $S \subseteq V(G)$ for which $G - S \in \mathcal{H}$, there exists $S_A \in \mathcal{S}$ such that for $S' := (S \setminus A) \cup S_A$ we have $|S'| \leq |S|$ and $G - S' \in \mathcal{H}$.

As a consequence of this definition, if $\mathcal{S}$ is A-exhaustive for $\mathcal{H}$-Deletion then there exists an optimal solution S to $\mathcal{H}$-Deletion with $S \cap A \in \mathcal{S}$.

The notion of A-exhaustive families is similar in spirit to that of q-representative families for matroids, which have been used in recent algorithms working on graph decompositions [80, 125] (cf. [48, §12.3]) to trim the set of partial solutions stored by a dynamic program while preserving the existence of an optimal solution. As the desired outcome of the replacement process in our case is not defined in terms of independence in a matroid and there is no particular importance of a given integer q, we use different terminology for our concept of exhaustive families. We reserve the name of a *representative family* to refer to a set of representatives for each equivalence class in the relation introduced in Definition 6.18.

The following observation shows how A-exhaustive families can be extended to supersets $A' \supseteq A$ by brute force. If $|A' \setminus A|$ is bounded, the increase in the

size of the exhaustive family can be controlled.

Observation 6.16. *Let $\mathcal{H}$ be a graph class, let G be a graph, let $A \subseteq V(G)$ and let $\mathcal{S} \subseteq 2^A$ be A-exhaustive for $\mathcal{H}$-Deletion on G. Then for any set $A' \supseteq A$, the family $\mathcal{S}' \subseteq 2^{A'}$ defined as follows has size at most $|\mathcal{S}| \cdot 2^{|A' \setminus A|}$ and is A'-exhaustive for $\mathcal{H}$-Deletion on G:*

$$\mathcal{S}' := \{S_1 \cup S^* \mid S_1 \in \mathcal{S} \wedge S^* \subseteq (A' \setminus A)\}.$$

A similar brute-force extension can be done when merging exhaustive families for disjoint subsets A_1, A_2 into an A-exhaustive family for a common superset $A \supseteq A_1 \cup A_2$. We will use this step to handle a variation of standard join bags in a tree decomposition.

Lemma 6.17. *Let $\mathcal{H}$ be a graph class and let G be a graph. Let $A_1, A_2 \subseteq V(G)$ be disjoint sets and let $\mathcal{S}_1, \mathcal{S}_2$ be A_1-exhaustive (respectively, A_2-exhaustive) for $\mathcal{H}$-Deletion on G. Then for any set $A' \supseteq A_1 \cup A_2$, the family $\mathcal{S}' \subseteq 2^{A'}$ defined as follows has size at most $|\mathcal{S}_1| \cdot |\mathcal{S}_2| \cdot 2^{|A' \setminus (A_1 \cup A_2)|}$ and is A'-exhaustive for $\mathcal{H}$-Deletion on G:*

$$\mathcal{S}' := \{S_1 \cup S_2 \cup S^* \mid S_1 \in \mathcal{S}_1 \wedge S_2 \in \mathcal{S}_2 \wedge S^* \subseteq (A' \setminus (A_1 \cup A_2))\}.$$

Proof. The bound on $|\mathcal{S}'|$ is clear from the definition. Consider an arbitrary optimal solution $S \subseteq V(G)$ to $\mathcal{H}$-Deletion on G. We will show there exists $\widehat{S} \in \mathcal{S}'$ such that $(S \setminus A') \cup \widehat{S}$ is an optimal solution. We use a two-step argument.

Since $\mathcal{S}_1$ is A_1-exhaustive, there exists $S_1 \in \mathcal{S}_1$ such that $S' := (S \setminus A_1) \cup S_1$ is again an optimal solution.

Applying a similar step to S', as $\mathcal{S}_2$ is A_2-exhaustive there exists $S_2 \in \mathcal{S}_2$ such that $S'' := (S' \setminus A_2) \cup S_2$ is an optimal solution.

Since A_1 and A_2 are disjoint, we have $S'' \cap A_1 = S_1$ and $S'' \cap A_2 = S_2$. Let $S^* := S'' \cap (A' \setminus (A_1 \cup A_2))$. It follows that the set $\widehat{S} = S_1 \cup S_2 \cup S^*$ belongs to $\mathcal{S}'$. Now note that $S \setminus A' = S'' \setminus A'$ as we have only replaced parts of the solution within A_1 and A_2, while $A' \supseteq A_1 \cup A_2$. Hence $(S \setminus A') \cup \widehat{S} = S''$ is an optimal solution, which concludes the proof. □

Note that Lemma 6.17 implies Observation 6.16 by letting $A_2 = \emptyset$, $\mathcal{S}_2 = \{\emptyset\}$.

Boundaried graphs. For a function $f\colon A \to B$ and a set $A' \subseteq A$, we denote by $f_{|A'}\colon A' \to B$ the restriction of f to A'. A k-boundaried graph is a triple $\widehat{G} = (G, X, \lambda)$, where G is a graph, $X \subseteq V(G)$, and $\lambda\colon [k] \to X$ is a bijection. We define $V(\widehat{G}) = V(G)$. Two k-boundaried graphs (G_1, X_1, λ_1),

(G_2, X_2, λ_2) are compatible if $\lambda_2 \circ \lambda_1^{-1}$ is a graph isomorphism between $G_1[X_1]$ and $G_2[X_2]$, that is, for each $i, j \in [k]$ it holds that $\lambda_1(i)\lambda_1(j) \in E(G_1[X_1])$ if and only if $\lambda_2(i)\lambda_2(j) \in E(G_2[X_2])$. Two k-boundaried graphs (G_1, X_1, λ_1), (G_2, X_2, λ_2) are isomorphic if there is a graph isomorphism $\pi\colon V(G_1) \to V(G_2)$, such that $\pi_{|X_1} = \lambda_2 \circ \lambda_1^{-1}$. The gluing operation $\oplus$ takes two compatible k-boundaried graphs (G_1, X_1, λ_1), (G_2, X_2, λ_2), creates their disjoint union, and then identifies the vertices $\lambda_1[i], \lambda_2[i]$ for each $i \in [k]$. Any tri-separation (A, X, B) of order k in a graph G allows G to be decomposed as the sum of two k-boundaried graphs $(G[A \cup X], X, \lambda) \oplus (G[B \cup X], X, \lambda)$ for an arbitrary bijection $\lambda\colon [|X|] \to X$.

Definition 6.18. Two k-boundaried graphs (G_1, X_1, λ_1) and (G_2, X_2, λ_2) are $(\mathcal{H}, k)$-equivalent if they are compatible and for every compatible k-boundaried graph (G_3, X_3, λ_3), it holds that $(G_1, X_1, \lambda_1) \oplus (G_3, X_3, \lambda_3) \in \mathcal{H} \iff (G_2, X_2, \lambda_2) \oplus (G_3, X_3, \lambda_3) \in \mathcal{H}$.

Observe that if (G_1, X_1, λ_1) and (G_2, X_2, λ_2) are compatible and G_1, G_2 do not belong to $\mathcal{H}$, they are $(\mathcal{H}, k)$-equivalent because we only consider hereditary classes of graphs. Therefore in each equivalence class of compatibility there can be only one $(\mathcal{H}, k)$-equivalence class that is comprised of k-boundaried graphs which do not belong to $\mathcal{H}$. The relevance of the $(\mathcal{H}, k)$-equivalence relation for solving $\mathcal{H}$-Deletion can be seen from the observation below.

Observation 6.19. *Consider k-boundaried graphs (G_1, X_1, λ_1), (G_2, X_2, λ_2), (H, X_3, λ_3), such that (G_1, X_1, λ_1), (G_2, X_2, λ_2) are $(\mathcal{H}, k)$-equivalent and compatible with (H, X_3, λ_3). Let $S \subseteq V(H) \setminus X_3$. Then $(H - S, X_3, \lambda_3) \oplus (G_1, X_1, \lambda_1) \in \mathcal{H}$ if and only if $(H - S, X_3, \lambda_3) \oplus (G_2, X_2, \lambda_2) \in \mathcal{H}$.*

The $\mathcal{H}$-Membership problem is the problem of deciding whether a given graph belongs to $\mathcal{H}$. We say that the $\mathcal{H}$-Membership problem is finite state if the relation of $(\mathcal{H}, k)$-equivalence has finitely many equivalence classes for each k. A k-boundaried graph (G, X, λ) is called a *minimal representative* in the relation of $(\mathcal{H}, k)$-equivalence if $G \in \mathcal{H}$ and there is no other k-boundaried graph which is $(\mathcal{H}, k)$-equivalent to (G, X, λ) and has a smaller number of vertices. We restrict our definition of a representative only to graphs in $\mathcal{H}$ as it will allow us to (a) avoid non-interesting corner cases and (b) exploit the properties of some classes $\mathcal{H}$ to bound the number of minimal representatives.

A family $\mathcal{R}_k^{\mathcal{H}}$ of k-boundaried graphs is called $(\mathcal{H}, k)$-representative if it contains a minimal representative from each $(\mathcal{H}, k)$-equivalence class where the underlying graphs belong $\mathcal{H}$. It will not matter how the ties are broken. A family $\mathcal{R}_{\leq k}^{\mathcal{H}}$ is called $(\mathcal{H}, \leq k)$-representative if it is a union of $(\mathcal{H}, t)$-representative

families for all $t \in [k]$. We define $\texttt{vol}(\mathcal{R}^{\mathcal{H}}_{\leq k}) = \sum_{R \in \mathcal{R}^{\mathcal{H}}_{\leq k}} |V(R)|$. Note that even though there may be many ways to construct such a family, the sum above is well-defined, as well as the size of the family. Let us note that we can effectively test $(\mathcal{H}, k)$-equivalence as long as some $(\mathcal{H}, k)$-representative family is provided.

Observation 6.20. *Let $\mathcal{R}^{\mathcal{H}}_k$ be an $(\mathcal{H}, k)$-representative family. Suppose that k-boundaried graphs (G_1, X_1, λ_1), (G_2, X_2, λ_2) are compatible and for every compatible k-boundaried graph (G_3, X_3, λ_3) from $\mathcal{R}^{\mathcal{H}}_k$ it holds that $(G_1, X_1, \lambda_1) \oplus (G_3, X_3, \lambda_3) \in \mathcal{H} \iff (G_2, X_2, \lambda_2) \oplus (G_3, X_3, \lambda_3) \in \mathcal{H}$. Then (G_1, X_1, λ_1), (G_2, X_2, λ_2) are $(\mathcal{H}, k)$-equivalent.*

We will be interested in upper bounding the maximal number of vertices of a graph in an $(\mathcal{H}, \leq k)$-representative family $\mathcal{R}^{\mathcal{H}}_{\leq k}$; we will denote this quantity by $r_{\mathcal{H}}(k)$. Since there are $2^{\mathcal{O}((r_{\mathcal{H}}(k))^2)}$ different graphs on $r_{\mathcal{H}}(k)$ vertices, this gives an immediate bound on $|\mathcal{R}^{\mathcal{H}}_{\leq k}|$ and $\texttt{vol}(\mathcal{R}^{\mathcal{H}}_{\leq k})$. What is more, we can construct an $(\mathcal{H}, \leq k)$-representative family within the same running time.

Lemma 6.21. *Let $\mathcal{H}$ be a graph class recognizable in polynomial time such that $\mathcal{H}$-Membership is finite state and there is a time-constructible[1] function $r(k) \geq k$, such that for every integer k and for every t-boundaried graph, where $t \in [k]$, there exists an $(\mathcal{H}, t)$-equivalent t-boundaried graph on at most $r(k)$ vertices. Then there exists an algorithm that, given an integer k, runs in time $2^{\mathcal{O}((r(k))^2)}$ and returns an $(\mathcal{H}, \leq k)$-representative family.*

Proof. Consider a process in which we begin from an edgeless graph on at most $r(k)$ vertices, fix t boundary vertices, and choose an adjacency matrix determining which pairs of vertices share an edge. By iterating over all possible adjacency matrices for graphs on $r(k)$ vertices and considering the first t rows to be the boundary vertices, we can generate all $2^{\mathcal{O}((r(k))^2)}$ t-boundaried graphs on at most $r(k)$ vertices, for all $t \in [k]$, in time $2^{\mathcal{O}((r(k))^2)}$.

Let $\mathcal{G}_t$ be the constructed set of graphs for each $t \in [k]$. We separate them into equivalence classes. For each pair of graphs in $\mathcal{G}_t$, test if they are compatible in $\mathcal{O}(t^2)$ time and, if so, test in $2^{\mathcal{O}((r(k))^2)}$ time if they are $(\mathcal{H}, t)$-equivalent by gluing all other graphs in $\mathcal{G}_t$ and testing if containment in $\mathcal{H}$ is the same for both glued graphs. The latter can be done in $r(k)^{\mathcal{O}(1)}$ time by assumption as $\mathcal{H}$ is recognizable in polynomial time. If the two are indeed

[1] A function $r \colon \mathbb{N} \to \mathbb{N}$ is time-constructible if there exists a Turing machine that, given a string 1^k, outputs the binary representation of $r(k)$ in time $\mathcal{O}(r(k))$. We add this condition so we can assume that the value of $r(k)$ is known to the algorithm. All functions of the form $r(k) = \alpha \cdot k^c$, where α, c are positive integers, are time-constructible.

equivalent, then the larger of the two can be removed from $\mathcal{G}_t$ (if their size is the same, it does not matter how ties are broken). Correctness is preserved due to Observation 6.20. In total this takes $\binom{|\mathcal{G}_t|}{2} \cdot \mathcal{O}(t^2) \cdot |\mathcal{G}_t| \cdot r(k)^{\mathcal{O}(1)}$ time, which is bounded by $2^{\mathcal{O}((r(k))^2)}$. In the end $\mathcal{G}_t$ has a single graph for each equivalence class. By joining the results for each $t \in [k]$, we get the claimed result. □

6.4.2 Dynamic programming with A-exhaustive families

We move on to designing a meta-algorithm for solving $\mathcal{H}$-Deletion parameterized by $\mathcal{H}$-treewidth. Let S be an optimal solution and t be a node in a tree $\mathcal{H}$-decomposition. We will associate some tri-separation (A, X, B) with t, where A stands for the set of vertices introduced below t and $|X| \leq k$ for $k-1$ being the width of the decomposition. The main idea is to consider all potential tri-separations (A_S, X_S, B_S) that may be obtained after removing S. For each $X_S \subseteq X$ we do the following. We enumerate all the k-boundaried graphs from an $(\mathcal{H}, \leq k)$-representative family and in each iteration glue the k-boundaried graph to $G[A \cup X_S]$ and seek for a minimal deletion set within A so that the resulting graph belongs to $\mathcal{H}$. Correctness will follow from the fact that one of the representatives is $(\mathcal{H}, k)$-equivalent to $(G[B_S \cup X_S], X_S)$. We will show that it suffices to consider all the representatives in the relation of $(\mathcal{H}, k)$-equivalence to construct an A-exhaustive family of partial solutions. We shall use the following problem to describe a subroutine for finding such deletion sets in the leaves of the decomposition.

Disjoint $\mathcal{H}$-Deletion	**Parameter:** s, ℓ
Input: A graph G, integers s, ℓ, and a subset $U \subseteq V(G)$ of size at most ℓ that is an $\mathcal{H}$-deletion set in G.	
Task: Either return a minimum-size $\mathcal{H}$-deletion set $S \subseteq V(G) \setminus U$ or conclude that no such set of size at most s exists.	

We introduce two parameters s, ℓ that control the solution size and the size of the set U. In the majority of cases we can adapt known algorithms for $\mathcal{H}$-Deletion to solve Disjoint $\mathcal{H}$-Deletion even without the assumptions on the size and structure of U. It is however convenient to impose such requirements on U because (1) this captures the subproblem we need to solve in Lemma 6.22 and (2) this allows us to adapt the known algorithm for $\mathcal{H} = \mathsf{planar}$ (Theorem 6.31).

Recall that vertices $u, v \in V(G)$ are *true-twins* if $N_G[u] = N_G[v]$. The following lemma shows that the requirement of having undeletable vertices is easy to overcome if $\mathcal{H}$ is closed under the addition of true twins. More

formally, if $G \in \mathcal{H}$ and G' is obtained from G by adding a new vertex $v \notin V(G)$ such that v is a true twin of some vertex $u \in V(G)$, then $G' \in \mathcal{H}$. Note that in the following lemma the value of parameter ℓ is insignificant. We do not optimize the construction of G' as in our applications (Chordal- and Interval Deletion) we already get an unspecified polynomial dependency on n.

Lemma 6.22. *Let $\mathcal{H}$ be a hereditary graph class closed under the addition of true twins and such that $\mathcal{H}$-Deletion parameterized by the solution size s can be solved in $f(n,s)$ time. Then the problem* Disjoint $\mathcal{H}$-Deletion *can be solved in $f(s \cdot n, s) + n^{\mathcal{O}(1)}$ time.*

Proof. Let G' be the graph obtained from G by making s new true-twin copies of every vertex in the undeletable set U (so including the original vertices, there are $s+1$ true-twins of every vertex in U). The graph G' can be constructed in $n^{\mathcal{O}(1)}$ time. Then for every set $S \subseteq V(G) \setminus U$ for which $G - S$ is a member of $\mathcal{H}$, the graph $G' - S$ is also in $\mathcal{H}$ since it can be obtained from a graph in $\mathcal{H}$ by repeatedly adding true twins. Conversely, any minimum-size set $S \subseteq V(G')$ of size at most s for which $G' - S \in \mathcal{H}$ does not contain any copies of vertices in U, as the budget of s vertices is insufficient to contain all copies of a vertex, while a solution that avoids one copy of a vertex can avoid all copies, since members of $\mathcal{H}$ are closed under the addition of true twins. Consequently, an optimal solution in G' of size at most s is disjoint from U, and is also a solution in the induced subgraph G of G' since $\mathcal{H}$ is hereditary. Hence to compute a set S as desired, it suffices to compute an optimal $\mathcal{H}$-deletion set in G' if there is one of size at most s, which can be done by assumption. □

The following lemma describes the construction of an A-exhaustive family for $\mathcal{H}$-Deletion when given a tri-separation (A, X, B) of a graph for which $G[A] \in \mathcal{H}$. It will be applied to the leaves of a tree $\mathcal{H}$-decomposition; the tri-separation for these nodes as described in Observation 6.11 satisfy $G[A] \in \mathcal{H}$ as A will consist of base components of the decomposition.

Lemma 6.23. *Suppose that $\mathcal{H}$-Membership is finite state, $\mathcal{H}$ is hereditary and union-closed, and* Disjoint $\mathcal{H}$-Deletion *admits an algorithm with running time $f(s,\ell) \cdot n^{\mathcal{O}(1)}$. Then there is an algorithm that, given a tri-separation (A, X, B) of order k in a graph G such that $G[A] \in \mathcal{H}$, and an $(\mathcal{H}, \leq k)$-representative family $\mathcal{R}^{\mathcal{H}}_{\leq k}$, runs in time $2^k \cdot f(k, r_{\mathcal{H}}(k)) \cdot \mathtt{vol}(\mathcal{R}^{\mathcal{H}}_{\leq k})^{\mathcal{O}(1)} \cdot n^{\mathcal{O}(1)}$ and outputs a family $\mathcal{S}$ of size at most $2^k \cdot |\mathcal{R}^{\mathcal{H}}_{\leq k}|$ that is A-exhaustive for $\mathcal{H}$-Deletion on G.*

Proof. Initialize $\mathcal{S} = \emptyset$. For each subset $X' \subseteq X$, fix an arbitrary bijection $\lambda\colon [|X'|] \to X'$ and consider the graph $G - (X \setminus X')$. It admits a tri-separation (A, X', B). For each representative $R \in \mathcal{R}_t^{\mathcal{H}}$, where $t = |X'|$, which is compatible with $(G[B \cup X'], X', \lambda)$, we perform the gluing operation $G_R = (G[A \cup X'], X', \lambda) \oplus R$ and execute the algorithm for Disjoint $\mathcal{H}$-Deletion on G_R with the set of undeletable vertices $U = V(G_R) \setminus A$ and parameters $(k, r_{\mathcal{H}}(k))$. In other words, we seek a minimum-size deletion set $A' \subseteq A$ of size at most k. If such a set is found, we add it to $\mathcal{S}$. Note that U is indeed an $\mathcal{H}$-deletion set in G_R because $G_R - U = G[A]$ which belongs to $\mathcal{H}$ by assumption. By the definition of an $(\mathcal{H}, \leq k)$-representative family we have that $|U| \leq |V(R)| \leq r_{\mathcal{H}}(k)$, so the created instance meets the specification of Disjoint $\mathcal{H}$-Deletion.

The constructed family clearly has size at most $2^k \cdot |\mathcal{R}^{\mathcal{H}}_{\leq k}|$. The running time can be upper bounded by $2^k \cdot \sum_{R \in \mathcal{R}^{\mathcal{H}}_{\leq k}} f(k, r_{\mathcal{H}}(k))(n + |V(R)|)^{\mathcal{O}(1)} = 2^k \cdot f(k, r_{\mathcal{H}}(k)) \cdot \mathtt{vol}(\mathcal{R}^{\mathcal{H}}_{\leq k})^{\mathcal{O}(1)} \cdot n^{\mathcal{O}(1)}$. It remains to show that $\mathcal{S}$ is indeed A-exhaustive.

Consider a minimum-size solution S to $\mathcal{H}$-Deletion on G. Define S_A, S_X, S_B as $S \cap A, S \cap X, S \cap B$, respectively, and let $X' := X \setminus S_X$, $|X'| = t$. Fix an arbitrary bijection $\lambda\colon [t] \to X'$. Since $\mathcal{H}$ is union-closed and $G[A] \in \mathcal{H}$, by Observation 6.4 we know that $|S_A| \leq k$. The graph $G[B \cup X'] - S_B$ is an induced subgraph of $G - S$ so it belongs to $\mathcal{H}$. The set $\mathcal{R}_t^{\mathcal{H}}$ contains a t-boundaried graph R that is $(\mathcal{H}, t)$-equivalent to $(G[B \cup X'] - S_B, X', \lambda)$. By Observation 6.19, S_A is an $\mathcal{H}$-deletion set for $(G[A \cup X'], X', \lambda) \oplus R$. As $(G[A \cup X'], X', \lambda) \oplus R$ contains an $\mathcal{H}$-deletion set within A of size at most k, some set S'_A with this property has been added to $\mathcal{S}$. Furthermore, S'_A is a minimum-size solution, so $|S'_A| \leq |S_A|$. Again by Observation 6.19, S'_A is an $\mathcal{H}$-deletion set for $(G[A \cup X'], X', \lambda) \oplus (G[B \cup X'] - S_B, X', \lambda)$. It means that $S' = (S \setminus A) \cup S'_A = S_B \cup S_X \cup S'_A$ is an $\mathcal{H}$-deletion set in G and $|S'| \leq |S|$, which finishes the proof. □

The next step is to propagate the partial solutions along the decomposition in a bottom-up manner. As we want to grow the sets A for which A-exhaustive families are computed, we can take advantage of Observation 6.16 and Lemma 6.17. However, after processing several nodes, the size of A-exhaustive families computed this way can become arbitrarily large. In order to circumvent this, we shall prune the A-exhaustive families after each step.

Lemma 6.24. *Suppose that $\mathcal{H}$-Membership is finite state and graphs in the class $\mathcal{H}$ can be recognized in polynomial time. There is an algorithm that, given a tri-separation (A, X, B) of order k in a graph G, a family $\mathcal{S}' \subseteq 2^A$ that is*

A-exhaustive for $\mathcal{H}$-Deletion on G, and an $(\mathcal{H}, \leq k)$-representative family $\mathcal{R}^{\mathcal{H}}_{\leq k}$, runs in time $2^k \cdot |\mathcal{S}'| \cdot \mathtt{vol}(\mathcal{R}^{\mathcal{H}}_{\leq k})^{\mathcal{O}(1)} \cdot n^{\mathcal{O}(1)}$ and outputs a family $\mathcal{S} \subseteq \mathcal{S}'$ of size at most $2^k \cdot |\mathcal{R}^{\mathcal{H}}_{\leq k}|$ that is A-exhaustive for $\mathcal{H}$-Deletion on G.

Proof. Initialize $\mathcal{S} = \emptyset$. For each subset $X' \subseteq X$, fix an arbitrary bijection $\lambda\colon [|X'|] \to X'$ and consider the graph $G - (X \setminus X')$. It admits a tri-separation (A, X', B). For each graph $R \in \mathcal{R}^{\mathcal{H}}_t$, where $t = |X'|$, which is compatible with $(G[B \cup X'], X', \lambda)$, we perform the gluing operation $G_R = (G[A \cup X'], X', \lambda) \oplus R$. Using the polynomial-time recognition algorithm we choose a minimum-size set $S_A \in \mathcal{S}'$ which is an $\mathcal{H}$-deletion set for G_R, if one exists, and add it to $\mathcal{S}$.

We construct at most $2^k \cdot |\mathcal{R}^{\mathcal{H}}_{\leq k}|$ graphs G_R. For each graph G_R we add at most one set to $\mathcal{S}$ and spend $|\mathcal{S}'| \cdot (n + |R|)^{\mathcal{O}(1)}$ time. In total, we perform at most $2^k \cdot |\mathcal{R}^{\mathcal{H}}_{\leq k}| \cdot |\mathcal{S}'| \cdot \sum_{R \in \mathcal{R}^{\mathcal{H}}_{\leq k}} (n + |V(R)|)^{\mathcal{O}(1)} = 2^k \cdot |\mathcal{S}'| \cdot \mathtt{vol}(\mathcal{R}^{\mathcal{H}}_{\leq k})^{\mathcal{O}(1)} \cdot n^{\mathcal{O}(1)}$ operations. It remains to show that $\mathcal{S}$ is indeed A-exhaustive.

Consider a minimum-size solution S to $\mathcal{H}$-Deletion on G. Define S_A, S_X, S_B as $S \cap A, S \cap X, S \cap B$, respectively, and let $X' := X \setminus S_X$, $|X'| = t$. Since $\mathcal{S}'$ is A-exhaustive on G, there exists $\widehat{S}_A \in \mathcal{S}'$ such that $\widehat{S} = (S \setminus A) \cup \widehat{S}_A$ is also a minimum-size solution to G, implying that $|\widehat{S}_A| \leq |S_A|$. Fix an arbitrary bijection $\lambda\colon [t] \to X'$. The set $\mathcal{R}^{\mathcal{H}}_t$ contains a t-boundaried graph R that is $(\mathcal{H}, t)$-equivalent to $(G[B \cup X'] - S_B, X', \lambda)$. By Observation 6.19, a set $A' \subseteq A$ is an $\mathcal{H}$-deletion set for $G' = (G[A \cup X'], X', \lambda) \oplus (G[B \cup X'] - S_B, X', \lambda)$ if and only if A' is an $\mathcal{H}$-deletion set for $(G[A \cup X'], X', \lambda) \oplus R$. Since $G' - \widehat{S}_A = G - \widehat{S} \in \mathcal{H}$, we know that $\widehat{S}_A \in \mathcal{S}'$ is such a set. Hence by the construction above, there exists some (possibly different) $S'_A \in \mathcal{S}$ with this property and minimum size; hence $|S'_A| \leq |\widehat{S}_A| \leq |S_A|$ and $S' = (S \setminus A) \cup S'_A = S_B \cup S_X \cup S'_A$ is an $\mathcal{H}$-deletion set in G and $|S'| \leq |S|$. The claim follows. □

We are ready to combine the presented subroutines in a general meta-algorithm to process tree $\mathcal{H}$-decompositions for every class $\mathcal{H}$ which satisfies three simple conditions.

Theorem 6.25. *Suppose that the class $\mathcal{H}$ satisfies the following:*

1. *$\mathcal{H}$ is hereditary and union-closed,*
2. *Disjoint $\mathcal{H}$-Deletion admits an algorithm with running time $f(s, \ell) \cdot n^{\mathcal{O}(1)}$,*
3. *$\mathcal{H}$-Membership is finite state and there is an algorithm computing an $(\mathcal{H}, \leq t)$-representative family with running time $v(t)$.*

Then $\mathcal{H}$-Deletion can be solved in time $2^{\mathcal{O}(k)} \cdot f(k, r_{\mathcal{H}}(k)) \cdot v(k)^{\mathcal{O}(1)} \cdot n^{\mathcal{O}(1)}$ when given a tree $\mathcal{H}$-decomposition of width $k-1$ consisting of $n^{\mathcal{O}(1)}$ nodes.

Proof. First, we construct an $(\mathcal{H}, \leq k)$-representative family $\mathcal{R}^{\mathcal{H}}_{\leq k}$ in time $v(k)$. Since the output size of the algorithm cannot exceed its running time, we have $|\mathcal{R}^{\mathcal{H}}_{\leq k}| \leq \mathtt{vol}(\mathcal{R}^{\mathcal{H}}_{\leq k}) \leq v(k)$.

The algorithm is based on a variant of dynamic programming in which bounded-size sets of partial solutions are computed, with the guarantee that at least one of the partial solutions which are stored can be completed to an optimal solution. More formally, for each node $t \in V(T)$ we are going to compute (recall κ, π from Definition 6.9) a set of partial solutions $\mathcal{S}_t \subseteq 2^{\kappa(t)}$ of size at most $2^k \cdot |\mathcal{R}^{\mathcal{H}}_{\leq k}|$ which is $\kappa(t)$-exhaustive for $\mathcal{H}$-Deletion in G. As $\kappa(r) = V(G)$ for the root node r by Observation 6.10, any minimum-size set $S \in \mathcal{S}_r$ for which $G - S \in \mathcal{H}$ is an optimal solution to the problem, and the property of $\kappa(r)$-exhaustive families guarantees that one exists. We do the computation bottom-up in the tree decomposition, using Lemma 6.24 to prune sets of partial solutions at intermediate steps to prevent them from becoming too large.

Let (T, χ, L) be the given tree $\mathcal{H}$-decomposition of width $k-1$. By Lemma 6.7 we may assume that the decomposition is nice and is rooted at some node r. For $t \in V(T)$, define $L_t := L \cap \chi(t)$. Process the nodes of T from bottom to top. We process a node t after having computed exhaustive families for all its children, as follows. Let $X_t := \chi(t) \cap \pi(t)$, let $A_t := \kappa(t)$ and let $B_t := V(G) \setminus (A_t \cup X_t)$. By Observation 6.11, the partition (A_t, X_t, B_t) is a tri-separation of G. The way in which we continue processing t depends on the number of children it has. As T is a nice decomposition, node t has at most two children.

Leaf nodes. For a leaf node $t \in V(T)$, we construct an exhaustive family of partial solutions $\mathcal{S}_t \subseteq 2^{\kappa(t)}$ as follows. By Definition 3.6, vertices of L_t do not occur in other bags than $\chi(t)$. Because the decomposition is nice, we have $\chi(t) \setminus L_t = \pi(t)$. Therefore $\kappa(t) = L_t$ and we have $(A_t, X_t, B_t) = (L_t, \chi(t) \setminus L_t, V(G) \setminus \chi(t))$. Furthermore, $|X_t| \leq k$ since the width of the decomposition is $k-1$. As $G[L_t] \in \mathcal{H}$, we can process the tri-separation (A_t, X_t, B_t) with Lemma 6.23 within running time $2^k \cdot f(k, r_{\mathcal{H}}(k)) \cdot v(k)^{\mathcal{O}(1)} \cdot n^{\mathcal{O}(1)}$. We obtain a $\kappa(t)$-exhaustive family of size at most $2^k \cdot |\mathcal{R}^{\mathcal{H}}_{\leq k}|$.

Nodes with a unique child. Let t be a node that has a unique child c, for which a $\kappa(c)$-exhaustive family $\mathcal{S}_c$ of size $2^k \cdot |\mathcal{R}^{\mathcal{H}}_{\leq k}|$ has already been computed. Recall that vertices of L only occur in leaf bags, so that $L_t = \emptyset$ and therefore $|\chi(t)| \leq k$. Observe that $\kappa(t) \setminus \kappa(c) \subseteq \chi(t)$, so that $|\kappa(t) \setminus \kappa(c)| \leq k$. (A tighter bound for most nodes is possible by exploiting the niceness property,

which we avoid for ease of presentation.) Compute the following set of partial solutions:

$$\mathcal{S}'_t := \{S_c \cup S^* \mid S_c \in \mathcal{S}_c, S^* \subseteq \kappa(t) \setminus \kappa(c)\}.$$

Since the number of choices for S_c is $2^k \cdot |\mathcal{R}^{\mathcal{H}}_{\leq k}|$, while the number of choices for S^* is 2^k, the set $\mathcal{S}'_t$ has size at most $2^{2k} \cdot |\mathcal{R}^{\mathcal{H}}_{\leq k}|$ and can be computed in time $2^{2k} \cdot |\mathcal{R}^{\mathcal{H}}_{\leq k}| \cdot n^{\mathcal{O}(1)}$. Since $\kappa(c) \subseteq \kappa(t)$ due to Observation 6.10, we can invoke Observation 6.16 to deduce that the family $\mathcal{S}'_t$ is $\kappa(t)$-exhaustive for $\mathcal{H}$-Deletion on G. As the last step for the computation of this node, we compute the desired exhaustive family $\mathcal{S}_t$ as the result of applying Lemma 6.24 to $\mathcal{S}'_t$ and the tri-separation (A_t, X_t, B_t) of G, which is done in time $2^{3k} \cdot v(k)^{\mathcal{O}(1)} \cdot n^{\mathcal{O}(1)}$ because $|\mathcal{R}^{\mathcal{H}}_{\leq k}| \leq v(k)$. As $A_t = \kappa(t)$, the lemma guarantees that $\mathcal{S}_t$ is $\kappa(t)$-exhaustive and it is sufficiently small.

Nodes with two children. The last type of nodes to handle are those with exactly two children. So let $t \in V(T)$ have two children c_1, c_2. Since t is not a leaf we have $L_t = \emptyset$. Let $K := \kappa(t) \setminus (\kappa(c_1) \cup \kappa(c_2))$ and observe that $K \subseteq \chi(t) \setminus L$. Therefore $|K| \leq k$.

Using the $\kappa(c_1)$-exhaustive set $\mathcal{S}_{c_1}$ and the $\kappa(c_2)$-exhaustive set $\mathcal{S}_{c_2}$ computed earlier in the bottom-up process, we define a set $\mathcal{S}'_t$ as follows:

$$\mathcal{S}'_t := \{S_1 \cup S_2 \cup S^* \mid S_1 \in \mathcal{S}_{c_1}, S_2 \in \mathcal{S}_{c_2}, S^* \subseteq K\}.$$

As $\mathcal{S}_{c_1}$ and $\mathcal{S}_{c_2}$ both have size $2^k \cdot |\mathcal{R}^{\mathcal{H}}_{\leq k}|$, while $|K| \leq 2^k$, we have $|\mathcal{S}'_t| = 2^{3k} \cdot |\mathcal{R}^{\mathcal{H}}_{\leq k}|^2$. By Observation 6.10 we have that $\kappa(c_1) \cap \kappa(c_2) = \emptyset$ and $\kappa(c_1) \cup \kappa(c_2) \subseteq \kappa(t)$, so we can apply Lemma 6.17 to obtain that the family $\mathcal{S}'_t$ is $\kappa(t)$-exhaustive for $\mathcal{H}$-Deletion on G. The desired exhaustive family $\mathcal{S}_t$ is obtained by applying Lemma 6.24 to $\mathcal{S}'_t$ and the tri-separation (A_t, X_t, B_t) of G, which is done in time $2^{4k} \cdot v(k)^{\mathcal{O}(1)} \cdot n^{\mathcal{O}(1)}$.

Wrapping up. Using the steps described above we can compute, for each node of $t \in V(T)$ in a bottom-up fashion, a $\kappa(t)$-exhaustive family $\mathcal{S}_t$ of size $2^k \cdot |\mathcal{R}^{\mathcal{H}}_{\leq k}|$. Since the number of nodes of t is $n^{\mathcal{O}(1)}$ the overall running time follows. As discussed in the beginning of the proof, an optimal solution can be found by taking any minimum-size solution from the family $\mathcal{S}_r$ for the root r. □

6.4.3 Hitting forbidden connected minors

As a first application of the meta-theorem, we consider classes defined by a finite set of forbidden connected minors. The seminal results of Robertson and Seymour [163] state that every minor-closed family $\mathcal{H}$ can be defined by

a finite set of forbidden minors. The $\mathcal{H}$-DELETION problem is FPT for such classes when parameterized by the solution size [2, 165] or by treewidth [12]. The requirement that all the forbidden minors are connected holds whenever $\mathcal{H}$ is union-closed.

Unlike the next sections, here we do not need to prove any claims about the structure of minor-closed classes and we can just take advantage of known results in a black-box manner. In order to apply Theorem 6.25, we first need to bound the sizes of representatives in $\mathcal{R}_k^{\mathcal{H}}$. To this end, we shall take advantage of the recent result of Baste, Sau, and Thilikos [11], who have studied optimal running times for $\mathcal{H}$-DELETION parameterized by treewidth. They define a relation of $(\leq h, k)$-equivalence: two k-boundaried graphs (G_1, X_1, λ_1), (G_2, X_2, λ_2) are $(\leq h, k)$-equivalent if they are compatible and for every graph F on at most h vertices and for every compatible k-boundaried graph (G_3, X_3, λ_3), F is a minor of $(G_1, X_1, \lambda_1) \oplus (G_3, X_3, \lambda_3)$ if and only if F is a minor of $(G_2, X_2, \lambda_2) \oplus (G_3, X_3, \lambda_3)$.

Theorem 6.26 ([11, Thm. 6.2])**.** *There is a computable function f, so that if (R, X, λ) is a k-boundaried graph and R is K_q-minor-free, then there exists a k-boundaried graph (R', X', λ') which is $(\leq h, k)$-equivalent to (R, X, λ) and $|V(R')| \leq f(q, h) \cdot k$.*

Let $\mathcal{H}$ be a class defined by a family of forbidden minors, which are all connected and have at most h vertices. Then whenever two k-boundaried graphs are $(\leq h, k)$-equivalent, they are also $(\mathcal{H}, k)$-equivalent. As we consider only representatives whose underlying graphs belong to $\mathcal{H}$, they must exclude K_h as a minor. This leads to the following corollary.

Corollary 6.27. *Let $\mathcal{H}$ be a class defined by a finite family of forbidden minors.*

There exists a constant $d_{\mathcal{H}}$ such that for every minimal representative R in the relation of $(\mathcal{H}, k)$-equivalence we have $|V(R)| \leq d_{\mathcal{H}} \cdot k$.

Lemma 6.28. *Let $\mathcal{H}$ be a class defined by a finite non-empty family of forbidden minors. There exists an algorithm that, given an integer k, runs in time $2^{\mathcal{O}(k \log k)}$ and returns an $(\mathcal{H}, \leq k)$-representative family.*

Proof. We take advantage of the fact that graphs in $\mathcal{H}$ are sparse, that is, there exists a constant $c_{\mathcal{H}}$ such that if $G \in \mathcal{H}$ then $|E(G)| \leq c_{\mathcal{H}} \cdot |V(G)|$ [138]. Any graph $G \in \mathcal{H}$ on n vertices can be represented by a set of at most $c_{\mathcal{H}} \cdot n$ pairs of vertices which share an edge. Therefore the number of such graphs is at most $(n+1)^{2c_{\mathcal{H}} \cdot n} = 2^{\mathcal{O}(n \log n)}$ (for each of the $2c_{\mathcal{H}} \cdot n$ edge endpoints there are $n+1$ possibilities, where the plus one allows the graph to have fewer than $c_{\mathcal{H}} \cdot n$ edges).

We proceed similarly as in Lemma 6.21, with a slight change in how t-boundaried graphs are generated. Since a representative family only consists of graphs that belong to $\mathcal{H}$, we generate only those t-boundaried graphs that have sufficiently few vertices to be potentially in $\mathcal{H}$. For each $t \in [k]$ we do the following. We generate all graphs on at most $d_{\mathcal{H}} \cdot k$ vertices and at most $c_{\mathcal{H}} \cdot d_{\mathcal{H}} \cdot k$ edges as described above (see Corollary 6.27), there are $2^{\mathcal{O}(k \log k)}$ such graphs. For each generated graph, verify that it is contained in $\mathcal{H}$ in time $\mathcal{O}(k^3)$ by the algorithm of Robertson and Seymour (see [48, Thm. 6.12]). Proceed by picking all $\binom{d_{\mathcal{H}} \cdot k}{t}$ ways of picking the t boundary vertices. In total this results in generation of $2^{\mathcal{O}(k \log k)}$ t-boundaried graphs. The construction of the equivalence classes and representative family follows as in Lemma 6.21. □

Furthermore, it turns out that the known algorithms for $\mathcal{H}$-Deletion parameterized by the solution size can be adapted to work with undeletable vertices [165, §7.2].

Theorem 6.29 ([165])**.** *Let $\mathcal{H}$ be a class defined by a finite family of forbidden minors. Then* Disjoint $\mathcal{H}$-Deletion *admits an algorithm with running time $2^{s^{\mathcal{O}(1)}} \cdot n^3$, where s is the solution size.*

In order to obtain a better final guarantee for the most important case $\mathcal{H} = \mathsf{planar}$, we need a concrete bound on the exponent in the running time for Disjoint Planar Deletion. An algorithm with this property was proposed by Jansen, Lokshtanov, and Saurabh [112] as a subroutine in the iterative compression step for solving Planar Deletion. This is the only place where we rely on the assumption that that the undeletable set U is an $\mathcal{H}$-deletion set of bounded size.

Theorem 6.30 ([112])**.** Disjoint Planar Deletion *admits an algorithm with running time $2^{\mathcal{O}((\ell+s)\log(\ell+s))} \cdot n$, where s is the solution size and ℓ is the size of the undeletable set.*

We are ready to combine all the ingredients and apply the meta-theorem.

Theorem 6.31. *Let $\mathcal{H}$ be a class defined by a finite family of forbidden connected minors. Then $\mathcal{H}$-*Deletion *can be solved in time $2^{k^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$ when given a tree $\mathcal{H}$-decomposition of width $k-1$ consisting of $n^{\mathcal{O}(1)}$ nodes. In the special case of $\mathcal{H} = \mathsf{planar}$ the running time is $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$.*

Proof. We check the conditions of Theorem 6.25. The class $\mathcal{H}$ is hereditary and union-closed because the forbidden minors are connected. By Theorem 6.29, Disjoint $\mathcal{H}$-Deletion can be solved in time $2^{s^{\mathcal{O}(1)}} \cdot n^3$ and, by Lemma 6.28,

there is an algorithm computing an $(\mathcal{H}, \leq k)$-representative family in time $v(k) = 2^{\mathcal{O}(k \log k)}$.

For the case $\mathcal{H} = \mathsf{planar}$ we additionally take advantage of Theorem 6.30 to solve Disjoint Planar Deletion in time $f(s,\ell) \cdot n$ where $f(s,\ell) = 2^{\mathcal{O}((\ell+s)\log(\ell+s))}$. By Corollary 6.27 we can bound $r_{\mathcal{H}}(k)$ by $\mathcal{O}(k)$. Hence, the running time in Theorem 6.25 becomes $2^{\mathcal{O}(k)} \cdot f(k, r_{\mathcal{H}}(k)) \cdot 2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)} = 2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$. □

Finally, we invoke the algorithm for computing a tree $\mathcal{H}$-decomposition of approximate width to infer the general tractability result.

Corollary 6.32. *Let $\mathcal{H}$ be a class defined by a finite family of forbidden connected minors. Then $\mathcal{H}$-Deletion can be solved in time $2^{k^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$ where $k = \mathbf{tw}_{\mathcal{H}}(G)$. In the special case of $\mathcal{H} = \mathsf{planar}$ the running time is $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$.*

Proof. Due to Theorem 6.29, a tree $\mathcal{H}$-decomposition of width $\mathcal{O}(k)$ can be obtained in time $2^{k^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$ by Theorem 5.1. For the case $\mathcal{H} = \mathsf{planar}$, due to Theorem 6.30 we can obtain a tree $\mathcal{H}$-decomposition of width $\mathcal{O}(k)$ in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ instead. We obtain the result by plugging the decompositions into Theorem 6.31. □

The running time for $\mathcal{H}$-Deletion where $\mathcal{H}$ is the class of planar graphs is ETH-tight due to [153].

6.4.4 Hitting forbidden connected induced subgraphs

In this section we deal with graph classes $\mathcal{H}$ characterized by a finite family $\mathcal{F}$ of forbidden induced subgraphs. The problem of hitting finite forbidden (induced) subgraphs, parameterized by treewidth, has been studied by several authors [51, 154, 164].

We introduce some terminology for working with induced subgraphs and isomorphisms. For graphs G and H, a function $f\colon V(H) \to V(G)$ is an *induced subgraph isomorphism from H to G* if f is injective and satisfies $xy \in E(H) \Leftrightarrow f(x)f(y) \in E(G)$ for all $x, y \in V(H)$. A function $f'\colon A \to B$ is an *extension* of a function $f\colon A' \to B$ if $f'_{|A'} = f$.

Definition 6.33. Let $\mathcal{F}$ be a finite family of graphs. The operation of *$\mathcal{F}$-pruning* a k-boundaried graph (G, X, λ) is defined as follows:

- Initialize all vertices of $V(G) \setminus X$ as unmarked.

- For each $F \in \mathcal{F}$, for each tri-separation (A_F, X_F, B_F) of F with $B_F \neq \emptyset$, for each induced subgraph isomorphism f from $F[X_F]$ to $G[X]$, if there exists an induced subgraph isomorphism from $F[X_F \cup B_F]$ to G that is an extension of f, then mark the vertex set $\{f'(b) \mid b \in B_F\}$ for one such extension f', chosen arbitrarily.
- Remove all vertices of $V(G) \setminus X$ which are not marked at the end of the process.

Observe that graph resulting from the operation of *$\mathcal{F}$-pruning* depends on the choices made for f'. Our statements and algorithms are valid regardless how these ties are broken. We remark that there is no need to consider implementation aspects of $\mathcal{F}$-pruning because we only use it for an existential bound on the sizes of representatives.

Lemma 6.34. *Let $\mathcal{H}$ be a class defined by a finite family $\mathcal{F}$ of forbidden induced subgraphs. Let (G_1, X, λ) be a k-boundaried graph and suppose that (G_2, X, λ) was obtained by $\mathcal{F}$-pruning (G_1, X, λ). Then (G_1, X, λ) and (G_2, X, λ) are $(\mathcal{H}, k)$-equivalent.*

Proof. First observe that these graphs are compatible because $\mathcal{F}$-pruning removes a subset of vertices from $V(G_1) \setminus X$. As G_2 is an induced subgraph of G_1, the forward implication of Definition 6.18 is trivial. We prove that for any compatible k-boundaried graph $\widehat{H}$ it holds that if $\widehat{H} \oplus (G_2, X, \lambda)$ is induced-$\mathcal{F}$-free, then also $\widehat{H} \oplus (G_1, X, \lambda)$ is induced-$\mathcal{F}$-free.

Assume for a contradiction that $G = \widehat{H} \oplus (G_1, X, \lambda)$ contains an induced subgraph isomorphic to F for some $F \in \mathcal{F}$. Let (A, X, B) be the tri-separation of G, so that $(G[A \cup X], X, \lambda)$ is isomorphic with $\widehat{H}$ and $(G[B \cup X], X, \lambda)$ is isomorphic with (G_1, X, λ). Let $B' \subseteq B$ be the set of vertices marked during $\mathcal{F}$-pruning $(G[B \cup X], X, \lambda)$. Let f be an induced subgraph isomorphism from F to G. We define a tri-separation of F based on f: let $A_F := \{v \in V(F) \mid f(v) \in A\}$, let $X_F := \{v \in V(F) \mid f(v) \in X\}$, and let $B_F := \{v \in V(F) \mid f(v) \in B\}$. Observe that if $B_F = \emptyset$, then the image of F is fully contained in $A \cup X$, and so $G[A \cup X \cup B'] = \widehat{H} \oplus (G_2, X, \lambda)$ contains F as an induced subgraph, which gives a contradiction. Assume from now on that $B_F \neq \emptyset$.

Note that $f_{|X_F}$ is an induced subgraph isomorphism from $F[X_F]$ to $G[X]$ (if $X_F = \emptyset$ this is an empty isomorphism, which is also considered during $\mathcal{F}$-pruning), and that $f_{|X_F \cup B_F}$ is an extension of $f_{|X_F}$ that forms an induced subgraph isomorphism from $F[X_F \cup B_F]$ to $G[X \cup B]$. Consequently, in the process of $\mathcal{F}$-pruning $(G[B \cup X], X, \lambda)$ we considered F, the function $f_{|X_F}$, and an extension f' of $f_{|X_F}$ that is an induced subgraph isomorphism from $F[X_F \cup$

$B_F]$ to $G[X \cup B]$. Hence the vertices $\{f'(b) \mid b \in B_F\}$ were marked during the pruning process and are preserved in B'. It follows that f' is also an induced subgraph isomorphism from $F[X_F \cup B_F]$ to $G[X \cup B']$. Now consider the function $f^*\colon V(F) \to V(G)$ such that $f^*_{|A_F \cup X_F} = f$ and $f^*_{|B_F} = f'$, and recall that $f_{|X_F} = f'_{|X_F}$. As the tri-separation of G ensures that $f(x)f(y) \notin E(G)$ for any $x \in A_F$ and $y \in B_F$, it can easily be verified that f^* is an induced subgraph isomorphism from F to $G[A \cup X \cup B']$. This graph is isomorphic with $\widehat{H} \oplus (G_2, X, \lambda)$, so it also contains an induced subgraph isomorphic to F: a contradiction to the assumption that $\widehat{H} \oplus (G_2, X, \lambda)$ is induced-$\mathcal{F}$-free. □

Since $\mathcal{F}$-pruning preserves the $(\mathcal{H}, k)$-equivalence class, we can assume that the minimal representatives cannot be reduced by $\mathcal{F}$-pruning. It then suffices to estimate the maximal number of vertices left after $\mathcal{F}$-pruning. For a graph F on c vertices, there are at most 3^c tri-separations of F as each vertex either belongs to A_F, X_F, or B_F. The number of induced subgraph isomorphisms from X_F to X_G is bounded by k^c, where $k = |X|$, as for each of the at most c vertices in X_F there are at most k options for their image. For each such induced subgraph isomorphism we mark at most c vertices.

Observation 6.35. *Let $\mathcal{F}$ be a finite family of graphs on at most c vertices each. The operation of $\mathcal{F}$-pruning a k-boundaried graph (G, X, λ) removes all but $|\mathcal{F}| \cdot 3^c \cdot k^c \cdot c$ vertices from $V(G) \setminus X$.*

Corollary 6.36. *Let $\mathcal{H}$ be a graph class defined by a finite set $\mathcal{F}$ of forbidden induced subgraphs on at most c vertices each. If (R, X, λ) is a minimal representative in the relation of $(\mathcal{H}, k)$-equivalence, then $|V(R)| = \mathcal{O}(k^c)$.*

As the next step, we need to provide an algorithm for $\mathcal{H}$-Deletion parameterized by the solution size, which works with undeletable vertices. This can be done via a straightforward application of the technique of bounded-depth search trees.

Lemma 6.37 (cf. [35])**.** *Let $\mathcal{H}$ be a graph class defined by a finite set $\mathcal{F}$ of forbidden induced subgraphs on at most c vertices each. Then* Disjoint $\mathcal{H}$-Deletion *admits an algorithm with running time $c^s \cdot n^{\mathcal{O}(1)}$, where s is the solution size.*

Proof. Given an input (G, s, ℓ, U) (the parameter ℓ is unused here), we start by finding an induced subgraph isomorphism f from some $H \in \mathcal{F}$ to G, if one exists. As c is constant, this can be done in time $n^{\mathcal{O}(1)}$ by brute force. If no such f exists, then output the empty set as the optimal solution. Otherwise, let $T := \{f(v) \mid v \in V(H)\}$ denote the vertices in the range of f. Any valid

solution has to include a vertex of $T \setminus U$. If $s = 0$ or $T \subseteq U$, then clearly no solution of size at most s exists and we report failure. Otherwise, for each of the at most c vertices $v \in T \setminus U$ we recurse on the instance $(G - v, s - 1, \ell, U)$. If all recursive calls report failure, then we report failure for this call as well. If at least one branch succeeds, then we take a minimum-size solution S' returned by a recursive call and add v to it to form the output.

Since the branching is exhaustive, it is easy to see that the algorithm is correct. As the depth of the recursion tree is at most s, while the algorithm branches on $|T \setminus U| \leq c$ vertices at every step, the claimed running time follows. □

We can now combine all the ingredients and plug them into the meta-theorem. Even though c is constant, we can keep track of how it affects the exponent at k, as it follows easily from the claims above. Observe that so far we never had to assume that the graphs in the family $\mathcal{F}$ are connected, but this requirement is crucial for the tractability.

Theorem 6.38. *Let $\mathcal{H}$ be a graph class defined by a finite set $\mathcal{F}$ of forbidden induced subgraphs on at most c vertices each, which are all connected. Then $\mathcal{H}$-Deletion can be solved in time $2^{\mathcal{O}(k^{2c})} \cdot n^{\mathcal{O}(1)}$ when given a tree $\mathcal{H}$-decomposition of width $k-1$ consisting of $n^{\mathcal{O}(1)}$ nodes.*

Proof. We check the conditions of Theorem 6.25. The class $\mathcal{H}$ is hereditary and closed under disjoint union of graphs because the forbidden subgraphs are connected. By Lemma 6.37, Disjoint $\mathcal{H}$-Deletion can be solved in time $c^s \cdot n^{\mathcal{O}(1)}$. Finally, by Corollary 6.36 and Lemma 6.21, an $(\mathcal{H}, \leq k)$-representative family can be computed in time $v(k) = 2^{\mathcal{O}(k^{2c})}$. □

Corollary 6.39. *For any graph class $\mathcal{H}$ which is defined by a finite set $\mathcal{F}$ of connected forbidden induced subgraphs on at most c vertices each, $\mathcal{H}$-Deletion can be solved in time $2^{\mathcal{O}(k^{2c})} \cdot n^{\mathcal{O}(1)}$ when parameterized by $k = \boldsymbol{tw}_{\mathcal{H}}(G)$.*

Proof. By Lemma 6.37 without any undeletable vertices, $\mathcal{H}$-Deletion parameterized by solution size s can be solved in time $\mathcal{O}(c^s \cdot n^{c+1})$. Note that $\mathcal{H}$ is hereditary and union-closed. By Theorem 5.1, we can construct a tree $\mathcal{H}$-decomposition of width $\mathcal{O}(k)$ in time $c^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ (dropping the constant c in the polynomial part of the running time). By plugging the decomposition into Theorem 6.38, we obtain the result. □

6.4.5 Chordal deletion

In this section we develop a dynamic-programming algorithm that solves Chordal Deletion using a tree chordal-decomposition. We again want to use the meta-algorithm presented in Theorem 6.25. To this end, we need to bound the sizes of representatives in the relation of $(\mathsf{chordal}, k)$-equivalence. We obtain it through a new criterion that tests whether a graph G is chordal based on several properties of a tri-separation (A, X, B) in G. We therefore first develop some theory of chordal graphs.

Recall that a *hole* in a graph G is an induced cycle of length at least four and that a graph is chordal if it does not contain any holes. We need the following observation, which follows easily from the alternative characterization of chordal graphs as intersection graphs of the vertex sets of subtrees of a tree [89].

Observation 6.40. *Chordal graphs are closed under edge contractions.*

A vertex v in a graph G is *simplicial* if the set $N_G(v)$ forms a clique in G, that is, $G[N_G(v)]$ induces a complete graph. This includes isolated vertices (vertices without any neighbors) and vertices with a single neighbor. Since a hole does not contain any simplicial vertices, we have the following.

Observation 6.41. *If v is a simplicial vertex in G, then G is chordal if and only if $G - v$ is chordal. Consequently, if a graph G' is obtained from a chordal graph G by inserting a new vertex whose neighborhood is a clique in G, then G' is chordal.*

Lemma 6.42 ([30, Thm. 5.1.1])**.** *Every chordal graph contains a simplicial vertex.*

The following structural property of chordal graphs will be used to bound the sizes of representatives, once their structure is revealed.

Lemma 6.43. *If G is a chordal graph and $A \cup B$ is a partition of $V(G)$ such that B is an independent set in G and no vertex of B is simplicial in G, then $|B| < |A|$.*

Proof. Proof by induction on $|A|$. If $|A| = 1$ then $B = \emptyset$, as vertices in the independent set B can either be isolated or adjacent to the unique vertex in A, which would make them simplicial.

For the induction step, let $|A| > 1$ and let $v \in V(G)$ be a simplicial vertex, which exists by Lemma 6.42. By the precondition to the lemma, $v \in A$. Let $B_v := N_G(v) \cap B$. Since B_v is a clique as v is simplicial, while $B \supseteq B_v$ is

an independent set by assumption, we have $|B_v| \leq 1$. Let $G' := G - (\{v\} \cup B_v)$. Then the vertices of $B \setminus B_v$ are not simplicial in G', as the non-edges in their neighborhood do not involve v. By inductive assumption the graph G' with its partition into $A' := A \setminus \{v\}$ and $B' := B \setminus B_v$ satisfies $|B'| = |B \setminus B_v| < |A'| = |A| - 1$. As $|B_v| \leq 1$, this implies the lemma. □

Recall that a walk from a vertex u to a vertex v in a graph G is a sequence of (not necessarily distinct) vertices, starting with u and ending with v, such that consecutive vertices are adjacent in G. The vertices u and v are the *endpoints* of the walk, all other vertices occurring on the walk are *internal* vertices. The following observation gives an easy way to certify that a graph is not chordal.

Observation 6.44 ([142, Proposition 3])**.** *If a graph G contains a vertex v with two nonadjacent neighbors $u_1, u_2 \in N_G(v)$, and a walk from u_1 to u_2 with all internal vertices in $V(G) \setminus N_G[v]$, then G contains a hole passing through v.*

We are ready to formulate an operation used to produce representatives of bounded size.

Definition 6.45. Let (G, X, λ) be a k-boundaried graph. The operation of *condensing* (G, X, λ) is defined as follows.

- For each connected component B_i of $G - X$ for which $G[N_G(B_i)]$ is a clique, called a *simplicial component*, remove all vertices of B_i.
- For each connected component B_i of $G - X$ for which $G[N_G(B_i))]$ is *not* a clique, called a *non-simplicial component*, contract B_i to a single vertex.

We want to show that condensing boundaried graphs preserves its equivalence class. To this end, we show that we can harmlessly contract any edge which is not incident with the separator X. Recall that the graph obtained by contracting an edge uv is denoted G/uv.

Lemma 6.46. *Let G be a graph, (A, X, B) be a tri-separation in G, and let $u, v \in B$, $uv \in E(G)$. Then G is chordal if and only if the following conditions hold:*

1. *The graphs $G[A \cup X]$ and $G[B \cup X]$ are chordal.*
2. *The graph G/uv is chordal.*

Proof. If G is chordal, then since chordal graphs are hereditary and closed under edge contractions by Observation 6.40, both conditions are satisfied.

We prove the reverse implication. Suppose that $G[A \cup X], G[B \cup X]$, and G/uv are chordal but G is not, that is, it contains a hole H. As both $G[A \cup X]$ and $G[B \cup X]$ are chordal, hole H contains some $a \in A$ and some $b \in B$.

Consider a vertex $a \in A$ that lies on hole H, and let p, q be the predecessor and successor of a on the hole. Then $p, q \in N_G(a)$ and therefore $p, q \notin B$ by the properties of a tri-separation. Furthermore, $pq \notin E(G)$ since a hole is chordless. The subgraph $P := H - \{a\}$ forms an induced path between p and q in G. Since $u, v \in B$ and $a \in A$, we have $u, v \notin N_G(a)$. As contractions preserve the connectivity of subgraphs, when contracting edge uv the path P turns into a (possibly non-simple) walk between nonadjacent $p, q \in N_G(a)$ in G/uv, whose internal vertices avoid $N_G[a]$ since neither of the contracted vertices is adjacent to a. By Observation 6.44, this implies G/uv contains a hole; a contradiction. □

Lemma 6.47. *Let (G_1, X, λ) be a k-boundaried graph, so that G_1 is chordal, and let (G_2, X, λ) be obtained by condensing (G_1, X, λ). Then (G_1, X, λ) and (G_2, X, λ) are $(\mathsf{chordal}, k)$-equivalent.*

Proof. The condensing operation does not affect the boundary X so these graphs are compatible. Since G_1 is chordal, the same holds for G_2 because chordal graphs are closed under contracting edges and removing vertices. Consider a k-boundaried graph $\widehat{H}$ compatible with (G_1, X, λ). By the same argument as above, if $\widehat{H} \oplus (G_1, X, \lambda)$ is chordal, then $\widehat{H} \oplus (G_2, X, \lambda)$ is as well.

We now prove the second implication. Suppose that $\widehat{H} \oplus (G_2, X, \lambda)$ is chordal, so the underlying graph in $\widehat{H}$ is chordal as well. Let (A, X, B) be a tri-separation of $G = \widehat{H} \oplus (G_1, X, \lambda)$, so that $(G[A \cup X], X, \lambda)$ is isomorphic with $\widehat{H}$ and $(G[B \cup X], X, \lambda)$ is isomorphic with (G_1, X, λ). By the definition of condensing, $\widehat{H} \oplus (G_2, X, \lambda)$ can be obtained from G by contracting each connected component of $G[B]$ to a single vertex – let us refer to this graph as G' – and then removing some simplicial vertices. Since G' can be obtained from the chordal graph $\widehat{H} \oplus (G_2, X, \lambda)$ by inserting simplicial vertices, then G' is chordal by Observation 6.41. Let $G = G^1, G^2, \ldots, G^m = G'$ be the graphs given by the series of edge contractions that transforms G into G'. We prove that if G^{i+1} is chordal, then G^i is as well. The graph G^i admits a tri-separation (A, X, B^i), so that $G^i[B^i \cup X]$ is obtained from $G[B \cup X]$ via edge contractions, therefore $G^i[B^i \cup X]$ is chordal. Moreover, $G^i[A \cup X]$ is isomorphic with the underlying graph in $\widehat{H}$, so it is also chordal, and G^{i+1} is obtained from G^i by contracting an edge in B^i. We can thus apply Lemma 6.46 to infer that G^i

is chordal. It follows that $G = \widehat{H} \oplus (G_1, X, \lambda)$ is chordal, which finishes the proof. □

Corollary 6.48. *If (R, X, λ) is a minimal representative in the relation of $(\mathsf{chordal}, k)$-equivalence, $k > 0$, then $|V(R)| \leq 2k - 1$.*

Proof. By Lemma 6.47 it follows that $(\mathsf{chordal}, k)$-equivalence is preserved by condensing. If (R, X, λ) is a minimal representative (so R is chordal), it must be condensed. Therefore $V(R) \setminus X$ is an independent set and no $v \in V(R) \setminus X$ is simplicial. From Lemma 6.43 we get that $|V(R) \setminus X| < |X|$ and therefore $|V(R)| \leq 2k - 1$. □

The machinery developed so far is sufficient to obtain an FPT algorithm for Chordal Deletion on a standard tree decomposition. To be able to accommodate tree chordal-decompositions, which can contain leaf bags with arbitrarily large chordal base components, we need to be able to efficiently compute exhaustive families for such base components. Towards this end, we will use the algorithm by Cao and Marx for the parameterization by the solution size as a subroutine.

Theorem 6.49 ([38, Thm. 1.1])**.** *There is an algorithm that runs in $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ time which decides, given a graph G and integer k, whether or not G has a chordal deletion set of size at most k.*

By self-reduction and some simple graph transformations, the above algorithm can be adapted to our setting (see Lemma 2.8).

Lemma 6.50. *There is an algorithm with running time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ that solves* Chordal Deletion *parameterized by the solution size k.*

In order to enforce the requirement that some vertices are not allowed to be part of a solution, we use the following consequence of the fact that holes do not contain vertices sharing the same closed neighborhood.

Observation 6.51. *Let G be a chordal graph and let $v \in V(G)$. If G' is obtained from G by making a true-twin copy of v, that is, by inserting a new vertex v' which becomes adjacent to $N_G[v]$, then G' is chordal.*

Theorem 6.52. *The* Chordal Deletion *problem can be solved in $2^{\mathcal{O}(k^2)} \cdot n^{\mathcal{O}(1)}$ time when given a tree* chordal*-decomposition of width $k - 1$ consisting of $n^{\mathcal{O}(1)}$ nodes.*

Proof. We check the conditions of Theorem 6.25. The class of chordal graphs is clearly closed under vertex deletion and disjoint union of graphs. Because of Lemma 6.50 and Observation 6.51, we can apply Lemma 6.22 to solve Disjoint Chordal Deletion in time $2^{\mathcal{O}(s \log s)} \cdot n^{\mathcal{O}(1)}$ by introducing s true twins for each undeletable vertex. Next, by Corollary 6.48 and Lemma 6.21, a $(\mathsf{chordal}, \leq k)$-representative family can be computed in time $v(k) = 2^{\mathcal{O}(k^2)}$. □

Corollary 6.53. *The* Chordal Deletion *problem can be solved in* $2^{\mathcal{O}(k^2)} \cdot n^{\mathcal{O}(1)}$ *time when parameterized by* $k = \mathbf{tw}_{\mathsf{chordal}}(G)$.

Proof. The Chordal Deletion problem parameterized by solution size s can be solved in time $2^{\mathcal{O}(s \log s)} \cdot n^{\mathcal{O}(1)}$ by Lemma 6.50. By Theorem 5.1 it follows that we can compute a tree chordal-decomposition of width $\mathcal{O}(k)$ in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$. We obtain the claimed result by plugging it into Theorem 6.52. □

6.4.6 Interval deletion

An interval graph is the intersection graph of intervals of the real line, we have already seen them in Chapter 1. In an interval model $\mathcal{I}_G = \{I(v) \mid v \in V(G)\}$ of a graph G, each vertex $v \in V(G)$ corresponds to a closed interval $I(v) = [\mathrm{lp}(v), \mathrm{rp}(v)]$, with left and right endpoints $\mathrm{lp}(v)$ and $\mathrm{rp}(v)$ such that $\mathrm{lp}(v) < \mathrm{rp}(v)$; there is an edge between vertices u and v if and only if $I(v) \cap I(u) \neq \emptyset$, that is, their corresponding intervals have a non-empty intersection. The goal of this section is to show that Interval Deletion is FPT parameterized by $k = \mathbf{tw}_{interval}(G)$. In order to apply Theorem 6.25 (with Lemma 6.21), we aim to bound the size of a minimal representative for the $(\mathsf{interval}, k)$-equivalence classes. We introduce some notation and definitions. Since an edge contraction can be seen as merging two overlapping intervals, we have the following observation.

Observation 6.54. *Interval graphs are closed under edge contractions.*

For $u, v \in V(G)$, we say that $I(u)$ is strictly right of $I(v)$ (equivalently $I(v)$ is strictly left of $I(u)$) if $\mathrm{lp}(u) > \mathrm{rp}(v)$. We denote this by $I(u) > I(v)$ (equivalently $I(v) < I(u)$) for short. An interval model is called *normalized* if no pair of distinct intervals shares an endpoint. Every interval graph has a normalized interval model that can be produced in linear time (see [36]). We use the following well known characterization of interval graphs. Three distinct vertices $u, v, w \in V(G)$ form an *asteroidal triple* (AT) in G if for any two of them there is a path between them avoiding the closed neighborhood of the third.

Theorem 6.55 ([126], see also [30])**.** *A graph G is an interval graph if and only if G is chordal and contains no AT.*

A vertex set $M \subseteq V(G)$ is a module of G if $N_G(u) \setminus M = N_G(v) \setminus M$ for all $u, v \in M$. A module M is *trivial* if $|M| \leq 1$ or $|M| = |V(G)|$, and *non-trivial* otherwise. Throughout the section, we use the following terminology. An *obstruction* in a graph G is an inclusion-minimal vertex set X such that $G[X]$ is not interval.

Lemma 6.56 ([37, Proposition 4.4])**.** *Let G be a graph and $M \subseteq V(G)$ be a module. If $X \subseteq V(G)$ is an obstruction and $|X| > 4$, then either $X \subseteq M$ or $|M \cap X| \leq 1$.*

Theorem 6.55 implies that all obstructions induce connected graphs, as the obstructions to chordality—chordless cycles and minimal subgraphs containing an AT—are easily seen to be connected. The only obstruction of no more than four vertices induces a C_4 (see [37]). We use the following consequence.

Lemma 6.57. *Let (A, X, B) be a tri-separation of G, and let $M \subseteq B$ be a module in G. If $G[X \cup B]$ is interval, then any obstruction in G contains at most one vertex of M. Furthermore, for each obstruction S intersecting M, for each $v \in M$, the set $(S \setminus M) \cup \{v\}$ is also an obstruction.*

Proof. We first derive the first part of the statement. For any obstruction S larger than four vertices, the statement follows from Lemma 6.56 since no obstruction can be fully contained in $M \subseteq B$ as $G[X \cup B]$ is interval. For the case of $G[S]$ isomorphic to C_4, at least one of its vertices must be in A as $G[X \cup B]$ is interval. Since X is a separator, it follows that $|M \cap S| \leq |B \cap S| \leq 1$.

For the second part, consider some obstruction S intersecting M. By the arguments above, this intersection is a single vertex, say, u. The statement for $u = v$ is clear as then $(S \setminus M) \cup \{v\} = S$. In all other cases, since none of $S \setminus \{u\}$ is part of M, by definition of a module it follows that v has the exact same neighborhood to $S \setminus \{u\}$ as u. Hence, the graph induced by $(S \setminus M) \cup \{v\}$ is isomorphic to $G[S]$. □

We arrive at the first useful observation about the structure of minimal representatives.

Lemma 6.58. *If the k-boundaried graph (G, X, λ) is a minimal representative in the relation of $(\mathsf{interval}, k)$-equivalence and G is interval, then G has no non-trivial module $M \subseteq V(G) \setminus X$.*

Proof. For the sake of contradiction, suppose that G has a non-trivial module $M \subseteq V(G) \setminus X$. Pick an arbitrary vertex $v \in M$. We argue that $(G' = G - (M \setminus \{v\}), X, \lambda)$ is $(\mathsf{interval}, k)$-equivalent to (G, X, λ). Consider a k-boundaried graph H compatible with (G, X, λ). Note that H is compatible with (G', X, λ) too.

First suppose that $H \oplus (G, X, \lambda)$ is not interval. Consider the tri-separation $(V(H) \setminus X, X, V(G) \setminus X)$ of $F = H \oplus (G, X, \lambda)$. Let $S \subseteq V(F)$ be an obstruction. By Lemma 6.57 we have that $|S \cap M| \leq 1$ and furthermore that $S' = (S \setminus M) \cup \{v\}$ is an obstruction. It follows that $H \oplus (G', X, \lambda)$ contains the obstruction S' and hence is not interval. Now suppose that $H \oplus (G, X, \lambda)$ is interval. Since interval graphs are hereditary, it follows that $H \oplus (G', X, \lambda)$ is also an interval graph. □

Marking scheme. For a k-boundaried graph (G, X, λ) we proceed by marking a set of $|X|^{\mathcal{O}(1)}$ vertices $Q \subseteq V(G)$ such that for any compatible k-boundaried graph H, the following holds: if $H \oplus (G, X, \lambda)$ contains an asteroidal triple, then it contains an AT (v_1, v_2, v_3) such that $\{v_1, v_2, v_3\} \cap V(G) \subseteq X \cup Q$. Our bound on the size of the minimal representative is then obtained by analyzing the size of $G - (X \cup Q)$. Before getting to the marking scheme, we introduce some definitions and notation.

For a path P and $x, y \in V(P)$ let $P[x, y]$ be the subpath of P from x to y. For a set $U \subseteq V(G)$ let $\mathcal{P}(P, U)$ be the family of maximal subpaths of P contained in U.

Observation 6.59. *Consider a vertex set $U \subseteq V(G)$. Let P be a path whose endpoints are contained in $N(V(G) \setminus U)$. Then for each $Q \in \mathcal{P}(P, U)$, the endpoints of Q are contained in $N(V(G) \setminus U)$.*

For a k-boundaried graph (G, X, λ) such that G is interval, and a normalized interval model $\mathcal{I} = \{I(v) \mid v \in V(G)\}$, we shortly say that $(G, X, \lambda, \mathcal{I})$ is a k-boundaried interval graph with a model. Given a k-boundaried graph with a model $(G, X, \lambda, \mathcal{I})$ and connected vertex set $A \subseteq V(G)$, let $I(A) = \bigcup_{a \in A} I(a)$ denote the union of intervals of the vertices in A. Since A is connected, $I(A)$ is itself an interval $[\mathrm{lp}(A), \mathrm{rp}(A)]$ with $\mathrm{lp}(A) = \min_{a \in A} \mathrm{lp}(a)$ and $\mathrm{rp}(A) = \max_{a \in A} \mathrm{rp}(a)$.

Lemma 6.60. *Let $(G, X, \lambda, \mathcal{I})$ be a k-boundaried interval graph with a model and H be a k-boundaried graph compatible with (G, X, λ). For a chordless path P in $H \oplus (G, X, \lambda)$, let $\mathcal{I}(P) = \{I(V(Q)) \mid Q \in \mathcal{P}(P, V(G))\}$. Then $\mathcal{I}(P)$ is a set of pairwise disjoint intervals. Furthermore, if P is disjoint from $N_G[u]$ for some $u \in V(G)$, then these intervals are disjoint from $I(u)$.*

Proof. First observe that for each $Q \in \mathcal{P}(P, V(G))$ we have that $V(Q)$ is a connected vertex set, namely a chordless path, and therefore $I(V(Q))$ is well-defined. If for any two distinct $Q, Q' \in \mathcal{P}(P, V(G))$, the intervals $I(V(Q))$ and $I(V(Q'))$ would overlap, then either the paths were not maximal subpaths of P, or P would not be chordless.

To see the second part, note that any overlap between $I(V(Q))$ and $I(u)$ would imply that u is adjacent (or equal) to some vertex of $V(Q) \subseteq V(P)$. □

We would like to encode all the relevant information about a path that connects two vertices (v_1, v_2) and avoids the neighborhood of a vertex u, so later we could argue that some vertex in an AT can be replaced with another one. Since the boundaried graph H is unknown, we want to encode the subpaths that might appear within G, in particular their starting and ending points in X. However there might be $\Omega(|X|)$ such subpaths and exponentially-many combinations of starting/ending points. We shall show that only the two subpaths including the vertices v_1, v_2 and the (at most) two subpaths closest to u in the interval model are relevant. This means we only need to encode $\mathcal{O}(1)$ subpaths which gives only $|X|^{\mathcal{O}(1)}$ combinations. We begin with formalizing the concept of encoding a path.

Definition 6.61. Let $(G, X, \lambda, \mathcal{I})$ be a k-boundaried interval graph with a model and H be a k-boundaried graph compatible with (G, X, λ). Furthermore, let $F = H \oplus (G, X, \lambda)$, $v_1, v_2, u \in V(F)$, and P be a chordless (v_1, v_2)-path in $F - N_F[u]$. The signature S of (P, u) with respect to $(G, X, \lambda, \mathcal{I})$ is defined as follows.

If $V(P) \cap X = \emptyset$ then S is *trivial*. Otherwise S is a triple $(x_1, x_2, \mathcal{X})$ where $x_1, x_2 \in X$ and $\mathcal{X}$ is a set of ordered pairs from X. Let x_1 be the first vertex of P starting from v_1 with $x_1 \in X$. Similarly let x_2 be the first vertex of P in X starting from v_2. If $u \notin V(G)$, then $\mathcal{X} = \emptyset$. Otherwise, if there exists $Q \in \mathcal{P}(P[x_1, x_2], V(G))$ such that $I(V(Q)) < I(u)$, choose such $Q_\ell = (w_1, \ldots, w_{|V(Q_\ell)|})$ with maximal $\mathrm{rp}(V(Q_\ell))$ and add the pair $(w_1, w_{|V(Q_\ell)|})$ to $\mathcal{X}$. Similarly add a pair for a path Q_r with minimal $\mathrm{lp}(V(Q_r))$ such that $I(u) < I(V(Q_r))$ if such Q_r exists.

Note that the definition above is well-defined due to Lemma 6.60. An example is shown in Figure 6.3. In a signature we may have $x_1 = x_2$ and for any ordered pair $(x, y) \in \mathcal{X}$, possibly $x = y$. By Observation 6.59 it follows that the pairs in $\mathcal{X}$ consist of elements of X. Since a non-trivial signature S with respect to (G, X, λ) can be represented as a sequence of at most six vertices of X, we observe the following.

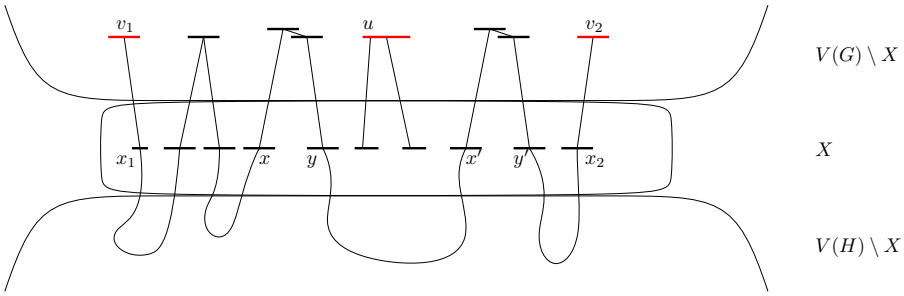


Figure 6.3: Schematic illustration of a graph $F = (G, X, \lambda) \oplus H$, where $(G, X, \lambda, \mathcal{I})$ is a k-boundaried interval graph with a model. A (v_1, v_2)-path P in $F - N_F[u]$ is shown. The signature of (P, u) is the triple $(x_1, x_2, \mathcal{X} = \{(x, y), (x', y')\})$.

Observation 6.62. *Let $\mathcal{S}(G, X, \lambda, \mathcal{I})$ be the family of all possible signatures with respect to $(G, X, \lambda, \mathcal{I})$. Then $|\mathcal{S}(G, X, \lambda, \mathcal{I})| = \mathcal{O}(|X|^6)$.*

We now define the obedience relation between a signature and a triple of vertices. We want to ensure the following properties: (1) if P is an (v_1, v_2)-path in $H \oplus (G, X, \lambda)$ avoiding the closed neighborhood of a vertex u, then (v_1, v_2, u) obeys the signature of (P, u), and (2) if two "similar" triples obey some signature, then for any choice of H the desired path exists either for both triples or for none of them.

A technical issue occurs when we want to consider triples of vertices not only from G but from $H \oplus (G, X, \lambda)$. Since our framework should be oblivious to the choice of H, we introduce the symbol $\perp$ as a placeholder for a vertex from $H - X$. In the definition below we assume $N_G[\perp] = \emptyset$.

Definition 6.63. Let $(G, X, \lambda, \mathcal{I})$ be a k-boundaried interval graph with a model and $v_1, v_2, u \in V(G) \cup \{\perp\}$. We say that (v_1, v_2, u) obeys the trivial signature if $v_1 = v_2 = \perp$ or $G - X - N_G[u]$ contains a (v_1, v_2)-path (the latter implies that $v_1, v_2 \in V(G) \setminus X$). We say that (v_1, v_2, u) obeys a non-trivial signature $S = (x_1, x_2, \mathcal{X}) \in \mathcal{S}(G, X, \lambda, \mathcal{I})$ if all the following conditions hold.

1. $v_1 = \perp$ or there is a (v_1, x_1)-path contained in $G - (X \setminus \{x_1\}) - N_G[u]$,
2. $v_2 = \perp$ or there is a (v_2, x_2)-path contained in $G - (X \setminus \{x_2\}) - N_G[u]$,
3. $u = \perp$ or for each $(x, y) \in \mathcal{X}$ there is an (x, y)-path in $G - N_G[u]$.

We give some intuition behind the obedience definition above. Consider some k-boundaried graph H compatible with (G, X, λ). Let (v_1, v_2, u) be an AT in $F = H \oplus (G, X, \lambda)$, so there is a (v_1, v_2)-path P in $F - N_F[u]$. Suppose we want to replace some vertex from (v_1, v_2, u) with another vertex from F, so that the new triple would still obey the signature of (P, u), and certify that an analogous path exists. If we replace v_1 or v_2 we will need to update the (v_1, x_1)-subpath (resp. (v_2, x_2)-subpath) of P. The first (resp. second) condition certifies that such an update is possible: if $u \in V(G)$ then it directly states that the new subpath avoids $N_F[u] \cap V(G) = N_G[u]$ and if $u \notin V(G)$ (this translates to $u = \perp$) then $N_F[u] \cap V(G) \subseteq X$ and we do not introduce any new vertices from X. If we aim at replacing u, then the third condition states that we can update the two subpaths of P which are closest to u in the interval model $\mathcal{I}$—we will show that this is sufficient.

Let $(G, X, \lambda, \mathcal{I})$ be a k-boundaried interval graph with a model. We set $v^\perp = v$ if $v \in V(G)$ or $\perp$ otherwise, assuming that G is clear from the context. We now show that the obedience relation satisfies the intuitive property that whenever a (v_1, v_2)-path P avoids the closed neighborhood of u then $(v_1^\perp, v_2^\perp, u^\perp)$ obeys the signature of (P, u).

Lemma 6.64. *Let $(G, X, \lambda, \mathcal{I})$ be a k-boundaried interval graph with a model and H be a k-boundaried graph compatible with (G, X, λ). Furthermore, let $F = H \oplus (G, X, \lambda)$, $v_1, v_2, u \in V(F)$, P be a chordless (v_1, v_2)-path in $F - N_F[u]$, and S be the signature of (P, u) with respect to $(G, X, \lambda, \mathcal{I})$. Then $(v_1^\perp, v_2^\perp, u^\perp)$ obeys S.*

Proof. We do a case distinction on $V(P) \cap X$. First suppose that $V(P) \cap X = \emptyset$. By Definition 6.61 we have that S is trivial. In the case that $V(P) \subseteq V(H) \setminus X$, then $v_1^\perp = v_2^\perp = \perp$ and therefore $(v_1^\perp, v_2^\perp, u^\perp)$ obeys S. In the case that $V(P) \subseteq V(G) \setminus X$, then P is a (v_1, v_2)-path in $G - X - N_G[u]$ and again $(v_1^\perp, v_2^\perp, u^\perp)$ obeys S.

Next, suppose that $V(P) \cap X \neq \emptyset$. Then by Definition 6.61 we have $S = (x_1, x_2, \mathcal{X})$. We check the obedience conditions of Definition 6.63 for $(v_1^\perp, v_2^\perp, u^\perp)$.

1. If $v_1 \notin V(G)$, then $v_1^\perp = \perp$ and the first condition clearly holds. Otherwise, $v_1 \in V(G)$, and x_1 is the first vertex of P starting from v_1 with $x_1 \in X$. If $u \notin V(G)$, then $u^\perp = \perp$ and by construction there is a (v_1, x_1)-path contained in $G - (X \setminus \{x_1\}) = G - (X \setminus \{x_1\}) - N_G[u^\perp]$. Otherwise, $u^\perp = u \in V(G)$ and $N_G[u] \subseteq N_F[u]$. Since P is disjoint from $N_F[u]$, it follows that there is a (v_1, x_1)-path contained in $G - (X \setminus \{x_1\}) - N_G[u^\perp]$.

2. The argument for the second condition is symmetric to the first condition.

3. If $u \notin V(G)$, then $u^\perp = \perp$ and the third condition clearly holds. Otherwise, $u^\perp = u \in V(G)$. By definition of $\mathcal{X}$, for each pair $(x, y) \in \mathcal{X}$, there is a subpath of P in $G - N_G[u]$ starting at x and ending at y. It follows that the third condition holds. □

In order to introduce the concept of replacing vertices in a triple, we formalize what we mean by saying that two triples are "similar". Simply speaking, we consider the endpoints of intervals of the boundary vertices and treat two vertices as equivalent if their intervals contain the same set of "boundary endpoints". We give the definition for a non-necessarily boundaried graph as later we will use it in a more general context.

Definition 6.65. Let G be an interval graph with a normalized interval model $\mathcal{I} = \{I(v) \mid v \in V(G)\}$ and $U \subseteq V(G)$. The endpoints of intervals of U partition the real line into $z = 2 \cdot |U| + 1$ regions. Let $(x_1, \ldots, x_{z-1})$ be an increasing order of these endpoints and let $x_0 = -\infty$ and $x_z = \infty$. We define the set of subsets $\mathcal{J}_U^{\mathcal{I}} = \{J_{i,j} \subseteq V(G) \setminus U \mid i \leq j \in [z]\}$, where $u \in V(G) \setminus U$ is in $J_{i,j}$ if and only if i is the smallest index such that the intervals $[\mathrm{lp}(u), \mathrm{rp}(u)]$ and $[x_{i-1}, x_i]$ have a non-empty intersection, and j is the largest index such that $[\mathrm{lp}(u), \mathrm{rp}(u)]$ and $[x_{j-1}, x_j]$ have a non-empty intersection.

The family $\mathcal{J}_U^{\mathcal{I}}$ clearly forms a partition of $V(G) \setminus U$. For a superset A of $V(G)$ and $x, y \in A$ we say x, y are $(G, X, \mathcal{I})$-equivalent if $x = y$ or $x, y \in V(G) \setminus X$ and x, y belong to the same set in the partition $\mathcal{J}_X^{\mathcal{I}}$ with respect to X. Two ℓ-tuples over A are $(G, X, \mathcal{I})$-equivalent if they are pointwise $(G, X, \mathcal{I})$-equivalent.

We now prove the main technical lemma which states that whenever two $(G, X, \mathcal{I})$-equivalent triples obey some signature, then the desired path exists either for both triples of vertices or for none of them.

Lemma 6.66. *Let $(G, X, \lambda, \mathcal{I})$ be a k-boundaried interval graph with a model and H be a k-boundaried graph compatible with (G, X, λ). Furthermore, let $F = H \oplus (G, X, \lambda)$, $v_1, v_2, u, w_1, w_2, z \in V(F)$, and P be a chordless (v_1, v_2)-path in $F - N_F[u]$. Suppose that triples (v_1, v_2, u) and (w_1, w_2, z) are $(G, X, \mathcal{I})$-equivalent and $(w_1^\perp, w_2^\perp, z^\perp)$ obeys the signature of (P, u) in $(G, X, \lambda, \mathcal{I})$. Then there exists a (w_1, w_2)-path in $F - N_F[z]$.*

Proof. Let S be the signature of (P, u). We do a case distinction on $V(P) \cap X$. First suppose that $V(P) \cap X = \emptyset$. By Definition 6.61 we have that S is trivial.

- In the case that $V(P) \subseteq V(H) \setminus X$, we have $v_1^\perp = v_2^\perp = \perp$. By the $(G, X, \mathcal{I})$-equivalence we have that $v_1 = w_1$ and $v_2 = w_2$. If $u \notin V(G) \setminus X$,

then by the $(G, X, \mathcal{I})$-equivalence we have $u = z$ and the path P satisfies the lemma. Otherwise $u \in V(G) \setminus X$ and by the $(G, X, \mathcal{I})$-equivalence we have $z \in V(G) \setminus X$ and therefore $N_F[z] \subseteq V(G)$. Since $V(P) \subseteq V(H) \setminus X$, again the path P satisfies the lemma.

- Next consider the case that $V(P) \subseteq V(G) \setminus X$. Since $v_1, v_2 \in V(G) \setminus X$, by the $(G, X, \mathcal{I})$-equivalence we have that $w_1, w_2 \in V(G) \setminus X$. Since $(w_1^\perp, w_2^\perp, z^\perp)$ obeys S, it follows that there is a (w_1, w_2)-path in $G - X - N_G[z^\perp]$.

Next, suppose that $V(P) \cap X \neq \emptyset$. By Definition 6.61 we have $S = (x_1, x_2, \mathcal{X})$. We transform P into the required path. We do a case distinction on the location of $u \in V(F)$.

- Suppose $u \in X$, then $u = z$ by the $(G, X, \mathcal{I})$-equivalence and therefore P is a (v_1, v_2)-path in $F - N_F[z]$. We first argue that there is a (w_1, x_1)-path in $F - N_F[z]$. This is trivially true if $w_1^\perp = \perp$, since then $w_1 = v_1$ by the $(G, X, \mathcal{I})$ equivalence and $P[v_1, x_1]$ is such a path. Otherwise, because of the first obedience condition it follows that there is a (w_1, x_1)-path P' contained in $G - (X \setminus \{x_1\}) - N_G[z^\perp = z]$. Analogously, there is a (x_2, w_2)-path P'' contained in $F - N_F[z]$. Then the concatenation of P', $P[x_1, x_2]$, and P'' is a (w_1, w_2)-path in $F - N_F[z]$, as desired.

- Suppose $u \in V(H) \setminus X$, then again $u = z$ by the $(G, X, \mathcal{I})$-equivalence and therefore P is a (v_1, v_2)-path in $F - N_F[z]$. The construction of the required path is identical to the previous case, but the argument requires one more observation here to show that $V(P')$ avoids $N_F[z]$ as $z^\perp = \perp$. Consider the case that $w_1^\perp \neq \perp$, then by the first obedience condition it follows that there is a (w_1, x_1)-path P' contained in $G - (X \setminus \{x_1\}) - N_G[z^\perp = \perp]$ (recall that $N_G[\perp] = \emptyset$). Observe that $V(P') \cap X = \{x_1\}$. Since $N_F[z] \cap V(G) \subseteq X$ as $z \in V(H) \setminus X$, and $u = z$ is not adjacent to x_1 as $x_1 \in V(P)$ and P is a path that avoids $N_F[u]$, it follows that P' is a (w_1, x_1)-path in $F - N_F[z]$. A symmetric argument shows that P'' avoids $N_F[z]$. Concatenating these with $P[x_1, x_2]$ yields a (w_1, w_2)-path in $F - N_F[z]$.

- Finally suppose $u \in V(G) \setminus X$. By the $(G, X, \mathcal{I})$-equivalence, we have $z \in V(G) \setminus X$ and $u = u^\perp$ and $z = z^\perp$ belong to the same set in the partition $\mathcal{J}_X^{\mathcal{I}}$. Obtain a (w_1, x_1)-path P' in F as in the previous case, possibly identical to $P[v_1, x_1]$, and a (x_2, w_2)-path P'' in F. By the obedience conditions of Definition 6.64, these can be taken disjoint from $N_G[z] = N_F[z]$.

Now it suffices to argue that there is a (x_1, x_2)-path disjoint from $N_F[z] = N_G[z]$. We transform $P[x_1, x_2]$ to the desired path, only modifying $\mathcal{P}(P[x_1, x_2], V(G))$. By Observation 6.59 it follows that $q_1, q_r \in X$ for each path $Q = (q_1, \ldots, q_r) \in \mathcal{P}(P[x_1, x_2], V(G))$. To complete the argument, we show that there is a (q_1, q_r)-path in $G - N_G[z]$ for each $Q = (q_1, \ldots, q_r) \in \mathcal{P}(P[x_1, x_2], V(G))$. Consider $\mathcal{I}(P[x_1, x_2]) = \{I(V(Q)) \mid Q \in \mathcal{P}(P[x_1, x_2], V(G))\}$. By Lemma 6.60 it follows that $\mathcal{I}(P[x_1, x_2])$ is a set of pairwise disjoint intervals. Consider the position of $I(u)$ with respect to the intervals in $\mathcal{I}(P[x_1, x_2])$. Since P is disjoint from $N_F[u] = N_G[u]$, it follows that $I(u)$ does not intersect $I(V(Q))$ for any $Q \in \mathcal{P}(P[x_1, x_2], V(G))$. Suppose there is a path $Q \in \mathcal{P}(P[x_1, x_2], V(G))$ with $I(V(Q)) < I(u)$. Let $Q_\ell = (\ell_1, \ldots, \ell_r)$ with $r = |V(Q_\ell)|$ be such that $I(V(Q_\ell)) < I(u)$ and $\mathrm{rp}(V(Q_\ell))$ is maximal. By Definition 6.61 we have that $(\ell_1, \ell_r) \in \mathcal{X}$. By the third obedience condition, there is a (ℓ_1, ℓ_r)-path in $G - N_G[z]$. Note that $I(\ell_1) < I(z)$ and $I(\ell_r) < I(z)$ since u and z are in the same set in the partition $\mathcal{J}_X^{\mathcal{I}}$. We argue that for any $Q = (q_1, \ldots, q_r) \in \mathcal{P}(P[x_1, x_2], V(G))$ with $I(V(Q)) < I(V(Q_\ell))$, the path Q is disjoint from $N_G[z]$. Suppose not, then there is some $s \in V(Q)$ such that s is adjacent to z. But since $I(s) < I(\ell_r) < I(z)$, this is not possible (refer to Figure 6.4 for an intuition). A symmetric argument shows the existence of a (q_1, q_r)-path for any $Q = (q_1, \ldots, q_r) \in \mathcal{P}(P[x_1, x_2], V(G))$ with $I(u) < I(V(Q))$. As each path of $\mathcal{P}(P[x_1, x_2], V(G))$ can be replaced by a path with the same endpoints that avoids $N_F[z]$, this yields the desired (x_1, x_2)-path avoiding $N_F[z]$ since the subpaths outside $P[x_1, x_2]$ are trivially disjoint from $N_F[z] = N_G[z] \subseteq V(G)$. □

We are ready to define the marking scheme and prove that we can always assume that a potential AT uses only the marked vertices. Since there are only $k^{\mathcal{O}(1)}$ signatures in a k-boundaried graph G, and every AT can be represented by three signatures, the replacement property from Lemma 6.66 allows us to use the same vertices in G for each of $k^{\mathcal{O}(1)}$ "types" of an AT.

Lemma 6.67. *Let $(G, X, \lambda, \mathcal{I})$ be a k-boundaried interval graph with a model. There exists a set $Q \subseteq V(G) \setminus X$ of $\mathcal{O}(k^{24})$ vertices so that for any k-boundaried graph H compatible with (G, X, λ), if $F = H \oplus (G, X, \lambda)$ contains some AT, then F contains an AT (w_1, w_2, w_3) such that $\{w_1, w_2, w_3\} \cap V(G) \subseteq X \cup Q$.*

Proof. For each triple (S_1, S_2, S_3) of signatures from $\mathcal{S}(G, X, \lambda, \mathcal{I})$, let the set $O(S_1, S_2, S_3)$ consist of triples (v_1, v_2, v_3) from $V(G) \cup \{\perp\}$ such that:

1. (v_2, v_3, v_1) obeys S_1,

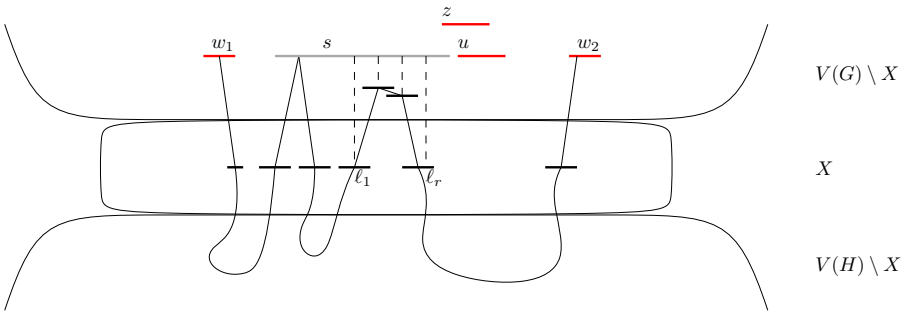


Figure 6.4: Illustration of the last case in Lemma 6.66. By replacing the interval u by z, we can only introduce adjacencies to the (ℓ_1, ℓ_r)-subpath of P; any potential neighbor $s \in N_F(z)$ in a path $P' \in \mathcal{P}(P[x_1, x_2], V(G))$ with $I(P') < I(P[\ell_1, \ell_r])$ would create a chord such as $s\ell_1$, which is impossible as the path P is assumed to be chordless.

2. (v_3, v_1, v_2) obeys S_2,

3. (v_1, v_2, v_3) obeys S_3.

Partition the triples $O(S_1, S_2, S_3)$ into equivalence classes based on the relation of $(G, X, \mathcal{I})$-equivalence among triples. Let $Q(S_1, S_2, S_3)$ contain an arbitrary triple from each class of $(G, X, \mathcal{I})$-equivalence within $O(S_1, S_2, S_3)$, as long as at least one such triple exists. We add to Q every vertex from $V(G) \setminus X$ that appears in any triple in $Q(S_1, S_2, S_3)$ for any (S_1, S_2, S_3). Observe that $|Q| = \mathcal{O}(k^{24})$ as there are $\mathcal{O}(k^{18})$ choices of (S_1, S_2, S_3) and $\mathcal{O}(k^6)$ equivalency classes.

We now argue that Q has the claimed property. So consider a k-boundaried graph H compatible with (G, X, λ) such that $F = H \oplus (G, X, \lambda)$ contains an AT (v_1, v_2, v_3). Let P_1 be a shortest (v_2, v_3)-path in $F - N_F[v_1]$. Note that P_1 is chordless. By Lemma 6.64 the triple $(v_2^\perp, v_3^\perp, v_1^\perp)$ obeys the signature S_1 of (P_1, v_1). The same holds for analogously defined P_2, P_3 and signatures S_2, S_3. The marking scheme picks some triple (b_1, b_2, b_3) with $b_i \in V(G) \cup \{\perp\}$ for each $i \in [3]$, which is $(G, X, \mathcal{I})$-equivalent to $(v_1^\perp, v_2^\perp, v_3^\perp)$ and such that (b_2, b_3, b_1) obeys S_1, (b_3, b_1, b_2) obeys S_2 and (b_1, b_2, b_3) obeys S_3.

Let (w_1, w_2, w_3) be defined as follows: for each $i \in [3]$, if $b_i = \perp$ then $w_i = v_i$, otherwise $w_i = b_i$. Note that $w_i \in V(F)$ for each $i \in [3]$. Then (w_1, w_2, w_3) is $(G, X, \mathcal{I})$-equivalent to (v_1, v_2, v_3) and $(w_1^\perp, w_2^\perp, w_3^\perp) = (b_1, b_2, b_3)$, so these triples behave the same for any signature. It follows that $(w_2^\perp, w_3^\perp, w_1^\perp)$ obeys

the signature S_1 so by Lemma 6.66 there exists a (w_2, w_3)-path in $F - N_F[w_1]$. By applying Lemma 6.66 to each of the above orderings of (w_1, w_2, w_3), it follows that (w_1, w_2, w_3) is an AT in F. By the definition of the marked set of vertices Q, $\{w_1, w_2, w_3\} \cap V(G) \subseteq X \cup Q$. □

Wrapping up. The last step of the proof is to give a bound on the size of a minimal representative (G, X, λ) in the relation of $(\mathsf{interval}, k)$-equivalence, by arguing that in such a graph only few vertices exist outside the set Q of bounded size. We achieve this by combining the module-free property (Lemma 6.58) and the former contraction property for chordal graphs (Lemma 6.46).

Lemma 6.68. *If the k-boundaried graph (G, X, λ) is a minimal representative in the relation of $(\mathsf{interval}, k)$-equivalence, then G has $\mathcal{O}(k^{48})$ vertices.*

Proof. By the definition, G is interval. Let $\mathcal{I}$ be a normalized interval model of G. Let $Q \subseteq V(G)$ be provided by Lemma 6.67. Obtain the partition $\mathcal{J}^{\mathcal{I}}_{X \cup Q} = \{J_{i,j} \subseteq V(G) \setminus (X \cup Q) \mid i \leq j \in [z]\}$ of $V(G) \setminus (X \cup Q)$ via Definition 6.65.

First consider the case that $J_{i,j}$ is an independent set for all $i \leq j \in [z]$. We show that G satisfies the lemma. For each $i \neq j \in [z]$, $G[J_{i,j}]$ induces a clique as the intervals intersect at some region border between the ith and jth region. Therefore $G[J_{i,j}]$ consists of a single vertex for each $i \neq j \in [z]$. We argue that $J_{i,i}$ has size $\mathcal{O}(|X \cup Q|)$ for each $i \in [z]$. If any two vertices in $J_{i,i}$ have the same open neighborhood, then they form a non-trivial module in $V(G) \setminus X$, which by Lemma 6.58 would contradict that (G, X, λ) is a minimal representative. It follows that each pair of vertices in $J_{i,i}$ must have different neighborhoods in $V(G) \setminus (X \cup Q)$ as they have the same neighborhoods in $X \cup Q$ by Definition 6.65. Since every vertex in $J_{i,i}$ is adjacent to every vertex in $J_{p,q}$ for each $p < i$ and $i < q$, they can only have neighborhood differences by adjacencies to $J_{p,i}$ or $J_{i,q}$, each of which consists of a single vertex as argued above. Since there are only $\mathcal{O}(|X \cup Q|)$ such vertices as $z = \mathcal{O}(|X \cup Q|)$, we have that $J_{i,i}$ has size $\mathcal{O}(|X \cup Q|)$. Since $|Q| = \mathcal{O}(|X|^{24})$ by Lemma 6.67, it follows that G at most $|X| + |Q| + \sum_{i \in [z]} |J_{i,i}| + \sum_{i < j \in [z]} |J_{i,j}| = \mathcal{O}(k^{48})$ vertices and the lemma holds.

In the remaining case suppose there is some edge uv for $\{u, v\} \subseteq J_{i,j}$ for some $i \leq j \in [z]$. We argue that (G, X, λ) is not a minimal representative. Let H be a k-boundaried graph compatible with (G, X, λ). Let $F = H \oplus (G, X, \lambda)$ with tri-separation $(A = V(H) \setminus X, X, B = V(G) \setminus X)$. We argue that F is interval if and only if F/uv is interval. Since interval graphs are closed under edge contractions by Observation 6.54, we have that if F is interval

then so is F/uv. In the other direction suppose that F is not interval. By Theorem 6.55 it follows that F contains a chordless cycle of length at least four or an asteroidal triple. Suppose F contains a chordless cycle of length at least four and $F' = F/uv$ does not. Since $F'[A \cup X] = F[A \cup X] = H$ we have that $F[A \cup X]$ is chordal. Since $F[B \cup X] = G$ and G is an interval graph we have that $F[B \cup X]$ is chordal. But then by Lemma 6.46 we have that F is chordal, contradicting that it contains a chordless cycle. It follows that F/uv also contains a chordless cycle and therefore F/uv is not interval. Finally suppose that F is chordal but contains an asteroidal triple (a, b, c). By Lemma 6.67 we can assume that $\{a, b, c\} \cap V(G) \subseteq X \cup Q$. Since $u, v \in J_{i,j}$ have the same neighborhood in $X \cup Q$, it follows that they are adjacent to the same vertices in $\{a, b, c\}$. Therefore any path in F that avoids the closed neighborhood of any vertex of the triple translates to a path in F/uv that avoids the closed neighborhood. It follows that (a, b, c) is an asteroidal triple in F/uv and therefore F/uv is not interval. Since $F/uv = H \oplus (G/uv, X, \lambda)$, we have shown that $(G/uv, X, \lambda)$ is in the same $(\mathsf{interval}, k)$-equivalence class, contradicting that (G, X, λ) is a minimal representative. □

As with Chordal Deletion in the previous section, we proceed by showing that we can solve Interval Deletion where some vertices become undeletable. As a first step, we observe that interval graphs are closed under addition of true twins, which follows from the fact that the added vertex can get the same interval in an interval model.

Observation 6.69. *Let G be an interval graph an let $v \in V(G)$. If G' is obtained from G by making a true-twin copy v' of v, then G' is an interval graph.*

Interval Deletion was shown to be FPT parameterized by solution size k by Cao and Marx. Their algorithm either returns a minimum cardinality solution, or decides that no solution of size at most k exists.

Theorem 6.70 ([37])**.** Interval Deletion *parameterized by the solution size k can be solved in time $10^k \cdot n^{\mathcal{O}(1)}$.*

Theorem 6.71. *The* Interval Deletion *problem can be solved in time $2^{\mathcal{O}(k^{96})} \cdot n^{\mathcal{O}(1)}$ when given a tree* $\mathsf{interval}$*-decomposition of width $k-1$ consisting of $n^{\mathcal{O}(1)}$ nodes.*

Proof. We check the conditions of Theorem 6.25. The class of interval graphs is clearly closed under vertex deletion and disjoint union of graphs. Because of Theorem 6.70 and Observation 6.69, we can apply Lemma 6.22 to solve Disjoint

Interval Deletion in time $10^s \cdot n^{\mathcal{O}(1)}$ by introducing s true twins for each undeletable vertex. Next, by Lemma 6.68 and Lemma 6.21, an $(\mathsf{interval}, \leq k)$-representative family can be computed in time $v(k) = 2^{\mathcal{O}(k^{96})}$. □

Corollary 6.72. *The* Interval Deletion *problem can be solved in time* $2^{\mathcal{O}(k^{96})} \cdot n^{\mathcal{O}(1)}$ *when parameterized by* $k = \boldsymbol{tw}_{interval}(G)$.

Proof. The Interval Deletion problem parameterized by the solution size can be solved in single-exponential time by Theorem 6.70. Since interval graphs are hereditary and union-closed, by Theorem 5.1 it follows that we can compute a tree $\mathsf{interval}$-decomposition of width $\mathcal{O}(k)$ in $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time. We obtain the result by plugging it in to Theorem 6.71. □

6.5 Hardness for non-union-closed classes

In this section we show why the restriction of Theorem 6.25 to graph classes which are closed under disjoint unions is necessary. To this end, we show that for $\mathcal{H}$ not closed under disjoint union, $\mathcal{H}$-Deletion can be NP-complete even for graphs of elimination distance 0 to a member of $\mathcal{H}$. Recall that K_5 is a clique on five vertices and $K_{1,3}$ is the claw. Let $H_{5+1,3} := K_5 + K_{1,3}$ denote the disjoint union of these two graphs. Let $\mathcal{H}_{5+1,3}$ denote the (hereditary) family of graphs which do not contain $H_{5+1,3}$ as an induced subgraph.

For our hardness proof, we will use the following result of Lewis and Yannakakis. Here a graph property is nontrivial in the class of planar graphs if there are infinitely many planar graphs that have the property, and infinitely many planar graphs which do not.

Theorem 6.73 ([127, Cor. 5])**.** *The node-deletion problem restricted to planar graphs for graph-properties that are hereditary on induced subgraphs and nontrivial on planar graphs is* NP*-complete.*

In particular, their result shows that Induced-$K_{1,3}$-free Deletion is NP-complete when restricted to planar graphs.

Theorem 6.74. $\mathcal{H}_{5+1,3}$ Deletion *is* NP*-complete when restricted to graphs whose elimination distance to* $\mathcal{H}_{5+1,3}$ *is* 0.

Proof. We give a reduction from an instance (G, k) of Induced-$K_{1,3}$-free Deletion on planar graphs, which asks whether the planar graph G can be made (induced) claw-free by removing at most k vertices.

Let G' be the disjoint union of G with $k+1$ copies of K_5. Observe that the elimination distance of G' to $\mathcal{H}_{5+1,3}$ is 0: each connected component of G' is

either isomorphic to K_5 (and does not contain $K_{1,3}$) or consists of the planar graph G and therefore does not contain K_5. Hence each connected component of G' belongs to $\mathcal{H}_{5+1,3}$.

We claim that the instance (G', k) of $\mathcal{H}_{5+1,3}$-Deletion is equivalent to the instance (G, k) of claw-free deletion. In one direction, any vertex set $S \subseteq V(G)$ for which $G - S$ is claw-free also ensures that $G' - S$ is claw-free (as the components isomorphic to K_5 do not contain any claws) and therefore implies $G' - S \in \mathcal{H}_{5+1,3}$. For the reverse direction, suppose $S' \subseteq V(G')$ is a set of size at most k such that $G' - S' \in \mathcal{H}_{5+1,3}$. Then $G' - S'$ must be claw-free, because if $G' - S'$ contains a claw, then this claw forms an induced copy of $H_{5+1,3}$ in $G' - S'$ together with one of the $k + 1$ copies of K_5 that contains no vertex of S'. So $G' - S'$ is claw-free, and therefore the induced subgraph $G - S'$ is claw-free as well, showing that (G, k) is a yes-instance.

As the transformation can easily be performed in polynomial time, this completes the proof. □

We remark that similar hardness proofs can be obtained for graph classes defined by a disconnected forbidden minor rather than a disconnected forbidden induced subgraph. For example, when H' is the disjoint union of the cycle C_9 on nine vertices and the graph K_5, one can show that Induced H'-Minor-free Deletion is NP-complete on graphs whose elimination distance to an H'-minor-free graph is 0. This can be seen from the fact that Feedback Vertex Set remains NP-complete on planar graphs of girth at least nine since subdividing edges does not change the answer, that planar graphs do not contain K_5 as a minor, while K_5 does not contain C_9 as a minor. As the purpose of these lower bounds is merely to justify our restriction to graph classes closed under disjoint union, we omit further details.

6.6 Conclusion

We presented a meta-theorem that can be used to solve $\mathcal{H}$-Deletion when given a tree $\mathcal{H}$-decomposition. Our results show that we can obtain ETH-tight parameter dependencies for $\mathcal{H}$-treewidth for $\mathcal{H}$ being the class of bipartite or planar graphs. Our work opens up a multitude of directions for future work. An obvious first one is to extend this list of graph classes for which we can give tight algorithms for $\mathcal{H}$-Deletion with hybrid parameters.

Beyond undirected graphs. On a conceptual level, the idea of solving a deletion problem parameterized by $\mathcal{H}$-treewidth (or $\mathcal{H}$-elimination distance)

is not restricted to undirected graphs. By developing notions of elimination distance for directed graphs, hypergraphs, or other discrete structures, similar questions could be pursued in those contexts. One could also consider undirected graphs with a distinguished set of terminal vertices, for example in an attempt to develop (uniform, single-exponential) FPT algorithms for Multiway Cut parameterized by the elimination distance to a graph where each component has at most one terminal.

Cross-parameterizations. Our main focus was on solving $\mathcal{H}$-Deletion parameterized by $\mathbf{tw}_{\mathcal{H}}$. However, $\mathcal{H}$-treewidth (or $\mathcal{H}$-elimination distance) can also be used as a parameterization away from triviality for solving other parameterized problems Π, for instance when using classes $\mathcal{H}$ in which Π is polynomial-time solvable. This can lead to interesting challenges of exploiting graph structure. For problems which are FPT parameterized by deletion distance to $\mathcal{H}$, does the tractability extend to elimination distance to $\mathcal{H}$? For example, is Undirected Feedback Vertex Set FPT when parameterized by the elimination distance to a subcubic graph or to a chordal graph? The problem is known to be FPT parameterized by the deletion distance to a chordal graph [114] or the edge-deletion distance to a subcubic graph [139].

As a step in this direction, Eiben et al. [62, Thm. 11] present a meta-theorem that yields non-uniform FPT algorithms when Π satisfies several conditions, which require a technical generalization of an FPT algorithm for Π parameterized by deletion distance to $\mathcal{H}$.

Capturing more vertex-deletion problems. There are some graph classes for which Theorem 5.1 provides a decomposition, but for which we currently have no follow-up algorithm to solve $\mathcal{H}$-Deletion on a given decomposition. A natural target for future work is to see whether the decompositions can be turned into vertex-deletion algorithms. For example, if $\mathcal{H}$ is characterized by a finite set of connected forbidden topological minors, is $\mathcal{H}$-Deletion FPT parameterized by $\mathbf{tw}_{\mathcal{H}}$?

Part III

Essential Vertices

Chapter 7

Search-Space Reduction via Essential Vertices

7.1 Introduction

Due to the enormous potential of preprocessing to speed up the algorithmic solution to combinatorial problems [1, 7, 8, 170, 171], a large body of work in theoretical computer science is devoted to understanding its power and limitations. Using the notion of *kernelization* [19, 68, 82, 99, 124, 133] from parameterized complexity [48, 60] it is possible to formulate a guarantee on the size of the instance after preprocessing based on the parameter of the original instance. Under this model, a good preprocessing algorithm is a *kernelization algorithm*: given a parameterized instance (x, k), it outputs an equivalent instance (x', k') of the same decision problem such that $|x'| + k' \leq f(k)$ for some function f that bounds the size of the kernel. Research into kernelization led to deep algorithmic insights, including connections to protrusions and finite-state properties [24], well-quasi ordering [79], and matroids [125]. These positive results were complemented by impossibility results [54, 61, 125] delineating the boundaries of tractability.

Results on kernelization led to profound insights into the limitations of polynomial-time data compression for NP-hard problems. However, as recently advocated [56], the definition of kernelization only gives guarantees on the *size* of the instance after preprocessing, which does not directly correspond to the running time of subsequently applied algorithms. If the preprocessed instance is not solved by brute force, but via a fixed-parameter tractable

algorithm whose running time is of the form $f(k) \cdot n^{\mathcal{O}(1)}$, then the exponential dependence in the running time is on the value of the *parameter* k, which is not guaranteed to decrease via kernelization. In fact, if $\mathsf{P} \neq \mathsf{NP}$ then no polynomial-time preprocessing algorithm can guarantee to always decrease the parameter of an NP-hard fixed-parameter tractable problem, as iterating the preprocessing algorithm would lead to its solution in polynomial time. In this chapter, we develop a new analysis of preprocessing aimed at reducing the search space of the follow-up algorithm. We apply this framework to combinatorial optimization problems on graphs, whose goal it is to find a minimum cardinality subset of vertices satisfying certain properties. The main idea behind the framework is to define formally what it means for a vertex to be *essential* for making reasonable solutions to the problem, and to prove that an efficient preprocessing algorithm can detect a subset of an optimal solution that contains all such essential vertices.

Before stating our results, we introduce and motivate the model. We consider vertex-subset minimization problems on (possibly directed) graphs, in which the goal is to find a minimum vertex subset having a certain property. Examples of the problems we study include Vertex Cover, Odd Cycle Transversal, and Dominating Set. The analysis of such minimization problems, parameterized by the size of the solution, has played an important role in the literature (cf. [39, 62, 110, 118, 159]). Our starting point is the idea that a good preprocessing algorithm should reduce the *search space*. Since many graph problems are known to be fixed-parameter tractable when parameterized by the size of the solution, we can reduce the search space of these FPT algorithms by finding one or more vertices which are part of an optimal solution, thereby decreasing the size of the solution still to be found in the reduced instance (i.e. the parameter value).

Since in general no polynomial-time algorithm can guarantee to identify at least one vertex that belongs to an optimal solution, the guarantee of the effectiveness of the preprocessing algorithm should be stated in a more subtle way. When solving problems by hand, one sometimes finds that certain vertices v are easily identified to belong to an optimal solution, as avoiding them would force the solution to contain a prohibitively large number of alternative vertices and therefore be suboptimal. Can an efficient preprocessing algorithm identify those vertices that cannot be avoided when making an optimal solution?

Since many NP-hard problems remain hard even when there is a unique solution [168], this turns out to be too much to ask as it would allow instances with unique solutions to be solved in polynomial time, which leads to $\mathsf{NP} = \mathsf{RP}$. We therefore have to relax the requirements on the preprocessing guarantee slightly, as follows. For an instance of a vertex-subset minimization problem Π

on a graph G, we denote the minimum size of a solution on G by $\mathrm{OPT}_\Pi(G)$. For a fixed $c \in \mathbb{R}_{\geq 1}$, we say a vertex $v \in V(G)$ is *c-essential* for Π on G if all feasible solutions $S \subseteq V(G)$ for Π whose total size is at most $c \cdot \mathrm{OPT}_\Pi(G)$ contain v. Based on this notion, we can ask ourselves: can an efficient preprocessing algorithm identify part of an optimal solution if there is at least one c-essential vertex?

Phrased in this way, the algorithmic task becomes more tractable. For example, for the Vertex Cover problem, selecting all vertices that receive the value 1 in an optimal half-integral solution to the standard LP-relaxation (see [48, Equation 2.1]) results in a set S which is contained in some optimal solution (by the Nemhauser-Trotter theorem [147], cf [48, §2.5]), and at the same time includes all 2-essential vertices: any vertex $v \notin S$ only has neighbors of value $\frac{1}{2}$ and 1, which implies that the set X of vertices other than v whose value in the LP relaxation is at least $\frac{1}{2}$, forms a feasible solution which avoids v. Its cardinality is at most twice the cost of the LP relaxation and therefore X is a 2-approximation. Hence S contains all 2-essential vertices.

This example shows that a preprocessing step that detects c-essential vertices without any additional information is sometimes possible. However, to be able to extend the scope of our results also to problems which *do not* have polynomial-time constant-factor approximations, we slightly relax the requirements on the preprocessing algorithm as follows. Let Π be a minimization problem on graphs whose solutions are vertex subsets and let $c \in \mathbb{R}_{\geq 1}$.

c-Essential detection for Π
Input: A graph G and integer k.
Task: Find a vertex set $S \subseteq V(G)$ such that:

G1 if $\mathrm{OPT}_\Pi(G) \leq k$, then there is an optimal solution in G containing all of S, and

G2 if $\mathrm{OPT}_\Pi(G) = k$, then S contains all c-essential vertices.

In this model, the preprocessing task is facilitated by supplying an additional integer k in the input. The correctness properties of the output S are formulated in terms of k. If $\mathrm{OPT}_\Pi(G) \leq k$, then the set S is required to be part of an optimal solution. The upper bound on $\mathrm{OPT}_\Pi(G)$ is useful to the algorithm: whenever the algorithm establishes that avoiding v would incur a cost of more than k, it is safe to add v to S. If $\mathrm{OPT}_\Pi(G) = k$, then the algorithm should guarantee that S contains all c-essential vertices. Knowing a lower bound on $\mathrm{OPT}_\Pi(G)$ is useful to the algorithm in case it can establish that any optimal solution containing v can be transformed into one avoiding v whose cost

is $(c-1)\cdot k$ larger, which yields a c-approximation if $(c-1)\cdot k \leq (c-1)\operatorname{OPT}_\Pi(G)$. Hence vertices for which such a replacement exists are not c-essential and may safely be left out of S.

Results. We present polynomial-time algorithms for c-Essential detection for Π for a range of vertex-deletion problems Π and small values of c, typically $c \in \{2, 3\}$. Example applications include Vertex Cover and Feedback Vertex Set, and also Chordal Vertex Deletion (for which no $\mathcal{O}(1)$-approximation is known), Odd Cycle Transversal (for which no $\mathcal{O}(1)$-approximation exists, assuming the Unique Games Conjecture [121, 172]), and even Directed Odd Cycle Transversal (which is W[1]-hard parameterized by solution size [137]).

The model of c-Essential detection for Π is chosen such that the detection algorithms whose correctness is formulated with respect to the value of k, can be used as a preprocessing step to optimally solve vertex-subset problems without any knowledge of the optimum. Let $\mathcal{E}_c^\Pi(G)$ denote the set of c-essential vertices in G, which is well-defined since all optimal solutions contain all c-essential vertices. By using a preprocessing step that detects a superset of the c-essential vertices in the solution, we can effectively improve the running-time guarantee for FPT algorithms parameterized by solution size from $f(\operatorname{OPT}_\Pi(G)) \cdot |V(G)|^{\mathcal{O}(1)}$, to $f(\operatorname{OPT}_\Pi(G) - |\mathcal{E}_c^\Pi(G)|) \cdot |V(G)|^{\mathcal{O}(1)}$. This leads to the following results.

Theorem 7.1. *For each problem Π with coefficient c and parameter dependence f listed in Table 7.1 that is not W[1]-hard, there is an algorithm that, given a graph G, outputs an optimal solution in time $f(\ell) \cdot |V(G)|^{\mathcal{O}(1)}$, where $\ell := \operatorname{OPT}_\Pi(G) - |\mathcal{E}_c^\Pi(G)|$ is the number of vertices in an optimal solution which are not c-essential.*

Hence the running time for solving these problems does not depend on the total size of an optimal solution, only on the part of the solution that does not consist of c-essential vertices. The theorem effectively shows that by employing c-Essential detection for Π as a preprocessing step, the size of the search space no longer depends on the total solution size but only on its non-essential vertices.

We also prove limitations to this approach. Assuming FPT $\neq$ W[1], for Dominating Set, Perfect Deletion (in which the goal is to obtain a perfect graph by vertex deletions), and Wheel-Free Deletion, there is no polynomial-time algorithm for c-Essential detection for any $c \in \mathbb{R}_{\geq 1}$. In fact, we can even rule out such algorithms running in time $f(k) \cdot |V(G)|^{\mathcal{O}(1)}$.

Table 7.1: For each problem Π, there is a polynomial-time algorithm for c-Essential detection for the stated value of c. Combined with the state of the art algorithm for the natural parameterization, this leads to an algorithm solving the problem in time $f(\ell) \cdot |V(G)|^{\mathcal{O}(1)}$ where $\ell = \mathrm{OPT}_\Pi(G) - |\mathcal{E}_c^\Pi(G)|$.

Problem	c	$f(\ell)$	Reference
Vertex Cover	2	1.2738^ℓ	[39]
Feedback Vertex Set	2	2.7^ℓ	[128]
Directed Feedback Vertex Set	2	$4^\ell \cdot \ell!$	[41]
Odd Cycle Transversal	2	2.3146^ℓ	[134]
Directed Odd Cycle Transversal	3	W[1]-hard	[137]
Chordal Vertex Deletion	13	$2^{\mathcal{O}(\ell \log \ell)}$	[38]

These results are based on FPT-inapproximability results for Dominating Set [118] and existing reductions [101, 130] to the mentioned vertex-deletion problems.

Techniques. The main work lies in the algorithms for c-Essential detection, which are all based on covering/packing duality for forbidden induced subgraphs to certain graph classes, or variations thereof in terms of (integer) solutions to certain linear programs and their (integer) duals. To understand the relation between detecting essential vertices and covering/packing duality, consider the Odd Cycle Transversal problem (OCT). Following the argumentation for the classic Erdős-Pósa theorem [65], in general there is no constant c such that any graph either has an odd cycle transversal of size $c \cdot k$, or a packing of k vertex-disjoint odd cycles (certified by Escher walls [158]). However, we show that a linear packing/covering relation holds in the following slightly different setting. If $G - v$ is bipartite (so all odd cycles of G intersect v), then the minimum size of an OCT avoiding v equals the maximum cardinality of a packing of odd cycles which pairwise intersect only at v. We can leverage this statement to prove that any vertex v which is not at the center of a flower (see Definition 7.3) of more than $\mathrm{OPT}_{\mathrm{oct}}(G)$ odd cycles, is not 2-essential: for any optimal solution X containing v, the graph $G' := G - (X \setminus \{v\})$ becomes bipartite after removal of v and by assumption does not contain a packing of more than $\mathrm{OPT}_{\mathrm{oct}}(G)$ odd cycles pairwise intersecting at v. So by covering/packing duality on G', it has an OCT X' of size at most $\mathrm{OPT}_{\mathrm{oct}}(G)$ avoiding v, so that $(X \setminus \{v\}) \cup X'$ is a 2-approximation in G which avoids v, showing that v is not 2-essential. Since v is clearly contained in an optimal solution whenever

there is a flower of more than $\mathrm{OPT}_{\mathrm{oct}}(G)$ odd cycles centered at v, this yields a method for c-Essential detection when using a known reduction [90] to Maximum Matching to test for a flower of odd cycles.

Organization. After presenting preliminaries in Section 7.2, we give algorithms to detect essential vertices based on covering/packing duality in Section 7.3 and based on integrality gaps in Section 7.4. In Section 7.5 we show how these detection subroutines can be used to improve the parameter dependence of FPT algorithms parameterized by solution size. The lower bounds are presented in Section 7.6. We conclude in Section 7.7.

7.2 Preliminaries for essential vertices

In this chapter we consider finite simple graphs, some of which are directed. The minimum size of an $\mathcal{H}$-modulator in G is denoted $\mathrm{OPT}_{\mathcal{H}}(G)$, that is, the size of an optimal solution for $\mathcal{H}$-Deletion.

Throughout this chapter we consider hereditary graph classes $\mathcal{H}$. Recall that these can be characterized by a (possibly infinite) set of forbidden induced subgraphs, we will denote them by $\mathsf{forb}(\mathcal{H})$. The $\mathcal{H}$-Deletion problem is equivalent to finding a minimum set $S \subseteq V(G)$ such that no induced subgraph of $G - S$ is isomorphic to a graph in $\mathsf{forb}(\mathcal{H})$. We say that such a set S hits all induced copies of $\mathsf{forb}(\mathcal{H})$ in G.

For a graph G and a set $T \subseteq V(G)$, a T-path is a path P in G of length at least one (so at least one edge), such that both endpoints of P are contained in T. A T-path is odd if it has an odd number of edges. There is a polynomial-time algorithm due to a theorem of Gallai that computes a maximum cardinality packing of pairwise vertex-disjoint T-paths.

Theorem 7.2 ([48, Theorem 9.2])**.** *Given a graph G, a set $T \subseteq V(G)$, and an integer s, one can in polynomial time either*

1. *find a family of $s+1$ pairwise vertex-disjoint T-paths, or*
2. *conclude that no such family exists and, moreover, find a set B of at most $2s$ vertices, such that in $G - B$ no connected component contains more than one vertex of T.*

The maximum cardinality k of such a packing of pairwise vertex-disjoint T-paths is the largest integer k such that the theorem above applied with integer $k-1$ results in the first outcome.

Whenever we refer to a (u, v)-separator, we mean a restricted $(\{u\}, \{v\})$-separator, that is, a set $S \subseteq V(G) \setminus \{u, v\}$ whose removal disconnects u from v.

Directed graphs. This chapter also covers some variants of $\mathcal{H}$-Deletion problems where the input graph is directed. Here the task is to find a minimum cardinality set of vertices whose removal results in a directed graph contained in $\mathcal{H}$, where $\mathcal{H}$ is a class of directed graphs. A directed graph G consists of a set of vertices $V(G)$ and a set of edges (or arcs) $E(G) \subseteq V(G) \times V(G)$. Here an edge $(u, v) \in E$ is an ordered pair indicating that the edge is directed from u to v. The main notions we need are those of walks, paths, cycles, and separators. Their definitions are similar to their undirected counterparts (see Chapter 2), but with directed arcs between consecutive pairs of vertices instead. For instance, a directed walk of length $\ell \geq 0$ in a graph G is a sequence $(v_1, \ldots, v_{\ell+1})$ of $\ell + 1$ vertices such that $(v_i, v_{i+1}) \in E(G)$ for each $i \in [\ell]$. Directed closed walks, directed paths, and directed cycles are defined analogously, with the slight alteration that directed graphs can have cycles of length one and two. Again these are called odd if their length (number of edges) is odd. Every directed graph that contains an odd directed closed walk contains an odd directed cycle. Finally a (u, v)-separator in a directed graph G is a set $S \subseteq V(G) \setminus \{u, v\}$ such that in $G - S$ there is no directed path from u to v. Menger's theorem also applies to directed graphs: Theorem 2.1 holds where 'graph' is replaced by 'directed graph' and 'path' by 'directed path'.

We consider the Directed Feedback Vertex Set and Directed Odd Cycle Transversal problem, that correspond to finding a minimum cardinality vertex set that intersects every directed cycle and every odd directed cycle respectively. Since a cycle of length one (a self-loop) is part of every optimal solution for both problems, without loss of generality we assume these are already removed from the graph.

7.3 Positive results via packing covering

In this section we provide polynomial-time algorithms for c-Essential detection for Π for various problems Π. The case for the Vertex Cover problem was given in Section 7.1. The results in this section are all based on packing/covering duality (cf. [42], [166, §73]). Towards this end, we generalize the notion of *flowers*, which played a key role in kernelization for Feedback Vertex Set [27]. While flowers were originally formulated as systems of cycles (forbidden structures for Feedback Vertex Set) pairwise intersecting in a

single common vertex, we generalize the notion to near-packings of arbitrary structures here.

Definition 7.3. Let $\mathcal{F}$ be a set of graphs. For a graph G and $v \in V(G)$, a *$(v,\mathcal{F})$-flower with p petals* in G is a set $\{C_1, C_2, \ldots, C_p\}$ of induced subgraphs of G such that each C_i (with $i \in [p]$) is isomorphic to some member of $\mathcal{F}$ and such that $V(C_i) \cap V(C_j) = \{v\}$ for all distinct $i, j \in [p]$. The *$\mathcal{F}$-flower number* of a vertex $v \in V(G)$, denoted $\Gamma_{\mathcal{F}}(G, v)$, is the largest integer p for which there is a $(v,\mathcal{F})$-flower in G with p petals.

We show a general theorem for finding 2-essential vertices for $\mathcal{H}$-Deletion if a maximum $(v,\mathsf{forb}(\mathcal{H}))$-flower can be computed in polynomial-time. It applies to those classes $\mathcal{H}$ where graphs with $G - v \in \mathcal{H}$ obey a min-max relation between $\mathcal{H}$-modulators avoiding v and packings of forbidden induced subgraphs intersecting only at v.

Theorem 7.4. *Let $\mathcal{H}$ be a hereditary graph class such that, for any graph G and vertex $v \in V(G)$ with $G - v \in \mathcal{H}$, the minimum size of an $\mathcal{H}$-modulator avoiding v in G equals $\Gamma_{\mathsf{forb}(\mathcal{H})}(G, v)$. Suppose there exists a polynomial-time algorithm $\mathcal{A}$ that, given a graph G and vertex $v \in V(G)$, computes $\Gamma_{\mathsf{forb}(\mathcal{H})}(G, v)$. Then there is a polynomial-time algorithm that solves* 2-Essential detection for $\mathcal{H}$-Deletion.

Proof. Apply algorithm $\mathcal{A}$ to each vertex $v \in V(G)$ and output the set S of vertices for which it finds that $\Gamma_{\mathsf{forb}(\mathcal{H})}(G, v) > k$. We argue that Properties **G1** and **G2** are satisfied. If $\mathrm{OPT}_{\mathcal{H}}(G) \leq k$, then every vertex in S is contained in every optimal solution for G since a size-k solution cannot hit a flower of $k+1$ petals from $\mathsf{forb}(\mathcal{H})$ without using v. Therefore Property **G1** is satisfied. Next suppose that $\mathrm{OPT}_{\mathcal{H}}(G) = k$ and let X be an optimal solution. We argue that each vertex $v \notin S$ is not 2-essential. Clearly this holds for any vertex $v \notin X$, so suppose that $v \in X$. Note that for every vertex $v \notin S$ we have $\Gamma_{\mathsf{forb}(\mathcal{H})}(G, v) \leq k$, which implies that $\Gamma_{\mathsf{forb}(\mathcal{H})}(G', v) \leq k$ where $G' := G - (X \setminus \{v\})$. Note that since $G' - v \in \mathcal{H}$, by assumption there exists an $\mathcal{H}$-modulator $X' \subseteq V(G') \setminus \{v\}$ in G' of size $\Gamma_{\mathsf{forb}(\mathcal{H})}(G', v) \leq k$. Observe that $(X \setminus \{v\}) \cup X'$ is an $\mathcal{H}$-modulator in G of size at most $2k$ that avoids v and therefore v is not 2-essential. □

We give two straightforward applications of Theorem 7.4, namely for the Feedback Vertex Set (FVS) problem and its directed variant. The FVS problem corresponds to $\mathcal{H}$-Deletion where $\mathcal{H}$ is the class of acyclic graphs and $\mathsf{forb}(\mathcal{H})$ is the set of cycles.

Lemma 7.5. *There is a polynomial-time algorithm for* 2-Essential detection for FVS.

Proof. We argue that the preconditions of Theorem 7.4 hold. The existence of a polynomial-time algorithm that computes $\Gamma_{\mathsf{forb}(\mathcal{H})}(G, v)$ follows from Theorem 7.2, since a flower of cycles through v corresponds to a collection of paths connecting pairs of distinct vertices of $N_G(v)$ to each other in the graph $G - v$. By applying Theorem 7.2 to $G - v$ with $T = N_G(v)$, we get that $\Gamma_{\mathsf{forb}(\mathcal{H})}(G, v)$ is the largest integer k such that the outcome of the theorem applied with the integer $k - 1$ is a family of k pairwise vertex-disjoint T-paths in $G - v$. For the first precondition of Theorem 7.4, we argue that when $G - v$ is acyclic, the maximum number of petals in a $(v,\mathsf{forb}(\mathcal{H}))$-flower is equivalent to the minimum cardinality of a set S that hits all cycles through v.

Claim 7.6 ([15, page 67])**.** *Let T be a tree and let $\mathcal{F}$ be a collection of connected subgraphs of T. The maximum size of a packing of vertex-disjoint members of $\mathcal{F}$ equals the minimum size of a vertex set intersecting all of $\mathcal{F}$.*

By noting that any cycle through v in G corresponds to a path between two neighbors in a connected component of $G - v$ which is a tree, the claim above directly implies the desired result. It follows there is a polynomial-time algorithm for 2-Essential detection for FVS by Theorem 7.4. □

The DFVS problem corresponds to a directed version of $\mathcal{H}$-Deletion where $\mathcal{H}$ is the set of directed acyclic graphs and $\mathsf{forb}(\mathcal{H})$ is the set of directed cycles. An algorithm for finding 2-essential vertices for this problem follows by a simple application of Menger's theorem.

Lemma 7.7. *There is a polynomial-time algorithms for* 2-Essential detection for DFVS.

Proof. The preconditions of Theorem 7.4 follow from known results, see for instance the work of Fleischer et al. [73], we repeat the argument for completeness. Consider any directed graph D and vertex $v \in V(D)$. Obtain a graph D' by splitting v into v_i and v_o. Attach every incoming arc of v to v_i and every outgoing arc to v_o. Compute a minimum (v_i, v_o)-separator $S \subseteq V(D') \setminus \{v_i, v_o\}$ in D'. By Menger's theorem, its size is equivalent to the maximum number of internally vertex-disjoint paths from v_i to v_o. Since any (v_i, v_o)-path in D' corresponds to a directed cycle containing v in D, it follows that $\Gamma_{\mathsf{forb}(\mathcal{H})}(D, v)$ can be computed in polynomial time. Finally, suppose that $D - v$ is a directed acyclic graph. Note that then the separator S constructed above is an $\mathcal{H}$-modulator avoiding v of size $\Gamma_{\mathsf{forb}(\mathcal{H})}(D, v)$. By Theorem 7.4 it follows that there is a polynomial-time algorithm for 2-Essential detection for DFVS. □

7.3.1 Odd cycle transversal

Next, we consider Odd Cycle Transversal (OCT), which corresponds to $\mathcal{H}$-Deletion where $\mathcal{H}$ is the class of bipartite graphs and $\mathsf{forb}(\mathcal{H})$ consists of all odd cycles. In order to apply Theorem 7.4 to OCT, we first argue that the class of bipartite graphs satisfies the preconditions. The proof is similar to that of Geelen et al. [90, Lemma 11] who reduce the problem of packing odd cycles containing v to a matching problem. We note that, although their result can be used to obtain a 3-essential detection algorithm, we will show (Lemma 7.11) how to efficiently detect 2-essential vertices as well. If the graph resulting from their construction has a large matching, then there is a large $(v,\mathsf{forb}(\mathcal{H}))$-flower. If on the other hand there is no large matching, then the Tutte-Berge formula is used to obtain a set of size $2k$ that hits all the odd cycles passing through v. We show that if the graph $G-v$ is bipartite instead, then this second argument can be improved to obtain a hitting set of size k by noting that a T-path is odd if and only if its endpoints are in different parts of a 2-coloring of $G-v$. We can then obtain the desired hitting set using Menger's theorem.

Lemma 7.8. *For any undirected graph G and set $T \subseteq V(G)$, a maximum packing of odd T-paths can be computed in polynomial time. Moreover, if G is bipartite then the cardinality of a maximum packing of odd T-paths is equal to the minimum size of a vertex set which intersects all odd T-paths.*

Proof. We reduce to matching as in [90, Lemma 11]. Construct a graph H as follows. For each $v \in V(G) \setminus T$, let $v' \notin V(G)$ be a copy of v. Let $V(H) = V(G) \cup \{v' \mid v \in V(G) \setminus T\}$ and $E(H) = E(G) \cup \{u'v' \mid uv \in E(G-T)\} \cup \{vv' \mid v \in V(G) \setminus T\}$. Note that the graph H consists of the disjoint union of G and a copy of $G-T$, with an added edge between $v \in V(G) \setminus T$ and its copy v'. Geelen et al. [90] mention that there is a 1-1 correspondence between odd T-paths in G and certain augmenting paths in H. For completeness we give a self-contained argument.

Claim 7.9. *Graph G contains k vertex-disjoint odd T-paths if and only if H has a matching M of size $|V(G) \setminus T| + k$. Furthermore, given a matching M in H of size $|V(G) \setminus T| + k$ we can compute a set of k vertex-disjoint odd T-paths in polynomial time.*

Proof. ($\Rightarrow$) Let $\mathcal{P} = (P_1, \ldots, P_k)$ be a set of k vertex-disjoint odd T-paths in G. Consider a path $P = (v_1, \ldots, v_{2\ell}) \in \mathcal{P}$, where $\ell \geq 1$. First note that we can assume that $V(P) \cap T = \{v_1, v_{2\ell}\}$, since if $v_i \in T$ for some $1 < i < 2\ell$, then either $(v_1, \ldots, v_i)$ or $(v_i, \ldots, v_{2\ell})$ is an odd T-path and we can update $\mathcal{P}$ accordingly. Construct a matching M in H as follows. For any

path $P = (v_1, \ldots, v_{2\ell}) \in \mathcal{P}$, add the edges $v_1v_2, v_2'v_3', \ldots, v_{2\ell-1}v_{2\ell}$, alternating between original vertices and copy vertices. This is possible as P is of odd length. For any vertex in $u \in V(G) \setminus T$ that is not contained in an odd T-path, we add uu' to M. Observe that at least $2|V(G) \setminus T| + 2k$ vertices are matched, therefore $|M| \geq |V(G) \setminus T| + k$ as desired.

($\Leftarrow$) Let M be a matching of size $|V(G) \setminus T| + k$ in H. If M contains both uv and $u'v'$ for $u, v \in V(G) \setminus T$, then update M by removing them and inserting uu' and vv' instead. If for $v \in V(G) \setminus T$ only one of v and v' is matched, and it is not matched to its copy, then match it to its copy instead. Afterwards let $E' := \{uv \in E(G) \mid uv \in M \vee u'v' \in M\}$. Observe that in $G[E']$, each vertex in $V(G) \setminus T$ has degree 0 or 2. For each $v \in V(G) \setminus T$ such that v has degree 0 in $G[E']$, add vv' to M if it is not in already. Note that all vertices of $H - T$ are matched. It follows that at least $2k$ vertices in T are matched by M and they have degree 1 in $G[E']$. Observe that $G[E']$ is a collection of paths and cycles with all degree-1 vertices in T. We get that there are k T-paths in G that are of odd length by construction (every even numbered edge in $G[E']$ originated from the copy part of H). Note that we can find them in polynomial time. ■

Since a maximum matching can be computed in polynomial time, by the claim above we get that a maximum packing of vertex-disjoint odd T-paths can be computed in polynomial time. Next we prove the second part of the statement.

Claim 7.10. *For a bipartite graph G and $T \subseteq V(G)$, the maximum cardinality of a packing of odd T-paths is equal to the minimum size of a vertex set which intersects all odd T-paths.*

Proof. Observe that any T-path is contained in a single connected component of G. Furthermore since G is bipartite, each connected component has a unique 2-coloring (up to reversal of colors). Consider a 2-coloring $c\colon V(G) \to \{1, 2\}$ of G and let $A = c^{-1}(1)$ and $B = c^{-1}(2)$. A T-path is odd if and only if one of its endpoints is in A and the other is contained in B. The claim then follows from Menger's theorem (Theorem 2.1): the maximum cardinality of a packing of odd T-paths is equivalent to a maximum cardinality packing of $(A \cap T, B \cap T)$-paths, which in turn is equal to the minimum size of a $(A \cap T, B \cap T)$-separator. ■

This concludes the proof of Lemma 7.8. □

By observing that odd T-paths in $G - v$ directly correspond to flowers with of cycles pairwise intersecting at v in G, Lemma 7.8 and Theorem 7.4 imply the following.

Lemma 7.11. *There is a polynomial-time algorithm for* 2-Essential detection for OCT.

7.3.2 Directed odd cycle transversal

The DOCT problem corresponds to $\mathcal{H}$-Deletion where $\mathsf{forb}(\mathcal{H})$ consists of all directed cycles of odd length. Using Menger's theorem on an auxiliary graph, we can detect 3-essential vertices for this problem.

Lemma 7.12. *There is a polynomial-time algorithm for* 3-Essential detection for DOCT.

Proof. Consider an instance (D, k) of 3-Essential detection for DOCT. Construct the so called label-extended graph (cf. [9, Section 3.1]) D' (initially empty) as follows.

1. For each $v \in V(D)$, add v' and v'' to $V(D')$.
2. For each directed arc $(u, v) \in E(D)$, add the arcs (u', v'') and (u'', v') to $E(D')$.

We can now solve the problem using D'. For each vertex $v \in V(D)$, compute a minimum (v', v'')-separator S_v in D'. Observe that v' and v'' are not adjacent in D' since D had no self-loops (see Section 7.2), and therefore these restricted (v', v'')-separators exist. Let $Q = \{v \in V(D) \mid |S_v| \geq 2k + 1\}$. We argue that Q satisfies the output requirements **G1** and **G2**.

First, suppose that $\mathrm{OPT}_{\mathrm{DOCT}}(G) \leq k$. Consider a vertex $v \in Q$. Since $|S_v| \geq 2k+1$, by Menger's theorem there is a collection $P'_1, \ldots, P'_{|S_v|}$ of internally vertex-disjoint (v', v'')-paths in D'. Let $P_i = \{u \mid \{u', u''\} \cap V(P'_i) \neq \emptyset\}$ and $\mathcal{P} = \bigcup_i \{P_i\}$.

Claim 7.13. *$D[P_i]$ contains an odd directed cycle for each $i \in [|S_v|]$. Furthermore, each $u \in V(D) \setminus \{v\}$ intersects at most two vertex sets of $\mathcal{P}$.*

Proof. Each (v', v'')-path P'_i in D' corresponds to a directed closed walk P_i in D. Since P'_i starts at v' and ends at v'', while every edge in D' switches between parities, it follows that P_i is an odd closed walk in D. This in turn implies that $D[V(P_i)]$ contains a directed odd cycle. To see the second point,

note that the paths in D' are vertex-disjoint and since we created two copies of each vertex, it follows that each vertex of $V(D) \setminus \{v\}$ intersects at most two vertex sets of $\mathcal{P}$. ■

By the claim above, any solution to DOCT avoiding $v \in Q$ has size at least $k+1$. It follows that each vertex in Q belongs to every optimal solution and therefore Property **G1** is satisfied. Next, suppose that $\mathrm{OPT}_{\mathrm{DOCT}}(G) = k$. We argue that Q contains all 3-essential vertices. Consider an optimal solution X and a vertex $v \notin Q$. If $v \notin X$, then clearly v is not 3-essential and there is nothing to show, so suppose that $v \in X$. Let $X' = \{u \in V(D) \mid \{u', u''\} \cap S_v \neq \emptyset\}$. Note that $|X'| \leq |S_v| \leq 2k$. Since any odd cycle containing v corresponds to a (v', v'')-path in D', and since S_v is a (v', v'')-separator in D', it follows that $(X \setminus \{v\}) \cup X'$ is a solution of size at most $k - 1 + 2k < 3k$ that avoids v. It follows that v is not 3-essential, therefore Q contains all 3-essential vertices and Property **G2** is satisfied. □

We cannot use the approach based on computing a maximum $(v, \mathcal{F})$-flower for the Chordal Deletion problem. Below we give a simple reduction from Disjoint Paths [162] that shows that it is NP-hard to compute a maximum $(v, \mathcal{F})$-flower where $\mathcal{F}$ is the set of chordless cycles of length at least four. In the next section, we will therefore use a different approach based on linear-programming relaxations to deal with Chordal Deletion.

Lemma 7.14. *Computing a maximum $(v, \mathcal{F})$-flower where $\mathcal{F}$ is the set of chordless cycles of length at least four is* NP*-hard.*

Proof. Consider the NP-hard Disjoint Paths problem. Here we are given a graph G and pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$ of vertices of G. The task is to decide if there exist paths $P_1, \ldots, P_\ell$ that are pairwise vertex disjoint such that P_i is a $s_i t_i$-path in G. Starting from an instance $(G, (s_1, t_1), \ldots, (s_\ell, t_\ell))$ of Disjoint Paths satisfying $s_i t_i \notin E(G)$ for all $i \in [\ell]$ (which is without loss of generality), obtain a graph G' by inserting a vertex v^* adjacent to $A = \bigcup_{i=1}^{\ell} \{s_i, t_i\}$ and inserting all edges between vertices in A except $s_i t_i$ for each $i \in [\ell]$. Then the Disjoint Paths instance is a yes-instance if and only if G' has a $(v^*, \mathcal{F})$-flower of size ℓ. □

7.4 Positive results via linear programming

Consider the following natural linear program for $\mathcal{H}$-Deletion for hereditary graph classes $\mathcal{H}$. The LP corresponding to an input graph G is defined on the variables $(x_u)_{u \in V(G)}$, as follows.

$\mathcal{H}$-Deletion LP
Objective: minimize $\sum_{u \in V(G)} x_u$.
Subject to:

- $\sum_{u \in V(H)} x_u \geq 1$ for each induced subgraph H of G isomorphic to a graph in $\mathsf{forb}(\mathcal{H})$, and
- $0 \leq x_u \leq 1$ for each $u \in V(G)$.

In the corresponding integer program, the constraint $0 \leq x_u \leq 1$ is replaced by $x_u \in \{0, 1\}$. We say that a minimization LP has *integrality gap* at most c for some $c \in \mathbb{R}$ if the cost of an integer optimum is at most c times the cost of a fractional optimum. In general, the number of constraints in the $\mathcal{H}$-Deletion LP can be exponential in the size of the graph. Using the ellipsoid method (cf. [166]), this can be handled using a separation oracle: a polynomial-time algorithm that, given an assignment to the variables, outputs a violated constraint if one exists. It is well-known (cf. [166, Thm. 5.10]) that linear programs with an exponential number of constraints can be solved in polynomial time using a polynomial-time separation oracle. To detect essential vertices, the integrality gap of a slightly extended LP will be crucial. We define the v-Avoiding $\mathcal{H}$-Deletion LP for a graph G and distinguished vertex $v \in V(G)$ as the $\mathcal{H}$-Deletion LP with the additional constraint that $x_v = 0$. Hence its integral solutions correspond to $\mathcal{H}$-modulators avoiding v.

Theorem 7.15. *Let $\mathcal{H}$ be a hereditary graph class such that for each graph G and $v \in V(G)$ satisfying $G - v \in \mathcal{H}$, the integrality gap of v-Avoiding $\mathcal{H}$-Deletion on G is at most $c \in \mathbb{R}_{\geq 1}$. If there is a polynomial-time separation oracle for the $\mathcal{H}$-Deletion LP, then there is a polynomial-time algorithm for $(c+1)$-Essential detection for $\mathcal{H}$-Deletion.*

Proof. Given G and k, the detection algorithm initializes an empty vertex set S and proceeds as follows. For each $v \in V(G)$, it solves the v-Avoiding $\mathcal{H}$-Deletion LP on G in polynomial time using the ellipsoid method via the polynomial-time separation oracle. If the linear program has cost more than k, we add v to S. After having considered all vertices $v \in V(G)$, the resulting set S is given as the output.

To see that the output satisfies Property **G1**, assume that $\mathrm{OPT}_{\mathcal{H}}(G) \leq k$ and consider some optimal $\mathcal{H}$-modulator X of size at most k. If $v \notin X$ then X induces an integer feasible solution to the $\mathcal{H}$-Deletion LP that satisfies the additional constraint $x_v = 0$, so that the cost of the v-Avoiding $\mathcal{H}$-Deletion LP is at most k and therefore $v \notin S$. By contraposition, all vertices of S are

indeed contained in some minimum $\mathcal{H}$-modulator, namely X.

To see that the algorithm also satisfies Property **G2**, assume $\mathrm{OPT}_{\mathcal{H}}(G) = k$ and let X be a minimum $\mathcal{H}$-modulator of size k. We prove that S contains all $(c+1)$-essential vertices. Consider an arbitrary $v \notin S$; we will argue it is not $(c+1)$-essential by exhibiting a $(c+1)$-approximate modulator avoiding v. Since $v \notin S$, the v-Avoiding $\mathcal{H}$-Deletion LP has a (fractional) solution $\mathbf{x} = (x_u)_{u \in V(G)}$ of cost at most k. If $v \notin X$ then v was not $(c+1)$-essential, and there is nothing to show. So assume $v \in X$.

Restricting the assignment $\mathbf{x}$ to the vertices of the graph $G' := G - (X \setminus \{v\})$ yields a feasible solution $\mathbf{x}'$ for the v-Avoiding $\mathcal{H}$-Deletion LP on G', whose cost is at most that of $\mathbf{x}$ and therefore at most k. Note that $G' - v$ equals $G - X$ and therefore belongs to $\mathcal{H}$. By the precondition to the theorem, the integrality gap for v-Avoiding $\mathcal{H}$-Deletion on G' is at most c. Hence the solution $\mathbf{x}'$ can be rounded to an integral solution X' on G' of cost at most $c \cdot k$ and $v \notin X'$ due to the constraint $x_v = 0$. Since $G - ((X \setminus \{v\}) \cup X') = G' - X' \in \mathcal{H}$, it follows that $(X \setminus \{v\}) \cup X'$ is an $\mathcal{H}$-modulator of size at most $c \cdot k + k = (c+1)k$ avoiding v, which is therefore a $(c+1)$-approximation. Hence v is not $(c+1)$-essential whenever $v \notin S$. □

Using known results on covering versus packing for chordless cycles in near-chordal graphs, the approach above can be used to detect essential vertices for Chordal Deletion. For the class of chordal graphs, the corresponding set of forbidden induced subgraphs is the class hole of all holes, i.e., induced chordless cycles of length at least four.

Lemma 7.16 ([113, Lemma 1.3])**.** *There is a polynomial-time algorithm that, given a graph G and a vertex v such that $G - v$ is chordal, outputs a (v,hole)-flower with p-petals and a set $S \subseteq V(G) \setminus \{v\}$ of size at most $12p$ such that $G - S$ is chordal.*

Using this covering/packing statement, we can bound the relevant integrality gap and thereby detect essential vertices for Chordal Deletion.

Lemma 7.17. *There is a polynomial-time algorithm for* 13-Essential detection for Chordal Deletion.

Proof. We first argue that the integrality gap for v-Avoiding Chordal Deletion is bounded by 12 when $G - v$ is chordal. By Lemma 7.16, there is a value of p such that G contains both a (v, hole)-flower $\{C_1, \ldots, C_p\}$ with p-petals and a v-avoiding chordal modulator S of size at most $12p$. Due to the constraint $x_v = 0$, any fractional solution to the linear program has a cost of at least p, since $\sum_{u \in C_i} x_u \geq 1$ while $x_v = 0$ and the holes only intersect at v. At

the same time, an integral solution has cost at most $12p$ as S is such a solution. Hence the integrality gap is at most 12.

To invoke Theorem 7.15, it suffices to argue that there is a polynomial-time separation oracle for the linear program. Such a separation oracle is known (cf. [113, §10.1]), we repeat it here for completeness. To test whether an assignment $\mathbf{x} = (x_u)_{u \in V(G)}$ satisfies all constraints, it suffices to do the following. For each $u \in V(G)$ we find the minimum total weight of any hole involving u, as follows. For each pair of distinct non-adjacent $p, q \in N_G(u)$ we use Dijkstra's algorithm to find the minimum weight W of a path P from p to q in the graph $G - (N_G[u] \setminus \{p, q\})$ where the values of $\mathbf{x}$ are used as the vertex weights. There is a hole of weight less than 1 through (p, u, q) if and only if $W + x_u < 1$. Moreover, if $W + x_u < 1$ then by extending path P with the vertex u we find a hole whose total weight is less than 1 and therefore a violated constraint. The fact that we remove all vertices of $N_G[u]$ other than p and q ensures that the cycle we get in this way is induced, while the non-adjacency of p and q ensures it has at least four vertices. Hence after iterating over all choices of u, p, q we either find a violated constraint or conclude that the assignment is feasible. This shows that Theorem 7.15 is applicable and concludes the proof. □

7.5 Consequences for parameterized algorithms

In this section we show how the algorithms for c-Essential detection from the previous section can be used to solve $\mathcal{H}$-Deletion for various classes $\mathcal{H}$, despite the fact that the detection algorithms only work when certain guarantees on k are met. The main theorem connecting the detection problem to solving $\mathcal{H}$-Deletion is the following.

Theorem 7.18. *Let $\mathcal{A}$ be an algorithm that, given a graph G and an integer k, runs in time $f(k) \cdot |V(G)|^{\mathcal{O}(1)}$ for some non-decreasing function f and returns a minimum-size $\mathcal{H}$-modulator if there is one of size at most k. Let $\mathcal{B}$ be a polynomial-time algorithm for c-Essential detection for $\mathcal{H}$-Deletion. Then there is an algorithm that, given a graph G, outputs a minimum-size $\mathcal{H}$-modulator in time $f(\ell) \cdot |V(G)|^{\mathcal{O}(1)}$, where $\ell = \mathrm{OPT}_{\mathcal{H}}(G) - |\mathcal{E}_c(G)|$ is the c-non-essentiality.*

Proof. First we describe the algorithm as follows. For each $0 \leq k \leq |V(G)|$, let S_k be the result of running $\mathcal{B}$ on (G, k), let $G_k := G - S_k$, and let $b_k := k - |S_k|$.

Letting L be the list of all triples (G_k, S_k, b_k) sorted in increasing order by their third component b_k, proceed as follows. For each $(G_k, S_k, b_k) \in L$, run $\mathcal{A}$ on (G_k, b_k) to find a minimum $\mathcal{H}$-modulator $S_{\mathcal{A}}$ of size at most b_k, if one exists. If $|S_{\mathcal{A}}| = b_k$, then output $S_{\mathcal{A}} \cup S_k$ as a minimum $\mathcal{H}$-modulator in G.

To analyze the algorithm, we first argue it always outputs a solution. For the call with $k^* = \mathrm{OPT}_{\mathcal{H}}(G)$, both conditions of the detection problem are met. Hence by Property **G1** the set S_{k^*} is contained in a minimum modulator in G, so that $\mathrm{OPT}_{\mathcal{H}}(G - S_{k^*}) = \mathrm{OPT}_{\mathcal{H}}(G) - |S_{k^*}| = k^* - |S_{k^*}|$. Therefore graph $G_{k^*} = G - S_{k^*}$ has a modulator of size at most $b_{k^*} = \mathrm{OPT}_{\mathcal{H}}(G - S_{k^*})$ and none which are smaller, so that $\mathcal{A}$ correctly outputs a modulator of size b_{k^*}. In turn, this causes the overall algorithm to terminate with a solution.

Having established that the algorithm outputs a solution, we proceed to show that it outputs a minimum-size modulator whenever it outputs a solution (which may be in an earlier iteration than for $k^* = \mathrm{OPT}_{\mathcal{H}}(G)$). Let k' be the value of k that is reached when the algorithm outputs a solution $S_{\mathcal{A}} \cup S_{k'}$. Then we know:

1. algorithm $\mathcal{A}$ found a minimum-size modulator $S_{\mathcal{A}}$ in $G_{k'}$ of size at most $b_{k'}$, and

2. the set $S_{\mathcal{A}} \cup S_{k'}$ is a modulator in G, since $S_{\mathcal{A}}$ is a modulator in $G_{k'} = G - S_{k'}$, and therefore $\mathrm{OPT}_{\mathcal{H}}(G) \leq b_{k'} + |S_{k'}| = k'$.

To see that the algorithm is correct, notice that, since $\mathrm{OPT}_{\mathcal{H}}(G) \leq k'$, the set $S_{k'}$ is contained in some minimum-size modulator for G (Property **G1** of $\mathcal{B}$). Hence $\mathrm{OPT}_{\mathcal{H}}(G_{k'}) = \mathrm{OPT}_{\mathcal{H}}(G) - |S_{k'}|$. Since $\mathcal{A}$ outputs a minimum-size modulator if there is one of size at most $b_{k'}$, we have $|S_{\mathcal{A}}| = \mathrm{OPT}_{\mathcal{H}}(G) - |S_{k'}|$, so that $S_{\mathcal{A}} \cup S_{k'}$ is a feasible modulator of size $\mathrm{OPT}_{\mathcal{H}}(G)$ and therefore optimal.

Now we prove the desired running-time bound. First of all, notice that we can determine the list L in polynomial time by running $\mathcal{B}$ once for each value of k (which is at most $|V(G)|$). By how we sorted L, we compute $\mathcal{A}(G_k, b_k)$ only when $b_k \leq b_{k^*}$, as we argued above that if the algorithm has not already terminated, it does so after reaching $k^* = \mathrm{OPT}_{\mathcal{H}}(G)$. Hence the calls to algorithm $\mathcal{A}$ are for values of the budget b_k which satisfy $b_k \leq b_{k^*}$. We bound the latter, as follows.

Since $k^* = \mathrm{OPT}_{\mathcal{H}}(G)$, the set S_{k^*} found by $\mathcal{B}$ is a superset of the set $\mathcal{E}_c(G)$ of all of the c-essential vertices in G (Property **G2**). This means that we have

$$b_{k^*} = \mathrm{OPT}_{\mathcal{H}}(G) - |S_{k^*}| \leq \mathrm{OPT}_{\mathcal{H}}(G) - |\mathcal{E}_c(G)| = \ell,$$

so the parameter of each call to $\mathcal{A}$ is at most ℓ, giving the total time bound $f(\ell) \cdot |V(G)|^{\mathcal{O}(1)}$. □

Theorem 7.1 now follows from Theorem 7.18 via the algorithms for c-Essential detection given in the previous sections and the state-of-the-art algorithms for the natural parameterizations listed in Table 7.1. Although the latter may originally be stated for the decision version, using self-reduction they can easily be adapted to output a minimum solution if there is one of size at most k.

7.6 Hardness results

Given the positive results we saw in Sections 7.3 and 7.4, it is natural to seek problems Π for which c-Essential detection for Π is intractable. Here we show that c-Essential detection for Dominating Set is intractable for any $c \in \mathcal{O}(1)$ and then use this as a starting point to prove similar results for Hitting Set, Perfect Deletion, and Wheel-free Deletion.

A dominating set is a vertex set whose closed neighborhood is the entire graph. The domination number of a graph is the size of a minimum dominating set. The starting point for our reductions is the following result which states that it is W[1]-hard to solve Dominating set parameterized by solution size even on instances which have 'solution-size gaps'.

Lemma 7.19 ([118], cf. [66, Thm. 4])**.** *Let $F, f : \mathbb{N} \to \mathbb{N}$ be any computable functions with $F(x) > 1$ for any $x \in \mathbb{N}$. Assuming FPT $\neq$ W[1], there does not exist an algorithm that, given a graph G and non-negative integer k, runs in time $f(k) \cdot |V(G)|^{\mathcal{O}(1)}$ and distinguishes between the following two cases:*

- *Completeness: G has a dominating set of size k.*
- *Soundness: Every dominating set of G is of size at least $k \cdot F(k)$.*

All of our reductions in this section share a leitmotif. We start with a *gap* instance (G, k) of Dominating Set and map it to an instance G' of c-Essential detection for Π (for appropriate Π) equipped with a distinguished vertex v^* with the following property: (1) if G has domination number at most k, then no optimal solution for Π in G' contains v^*; (2) if G has domination number strictly greater than $c \cdot F(k)$ (for some appropriate F), then v^* is contained in every solution for Π of size at most $c \cdot F(k)$ in G'. Thus our hardness results will follow by combining reductions of this kind with Lemma 7.19.

Lemma 7.20. *There is a polynomial-time algorithm R that, given a graph G and integer k, outputs a graph $R(G, k)$ containing a distinguished vertex v^* such that:*

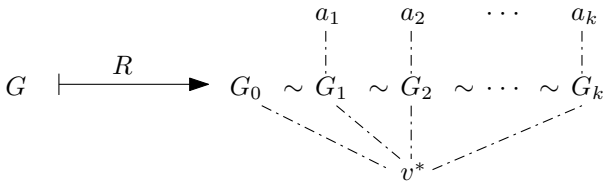


Figure 7.1: Reduction of Dominating Set to c-Essential detection for Dominating Set. For each $x \in V(G)$ and any two distinct copies x_i and x_j of x in G_i and G_j, x_i and x_j are true twins in $R(G,k)[V(G_i) \cup V(G_j)]$. Each a_i is an apex vertex to G_i (i.e. $N(a_i) = V(G_i)$) and v^* is an apex to the whole graph except $a_1, \ldots, a_k$.

- *if G has domination number at most k, then the domination number of $R(G,k)$ is exactly k and every optimal dominating set avoids v^*;*
- *if G has domination number strictly greater than $c \cdot (k+1)$ for some $c \in \mathbb{R}_{\geq 1}$, then $R(G,k)$ has domination number $k+1$ and the distinguished vertex v^* is contained in all $R(G,k)$-dominating sets of size at most $c \cdot (k+1)$.*

Proof. The graph $R(G,k)$ (see Figure 7.1) is defined formally as follows:

- Initialize $R(G,k)$ as the graph on vertex set $\{v_i \mid v \in V(G), 0 \leq i \leq k\}$ with edges $\{v_i u_j \mid uv \in E(G), 0 \leq i,j \leq k\} \cup \{v_i v_j \mid v \in V(G), 0 \leq i < j \leq k\}$.
- For each $i \in \{1, \ldots, k\}$ insert an apex a_i which is adjacent to $\{v_i \mid v \in V(G)\}$.
- Insert a vertex v^* which is adjacent to $\{v_i \mid v \in V(G), 0 \leq i \leq k\}$.

For our notational convenience, notice that the graph $R(G,k)$ contains $k+1$ isomorphic copies of G denoted as $G_0, \ldots, G_k$ where each G_i is defined as the induced subgraph $R(G,k)[\{v_i \mid v \in V(G)\}]$ of $R(G,k)$.

Notice that $R(G,k)$ has domination number at least k: since $\{a_1, \ldots, a_k\}$ is an independent set and the neighborhoods of each a_i are disjoint in $R(G,k)$, we have that any dominating set of $R(G,k)$ is forced to pick at least one vertex from $V(G_i) \cup \{a_i\}$ for each $1 \leq i \leq k$. In particular, this means that every dominating set that contains v^* must have cardinality at least $k+1$ (since v^* does not dominate $a_1, \ldots, a_k$).

Now, to prove that $R(G,k)$ has the desired properties, start by supposing that G has a dominating set $\{s_1,\ldots,s_\ell\}$ with ℓ at most k. We claim that, by 'spreading-out' this dominating set over $G_1,\ldots,G_\ell$ (i.e. pick the copy of s_i in G_i for each i) we obtain a vertex-subset S' of $R(G,k)$ which dominates v^*, $\{a_1,\ldots,a_\ell\}$ and every vertex of $G_0,\ldots,G_k$. The first two observations are immediate and the last follows since each G_i is a 'twin' of G_j in $R(G,k)$: any vertex y in G is dominated by at least one vertex, s_i of S and, by the construction of $R(G,k)$, the copy of s_i in G_i dominates all of the copies $y_0,\ldots,y_k$ of y in $G_0,\ldots,G_k$. It follows immediately from the definition of $R(G,k)$ that $S'\cup\{a_i : \ell < i \leq k\}$ is a dominating set of $R(G,k)$ of size k (which is smallest-possible by our previous arguments). Thus, by our previous discussion, all minimum-size dominating sets of $R(G,k)$ avoid v^*.

Now suppose there exists a constant $c \geq 1$ such that G has domination number greater than $c\cdot(k+1)$. Take any dominating set S of $R(G,k)$ not containing v^*. Since G_0 is not adjacent to any one of the vertices $a_1,\ldots,a_k$ and $v^*\notin S$, every vertex of G_0 must be dominated by some vertex of $G_0\uplus\cdots\uplus G_k$ in $R(G,k)$. Thus the obvious projection of $S\setminus\{a_1,\ldots,a_k,v^*\}$ onto G is a dominating set for G. Hence, if S avoids v^*, then $|S| > c\cdot(k+1)$ by our assumption on the domination number of G. In other words, every dominating set of size at most $c\cdot(k+1)$ must contain v^*. This, combined with the fact that any dominating set of $R(G,k)$ is forced to pick at least one vertex from $V(G_i)\cup\{a_i\}$ for each $1\leq i\leq k$ (as we observe earlier), implies that $\{v^*,a_1,\ldots,a_k\}$ is a minimum-size dominating set of $R(G,k)$ (where each a_i dominates itself and v^* dominates everything else) which has size $k+1$. □

Lemma 7.19 combined with the reduction provided by Lemma 7.20 yields the following.

Theorem 7.21. *Unless FPT = W[1], there is no FPT-time algorithm for* c-Essential detection for Dominating Set *parameterized by* k *for any* $c\in\mathbb{R}_{\geq 1}$.

Proof. Suppose such an algorithm $\mathcal{A}$ exists for c; we will use it with Lemma 7.19 to show FPT = W[1] for the function $F(k)=c(k+1)$.

Given an input instance (G,k) of Dominating Set in which the goal is to distinguish between the completeness and soundness cases, the algorithm proceeds as follows. Using the reduction R of Lemma 7.20, consider the graph $R(G,k)$ and let S be the output of an algorithm for c-Essential detection for Dominating Set on the pair $(R(G,k),k+1)$; note that the solution size for which we ask is $k+1$ rather than k. Without loss of generality we may assume $k\geq 2$, as the distinction can trivially be made otherwise. We will show

that in the completeness case we have $v^* \notin S$, while in the soundness case we have $v^* \in S$, which allows us to distinguish between these cases by checking whether v^* belongs to the output of $\mathcal{A}(R(G,k), k+1)$.

For the completeness case, suppose G has domination number at most k. Then, by Lemma 7.20, so does $R(G,k)$. This means that Property **G1** holds for the call to $\mathcal{A}(R(G,k), k+1)$, so that there is some optimal solution S' of size k which contains S and hence we have $v^* \notin S$ by Lemma 7.20.

For the soundness case, suppose G has domination number at least $k \cdot F(k) = c(k+1)k > c(k+1)$ (we use $k \geq 2$ here). Then by Lemma 7.20, graph $R(G,k)$ has domination number $k+1$ and v^* is contained in all its dominating sets of size at most $c(k+1)$. In other words: v^* is c-essential in $R(G,k)$. Consequently, $v^* \in S$ by Property **G2** since the argument $k+1$ we supplied to $\mathcal{A}$ coincides with the optimum in $R(G,k)$ in this case.

If $\mathcal{A}$ runs in FPT-time, then the overall procedure runs in FPT-time which implies FPT = W[1] by Lemma 7.19. □

In order to keep using the language of graphs, we view Hitting Set instances with universe U and collection of sets $\mathcal{F}$ as a hypergraph where U is the vertex set and $\mathcal{F}$ is the set of hyperedges. Consider the closed-neighborhood mapping S, that is, the standard polynomial-time reduction from Dominating Set to Hitting Set which maps each graph G to the hypergraph $S(G)$ that is defined as follows.

Definition 7.22. For a graph G, the hypergraph $S(G)$ has vertex set $V(G)$ and hyperedges $\{N_G[x] \mid x \in V(G)\}$.

The following observation captures the relation between dominating sets of G and hitting sets of $S(G)$.

Observation 7.23. *Let G be a graph and $X \subseteq V(G)$. Then X is a dominating set of G if and only if X is a hitting set of $S(G)$.*

Note that, given the reduction R of Lemma 7.20, the composite mapping $S \circ R$ relates c-essentiality to gaps in solution quality in much the same way as R did.

Now consider the parameter-preserving polynomial-time parameterized reduction P (due to Heggernes et al. [101, Thm. 1]) taking instances of Hitting Set with hyperedges of size at least two to Perfect Deletion. The reduction was also covered in Lemma 4.17, we repeat it for completeness.

Definition 7.24. Let H be a hypergraph with hyperedges of size at least two. Then $P(H)$ is the graph defined as follows:

- Initialize $P(H)$ as an edgeless graph on vertex set $V(H)$.
- For each hyperedge $e = \{x_1, \ldots, x_\ell\}$ in H, create $\ell + 1$ new set gadget vertices $\mathcal{G}_e = \{e_1, \ldots, e_{\ell+1}\}$ and add edges $\{e_1 e_{\ell+1}\} \cup \{x_i e_j \mid x_i \in e, j \in \{i, i+1\}\}$ to $P(H)$. Note that $\mathcal{G}_e \cup e$ induces an odd hole in $P(H)$ of length at least five.
- Take the pairwise join of the set gadgets, that is, make all vertices of $\mathcal{G}_e$ adjacent to all vertices of $\mathcal{G}_{e'}$ for each distinct $e, e' \in E(H)$.

The following properties of $P(H)$ are important for our hardness proof.

Lemma 7.25 ([101, Thm. 1 and Claims 1–3])**.** *Given any* Hitting Set *instance (H, k) with each hyperedge of size at least 2, the graph $P(H)$ defined in Definition 7.24 satisfies the following properties:*

P1 *given any vertex-subset X of $P(H)$ such that $P(H) - X$ is perfect; if there is an $x \in X$ such that $x \notin V(P(H)) \cap V(H)$, then x lies on exactly one odd hole C_e, and, for any vertex $y \in V(C_e) \cap V(H)$ we have that $P(H) - (X \cup \{y\}) \setminus \{x\}$ is perfect,*

P2 *for any $X \subseteq V(H)$, the set X is a hitting set in H if and only if $P(H) - X$ is perfect.*

Lemma 7.25 ensures that we can chain the reductions R, S and P (found respectively in Lemma 7.20, Definition 7.22, and Definition 7.24) to obtain a polynomial-time reduction from Dominating Set to the detection problem c-Essential detection for PerfDel with sufficient guarantees to be able to then infer the intractability of the latter problem using Lemma 7.19. Here we use shorthand PerfDel for Perfect Deletion.

Theorem 7.26. *Unless FPT = W[1], there is no FPT-time algorithm for c-*Essential detection for PerfDel *parameterized by k for any $c \geq 1$.*

Proof. Fix any gap instance (G, k) with $k \geq 2$ of Dominating Set which has domination number at most k or at least $ck(k+1) > c(k+1)$. Then, by Observation 7.23 and Lemma 7.20, the hypergraph $S(R(G, k))$ has a distinguished vertex v^* such that following properties are satisfied:

C1 if G has domination number at most k, then $S(R(G, k))$ has hitting set number k and all of its minimum hitting sets avoid v^*;

C2 if G has domination number strictly greater than $c(k+1)$, then $S(R(G, k))$ has hitting set number $k + 1$ and its distinguished vertex v^* is contained in every solution of size at most $c(k + 1)$.

Let Q be the set returned by any algorithm for c-Essential detection for PerfDel on input $\big((P \circ S \circ R)(G,k), k+1\big)$. Notice that, by Lemma 7.25, in both Case **C1** and Case **C2** we have that the optimum for PerfDel on $(P \circ S \circ R)(G,k)$ coincides with the optimum for HittingSet on $S(R(G,k))$.

With this observation in mind, consider Case **C1**. Here we have that Q is contained in an optimum solution Q' of size k (by Property **G1**). However, by the second point in Lemma 7.25 (and since all size-k hitting sets of $S(R(G,k))$ avoid v^*) we deduce that any such Q' (and hence Q) cannot contain v^* (which is a vertex of $S(R(G,k))$ and hence also a vertex of $(P \circ S \circ R)(G,k)$).

For Case **C2**, consider any perfect deletion set X in $(P \circ S \circ R)(G,k)$ of size at most $c(k+1)$. If we can show that X must always contain v^*, then we are done since it would imply that v^* is c-essential which, by Property **G2** would allow us to decide whether G has domination number either k or greater than $c(k+1)$ simply by checking whether v^* is in Q or not. So, seeking a contradiction, suppose $v^* \notin X$. By Property **P1** of Lemma 7.25 and since each hyperedge in $S(R(G,k))$ has size at least 2, we can find a subset Y of the vertices of $S(R(G,k)) - v^*$ of size at most $|X|$ such that $(P \circ S \circ R)(G,k) - Y$ is perfect. But then, by Property **P2** of Lemma 7.25, we have that Y is a hitting set of size at most $|X| = c(k+1) \leq ck(k+1)$ in $S(R(G,k))$ which avoids v^*. However, this is a contradiction since Case **C2** guarantees that v^* is contained in every hitting set for $S(R(G,k))$ of size at most $ck(k+1)$. □

In a similar vein to the reduction of Heggernes et al. [101, Thm. 1] (Definition 7.24) there is a reduction L from Hitting Set to Wheel-free Deletion (WheelDel for short) due to Lokshtanov [130]. A wheel W_q consists of a cycle of length $q \geq 3$ and a 'center' vertex adjacent to every vertex of the cycle. The WheelDel problem therefore asks to remove all such structures from the graph. The original reduction is defined as follows. Given an instance (H,k) of Hitting Set with $|V(H)| = n$, for each universe element $x \in V(H)$ add two vertices x_1 and x_2 and add an edge between them. Then for each hyperedge $e \in E(H)$, construct the wheel W_{3n} and distinguish an induced matching of length n in the cycle of the wheel. To each edge uv of the matching, assign an element of the universe $V(H)$, say x. If e contains x, also add special edges ux_1 and vx_2. Finally, contract all special edges. A sketch is given in Figure 7.2. With the goal of proving hardness for detecting essential vertices in mind, we need a slight modification to the reduction as currently solutions to WheelDel may interchangeably use either x_1 and x_2 for any $x \in V(H)$. This would cause our special vertex v^* in H to lose its essential properties. The reduction we need is defined as follows.

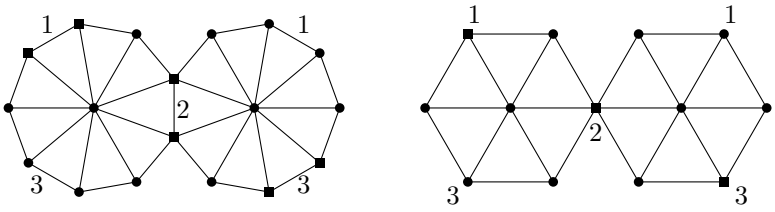


Figure 7.2: Sketch of reduction from Hitting Set to WheelDel for an instance with universe [3] and hyperedges $\{1, 2\}$ and $\{2, 3\}$. Square vertices indicate elements initally added for $V(H)$ that result from contracting special edges. Left the reduction due to Lokshtanov [130], where for each wheel the universe elements are mapped to an induced matching. Right the reduction as in Definition 7.27, where universe elements are mapped to every other vertex of the cycle.

Definition 7.27. Let H be a hypergraph with $n \geq 2$ vertices. Then $L(H)$ is the graph defined as follows.

- Initialize $L(H)$ as the edgeless graph on $V(H)$.
- For each hyperedge $e \in E(H)$, add a disjoint copy of a wheel W_{2n} with cycle $(v_1, \ldots, v_{2n})$. To each v_i with i being odd, assign a universe element of $V(H)$, say x, to v_i. If e contains x, then add the special edge $v_i x$.
- Finally contract all special edges into the universe element (the new vertex resulting from each contraction $v_j x$ should get label x so that $V(H) \subseteq V(L(H))$).

Again see Figure 7.2 for a sketch. The following properties are analogous to those in Lemma 7.25 for Perfect Deletion. Since the reduction is slightly changed, we briefly argue its correctness following the arguments in [130].

Lemma 7.28. *Given any* Hitting Set *instance (H, k) with at least 2 vertices, the graph $L(H)$ defined in Definition 7.27 satisfies the following properties:*

L1 *Given any vertex-subset X of $L(H)$ such that $L(H) - X$ is wheel-free; if there is an $x \in X$ such that $x \notin V(L(H)) \cap V(H)$, then x lies on exactly one wheel W, and, for any vertex $y \in V(W) \cap V(H)$ we have that $L(H) - (X \cup \{y\}) \setminus \{x\}$ is wheel-free.*

L2 *For any $X \subseteq V(H)$, the set X is a hitting set in H if and only if $L(H) - X$ is wheel-free.*

Proof. We first argue that the only vertices of $L(H)$ that can be centers of a wheel are the centers of W_{2n} that were added for each $e \in E(H)$. Clearly for each v_i of the cycle with i even, no special edges are incident to it, and so its neighborhood before contracting special edges is contained in W_{2n}. Therefore, after contracting special edges, the neighborhood of v_i induces a P_3 (path of length 2) in $L(H)$. This also holds for any v_i with i being odd that corresponds to a universe element that was not in the hyperedge e and therefore did not get any special edges. In the remaining case, after contracting special edges it may be that v_i (which corresponds to a universe element $x \in V(H)$) has neighbors in multiple wheels. This results in a neighborhood that is a disjoint union of P_3's, hence the vertex cannot be the center of a wheel. Since the neighborhood of the center of W_{2n} is an induced cycle, it follows that every vertex of $V(L(H)) \setminus V(H)$ is part of exactly one wheel and therefore we get Property **L1**.

For Property **L2**, suppose that $X \subseteq V(H)$ is a hitting set in H. To see that $L(H) - X$ is wheel-free, by the previous paragraph we only have to argue that the wheels W_{2n} that were added for each hyperedge are hit. By contracting the special edges, the cycle of each wheel contains all elements of $e \subseteq V(H)$. Since X hits e, it follows X hits W_{2n} and therefore $L(H) - X$ is wheel-free. In the other direction suppose $X \subseteq V(H)$ is a wheel-free deletion set of $L(H)$ (which exists due to Property **L1**). For every wheel W_{2n} that was added for hyperedge e, X contains at least one element of $e \subseteq V(H)$ that hits it and therefore X is a hitting set of H. □

Given Lemma 7.28, we can argue in similar vein to Theorem 7.26 to show the intractability of c-Essential detection for WheelDel. We leave out the proof as it would be identical to that of Theorem 7.26 with Lemma 7.25 replaced by Lemma 7.28.

Theorem 7.29. *Unless FPT = W[1], there is no FPT-time algorithm for* c-Essential detection for WheelDel *parameterized by* k *for any* c.

7.7 Conclusion

We introduced the notion of c-essential vertices for vertex-subset minimization problems on graphs, to formalize the idea that a vertex belongs to all *reasonable* solutions. Using a variety of approaches centered around the theme of covering/packing duality, we gave polynomial-time algorithms that detect a subset of an optimal solution containing all c-essential vertices. This decreased

the search space of parameterized algorithms from exponential in the size of the solution, to exponential in the number of non-essential vertices in the solution.

Throughout this chapter we have restricted ourselves to working with unweighted problems. However, many of the same ideas can be applied in the setting where each vertex has a positive integer weight of magnitude $\mathcal{O}(|V(G)|^{\mathcal{O}(1)})$ and we search for a minimum-weight solution. Since integral vertex weights can be simulated for many problems by making twin-copies of a vertex, our approach can be extended to Weighted Vertex Cover, Weighted Odd Cycle Transversal, and Weighted Chordal Deletion.

Our results shed a new light on which instances of NP-hard problems can be solved efficiently. FPT algorithms for parameterizations by solution size show that instances are easy when their optimal solutions are small. Theorem 7.1 refines this view: it shows that instances with large optimal solutions can still be easy, as long as only a small number of vertices in the optimum is not c-essential.

We remark that there is an alternative route to algorithms for c-Essential detection, which is applicable to $\mathcal{H}$-Deletion problems which admit a constant-factor approximation. If there is a polynomial-time algorithm that, given a graph G and vertex v, outputs a c-approximation for the problem of finding a minimum-size $\mathcal{H}$-modulator avoiding v, it can be used for c-Essential detection. A valid output S for the detection problem with input (G, k) is obtained by letting S contain all vertices for which the approximation algorithm outputs a v-avoiding modulator of size more than $c \cdot k$. Using this approach (cf. [79]) one can solve $\max_{F \in \mathcal{F}} |V(F)|^{\mathcal{O}(1)}$-Essential detection for $\mathcal{F}$-Minor-Free Deletion for any finite family $\mathcal{F}$ containing a planar graph. As the results for problems for which no constant-factor approximation exists are more interesting, we focused on those.

Our work opens up several questions for future work. Is the integrality gap for v-Avoiding Planar Vertex Deletion constant, when $G - v$ is planar? Can $\mathcal{O}(1)$-Essential detection for Planar Vertex Deletion be solved in polynomial time? Can 2-Essential detection for Chordal Deletion be solved in polynomial time? Can the constant c for which we can detect c-essential vertices be lowered below 2?

Considering a broader horizon, it would be interesting to investigate whether there are less restrictive notions than c-essentiality which can be used as the basis for guaranteed search-space reduction.

Part IV

Conclusion

Chapter 8

Concluding Remarks

In this thesis we have looked at $\mathcal{H}$-Deletion problems, where the task is to remove a minimum number of vertices to obtain a graph in a given graph class $\mathcal{H}$. Many of these problems are known to be FPT parameterized by the solution size, implying that instances with small solutions remain tractable in some sense. We explored two extensions of these tractable instances. In Part II we considered solutions which may be large, but are structured in some way. We looked at algorithms that use recently introduced hybrid parameters such as $\mathcal{H}$-treewidth, where essentially large induced subgraphs that belong to $\mathcal{H}$ with small neighborhoods are not counted towards the width. We have given an algorithm that computes an 8-approximate tree $\mathcal{H}$-decomposition whenever $\mathcal{H}$-Deletion is FPT by solution size (Theorem 5.1). Using dynamic programming techniques over these decompositions we are then able to solve $\mathcal{H}$-Deletion parameterized by $\mathbf{tw}_{\mathcal{H}}$ (Theorem 6.1). In their respective chapters we have mentioned possibilities for further work, such as improving approximation ratios, parameter dependencies, and polynomial factors of the given algorithms. The parameter dependency of our algorithm for $\mathcal{H}$-Deletion for $\mathcal{H}$ being the class of bipartite graphs or the class of planar graphs is as good as the best-known dependency on the natural parameter. It would be great to add more graph classes to this list. A first concrete candidate would be the Chordal Deletion problem. Our algorithm under the hybrid parameterization has a dependency of $2^{\mathcal{O}(k^2)}$ where $k = \mathbf{tw}_{\mathcal{H}}$, whereas under the natural parameterization there is an algorithm with parameter dependency $2^{\mathcal{O}(k \log k)}$ [38].

In Part III we looked at algorithms that identify part of an optimal solution, with the aim of speeding up a follow-up FPT algorithm that uses the natural solution-size parameterization. Our algorithms try to identify of vertices that

are somehow essential to creating good solutions. Roughly speaking, for a constant c depending on the graph class $\mathcal{H}$, for some $\mathcal{H}$-Deletion problems we are able to identify vertices that belong to every c-approximation in polynomial time. This results in algorithms for $\mathcal{H}$-Deletion where the parameter is the number of non-essential vertices of the instance rather than the total solution size. Section 7.7 mentions some possibilities for further work in this paradigm such as improving these constants c. One thing in particular is that the value of c for which we can find c-essential vertices is typically 2 or 3 (see Theorem 7.1). It remains open whether this is a limit, or if say 1.5-essential vertices can somehow be found for certain graph problems.

We conclude with some possible research directions that do not directly relate to the results of this thesis.

Superclass parameterization. The solution-size parameterization of $\mathcal{H}$-Deletion is essentially a parameterization by distance to $\mathcal{H}$. Another method to obtain parameters that are smaller than solution size is to parameterize by the distance to some superclass $\mathcal{G}$ of $\mathcal{H}$. Consider for instance Vertex Cover parameterized by distance to bipartite (observe that every edgeless graph is bipartite). By Kőnig's theorem [48, Thm. 2.11] a minimum vertex cover can be computed in polynomial time in bipartite graphs. Given a graph G and a bipartite deletion set S, a minimum vertex cover can be computed by the following procedure: for every subset X of S for which $S \setminus X$ is independent, compute a minimum vertex cover Y_X in the bipartite graph $G - S - N_G(S \setminus X)$ and output the minimum cardinality of $|X| + |N_G(S \setminus X) \setminus X| + |Y_X|$ over all options X. This simple branching algorithm implies that Vertex Cover is FPT parameterized by distance to bipartite. The same applies for any graph class $\mathcal{G}$ in which Vertex Cover is polynomial time solvable and for which computing a minimum $\mathcal{G}$-modulator is FPT parameterized by solution size, which holds for instance for chordal graphs [70, §4]. It would be interesting to see if there are more positive results of this type. Since Interval Deletion is NP-hard in chordal graphs, the parameterization 'distance to chordal' would not give positive results for this problem. On the other hand, Split Deletion (obtaining a graph that can be partitioned into an independent set and a clique) parameterized by distance to chordal or Cluster Deletion (obtaining a graph that is a disjoint union of cliques) parameterized by distance to interval could be considered. Preliminary investigation suggests that the former one is FPT.

The parameterization described in this paragraph is similar in spirit to that of 'above guarantee' parameterizations [120]. The difference is that the

parameter described here is considered to be the distance to $\mathcal{G}$, whereas the above guarantee parameterization would consider the difference between this lower bound (distance to $\mathcal{G}$) and the target solution size as the parameter.

Filling in the gaps for the natural parameterization. While this thesis focuses on obtaining FPT results for $\mathcal{H}$-Deletion for smaller parameters than the natural one, even for the natural parameter there remain some interesting open problems. Due to the work of Cai [35], it follows that $\mathcal{H}$-Deletion is FPT for any hereditary graph class $\mathcal{H}$ which is characterized by a finite set of forbidden induced subgraphs. No such general theorem is known for hereditary classes whose forbidden induced subgraph characterization is not finite. Results for these problems are obtained on a case by case basis, with a recent addition of an FPT result for the class of bipartite permutation graphs [29]. Hardness results for the class of perfect graphs [101] and wheel-free graphs [130] show that FPT algorithms cannot always be obtained. Is it possible to obtain a dichotomy that characterizes the classes $\mathcal{H}$ for which $\mathcal{H}$-Deletion is FPT parameterized by the natural parameter? Even for subclasses of perfect graphs such a classification would be interesting.

Other problems. This thesis heavily focuses on the $\mathcal{H}$-Deletion problem, but there are plenty of other interesting graph problems that are NP-hard in general graphs. First of all, $\mathcal{H}$-Deletion is part of a larger class of problems known as graph modification problems. Besides vertex deletions, you could also consider other operations such as edge removal, edge addition, and edge contraction to reach a graph with a certain desired property. These edge modification problems are typically more difficult to solve than their vertex-deletion counterpart. A survey of Crespelle et al. [47] gives an overview of the state of the art and lists many open problems in this paradigm.

Besides graph modification problems, there are many other interesting graph problems that are hard to solve in general graphs. A lot of work in classical complexity has resulted in restrictions of graph classes in which these problems remain polynomial time solvable. Consider for instance the Hamiltonian Cycle problem, which asks for the existence of a cycle that visits all vertices exactly once. The problem is NP-hard in general graphs, but is solvable in polynomial time in co-comparability graphs [55] (yet another subclass of perfect graphs). In the spirit of 'distance to triviality' parameterizations [98], it would be interesting to see if such results extend to FPT algorithms parameterized by distance to the graph class $\mathcal{H}$ in which they are polynomial time solvable. It is currently not known if Co-comparability Deletion parameterized by

solution size is FPT. Any positive results of this type could then be considered as a problem to be tackled using the techniques of Part II as mentioned in Section 6.6; do the 'deletion distance to $\mathcal{H}$' results also extend to the 'elimination distance to $\mathcal{H}$' parameterization?

Part V

Back Matter

Bibliography

[1] Tobias Achterberg, Robert E. Bixby, Zonghao Gu, Edward Rothberg, and Dieter Weninger. Presolve reductions in mixed integer programming. *INFORMS J. Comput.*, 32(2):473–506, 2020. doi:10.1287/ijoc.2018.0857.

[2] Isolde Adler, Martin Grohe, and Stephan Kreutzer. Computing excluded minors. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008*, pages 641–650. SIAM, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347153.

[3] Akanksha Agrawal, Lawqueen Kanesh, Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Deleting, eliminating and decomposing to hereditary classes are all FPT-equivalent. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 1976–2004. SIAM, 2022. doi:10.1137/1.9781611977073.79.

[4] Akanksha Agrawal, Lawqueen Kanesh, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. A fixed-parameter tractable algorithm for elimination distance to bounded degree graphs. *SIAM J. Discret. Math.*, 36(2):911–921, 2022. doi:10.1137/21m1396824.

[5] Akanksha Agrawal and M. S. Ramanujan. On the parameterized complexity of clique elimination distance. In *Proceedings of the 15th International Symposium on Parameterized and Exact Computation*, volume 180 of *LIPIcs*, pages 1:1–1:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.IPEC.2020.1.

[6] Akanksha Agrawal and M. S. Ramanujan. Distance from triviality 2.0: Hybrid parameterizations. In *Proceedings of the 33rd International*

Workshop on Combinatorial Algorithms, IWOCA 2022, volume 13270 of *Lecture Notes in Computer Science*, pages 3–20. Springer, 2022. doi:10.1007/978-3-031-06678-8_1.

[7] 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.

[8] Jochen Alber, Nadja Betzler, and Rolf Niedermeier. Experiments on data reduction for optimal domination in networks. *Annals OR*, 146(1):105–117, 2006. doi:10.1007/s10479-006-0045-4.

[9] Vedat Levi Alev and Lap Chi Lau. Approximating unique games using low diameter graph decomposition. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017*, volume 81 of *LIPIcs*, pages 18:1–18:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.18.

[10] Stefan Arnborg, Bruno Courcelle, Andrzej Proskurowski, and Detlef Seese. An algebraic theory of graph reduction. In *Graph Grammars and Their Application to Computer Science, Graph Grammars 1990*, pages 70–83, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg. doi:10.1007/BFb0017382.

[11] Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. A complexity dichotomy for hitting connected minors on bounded treewidth graphs: the chair and the banner draw the boundary. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*, pages 951–970. SIAM, 2020. doi:10.1137/1.9781611975994.57.

[12] Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. Hitting minors on bounded treewidth graphs. I. General upper bounds. *SIAM Journal on Discrete Mathematics*, 34(3):1623–1648, 2020. doi:10.1137/19M1287146.

[13] Mahdi Belbasi and Martin Fürer. A space-efficient parameterized algorithm for the hamiltonian cycle problem by dynamic algebraization. In *Proceedings of the 14th International Computer Science Symposium in Russia on Computer Science - Theory and Applications, CSR 2019*, volume 11532 of *Lecture Notes in Computer Science*, pages 38–49. Springer, 2019. doi:10.1007/978-3-030-19955-5_4.

[14] Mark de Berg, Kevin Buchin, Bart M. P. Jansen, and Gerhard J. Woeginger. Fine-grained complexity analysis of two classic TSP variants. *ACM Trans. Algorithms*, 17(1):5:1–5:29, 2021. doi:10.1145/3414845.

[15] Claude Berge. *Hypergraphs - Combinatorics of finite sets*, volume 45 of *North-Holland mathematical library*. North-Holland, 1989.

[16] Ivan Bliznets and Nikolay Karpov. Parameterized algorithms for partitioning graphs into highly connected clusters. In *Proceedings of the 42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017*, volume 83 of *LIPIcs*, pages 6:1–6:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.MFCS.2017.6.

[17] 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.

[18] 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.

[19] Hans L. Bodlaender. Lower bounds for kernelization. In *Proceedings of the 9th International Symposium on Parameterized and Exact Computation, IPEC 2014*, volume 8894 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2014. doi:10.1007/978-3-319-13524-3_1.

[20] Hans L. Bodlaender, Paul S. Bonsma, and Daniel Lokshtanov. The fine details of fast dynamic programming over tree decompositions. In *Proceedings of the 8th International Symposium on Parameterized and Exact Computation, IPEC 2013*, volume 8246 of *Lecture Notes in Computer Science*, pages 41–53. Springer, 2013. doi:10.1007/978-3-319-03898-8_5.

[21] Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.

[22] Hans L. Bodlaender, Jitender S. Deogun, Klaus Jansen, Ton Kloks, Dieter Kratsch, Haiko Müller, and Zsolt Tuza. Rankings of graphs. *SIAM J. Discret. Math.*, 11(1):168–181, 1998. doi:10.1137/S0895480195282550.

[23] Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michał Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016. doi:10.1137/130947374.

[24] 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.

[25] Hans L. Bodlaender and Arie M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, 2008. doi:10.1093/comjnl/bxm037.

[26] Hans L. Bodlaender and Babette van Antwerpen-de Fluiter. Reduction algorithms for graphs of small treewidth. *Inf. Comput.*, 167(2):86–119, 2001. doi:10.1006/inco.2000.2958.

[27] 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.

[28] Édouard Bonnet, Yoichi Iwata, Bart M. P. Jansen, and Lukasz Kowalik. Fine-grained complexity of k-OPT in bounded-degree graphs for solving TSP. In *Proceedings of the 27th Annual European Symposium on Algorithms, ESA 2019*, volume 144 of *LIPIcs*, pages 23:1–23:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.23.

[29] Lukasz Bozyk, Jan Derbisz, Tomasz Krawczyk, Jana Novotná, and Karolina Okrasa. Vertex deletion into bipartite permutation graphs. *Algorithmica*, 84(8):2271–2291, 2022. doi:10.1007/s00453-021-00923-7.

[30] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph classes: a survey*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

[31] Binh-Minh Bui-Xuan, Jan Arne Telle, and Martin Vatshelle. Boolean-width of graphs. *Theor. Comput. Sci.*, 412(39):5187–5204, 2011. doi:10.1016/j.tcs.2011.05.022.

[32] Jannis Bulian and Anuj Dawar. Graph isomorphism parameterized by elimination distance to bounded degree. *Algorithmica*, 75(2):363–382, 2016. doi:10.1007/s00453-015-0045-3.

[33] Jannis Bulian and Anuj Dawar. Fixed-parameter tractable distances to sparse graph classes. *Algorithmica*, 79(1):139–158, 2017. doi:10.1007/s00453-016-0235-7.

[34] Benjamin Merlin Bumpus, Bart M. P. Jansen, and Jari J. H. de Kroon. Search-space reduction via essential vertices. In *Proceedings of the 30th Annual European Symposium on Algorithms, ESA 2022*, volume 244 of *LIPIcs*, pages 30:1–30:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.30.

[35] Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, 58(4):171–176, 1996. doi:10.1016/0020-0190(96)00050-6.

[36] Yixin Cao. Linear recognition of almost interval graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*, pages 1096–1115. SIAM, 2016. doi:10.1137/1.9781611974331.ch77.

[37] Yixin Cao and Dániel Marx. Interval deletion is fixed-parameter tractable. *ACM Trans. Algorithms*, 11(3):21:1–21:35, 2015. doi:10.1145/2629595.

[38] Yixin Cao and Dániel Marx. Chordal editing is fixed-parameter tractable. *Algorithmica*, 75(1):118–137, 2016. doi:10.1007/s00453-015-0014-x.

[39] 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.

[40] Jianer Chen, Yang Liu, and Songjian Lu. An improved parameterized algorithm for the minimum node multiway cut problem. *Algorithmica*, 55(1):1–13, 2009. doi:10.1007/s00453-007-9130-6.

[41] Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM*, 55(5):21:1–21:19, 2008. doi:10.1145/1411509.1411511.

[42] Maria Chudnovsky, Jim Geelen, Bert Gerards, Luis A. Goddyn, Michael Lohman, and Paul D. Seymour. Packing non-zero A-paths in group-labelled graphs. *Comb.*, 26(5):521–532, 2006. doi:10.1007/s00493-006-0030-1.

[43] Maria Chudnovsky, Neil Robertson, Paul Seymour, and Robin Thomas. The strong perfect graph theorem. *Ann. Math. (2)*, 164(1):51–229, 2006. doi:10.4007/annals.2006.164.51.

[44] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. URL: http://mitpress.mit.edu/books/introduction-algorithms.

[45] Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2012. doi:10.1017/CBO9780511977619.

[46] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000. doi:10.1007/s002249910009.

[47] Christophe Crespelle, Pål Grønås Drange, Fedor V. Fomin, and Petr A. Golovach. A survey of parameterized algorithms and the complexity of edge modification. *CoRR*, abs/2001.06867, 2020. arXiv:2001.06867.

[48] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

[49] Marek Cygan, Lukasz Kowalik, and Arkadiusz Socala. Improving TSP tours using dynamic programming over tree decompositions. *ACM Trans. Algorithms*, 15(4):54:1–54:19, 2019. doi:10.1145/3341730.

[50] Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast Hamiltonicity checking via bases of perfect matchings. *J. ACM*, 65(3):12:1–12:46, 2018. doi:10.1145/3148227.

[51] Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michał Pilipczuk. Hitting forbidden subgraphs in graphs of bounded treewidth. *Inf. Comput.*, 256:62–82, 2017. doi:10.1016/j.ic.2017.04.009.

[52] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. *ACM Trans. Algorithms*, 18(2):17:1–17:31, 2022. doi:10.1145/3506707.

[53] Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. On multiway cut parameterized above lower bounds. *ACM Trans. Comput. Theory*, 5(1):3:1–3:11, 2013. doi:10.1145/2462896.2462899.

[54] 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, 2014. doi:10.1145/2629620.

[55] Jitender S. Deogun and George Steiner. Polynomial algorithms for Hamiltonian cycle in cocomparability graphs. *SIAM J. Comput.*, 23(3):520–552, 1994. doi:10.1137/S0097539791200375.

[56] Huib Donkers and Bart M. P. Jansen. Preprocessing to reduce the search space: Antler structures for feedback vertex set. In *Proceedings of the 47th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2021*, volume 12911 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2021. doi:10.1007/978-3-030-86838-3_1.

[57] Huib Donkers, Bart M. P. Jansen, and Jari J. H. de Kroon. Finding k-secluded trees faster. In *Proceedings of the 48th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2022*, volume 13453 of *Lecture Notes in Computer Science*, pages 173–186. Springer, 2022. doi:10.1007/978-3-031-15914-5_13.

[58] Frederic Dorn, Eelko Penninkx, Hans L. Bodlaender, and Fedor V. Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica*, 58(3):790–810, 2010. doi:10.1007/s00453-009-9296-1.

[59] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. doi:10.1007/978-1-4612-0515-9.

[60] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

[61] Andrew Drucker. New limits to classical and quantum instance compression. *SIAM J. Comput.*, 44(5):1443–1479, 2015. doi:10.1137/130927115.

[62] Eduard Eiben, Robert Ganian, Thekla Hamm, and O-joung Kwon. Measuring what matters: A hybrid approach to dynamic programming with treewidth. *J. Comput. Syst. Sci.*, 121:57–75, 2021. doi:10.1016/j.jcss.2021.04.005.

[63] Eduard Eiben, Robert Ganian, and Stefan Szeider. Meta-kernelization using well-structured modulators. *Discret. Appl. Math.*, 248:153–167, 2018. doi:10.1016/j.dam.2017.09.018.

[64] Eduard Eiben, Robert Ganian, and Stefan Szeider. Solving problems on graphs of high rank-width. *Algorithmica*, 80(2):742–771, 2018. doi:10.1007/s00453-017-0290-8.

[65] P. Erdős and L. Pósa. On independent circuits contained in a graph. *Canadian Journal of Mathematics*, 17:347–352, 1965. doi:10.4153/CJM-1965-035-8.

[66] Andreas Emil Feldmann, Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6):146, 2020. doi:10.3390/a13060146.

[67] M. R. Fellows and M. A. Langston. An analogue of the Myhill-Nerode theorem and its use in computing finite-basis characterizations. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science, FOCS 1989*, pages 520–525. IEEE Computer Society, 1989. doi:10.1109/SFCS.1989.63528.

[68] Michael R. Fellows. The lost continent of polynomial time: Preprocessing and kernelization. In *Proceedings of the Second International Workshop on Parameterized and Exact Computation, IWPEC 2006*, volume 4169 of *Lecture Notes in Computer Science*, pages 276–277. Springer, 2006. doi:10.1007/11847250_25.

[69] Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances A. Rosamond, Saket Saurabh, and Yngve Villanger. Local search: Is brute-force avoidable? *J. Comput. Syst. Sci.*, 78(3):707–719, 2012. doi:10.1016/j.jcss.2011.10.003.

[70] Michael R. Fellows, Bart M. P. Jansen, and Frances A. Rosamond. Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. *Eur. J. Comb.*, 34(3):541–566, 2013. doi:10.1016/j.ejc.2012.04.008.

[71] Michael R. Fellows and Michael A. Langston. On search, decision and the efficiency of polynomial-time algorithms (extended abstract). In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, STOC 1989*, pages 501–512. ACM, 1989. doi:10.1145/73007.73055.

[72] Samuel Fiorini, Nadia Hardy, Bruce A. Reed, and Adrian Vetta. Planar graph bipartization in linear time. *Discret. Appl. Math.*, 156(7):1175–1180, 2008. doi:10.1016/j.dam.2007.08.013.

[73] Rudolf Fleischer, Xi Wu, and Liwei Yuan. Experimental study of FPT algorithms for the directed feedback vertex set problem. In *Proceedings of the 17th Annual European Symposium on Algorithms, ESA 2009*, volume 5757 of *Lecture Notes in Computer Science*, pages 611–622. Springer, 2009. doi:10.1007/978-3-642-04128-0_55.

[74] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. doi:10.1007/3-540-29953-X.

[75] 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.

[76] Fedor V. Fomin, Petr A. Golovach, and Dimitrios M. Thilikos. Parameterized complexity of elimination distance to first-order logic properties. *ACM Trans. Comput. Log.*, 23(3):17:1–17:35, 2022. doi:10.1145/3517129.

[77] Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2010. doi:10.1007/978-3-642-16533-7.

[78] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, M. S. Ramanujan, and Saket Saurabh. Solving d-SAT via backdoors to small treewidth. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 630–641. SIAM, 2015. doi:10.1137/1.9781611973730.43.

[79] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar F-deletion: Approximation, kernelization and optimal FPT algorithms. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012*, pages 470–479. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.62.

[80] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.

[81] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Hitting topological minors is FPT. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*, pages 1317–1326. ACM, 2020. doi:10.1145/3357713.3384318.

[82] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. doi:10.1017/9781107415157.

[83] Fedor V. Fomin, Ioan Todinca, and Yngve Villanger. Large induced subgraphs via triangulations and CMSO. *SIAM J. Comput.*, 44(1):54–87, 2015. doi:10.1137/140964801.

[84] Fedor V. Fomin and Yngve Villanger. Treewidth computation and extremal combinatorics. *Comb.*, 32(3):289–308, 2012. doi:10.1007/s00493-012-2536-z.

[85] Martin Fürer and Huiwen Yu. Space saving by dynamic algebraization based on tree-depth. *Theory Comput. Syst.*, 61(2):283–304, 2017. doi:10.1007/s00224-017-9751-3.

[86] Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Backdoor treewidth for SAT. In *Proceedings of the 20th International Conference on Theory and Applications of Satisfiability Testing, SAT 2017*, volume 10491 of *Lecture Notes in Computer Science*, pages 20–37. Springer, 2017. doi:10.1007/978-3-319-66263-3_2.

[87] Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Combining treewidth and backdoors for CSP. In *Proceedings of the 34th Symposium on Theoretical Aspects of Computer Science, STACS 2017*, volume 66 of *LIPIcs*, pages 36:1–36:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.STACS.2017.36.

[88] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[89] Fănică Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974. doi:10.1016/0095-8956(74)90094-X.

[90] Jim Geelen, Bert Gerards, Bruce A. Reed, Paul D. Seymour, and Adrian Vetta. On the odd-minor variant of Hadwiger's conjecture. *J. Comb. Theory, Ser. B*, 99(1):20–29, 2009. doi:10.1016/j.jctb.2008.03.006.

[91] 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.

[92] Petr A. Golovach and Dimitrios M. Thilikos. Clustering to given connectivities. In *Proceedings of the 14th International Symposium on Parameterized and Exact Computation, IPEC 2019*, volume 148 of *LIPIcs*, pages 18:1–18:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.IPEC.2019.18.

[93] Martin Grohe. Local tree-width, excluded minors, and approximation algorithms. *Comb.*, 23(4):613–632, 2003. doi:10.1007/s00493-003-0037-9.

[94] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, Jun 1981. doi:10.1007/BF02579273.

[95] Sylvain Guillemot. FPT algorithms for path-transversal and cycle-transversal problems. *Discret. Optim.*, 8(1):61–71, 2011. doi:10.1016/j.disopt.2010.05.003.

[96] Jiong Guo, Sepp Hartung, Rolf Niedermeier, and Ondrej Suchý. The parameterized complexity of local search for TSP, more refined. *Algorithmica*, 67(1):89–110, 2013. doi:10.1007/s00453-012-9685-8.

[97] Jiong Guo, Danny Hermelin, and Christian Komusiewicz. Local search for string problems: Brute-force is essentially optimal. *Theor. Comput. Sci.*, 525:30–41, 2014. doi:10.1016/j.tcs.2013.05.006.

[98] Jiong Guo, Falk Hüffner, and Rolf Niedermeier. A structural view on parameterizing problems: Distance from triviality. In *Proceedings of the First International Workshop on Parameterized and Exact Computation, IWPEC 2004*, volume 3162 of *Lecture Notes in Computer Science*, pages 162–173. Springer, 2004. doi:10.1007/978-3-540-28639-4_15.

[99] Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, 38(1):31–45, 2007. doi:10.1145/1233481.1233493.

[100] 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, SODA 2019*, pages 1731–1749. SIAM, 2019. doi:10.1137/1.9781611975482.104.

[101] Pinar Heggernes, Pim van 't Hof, Bart M. P. Jansen, Stefan Kratsch, and Yngve Villanger. Parameterized complexity of vertex deletion into perfect graph classes. *Theor. Comput. Sci.*, 511:172–180, 2013. doi:10.1016/j.tcs.2012.03.013.

[102] Petr Hlinený, Sang-il Oum, Detlef Seese, and Georg Gottlob. Width parameters beyond tree-width and their applications. *Comput. J.*, 51(3):326–362, 2008. doi:10.1093/comjnl/bxm052.

[103] Falk Hüffner, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. Fixed-parameter algorithms for cluster vertex deletion. *Theory Comput. Syst.*, 47(1):196–217, 2010. doi:10.1007/s00224-008-9150-x.

[104] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

[105] Yoichi Iwata, Magnus Wahlström, and Yuichi Yoshida. Half-integrality, LP-branching, and FPT algorithms. *SIAM J. Comput.*, 45(4):1377–1411, 2016. doi:10.1137/140962838.

[106] Yoichi Iwata, Yutaro Yamaguchi, and Yuichi Yoshida. 0/1/All CSPs, half-integral A-path packing, and linear-time FPT algorithms. In *Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, pages 462–473. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00051.

[107] Ashwin Jacob, Jari J. H. de Kroon, Diptapriyo Majumdar, and Venkatesh Raman. Parameterized complexity of deletion to scattered graph classes. *CoRR*, abs/2105.04660, 2021. arXiv:2105.04660.

[108] Bart M. P. Jansen and Jari J. H. de Kroon. FPT algorithms to compute the elimination distance to bipartite graphs and more. In *Proceedings of the 47th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2021*, volume 12911 of *Lecture Notes in Computer Science*, pages 80–93. Springer, 2021. doi:10.1007/978-3-030-86838-3_6.

[109] Bart M. P. Jansen and Jari J. H. de Kroon. Preprocessing vertex-deletion problems: Characterizing graph properties by low-rank adjacencies. *J. Comput. Syst. Sci.*, 126:59–79, 2022. doi:10.1016/j.jcss.2021.12.003.

[110] Bart M. P. Jansen, Jari J. H. de Kroon, and Michał Włodarczyk. Vertex deletion parameterized by elimination distance and even less. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2021*, pages 1757–1769. ACM, 2021. doi:10.1145/3406325.3451068.

[111] Bart M. P. Jansen, Jari J. H. de Kroon, and Michał Włodarczyk. Vertex deletion parameterized by elimination distance and even less. *CoRR*, abs/2103.09715, 2021. arXiv:2103.09715.

[112] Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. A near-optimal planarization algorithm. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014*, pages 1802–1811. SIAM, 2014. doi:10.1137/1.9781611973402.130.

[113] 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.

[114] Bart M. P. Jansen, Venkatesh Raman, and Martin Vatshelle. Parameter ecology for feedback vertex set. *Tsinghua Science and Technology*, 19(4):387–409, 2014. Special Issue dedicated to Jianer Chen. doi:10.1109/TST.2014.6867520.

[115] Bart M. P. Jansen and Jules J. H. M. Wulms. Lower bounds for protrusion replacement by counting equivalence classes. *Discret. Appl. Math.*, 278:12–27, 2020. doi:10.1016/j.dam.2019.02.024.

[116] Vít Jelínek. The rank-width of the square grid. *Discret. Appl. Math.*, 158(7):841–850, 2010. doi:10.1016/j.dam.2009.02.007.

[117] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

[118] Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. *J. ACM*, 66(5):33:1–33:38, 2019. doi:10.1145/3325116.

[119] Ken-ichi Kawarabayashi. Planarity allowing few error vertices in linear time. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009*, pages 639–648. IEEE Computer Society, 2009. doi:10.1109/FOCS.2009.45.

[120] Leon Kellerhals, Tomohiro Koana, and Pascal Kunz. Vertex cover and feedback vertex set above and below structural guarantees. In *Proceedings of the 17th International Symposium on Parameterized and Exact Computation, IPEC 2022*, volume 249 of *LIPIcs*, pages 19:1–19:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.IPEC.2022.19.

[121] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, STOC 2002*, pages 767–775. ACM, 2002. doi:10.1145/509907.510017.

[122] Ton Kloks. *Treewidth: Computations and Approximations*, volume 842. Springer Science & Business Media, 1994. doi:10.1007/BFb0045375.

[123] Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 184–192. IEEE, 2021. doi:10.1109/FOCS52979.2021.00026.

[124] Stefan Kratsch. Recent developments in kernelization: A survey. *Bull. EATCS*, 113, 2014. URL: https://eatcs.org/images/bulletin/beatcs113.pdf.

[125] Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. *J. ACM*, 67(3):16:1–16:50, 2020. doi:10.1145/3390887.

[126] Boland J. Lekkerkerker C. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962. URL: http://eudml.org/doc/213681.

[127] 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.

[128] Jason Li and Jesper Nederlof. Detecting feedback vertex sets of size k in $O^*(2.7^k)$ time. *ACM Trans. Algorithms*, 18(4):34:1–34:26, 2022. doi:10.1145/3504027.

[129] Alexander Lindermayr, Sebastian Siebertz, and Alexandre Vigny. Elimination distance to bounded degree on planar graphs. In *Proceedings of the 45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020*, volume 170 of *LIPIcs*, pages 65:1–65:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.65.

[130] Daniel Lokshtanov. Wheel-free deletion is W[2]-hard. In *Proceedings of the Third International Workshop on Parameterized and Exact Computation, IWPEC 2008*, volume 5018 of *Lecture Notes in Computer Science*, pages 141–147. Springer, 2008. doi:10.1007/978-3-540-79723-4_14.

[131] Daniel Lokshtanov and Dániel Marx. Clustering with local restrictions. *Inf. Comput.*, 222:278–292, 2013. doi:10.1016/j.ic.2012.10.016.

[132] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Trans. Algorithms*, 14(2):13:1–13:30, 2018. doi:10.1145/3170442.

[133] Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Kernelization - preprocessing with a guarantee. In *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *Lecture Notes in Computer Science*, pages 129–161. Springer, 2012. doi:10.1007/978-3-642-30891-8_10.

[134] 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.

[135] Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Reducing CMSO model checking to highly connected graphs. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, volume 107 of *LIPIcs*, pages 135:1–135:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.135.

[136] Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Reducing CMSO model checking to highly connected graphs. *CoRR*, abs/1802.01453, 2018. arXiv:1802.01453.

[137] Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Parameterized complexity and approximability of directed odd cycle transversal. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*, pages 2181–2200. SIAM, 2020. doi:10.1137/1.9781611975994.134.

[138] W. Mader. Homomorphieeigenschaften und mittlere kantendichte von graphen. *Mathematische Annalen*, 174:265–268, 1967. doi:10.1007/BF01364272.

[139] Diptapriyo Majumdar and Venkatesh Raman. Structural parameterizations of undirected feedback vertex set: FPT algorithms and kernelization. *Algorithmica*, 80(9):2683–2724, 2018. doi:10.1007/s00453-018-0419-4.

[140] Dániel Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006. doi:10.1016/j.tcs.2005.10.007.

[141] Dániel Marx. Searching the k-change neighborhood for TSP is W[1]-hard. *Oper. Res. Lett.*, 36(1):31–36, 2008. doi:10.1016/j.orl.2007.02.008.

[142] Dániel Marx. Chordal deletion is fixed-parameter tractable. *Algorithmica*, 57(4):747–768, 2010. doi:10.1007/s00453-008-9233-8.

[143] Dániel Marx. Important separators and parameterized algorithms. In *Proceedings of the 37th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2011*, volume 6986 of *Lecture Notes in Computer Science*, pages 5–10. Springer, 2011. doi:10.1007/978-3-642-25870-1_2.

[144] Dániel Marx and Ildikó Schlotter. Obtaining a planar graph by vertex deletion. *Algorithmica*, 62(3-4):807–822, 2012. doi:10.1007/s00453-010-9484-z.

[145] Laure Morelle, Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. Faster parameterized algorithms for modification problems to minor-closed classes. *CoRR*, abs/2210.02167, 2022. arXiv:2210.02167.

[146] Jesper Nederlof. Algorithms for NP-hard problems via rank-related parameters of matrices. In *Treewidth, Kernels, and Algorithms - Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, volume 12160 of *Lecture Notes in Computer Science*, pages 145–164. Springer, 2020. doi:10.1007/978-3-030-42071-0_11.

[147] George L. Nemhauser and Leslie E. Trotter Jr. Vertex packings: Structural properties and algorithms. *Math. Program.*, 8(1):232–248, 1975. doi:10.1007/BF01580444.

[148] Jaroslav Nesetril and Patrice Ossona de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *Eur. J. Comb.*, 27(6):1022–1041, 2006. doi:10.1016/j.ejc.2005.01.010.

[149] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. doi:10.1093/acprof:oso/9780198566076.001.0001.

[150] Rolf Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010*, volume 5 of *LIPIcs*, pages 17–32. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010. doi:10.4230/LIPIcs.STACS.2010.2495.

[151] Sang-il Oum. Rank-width: Algorithmic and structural results. *Discret. Appl. Math.*, 231:15–24, 2017. doi:10.1016/j.dam.2016.08.006.

[152] Sang-il Oum and Paul D. Seymour. Approximating clique-width and branch-width. *J. Comb. Theory, Ser. B*, 96(4):514–528, 2006. doi:10.1016/j.jctb.2005.10.006.

[153] Marcin Pilipczuk. A tight lower bound for vertex planarization on graphs of bounded treewidth. *Discret. Appl. Math.*, 231:211–216, 2017. doi:10.1016/j.dam.2016.05.019.

[154] Michał Pilipczuk. Problems parameterized by treewidth tractable in single exponential time: A logical approach. In *Proceedings of the 36th International Symposium on Mathematical Foundations of Computer Science, MFCS 2011*, volume 6907 of *Lecture Notes in Computer Science*, pages 520–531. Springer, 2011. doi:10.1007/978-3-642-22993-0_47.

[155] Michał Pilipczuk and Marcin Wrochna. On space efficiency of algorithms working on structural decompositions of graphs. *TOCT*, 9(4):18:1–18:36, 2018. doi:10.1145/3154856.

[156] M. S. Ramanujan, Abhishek Sahu, Saket Saurabh, and Shaily Verma. An exact algorithm for knot-free vertex deletion. In *Proceedings of the 47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022*, volume 241 of *LIPIcs*, pages 78:1–78:15. Schloss

Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.MFCS.2022.78.

[157] B. A. Reed. *Algorithmic Aspects of Tree Width*, pages 85–107. Springer New York, New York, NY, 2003. doi:10.1007/0-387-22444-0_4.

[158] Bruce A. Reed. Mangoes and blueberries. *Comb.*, 19(2):267–296, 1999. doi:10.1007/s004930050056.

[159] Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004. doi:10.1016/j.orl.2003.10.009.

[160] Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. A faster parameterized algorithm for treedepth. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming, ICALP 2014*, volume 8572 of *Lecture Notes in Computer Science*, pages 931–942. Springer, 2014. doi:10.1007/978-3-662-43948-7_77.

[161] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986. doi:10.1016/0196-6774(86)90023-4.

[162] 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.

[163] Neil Robertson and Paul D Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004. doi:10.1016/j.jctb.2004.08.001.

[164] Ignasi Sau and Uéverton dos Santos Souza. Hitting forbidden induced subgraphs on bounded treewidth graphs. *Inf. Comput.*, 281:104812, 2021. doi:10.1016/j.ic.2021.104812.

[165] Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. k-apices of minor-closed graph classes. II. Parameterized algorithms. *ACM Trans. Algorithms*, 18(3):21:1–21:30, 2022. doi:10.1145/3519028.

[166] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.

[167] Jan Arne Telle and Yngve Villanger. Connecting terminals and 2-disjoint connected subgraphs. In *Proceedings of the 39th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2013*, volume 8165 of *Lecture Notes in Computer Science*, pages 418–428. Springer, 2013. doi:10.1007/978-3-642-45043-3_36.

[168] Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986. doi:10.1016/0304-3975(86)90135-0.

[169] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2001. doi:10.1007/978-3-662-04565-7.

[170] Karsten Weihe. Covering trains by stations or the power of data reduction. In *Algorithms and Experiments (ALEX98)*, pages 1–8, 1998. URL: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.57.2173.

[171] Karsten Weihe. On the differences between "practical" and "applied". In Stefan Näher and Dorothea Wagner, editors, *Proceedings of the 4th International Workshop on Algorithm Engineering, WAE 2000*, volume 1982 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2000. doi:10.1007/3-540-44691-5_1.

[172] Sebastian Wernicke. *On the algorithmic tractability of single nucleotide polymorphism (SNP) analysis and related problems*. diplom.de, 2014.

[173] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. URL: http://www.cambridge.org/de/knowledge/isbn/item5759340/.

Summary

Parameterized Graph Modification Beyond the Natural Parameter

For the class of NP-hard problems, assuming $\mathsf{P} \neq \mathsf{NP}$, no algorithm is able to solve all instances with a running time that is bounded by some polynomial in the input size. There are several ways of dealing with these problems. In the field of parameterized complexity, one takes a fine-grained approach. Rather than just focusing on input size, additional aspects of the input are taken into account. The goal is then to come up with algorithms whose exponential dependency is restricted to these additional aspects, known as parameters, while the running time remains polynomial in the input size. Whenever there exists such an algorithm for a problem and parameter pair, we say that the problem is fixed-parameter tractable (FPT) for this parameter.

In this thesis we consider a certain class of graph problems known as vertex-deletion problems. In these graph modification problems, the task is to do a minimum number of vertex deletions such that the resulting graph belongs to some graph class $\mathcal{H}$ such as bipartite, edgeless, or chordal graphs. In computer science, these $\mathcal{H}$-Deletion problems have been studied for half a century and are a prominent class of problems in parameterized complexity. An obvious choice for a parameter is the number of vertex-deletions, which is often referred to as the natural parameter. Many vertex-deletion problems turn out to be FPT with this parameter. In this work we try to obtain algorithms using even smaller parameters, that is, we aim to do parameterized graph modification beyond the natural parameter. We do so in two directions.

In the first direction our aim is to combine the solution-size parameter with structural parameters. Many problems become efficiently solvable if certain structural graph parameters, such as treewidth or treedepth, are bounded. Extending these definitions slightly based on some graph class $\mathcal{H}$ results in

parameters $\mathcal{H}$-treewidth ($\mathbf{tw}_{\mathcal{H}}$) and $\mathcal{H}$-elimination distance ($\mathbf{ed}_{\mathcal{H}}$), which are smaller than both the solution size and respectively treewidth and treedepth. These hybrid parameters were recently introduced in the literature. As a first step we give classification algorithms showing for which graph classes $\mathcal{H}$ these parameters can be computed in FPT time, most notably for the class of bipartite graphs. Next, we focus our attention to $\mathbf{tw}_{\mathcal{H}}$, where we compute approximately optimal decompositions and use them to solve $\mathcal{H}$-Deletion building on widely used dynamic programming techniques. We show that if $\mathcal{H}$ is hereditary (closed under vertex-deletions), closed under taking disjoint union of graphs, and $\mathcal{H}$-Deletion parameterized by solution size is FPT, then we can construct a so-called tree $\mathcal{H}$-decomposition whose width is at most 8 times $\mathbf{tw}_{\mathcal{H}}(G)$ in time that is FPT with respect to $\mathbf{tw}_{\mathcal{H}}$. For our dynamic program over these decompositions, we additionally need that we can solve a special type of $\mathcal{H}$-Deletion with undeletable vertices. Finally, using (either known or new) bounds on a certain equivalence relation regarding containment in $\mathcal{H}$ with respect to gluing boundaried graphs, we come up with algorithms for $\mathcal{H}$-Deletion that are FPT with respect to $\mathbf{tw}_{\mathcal{H}}$. As an example, we obtain an algorithm that solves Chordal Deletion in time $2^{\mathcal{O}(k^2)} \cdot n^{\mathcal{O}(1)}$ where $k = \mathbf{tw}_{\mathsf{chordal}}(G)$.

In the second direction we take a slightly different approach. By identifying part of an optimal solution to $\mathcal{H}$-Deletion in a preprocessing phase, the follow-up FPT algorithm that uses the solution-size parameterization will be sped up. We come up with algorithms that allow to find part of the solution in polynomial time under certain conditions. This condition is based on c-essentiality, where a vertex is c-essential if it belongs to every solution of at most c times the optimum size. The intuition being that these vertices should stand out and are therefore more easily detected. Among others, we show that in polynomial time all 2-essential vertices can be detected for Odd Cycle Transversal. This results in an algorithm for OCT that runs in time $2.3146^{\ell} \cdot n^{\mathcal{O}(1)}$ where ℓ is the number of vertices that are not 2-essential, where previously this exponent was the total solution size.

Curriculum Vitae

Jari J. H. de Kroon was born on the 24th of April in 1996 in Tilburg, the Netherlands. He completed his secondary education in 2014 at Jacob-Roelandslyceum in Boxtel. He studied Computer Science and Engineering at Eindhoven University of Technology and obtained his Bachelor's degree (cum laude) in 2017 and his Master's degree (cum laude) in 2019. Afterwards he continued there as a PhD student under the supervision of dr. Bart M. P. Jansen. His research during this time is represented by the results in this dissertation.