Faster parameterized algorithms for modification problems to minor-closed classes

Laure Morelle, Giannos Stamoulis, Ignasi Sau, Dimitrios M. Thilikos

TheoretiCS

Faster parameterized algorithms for modification problems to minor-closed classes

Laure Morelle [a] ✉

Ignasi Sau [a] ✉

Giannos Stamoulis [b] ✉

Dimitrios M. Thilikos [a] ✉

a LIRMM, Université de Montpellier, CNRS, Montpellier, France

b Institute of Informatics, University of Warsaw, Poland

ABSTRACT. Let $\mathcal{G}$ be a minor-closed graph class and let G be an n-vertex graph. We say that G is a *k-apex* of $\mathcal{G}$ if G contains a set S of at most k vertices such that $G \setminus S$ belongs to $\mathcal{G}$. Our first result is an algorithm that decides whether G is a k-apex of $\mathcal{G}$ in time $2^{\mathsf{poly}(k)} \cdot n^2$, improving a previous algorithm by Sau, Stamoulis, and Thilikos [ICALP 2020, TALG 2022] whose running time was $2^{\mathsf{poly}(k)} \cdot n^3$. The *elimination distance* of G to $\mathcal{G}$, denoted by $\mathsf{ed}_{\mathcal{G}}(G)$, is the minimum number of rounds required to reduce each connected component of G to a graph in $\mathcal{G}$ by removing one vertex from each connected component in each round. Bulian and Dawar [Algorithmica 2017] provided an FPT-algorithm, with parameter k, to decide whether $\mathsf{ed}_{\mathcal{G}}(G) \leq k$. The class of graphs with $\mathsf{ed}_{\mathcal{G}} \leq k$ is minor-closed, hence characterized by a finite set of excluded minors. The algorithm of Bulian and Dawar is based on the computability of this finite minor-obstruction set and its dependence on k is not explicit. We extend the techniques used in our first algorithm to decide whether $\mathsf{ed}_{\mathcal{G}}(G) \leq k$ in time $2^{2^{2^{\mathsf{poly}(k)}}} \cdot n^2$. This is the first algorithm for this problem with an explicit parametric dependence in k. In the special case where $\mathcal{G}$ excludes some apex-graph as a minor, we give two alternative algorithms, one running in time $2^{2^{O(k^2 \log k)}} \cdot n^2$ and one running in time $2^{\mathsf{poly}(k)} \cdot n^3$. As a stepping stone for these algorithms, we provide an algorithm that decides whether $\mathsf{ed}_{\mathcal{G}}(G) \leq k$ in time $2^{O(\mathsf{tw} \cdot k + \mathsf{tw} \log \mathsf{tw})} \cdot n$, where tw

is the treewidth of G. This algorithm combines the dynamic programming framework of Reidl, Rossmanith, Villaamil, and Sikdar [ICALP 2014] for the particular case where $\mathcal{G}$ contains only the empty graph (i.e., for treedepth) with the representative-based techniques introduced by Baste, Sau, and Thilikos [SODA 2020]. In the complexities above, poly is a polynomial function whose degree depends on $\mathcal{G}$, and the hidden constants also depend on $\mathcal{G}$. Finally, we provide explicit upper bounds on the size of the graphs in the minor-obstruction set of the class of graphs $\mathcal{E}_k(\mathcal{G}) = \{G \mid \mathsf{ed}_{\mathcal{G}}(G) \leq k\}$.

1.   Introduction

The *distance from triviality* is a concept formalized by Guo, Hüffner, and Niedermeier [35] to express the closeness of a graph to a supposedly "simple" target graph class. One such a measure of closeness is, for instance, the number of vertices or edges that one must delete/add from/to a graph G to obtain a graph in the target graph class. This concept of distance to a graph class has recently gained the interest of the parameterized complexity community. The motivation is that, if a problem is tractable on a graph class $\mathcal{G}$, it is natural to study other classes of graphs according to their "distance to $\mathcal{G}$". In this paper, we focus on two such measures of distance from triviality: Given a target graph class $\mathcal{G}$, we consider the *vertex deletion distance* to $\mathcal{G}$ and the *elimination distance* to $\mathcal{G}$, which we formalize next.

Given a target graph class $\mathcal{G}$ and a non-negative integer k, we define $\mathcal{A}_k(\mathcal{G})$ as the set of all graphs containing a set S of at most k vertices whose removal results in a graph in $\mathcal{G}$. If $G \in \mathcal{A}_k(\mathcal{G})$, then we say that G is a *k-apex* of $\mathcal{G}$. We refer to S as a *k-apex set of G for the class $\mathcal{G}$*. In other words, we consider the following meta-problem for a fixed class $\mathcal{G}$.

VERTEX DELETION TO $\mathcal{G}$

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

Objective: Find, if it exists, a k-apex set of G for the class $\mathcal{G}$.

Throughout the paper, we denote by n the number of vertices of the input graph of the problem under consideration. The importance of VERTEX DELETION TO $\mathcal{G}$ can be illustrated by the variety of graph modification problems that it encompasses (hence the term of meta-problem). For instance, if $\mathcal{G}$ is the class of edgeless (resp. acyclic, planar, bipartite, (proper) interval, chordal) graphs, then we obtain the VERTEX COVER (resp. FEEDBACK VERTEX SET, VERTEX PLANARIZATION, ODD CYCLE TRANSVERSAL, (PROPER) INTERVAL VERTEX DELETION, CHORDAL VERTEX DELETION) problem.

The second measure of distance from triviality that we study was recently introduced by Bulian and Dawar [17, 16]. Given a graph class $\mathcal{G}$, we define the *elimination distance* of a graph

G to $\mathcal{G}$, denoted by $\mathrm{ed}_{\mathcal{G}}(G)$, as follows:

$$\mathrm{ed}_{\mathcal{G}}(G) = \begin{cases} 0 & \text{if } G \in \mathcal{G}, \\ 1 + \min\{\mathrm{ed}_{\mathcal{G}}(G \setminus \{v\}) \mid v \in V(G)\} & \text{if } G \text{ is connected,} \\ \max\{\mathrm{ed}_{\mathcal{G}}(H) \mid H \text{ is a connected component of } G\} & \text{otherwise.} \end{cases}$$

Given that $\mathrm{ed}_{\mathcal{G}}(G) \leq k$, a set $S \subseteq V(G)$ of vertices recursively deleted from G to achieve $\mathrm{ed}_{\mathcal{G}}(G)$ is called a *k-elimination set of G for $\mathcal{G}$*. We define the (parameterized) class of graphs $\mathcal{E}_k(\mathcal{G}) = \{G \mid \mathrm{ed}_{\mathcal{G}}(G) \leq k\}$. The above notion can be seen as a natural generalization of *treedepth* (denoted by td), which corresponds to the case where $\mathcal{G}$ contains only the empty graph. Treedepth, along with treewidth, are two of the most studied and widely used parameters to measure the structural complexity of a graph [19, 48, 57]. The second meta-problem that we consider is the following, again for a fixed class $\mathcal{G}$.

ELIMINATION DISTANCE TO $\mathcal{G}$

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

Objective: Find, if it exists, a k-elimination set of G for the class $\mathcal{G}$.

Unsurprisingly, VERTEX DELETION TO $\mathcal{G}$ is NP-hard for every non-trivial graph class $\mathcal{G}$ [54], while ELIMINATION DISTANCE TO $\mathcal{G}$ is NP-hard even when $\mathcal{G}$ contains only the empty graph [60]. To circumvent this intractability, we study both problems from the parameterized complexity point of view and consider their parameterizations by k. In this setting, the most desirable behavior is the existence of an algorithm running in time $f(k) \cdot n^{O(1)}$, where f is a computable function depending only on k. Such an algorithm is called *fixed-parameter tractable*, or FPT-*algorithm* for short, and a parameterized problem admitting an FPT-algorithm is said to belong to the parameterized complexity class FPT. Also, the function f is called *parametric dependence* of the corresponding FPT-algorithm, and the challenge is to design FPT-algorithms with small parametric dependencies and with a polynomial factor of small degree [19, 21, 25, 58]. We may also consider XP-*algorithms*, i.e., algorithms running in time $O(f(k) \cdot n^{g(k)})$ for some computable functions f and g depending only on k.

In general, for any of the two considered problems, we cannot expect FPT-algorithms for every graph class $\mathcal{G}$. For instance, the two problems are NP-hard, even for $k = 0$, for every graph class $\mathcal{G}$ whose recognition problem is NP-hard. This is the case of 3-colorable graphs, which is a class closed under taking (induced) subgraphs. In this paper, we focus on a family of graph classes that exhibits a nice behavior with respect to the considered problems (and many others): we consider $\mathcal{G}$ to be a *minor-closed* graph class, i.e., such that every minor of a graph in $\mathcal{G}$ (that is, obtained from a subgraph of a graph in $\mathcal{G}$ by contracting edges; see Subsection 2.1 for the formal definition) is also in $\mathcal{G}$. Indeed, it turns out that, for every such a family $\mathcal{G}$, the problems become fixed-parameter tractable, as we proceed to discuss.

The *minor-obstruction set* (in short *obstruction set*) of $\mathcal{G}$ is the set of minor-minimal graphs that do not belong to $\mathcal{G}$, and is denoted by $\text{obs}(\mathcal{G})$. Notice that $\text{obs}(\mathcal{G})$ gives a complete characterization of $\mathcal{G}$ as, for every graph G, it holds that $G \in \mathcal{G}$ if and only if, for every $H \in \text{obs}(\mathcal{G})$, H is not a minor of G. Because of Robertson and Seymour's theorem [65], $\text{obs}(\mathcal{G})$ is *finite* for every minor-closed graph class. As checking whether an h-vertex graph H is a minor of G can be done in time $f(h) \cdot n^2$ [63, 43], the finiteness of $\text{obs}(\mathcal{G})$ along with the above characterization imply that, for every minor-closed graph class $\mathcal{G}$, checking whether $G \in \mathcal{G}$ can be done in time $c \cdot n^2$, where c is a constant depending on the graph class $\mathcal{G}$. This meta-theorem implies the *existence* of FPT-algorithms for a wide family of problems, including VERTEX DELETION TO $\mathcal{G}$ and ELIMINATION DISTANCE TO $\mathcal{G}$. Indeed, this follows by observing that if $\mathcal{G}$ is minor-closed, then for every non-negative integer k, the classes $\mathcal{A}_k(\mathcal{G})$ and $\mathcal{E}_k(\mathcal{G})$ are also minor-closed.

As Robertson and Seymour's theorem [65] does *not* give any way to construct the corresponding obstruction sets, the aforementioned argument is not constructive, i.e., it is not able to construct the obstruction sets required for the corresponding FPT-algorithms. Moreover, these algorithms are *non-uniform* in k, meaning that we have a *distinct* algorithm for every value of k. Important steps towards the constructibility of such FPT-algorithms were done by Adler, Grohe, and Kreutzer [2] and Bulian and Dawar [16], who respectively proved that $\text{obs}(\mathcal{A}_k(\mathcal{G}))$ and $\text{obs}(\mathcal{E}_k(\mathcal{G}))$ are effectively computable. Hence, for both problems, it is possible to construct uniform (in k) algorithms running in time $f(k) \cdot n^2$ for some computable function f. However, this does not imply any reasonable, or even explicit, parametric dependence of the obtained algorithms.

The main focus of this paper is on the parametric and polynomial dependence of FPT-algorithms to solve VERTEX DELETION TO $\mathcal{G}$ and ELIMINATION DISTANCE TO $\mathcal{G}$, i.e., for recognizing the classes $\mathcal{A}_k(\mathcal{G})$ and $\mathcal{E}_k(\mathcal{G})$, when $\mathcal{G}$ is a minor-closed graph class.

Concerning VERTEX DELETION TO $\mathcal{G}$, after a number of articles for particular cases of minor-closed classes $\mathcal{G}$, such as graphs of bounded treewidth [28, 46], planar graphs [55, 38], or graphs of bounded genus [49], an explicit FPT-algorithm for *any* minor-closed graph $\mathcal{G}$ was recently proposed by Sau, Stamoulis, and Thilikos [70], running in time $2^{O(k^c)} \cdot n^3$, where c is a constant that depends on the maximum size of a graph in the obstruction set of $\mathcal{G}$. Moreover, in the case where $\text{obs}(\mathcal{G})$ contains some apex-graph (that is, a 1-apex for the class of planar graphs), Sau, Stamoulis, and Thilikos [70] gave an improved running time of $2^{O(k^c)} \cdot n^2$. Note also that the more general variant where $\mathcal{G}$ is a topological-minor-closed graph class is in FPT as well [29].

As for ELIMINATION DISTANCE TO $\mathcal{G}$ when $\mathcal{G}$ is minor-closed, no explicit parametric dependence was known, with the notable exception of treedepth, for which Reidl, Rossmanith, Villaamil, and Sikdar [61] gave an algorithm deciding whether $\text{td}(G) \leq k$ in time $2^{O(k \cdot \text{tw})} \cdot n$, where $\text{tw} := \text{tw}(G)$ (see also [15]). Using our terminology, and given that $\text{tw}(G) \leq \text{td}(G)$ for every graph G, this yields an FPT-algorithm for ELIMINATION DISTANCE TO $\mathcal{G}_\emptyset$, where $\mathcal{G}_\emptyset$ is the class consisting of the empty graph, running in time $2^{O(k^2)} \cdot n$. Note that this algorithm [61],

combined with the fact that $\mathrm{td}(G) \leq \log(n) \cdot \mathrm{tw}(G)$ (see [15]), imply an XP-algorithm for the problem of computing td when parameterized by tw, namely an algorithm that computes the value of $\mathrm{td}(G)$ in time $n^{O(\mathrm{tw}(G)^2)}$. To the best of our knowledge, it is open whether computing td parameterized by tw is in FPT.

Before describing our results, let us mention some recent relevant results dealing with ELIMINATION DISTANCE TO $\mathcal{G}$ for classes $\mathcal{G}$ that are not necessarily minor-closed. Agrawal and Ramanujan [6] (resp. Agrawal, Kanesh, Panolan, Ramanujan, and Saurabh [5]) provided FPT-algorithms, with parameter k, when $\mathcal{G}$ is the class of cliques (resp. graphs of bounded degree). Fomin, Golovach, and Thilikos [27] identified sufficient and necessary conditions for the existence of FPT-algorithms when $\mathcal{G}$ is definable in first-order logic (such as having bounded degree). Jansen, de Kroon, and Włodarczyk [37] proved, among a number of other results, that if $\mathcal{G}$ is a hereditary union-closed graph class and VERTEX DELETION TO $\mathcal{G}$ can be solved in time $2^{k^{O(1)}} \cdot n^{O(1)}$ (as it is the case for every minor-closed class $\mathcal{G}$ by the results of [70]), then there is an algorithm that, given an n-vertex graph G, computes an $O(\mathrm{ed}_{\mathcal{G}}(G)^3)$-elimination set of G for $\mathcal{G}$ in time $2^{\mathrm{ed}_{\mathcal{G}}(G)^{O(1)}} \cdot n^{O(1)}$. Therefore, for union-closed minor-closed graph classes $\mathcal{G}$, the result of [37] yields an FPT-approximation algorithm for ELIMINATION DISTANCE TO $\mathcal{G}$.

Note that, for every graph class $\mathcal{G}$ and every graph G, it holds that $\mathrm{ed}_{\mathcal{G}}(G)$ is not larger than the smallest k such that G admits a k-apex for $\mathcal{G}$. Thus, if ELIMINATION DISTANCE TO $\mathcal{G}$ is in FPT, so is VERTEX DELETION TO $\mathcal{G}$. Agrawal, Kanesh, Lokshtanov, Panolan, Ramanujan, Saurabh, and Zehavi [4] showed, among other results, that in many cases the reverse implication also holds. Namely, they proved that if $\mathcal{G}$ is hereditary, union-closed, and definable in monadic second-order logic, and VERTEX DELETION TO $\mathcal{G}$ is in FPT, then ELIMINATION DISTANCE TO $\mathcal{G}$ is also (non-uniformly) in FPT. Incidentally, they also showed that if $\mathcal{G}$ is defined by excluding a finite number of connected topological minors, then ELIMINATION DISTANCE TO $\mathcal{G}$ is (uniformly) in FPT. We note that the results of [4] do not provide explicit parametric dependencies for these FPT-algorithms. Also, let us mention that it was conjectured in [4] that ELIMINATION DISTANCE TO $\mathcal{G}$ is in FPT parameterized by a generalization of treewidth called $\mathcal{G}$-*treewidth* (see [37, 4, 23]). Note that, if true, this conjecture would answer the open problem mentioned above of whether computing td parameterized by tw is in FPT.

Our results. In this paper, we provide explicit FPT-algorithms for VERTEX DELETION TO $\mathcal{G}$ and ELIMINATION DISTANCE TO $\mathcal{G}$ for *every* fixed minor-closed graph class $\mathcal{G}$. Our first result is the following.

THEOREM 1.1. *For every minor-closed graph class $\mathcal{G}$, there exists an algorithm that solves* VERTEX DELETION TO $\mathcal{G}$ *in time* $2^{k^{O(1)}} \cdot n^2$.

The degree of k in the running time of Theorem 1.1, as well as the constants hidden in the O-notation in the running time of the algorithms of the results below, depend on the maximum

size of a graph in $\mathrm{obs}(\mathcal{G})$. Thus, the algorithm of Theorem 1.1, while being uniformly FPT in k, is *not* uniform in the target class $\mathcal{G}$, as one needs to know an upper bound on the size of the minor-obstructions. This "meta-non-uniformity" applies to all the algorithms presented in this paper, and it is also the case, among many others, of the FPT-algorithms in [70]. The algorithm of Theorem 1.1 improves the algorithm of [70] from cubic to quadratic complexity in n while keeping the same parametric dependence on k. This answers positively one of the open problems posed in [70].

Our next algorithmic results concern ELIMINATION DISTANCE TO $\mathcal{G}$ and provide, to the authors' knowledge, the first FPT-algorithms for this problem, when $\mathcal{G}$ is minor-closed, with an *explicit parametric dependence.*

THEOREM 1.2. *For every minor-closed graph class $\mathcal{G}$, there exists an algorithm that solves* ELIMINATION DISTANCE TO $\mathcal{G}$ *in time* $2^{2^{2^{2^{k^{O(1)}}}}} \cdot n^2$. *In the particular case where* $\mathrm{obs}(\mathcal{G})$ *contains an apex-graph, this algorithm runs in time* $2^{2^{O(k^2 \log k)}} \cdot n^2$.

As examples of classes $\mathcal{G}$ where $\mathrm{obs}(\mathcal{G})$ contains an apex-graph, we may consider $\mathcal{G}$ whose graphs have bounded Euler genus, such as planar graphs. Our next result improves the parametric dependence of the algorithm of Theorem 1.2 when $\mathrm{obs}(\mathcal{G})$ contains an apex-graph, but with a worse polynomial factor.

THEOREM 1.3. *For every minor-closed graph class $\mathcal{G}$ such that* $\mathrm{obs}(\mathcal{G})$ *contains an apex-graph, there exists an algorithm that solves* ELIMINATION DISTANCE TO $\mathcal{G}$ *in time* $2^{k^{O(1)}} \cdot n^3$.

As discussed later, a crucial ingredient in the algorithms of Theorem 1.2 and Theorem 1.3 is to solve ELIMINATION DISTANCE TO $\mathcal{G}$ parameterized by the treewidth of the input graph. The following result, which may be of independent interest, deals with this case.

THEOREM 1.4. *For every minor-closed graph class $\mathcal{G}$, there exists an algorithm that solves* ELIMINATION DISTANCE TO $\mathcal{G}$ *in time* $2^{O(k \cdot \mathrm{tw} + \mathrm{tw} \log \mathrm{tw})} \cdot n$, *where* tw *denotes the treewidth of the input graph.*

The algorithm of Theorem 1.4 can be seen as a generalization of the algorithm of Reidl, Rossmanith, Villaamil, and Sikdar [61] deciding whether $\mathrm{td}(G) \leq k$ in time $2^{O(k \cdot \mathrm{tw})} \cdot n$. Since, for any graph G and any graph class $\mathcal{G}$, $\mathrm{ed}_{\mathcal{G}}(G) \leq \mathrm{td}(G) \leq \mathrm{tw}(G) \cdot \log n$, Theorem 1.4 implies the existence of an XP-algorithm for ELIMINATION DISTANCE TO $\mathcal{G}$ parameterized by treewidth, when $\mathcal{G}$ is minor-closed, running in time $n^{O(\mathrm{tw}^2)}$. Given that the conjecture of [4] is still open, this is the best type of algorithm that one can expect for ELIMINATION DISTANCE TO $\mathcal{G}$ parameterized by treewidth. Furthermore, since $\mathrm{tw}(G) \leq \mathrm{td}(G)$ for any graph G, Theorem 1.4 implies an FPT-algorithm for ELIMINATION DISTANCE TO $\mathcal{G}$ parameterized by treedepth, running in time $2^{O(\mathrm{td}^2)} \cdot n$.

Finally, for any minor-closed graph class $\mathcal{G}$, we provide an upper bound on the size of the graphs in the obstruction set of $\mathcal{E}_k(\mathcal{G})$.

THEOREM 1.5. *For every minor-closed graph class $\mathcal{G}$ and for every positive integer k, each graph in* $\mathrm{obs}(\mathcal{E}_k(\mathcal{G}))$ *has at most* $2^{2^{2^{2^{2^{k^{O(1)}}}}}}$ *vertices. Moreover, if* $\mathrm{obs}(\mathcal{G})$ *contains an apex-graph, this bound drops to* $2^{2^{k^{O(1)}}}$ *.*

The only previously known bound for the graphs in $\mathrm{obs}(\mathcal{E}_k(\mathcal{G}))$ is the one for treedepth by Dvořák, Giannopoulou, and Thilikos [22], who proved that every graph in $\mathrm{obs}(\mathcal{E}_k(\mathcal{G}_\emptyset))$ has size at most $2^{2^{k-1}}$. Theorem 1.5 can be seen as a generalization of the results of Sau, Stamoulis, and Thilikos [69], who provided similar upper bounds for the graphs in $\mathrm{obs}(\mathcal{A}_k(\mathcal{G}))$.

These two results are, to the authors' knowledge, the first upper bounds on the size of the graphs in the obstruction set for the treedepth and the elimination distance parameters, and give, as an immediate consequence, the first known upper bound for the size of these obstruction sets.

Our techniques. We now proceed to provide a high-level overview of the main tools used to prove our results, without getting into technical details. This paper builds heavily on the techniques recently introduced in [70] in order to deal with VERTEX DELETION TO $\mathcal{G}$, which are based on exploiting the Flat Wall Theorem of Robertson and Seymour [63], namely the version proved by Kawarabayashi, Thomas, and Wollan [44] and its recent restatement by Sau, Stamoulis, and Thilikos [68]. In a nutshell, the idea of Theorem 1.1, Theorem 1.2, and Theorem 1.3 is that, as far as the treewidth of the input graph is sufficiently large as an appropriate function of k, it is possible to either "branch" into a number of subproblems that depends only on k and where the value of the parameter is strictly smaller, or to find an irrelevant vertex (i.e., a vertex that does not change the answer to the considered problem) and remove it from the graph. The irrelevant vertex technique originates from Robertson and Seymour [63] and is further developed in [68, 69, 70]. Once the treewidth is bounded, what remains is to apply the most efficient possible algorithm to solve the problem via dynamic programming on tree decompositions.

Let us focus more particularly on the techniques we use to prove Theorem 1.1. Contrary to the algorithm of [70] that solves VERTEX DELETION TO $\mathcal{G}$ for any minor-closed class $\mathcal{G}$, we *avoid using iterative compression*. This explains the improvement from cubic to quadratic complexity in n. The algorithm of Theorem 1.1 can be seen as an extension of the algorithm of [70] that solves VERTEX DELETION TO $\mathcal{G}$ in the particular case where $\mathrm{obs}(\mathcal{G})$ contains some apex-graph, and uses ideas that date back to the work of Marx and Schlotter [55] for the PLANARIZATION problem, that is, when $\mathcal{G}$ is the class of planar graphs. In Section 3 we provide a sketch of the algorithms claimed in Theorem 1.1, Theorem 1.2, and Theorem 1.3, and in Section 6, Section 8, and Section 9, respectively, we present the algorithms in full detail, along with a proof of their correctness.

The proof of Theorem 1.4 consists of a dynamic programming algorithm that combines the framework of [61] for the particular case where $\mathcal{G}$ contains only the empty graph (i.e., for treedepth) with the representative-based techniques introduced in [9]. A bit more precisely, the idea is to encode the partial solutions (called *characteristic*) via sets of annotated trees with some additional properties. Here, the trees correspond to partial elimination trees and the annotations indicate the representatives, in the leaves of the elimination trees, with respect to the canonical equivalence relation defined for the target class $\mathcal{G}$. The size of the characteristic (cf. Lemma 7.6) dominates the running time of the whole algorithm. As usual when dealing with dynamic programming, the formal description of the algorithm, given in Section 7, is quite technical and lengthy.

Finally, to obtain the upper bound on the size of a graph $G \in \text{obs}(\mathcal{E}_k(\mathcal{G}))$ claimed in Theorem 1.5, we proceed in two steps. First, we bound the treewidth of G by a function of k. To do so, we observe that if the treewidth of G is big enough, then there is a big enough wall in G, and we find an irrelevant vertex v for ELIMINATION DISTANCE TO $\mathcal{G}$ in G. However, $G \setminus \{v\} \in \mathcal{E}_k(\mathcal{G})$ and $G \notin \mathcal{E}_k(\mathcal{G})$, hence we reach a contradiction. The second step is to bound the size of a minor-minimal obstruction of small treewidth. This uses the classic technique of Lagergren [52] (see also [31, 30, 39, 40, 32, 51, 53, 69]) combined with the encoding of the tables of the dynamic programming algorithm that we use to prove Theorem 1.4; see Section 10.

2.   Basic definitions and restatement of the problems

In Subsection 2.1 we give some basic definitions on graphs and minors and in Subsection 2.2 we redefine the problems in a more convenient way and we establish some conventions that we will use throughout the paper.

2.1   Basic definitions

Sets and integers. We denote by $\mathbb{N}$ the set of non-negative integers. Given two integers p and q, the set $[p, q]$ contains every integer r such that $p \leq r \leq q$. For an integer $p \geq 1$, we set $[p] = [1, p]$ and $\mathbb{N}_{\geq p} = \mathbb{N} \setminus [0, p - 1]$. Given a non-negative integer x, we denote by $\text{odd}(x)$ the smallest odd number that is not smaller than x. For a set S, we denote by 2^S the set of all subsets of S and, given an integer $r \in [|S|]$, we denote by $\binom{S}{r}$ the set of all subsets of S of size r and by $\binom{S}{\leq r}$ (resp. $\binom{S}{<r}$) the set of all subsets of S of size at most r (resp. $r - 1$). If $\mathcal{S}$ is a collection of objects where the operation $\cup$ is defined, then we denote $\bigcup \mathcal{S} = \bigcup_{X \in \mathcal{S}} X$.

Basic concepts on graphs. All graphs considered in this paper are undirected, finite, and without loops or multiple edges. We use standard graph-theoretic notation and we refer the reader to [20] for any undefined terminology. For convenience, we use uv instead of $\{u, v\}$ to denote an edge of a graph. Let G be a graph. In the rest of this paper we always use $n = |G|$

for the size of G, i.e., the cardinality of $V(G)$, and m for the cardinality of $E(G)$, where G is the input graph of the problem under consideration. We say that a pair $(L, R) \in 2^{V(G)} \times 2^{V(G)}$ is a *separation* of G if $L \cup R = V(G)$ and there is no edge in G between $L \setminus R$ and $R \setminus L$. The *order* of (L, G) is $|L \cap G|$. Given a vertex $v \in V(G)$, we denote by $N_G(v)$ the set of vertices of G that are adjacent to v in G. A vertex $v \in V(G)$ is *isolated* if $N_G(v) = \emptyset$. For $S \subseteq V(G)$, we set $G[S] = (S, E \cap \binom{S}{2})$ and use the shortcut $G \setminus S$ to denote $G[V(G) \setminus S]$. We may also use $G \setminus v$ instead of $G \setminus \{v\}$ for $v \in V(G)$. For $A, B \subseteq V(G)$, $E(A, B)$ denotes the set of edges of G with one extremity in A and the other in B. We use $\mathsf{cc}(G)$ to denote the set of connected components of G.

Dissolutions and subdivisions. Given a vertex $v \in V(G)$ of degree two with neighbors u and w, we define the *dissolution* of v to be the operation of deleting v and, if u and w are not adjacent, adding the edge uw. Given two graphs H and G, we say that H is a *dissolution* of G if H can be obtained from G after dissolving vertices of G. Given an edge $e = uv \in E(G)$, we define the *subdivision* of e to be the operation of deleting e, adding a new vertex w and making it adjacent to u and v. Given two graphs H and G, we say that H is a *subdivision* of G if H can be obtained from G after subdividing edges of G. Observe that G is a subdivision of H iff H is a dissolution of G.

Contractions and minors. The *contraction* of an edge $e = uv$ of a simple graph G results in a simple graph G' obtained from $G \setminus \{u, v\}$ by adding a new vertex w adjacent to all the vertices in the set $N_G(u) \cup N_G(v) \setminus \{u, v\}$. A graph G' is a *minor* of a graph G, denoted by $G' \preceq G$, if G' can be obtained from G by a sequence of vertex removals, edge removals, and edge contractions. If only edge contractions are allowed, we say that G' is a *contraction* of G. Let H be a graph that is a minor of a graph G. We call any subgraph M of G that can be contracted to H a *model* of H in G. Given two graphs H and G, if H is a minor of G then for every vertex $v \in V(H)$ there is a set of vertices in G that are the endpoints of the edges of G contracted towards creating v. We call this set *model* of v in G. Given a finite collection of graphs $\mathcal{F}$ and a graph G, we use notation $\mathcal{F} \preceq G$ to denote that some graph in $\mathcal{F}$ is a minor of G.

Minor obstructions. Let $\mathcal{G}$ be a graph class that is closed under taking minors. Recall that the *minor obstruction set* of $\mathcal{G}$ is defined as the set of all minor-minimal graphs that are not in $\mathcal{G}$, and is denoted by $\mathsf{obs}(\mathcal{G})$. Given a finite non-empty collection of non-empty graphs $\mathcal{F}$, we denote by $\mathsf{exc}(\mathcal{F})$ the set containing every graph G that excludes all graphs in $\mathcal{F}$ as minors. We call each graph in $\mathsf{exc}(\mathcal{F})$ $\mathcal{F}$-*minor-free*. We use $\mathcal{G}_\emptyset$ for the graph class containing only the empty graph $G_\emptyset$. Notice that $\mathsf{obs}(\mathcal{G}_\emptyset) = \{K_1\}$.

2.2   Restating the problems

Let $\mathcal{G}$ be a minor-closed graph class and $\mathcal{F}$ be its obstruction set. Clearly, Vertex Deletion to $\mathcal{G}$ is the same problem as asking, given a graph G and some $k \in \mathbb{N}$, for a vertex set $S \subseteq V(G)$ of at most k vertices such that $G \setminus S \in \mathsf{exc}(\mathcal{F})$. Following the terminology of [10, 11, 12, 9, 29, 28, 46, 47, 70], we call this problem $\mathcal{F}$-M-Deletion. Likewise, Elimination Distance to $\mathcal{G}$ is the same problem as asking whether $\mathsf{ed}_{\mathsf{exc}(\mathcal{F})}(G) \leq k$. We will thus follow a similar notation and call this problem $\mathcal{F}$-M-Elimination Distance. Using the notation, $\{K_1\}$-M-Elimination Distance is the problem of asking whether $\mathsf{td}(G) \leq k$. We say that $\mathcal{F}$ is *non-trivial* when all graphs in $\mathcal{F}$ contain at least two vertices.

Some conventions. In the rest of the paper, we fix $\mathcal{G}$ to be a minor-closed graph class and $\mathcal{F}$ to be the set $\mathsf{obs}(\mathcal{G})$. From Robertson and Seymour's theorem [65], we know that $\mathcal{F}$ is a finite collection of graphs. Given a graph G, we define its *apex number* to be the smallest integer a for which there is a set $A \subseteq V(G)$ of size at most a such that $G \setminus A$ is planar. An *apex-graph* is a graph with apex number one. Also, we define the *detail* of G, denoted by $\mathsf{detail}(G)$, to be the maximum among $|E(G)|$ and $|V(G)|$. We define three constants depending on $\mathcal{F}$ that will be used throughout the paper whenever we consider such a collection $\mathcal{F}$. We define $a_{\mathcal{F}}$ as the minimum apex number of a graph in $\mathcal{F}$, we set $s_{\mathcal{F}} := \max\{|V(H)| \mid H \in \mathcal{F}\}$, and we set $\ell_{\mathcal{F}} := \max\{\mathsf{detail}(H)| \mid H \in \mathcal{F}\}$. Given a tuple $\mathbf{t} = (x_1, \ldots, x_\ell) \in \mathbb{N}^\ell$ and two functions $\chi, \psi : \mathbb{N} \to \mathbb{N}$, we write $\chi(n) = O_{\mathbf{t}}(\psi(n))$ in order to denote that there exists a computable function $\phi : \mathbb{N}^\ell \to \mathbb{N}$ such that $\chi(n) = O(\phi(\mathbf{t}) \cdot \psi(n))$. Notice that $s_{\mathcal{F}} \leq \ell_{\mathcal{F}} \leq s_{\mathcal{F}}(s_{\mathcal{F}} - 1)/2$, and thus $O_{\ell_{\mathcal{F}}}(\cdot) = O_{s_{\mathcal{F}}}(\cdot)$. Observe also that $\mathcal{A}_k(\mathcal{G})$ and $\mathcal{E}_k(\mathcal{G})$ are $K_{s_{\mathcal{F}}+k}$-minor-free graph classes, and thus, due to [72], we can always assume that G has $O_{s_{\mathcal{F}}}(k\sqrt{\log k} \cdot n)$ edges, otherwise we can directly conclude that (G, k) is a no-instance for both problems.

3.   Sketch of the algorithms

Before going further through the definitions, let us provide a sketch of the algorithms claimed in Theorem 1.1, Theorem 1.2 and Theorem 1.3. As mentioned in the introduction, Theorem 1.1 can be seen as a generalization of the algorithm of [70] that solves $\mathcal{F}$-M-Vertex Deletion in the particular case where $\mathcal{F}$ contains some apex-graph. While many techniques taken from [70] remain the same, some new ingredients are needed so as to deal with the possible existence of many apices in all graphs in $\mathcal{F}$. On the other hand, Theorem 1.2 and Theorem 1.3 can be seen as an adaptation of Theorem 1.1 to $\mathcal{F}$-M-Elimination Distance. Since these three algorithms follow a common streamline, we sketch all of them simultaneously while pointing out the steps where they differ. Moreover, the full proofs of Theorem 1.1, Theorem 1.2, and Theorem 1.3 are given in Section 6, Section 8, and Section 9, respectively.

Walls and flat walls. In this paper we extensively deal with walls and flat walls, following the framework of [68]. Unfortunately, almost ten pages are required to provide all the technical notions to correctly present all this framework, that is necessary to use the tools developed in [68, 69, 70]. Thus, we only give some intuition on those definitions for the sketch of the algorithm, while the formal definitions are deferred to Section 5. More precisely, in Subsection 5.1, we introduce walls and several notions concerning them (just look at Figure 2 to understand what a wall is). In Subsection 5.2 we provide the definitions of a rendition and a painting, which are not crucial for understanding this section. Using the above notions, in Subsection 5.3 we define flat walls and flatness pairs. There are a number of technical terms (such as tilts, influence, regular flatness pairs, ...) that are not the main focus of the sketch. Let us just mention that the perimeter of a flat wall of a graph G separates $V(G)$ into two sets X and Y with Y containing the wall. The *compass* of a flat wall is $G[Y]$. See Figure 1: X is the set of vertices in the light green part, and Y the set of vertices in the pink part.
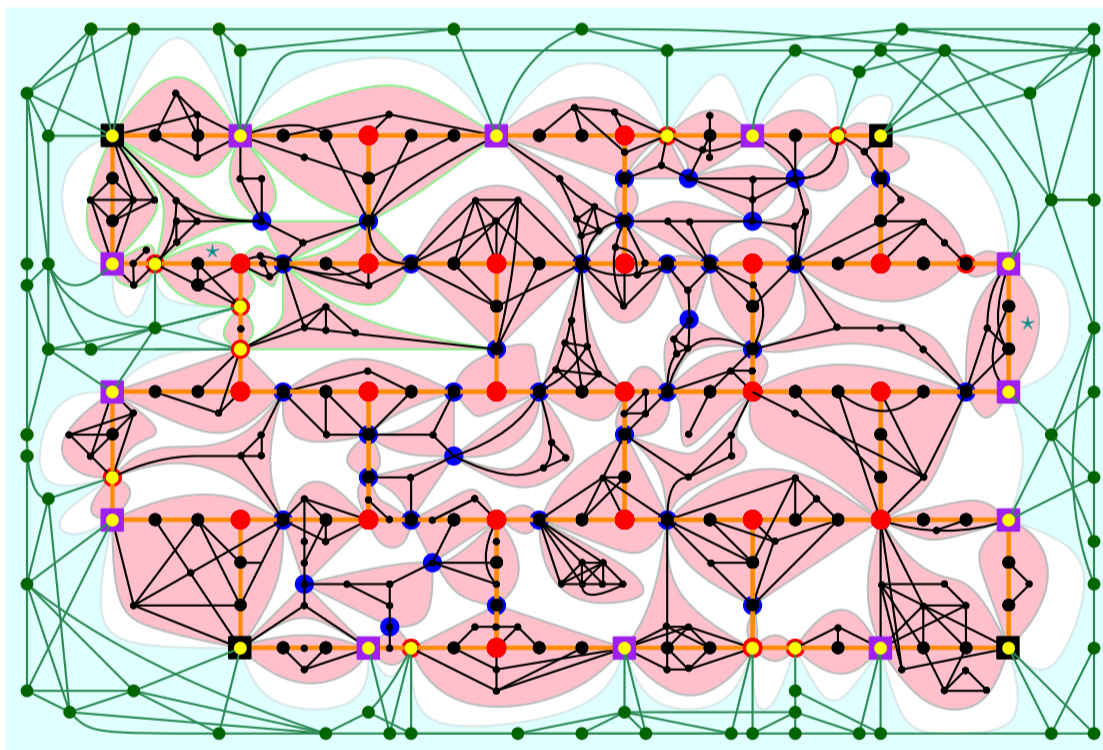
Figure 1. Illustration of a flat wall W inside a graph G. The edges of W are depicted in orange and the compass of W in G is the union of all parts of G that are drawn in pink cells. The graph in each such pink cell corresponds to a flap of the flat wall.

In Subsection 5.4 and Subsection 5.5 we define canonical partitions and the notion of bidimensionality. Informally speaking, a *canonical partition* of a graph with respect to some wall W refers to a partition of the vertex set of a graph in bags that follow the structure of a wall subgraph of the given graph; see Figure 3 for an illustration. The *bidimensionality* of a vertex set X with respect to a wall W of a graph G intuitively expresses the "spread" of a set X

in a W-canonical partition of G. The crucial idea is that a set X of small bidimensionality cannot "destroy" a large (flat) wall too much.

Finally, in Subsection 5.6 we present homogeneous walls. Intuitively, *homogeneous flat walls* are flat walls that allow the routing of the same set of (topological) minors in the augmented flaps (i.e., the flaps together with the apex set) "cropped" by each one of their bricks. Such a homogeneous wall can be detected in a big enough flat wall (Proposition 3.5) and this "homogeneity" property implies that some central part of a big enough homogeneous wall can be declared irrelevant (Proposition 3.6).

The first common step is to run the following algorithm that states that a graph G in $\mathcal{A}_k(\mathrm{exc}(\mathcal{F}))$ or $\mathcal{E}_k(\mathrm{exc}(\mathcal{F}))$ either has bounded treewidth (see Subsection 4.2 for the formal definition) or contains a large wall. This result was proved in [70] in the case of $\mathcal{F}$-M-DELETION. The proof in the case of $\mathcal{F}$-M-ELIMINATION DISTANCE, necessary for Theorem 1.2 and Theorem 1.3, can be found in Subsection 8.1.

PROPOSITION 3.1 ([70], Subsection 8.1). *Let $\mathcal{F}$ be a finite collection of graphs. There exist a function $f_1 : \mathbb{N} \to \mathbb{N}$ and an algorithm with the following specifications:*

Find-Wall(G, r, k)

Input: *A graph G, an odd $r \in \mathbb{N}_{\geq 3}$, and $k \in \mathbb{N}$.*

Output: *One of the following:*

— **Case 1:** *Either a report that (G, k) is a* no*-instance of $\mathcal{F}$-M-DELETION (resp. $\mathcal{F}$-M-ELIMINATION DISTANCE), or*

— **Case 2:** *a report that G has treewidth at most $f_1(s_\mathcal{F}) \cdot r + k$, or*

— **Case 3:** *an r-wall W of G.*

Moreover, $f_1(s_\mathcal{F}) = 2^{O(s_\mathcal{F}^2 \cdot \log s_\mathcal{F})}$, and the algorithm runs in time $2^{O_{\ell_\mathcal{F}}(r^2 + (k+r) \cdot \log(k+r))} \cdot n$ (resp. $2^{O_{\ell_\mathcal{F}}(r^2 + k^2)} \cdot n$).

In **Case 1**, we can immediately conclude. In **Case 2**, since the treewidth of G is bounded, we use a dynamic programming algorithm to solve the corresponding problem. Namely, we solve $\mathcal{F}$-M-DELETION on instances of bounded treewidth using the main result from [9].

PROPOSITION 3.2 ([9]). *For every finite collection of graphs $\mathcal{F}$, there exists an algorithm that, given a triple (G, tw, k) where G is a graph of treewidth at most tw and k is a non-negative integer, solves $\mathcal{F}$-M-DELETION in time $2^{O_{\ell_\mathcal{F}}(\mathrm{tw} \cdot \log \mathrm{tw})} \cdot n$.*

For $\mathcal{F}$-M-ELIMINATION DISTANCE, we use Theorem 1.4 to conclude. The proof of this (quite technically involved) dynamic programming algorithm is given in Section 7.

Therefore, it only remains to deal with **Case 3**. Given an r-wall W of G, we want to reduce the size of G. To do so, we observe that we can either:

— **Case 3a:** find a subwall W_a of W and an apex set A_a such that W_a is flat in $G \setminus A_a$ and has a compass of bounded treewidth, or

— **Case 3b:** find a subwall W_b of W that is very "well connected" to an apex set A_b of small size.

The above distinction is done using two algorithmic versions of the Flat Wall Theorem consecutively. The first one comes from [44, Theorem 7.7] and is translated here in the new framework with tilts of [68]. Informally, we say that a graph H is *grasped* by a wall W in a graph G if there is a model of H in G such that the model of every node of H intersects W. See Section 5 for the formal definition.

PROPOSITION 3.3 ([44]). *There are two functions $f_2, f_3 : \mathbb{N} \to \mathbb{N}$, such that the images of f_2 are odd integers, and an algorithm with the following specifications:*

Grasped-or-Flat(G, r, t, W)

Input: A graph G, an odd $r \in \mathbb{N}_{\geq 3}$, $t \in \mathbb{N}_{\geq 1}$, and an $f_2(t) \cdot r$-wall W of G.
Output: One of the following:

— *Either a model of a K_t-minor in G grasped by W, or*

— *a set $A \subseteq V(G)$ of size at most $f_3(t)$ and a flatness pair $(W', \mathfrak{R}')$ of $G \setminus A$ of heigth r such that W' is a $\tilde{W}'$-tilt of some subwall $\tilde{W}'$ of W.*

Moreover, $f_2(t) = O(t^{26})$, $f_3(t) = O(t^{24})$, and the algorithm runs in time $O(t^{24}m + n)$.

We would like to mention that the notion of being grasped by a wall is one of the new main arguments yileding the improvement of the complexity for $\mathcal{F}$-M-Deletion compared to [70].

The second one comes from [70] and adds the condition that W' has a compass of bounded treewidth, at the price of dropping the condition that the model of K_t is grasped by W.

PROPOSITION 3.4 ([70]). *There exist a function $f_4 : \mathbb{N} \to \mathbb{N}$ and an algorithm with the following specifications:*

Clique-Or-twFlat(G, r, t)

Input: A graph G, an odd $r \in \mathbb{N}_{\geq 3}$, and $t \in \mathbb{N}_{\geq 1}$.
Output: One of the following:

— *Either a report that K_t is a minor of G, or*

— *a tree decomposition of G of width at most $f_4(t) \cdot r$, or*

— *a set $A \subseteq V(G)$ of size at most $f_3(t)$ and a regular flatness pair $(W', \mathfrak{R}')$ of $G \setminus A$ of height r whose $\mathfrak{R}'$-compass has treewidth at most $f_4(t) \cdot r$.*

Moreover, $f_4(t) = 2^{O(t^2 \log t)}$ and this algorithm runs in time $2^{O_t(r^2)} \cdot n$. The algorithm can be modified to obtain an explicit dependence on t in the running time, namely $2^{2^{O(t^2 \log t)} \cdot r^3 \log r} \cdot n$.

Grasped-or-Flat is used to find a big enough complete graph "controlled" by the input wall, while we need Clique-or-twFlat to find a flat wall whose compass has bounded treewidth.

Unfortunately, we cannot obtain both conditions simultaneously, and this is why we need both results. If, after using both algorithms, we obtain a flatness pair $(\tilde{W}', \mathfrak{R}')$ of $G \setminus A_a$ of heigth r_a whose compass has bounded treewidth, then we are in **Case 3a**. In that case, the following result from [70] provides an algorithm that, given a flatness pair of big enough height, outputs a homogeneous flatness pair.

PROPOSITION 3.5 ([70]). *There is a function $f_5 : \mathbb{N}^4 \to \mathbb{N}$, whose images are odd integers, and an algorithm with the following specifications:*

Homogeneous$(r, \tilde{a}, a, \ell, t, G, A, W, \mathcal{R})$
Input: *Five integers $r \in \mathbb{N}_{\geq 3}$, $\tilde{a}, a, \ell, t \in \mathbb{N}$, where $\tilde{a} \leq a$, a graph G, a set $A \subseteq V(G)$ of size at most a, and a flatness pair $(W, \mathfrak{R})$ of $G \setminus A$ of height $f_5(r, a, \tilde{a}, \ell)$ whose $\mathfrak{R}$-compass has treewidth at most t.*
Output: *A flatness pair $(\check{W}, \check{\mathfrak{R}})$ of $G \setminus A$ of height r that is ℓ-homogeneous with respect to $\binom{A}{\leq \tilde{a}}$ and is a W'-tilt of $(W, \mathfrak{R})$ for some subwall W' of W.*
Moreover, $f_5(r, \tilde{a}, a, \ell) = O(r^{f_6(\tilde{a}, a, \ell)})$ where $f_6(\tilde{a}, a, \ell) = 2^{a^{\tilde{a}} \cdot 2^{O((\tilde{a}+\ell) \cdot \log(\tilde{a}+\ell))}}$ and the algorithm runs in time $2^{O(f_6(\tilde{a}, a, \ell) \cdot r \log r + t \log t)} \cdot (n + m)$.

Then we use the next result, that essentially says that the central vertex v of a big enough homogeneous wall is irrelevant, i.e., (G, k) and $(G \setminus v, k)$ are equivalent instances of the corresponding problem. Here, $\mathrm{bid}_{G \setminus A, W}(X)$ denotes the bidimensionality of a set X in the wall W with apex set A. The combinatorial version of this result is stated in [69, Lemma 16] and can be algorithmized using [68, Theorem 5] (Proposition 5.4). Before presenting the next result we give some insight on the Unique Linkage Theorem.

A *linkage L* of *order k* in a graph G is the union of a collection of k pairwise-disjoint paths of G. The set of pairs of vertices corresponding to the endpoints of these paths is the *pattern* of L. The Unique Linkage Theorem, proven in [67, 66] and also [45], asserts that there is a function f_{ul} such that if L is a linkage of pattern $\mathcal{P}$ of order k in a graph G with $V(G) = V(L)$ and L is unique with pattern $\mathcal{P}$, then the treewidth of G is at most $f_{\mathsf{ul}}(k)$. The linkage function appears in the general dependency of several results related to the application of the irrelevant vertex technique (see [68, 69, 13, 33, 3, 26, 34]).

PROPOSITION 3.6 ([68, 69]). *Let $\mathcal{F}$ be a finite collection of graphs. There exist two functions $f_7 : \mathbb{N}^4 \to \mathbb{N}$ and $f_8 : \mathbb{N}^2 \to \mathbb{N}$, and an algorithm with the following specifications:*

Find-Irrelevant-Vertex$(k, a, G, A, W, \mathcal{R})$
Input: *Two integers $k, a \in \mathbb{N}$, a graph G, a set $A \subseteq V(G)$, and a regular flatness pair $(W, \mathcal{R})$ of $G \setminus A$ of height at least $f_7(a, \ell_{\mathcal{F}}, 3, k)$ that is $f_8(a, \ell_{\mathcal{F}})$-homogeneous with respect to $\binom{A}{\leq a}$.*
Output: *A vertex v of $G \setminus A$ such that for every set $X \subseteq V(G)$ with $\mathrm{bid}_{G \setminus A, W}(X) \leq k$ and $|A \setminus X| \leq a$, it holds that $G \setminus X \in \mathrm{exc}(\mathcal{F})$ if and only if $G \setminus (X \setminus v) \in \mathrm{exc}(\mathcal{F})$.*

Moreover, $f_7(a, \ell_{\mathcal{F}}, q, k) = O(k \cdot (f_{ul}(16a + 12\ell_{\mathcal{F}}))^3 + q)$, where f_{ul} is the function of the Unique Linkage Theorem ([45, Theorem 1]) and $f_8(a, \ell_{\mathcal{F}}) = a + \ell_{\mathcal{F}} + 3$, and this algorithm runs in time $O(n + m)$.

We can prove that both k-apex sets and k-elimination sets have small bidimensionality (cf. Observation 5.7 and Lemma 5.9). If, for every k-apex set S, $G \setminus S \in \text{exc}(\mathcal{F})$ if and only if $G \setminus (S \setminus v) \in \text{exc}(\mathcal{F})$, then it is straightforward to see that v is irrelevant for $\mathcal{F}$-M-DELETION. It is slightly less trivial to prove that, for each k-elimination set S, we can find some superset $X \supseteq S$ of small bidimensionality such that a similar statement holds. Additional details are available in Subsection 5.5 of Section 5.

Therefore we can recursively solve the problems on the instance $(G \setminus v, k)$.

If no flatness pair whose compass has bounded treewidth was found, then we are in **Case 3b**. In this case, inspired by [55] and [70], we use the following result of [69] that basically says that if there is a big enough flat wall W and an apex set A' of $a_{\mathcal{F}}$ vertices that are all adjacent to many bags of a canonical partition of W, then each k-apex set or k-elimination set intersects A'.

PROPOSITION 3.7 ([69]). *There exist three functions $f_9, f_{10}, f_{11} : \mathbb{N}^3 \to \mathbb{N}$, such that if G is a graph, $k \in \mathbb{N}$, A is a subset of $V(G)$, $(W, \mathfrak{R})$ is a flatness pair of $G \setminus A$ of height at least $f_9(a_{\mathcal{F}}, s_{\mathcal{F}}, k)$, $\tilde{Q}$ is a W-canonical partition of $G \setminus A$, A' is a subset of vertices of A that are adjacent, in G, to vertices of at least $f_{10}(a_{\mathcal{F}}, s_{\mathcal{F}}, k)$ $f_{11}(a_{\mathcal{F}}, s_{\mathcal{F}}, k)$-internal bags of $\tilde{Q}$, and $|A'| \geq a_{\mathcal{F}}$, then for every set $X \subseteq V(G)$ such that $G \setminus X \in \text{exc}(\mathcal{F})$ and $\text{bid}_{G \setminus A, W}(X) \leq k$, it holds that $X \cap A' \neq \emptyset$. Moreover, $f_9(a, s, k) = O(2^a \cdot s^{5/2} \cdot k^{5/2})$, $f_{10}(a, s, k) = O(2^a \cdot s^3 \cdot k^3)$, and $f_{11}(a, s, k) = O((a^2 + k) \cdot s)$, where $a = a_{\mathcal{F}}$ and $s = s_{\mathcal{F}}$.*

For the $\mathcal{F}$-M-DELETION problem, if we find such a set A', then we can branch by guessing which vertex $v \in A'$ belongs to a k-apex set and recursively solving $(G \setminus v, k - 1)$. Given that A' has size $a_{\mathcal{F}}$ and that k decreases after each guess, this step is applied at most $a_{\mathcal{F}}^k$ times.

However, for $\mathcal{F}$-M-ELIMINATION DISTANCE, k does not decrease, given that the size of a k-elimination set may not depend on k. Thus, this step may be done $a_{\mathcal{F}}^n$ times, which does not give an FPT-algorithm. To circumvent this problem, we propose two alternatives:

Option 1: The first alternative is to only use **Case 3a**. This is possible given that $(K_{s_{\mathcal{F}}+k}, k)$ is a no-instance of both problems. Thus, when using the algorithms `Grasped-or-Flat` and `Clique-or-twFlat`, we force the outcome to be an apex set A and a flatness pair of $G \setminus A$. However, the bound on the size of A now depends on k, and thus, so does the variable a in the input of the algorithm `Homogeneous`. This explains the triple-exponential parametric dependence on k in Theorem 1.2. Interestingly, a precise analysis of the time complexity, which can be found in Section 8, shows that if $a_{\mathcal{F}} = 1$, i.e., when $\mathcal{F}$ contains an apex graph, the parametric dependence is only double-exponential on k (cf. Theorem 1.2).

Option 2: The second alternative is to restrict ourselves to the case where $a_{\mathcal{F}} = 1$. Thus, in **Case 3b**, we find a vertex v that belongs to every k-elimination set. There is no need to branch, and this step is done at most n times. However, the fact that the time complexity of this step is quadratic in n explains the cubic complexity of the algorithm in Theorem 1.3.

It remains to show that if no flatness pair whose compass has bounded treewidth was found, then we can find a flatness pair and a set A' satisfying the conditions of Proposition 3.7. To do so, using flow techniques, we find the set A of vertices with sufficiently many internally-disjoint paths to W, independently from one another. If this set is too large, we can safely declare a no-instance. Otherwise, we extend the canonical partition of W and just check whether $a_{\mathcal{F}}$ vertices of A are adjacent to many vertices of this new canonical partition. If this happens, then we can safely use Proposition 3.7. The second main improvement with respect to the algorithm in [70] is the new argument that the extension of the canonical partition of W can be done in a totally arbitrary manner. The quadratic complexity of this step stems from the search for internally-disjoint paths for every vertex of the input graph.

4. More definitions

In this section, we give some more definitions. Namely, in Subsection 4.1, we define elimination trees, that is an alternative way to define elimination distance of a graph to a graph class. Next, in Subsection 4.2, we define (nice) tree decompositions and we present some preliminary results. Finally, in Subsection 4.3 we define boundaried graphs, an equivalence relation between them, and the notion of representatives.

4.1 $\mathcal{F}$-elimination trees

We start this subsection by defining some notions on (rooted) trees.

Trees and rooted trees. Let T be a tree and u, v be two nodes of T. We denote by uTv the path in T u and v. A *rooted tree* is a pair (T, r) where T is a tree and r is a node of T called *root* of (T, r). Let (T, r) be a rooted tree and let u be a node of T. We define the *descendants* of u in (T, r) by $\mathsf{Desc}_{T,r}(u) = \{x \in V(T) \mid u \in V(xTr)\}$ and the *ancestors* of u in (T, r) by $\mathsf{Anc}_{T,r}(u) = V(rTu) \setminus \{u\}$. We define the *leaves* in (T, r) by $\mathsf{Leaf}(T, r) = \{u \in V(T) \mid \mathsf{Desc}_{T,r}(u) = \{u\}\}$ and the *internal nodes* in (T, r) by $\mathsf{Int}(T, r) = V(T) \setminus \mathsf{Leaf}(T, r)$. If $u \neq r$ then we denote by $\mathsf{Par}_{T,r}(u)$ the unique node in $\mathsf{Anc}_{T,r}(u) \cap N_T(u)$. We also agree that $\mathsf{Par}_{T,r}(r) = \text{void}$. We denote by $\mathsf{Ch}_{T,r}(u) = \mathsf{Desc}_{T,r}(u) \cap N_G(u)$ the set of the children of u (certainly $\mathsf{Ch}_{T,r}(u) = \emptyset$ if u is a leaf of T). Given $K \subseteq V(T)$, the *least common ancestor* of K in (T, r) is the node u such that $K \subseteq \mathsf{Desc}_{T,r}(u)$ and there is no child v of u such that $K \subseteq \mathsf{Desc}_{T,r}(v)$.

The *height function* $\text{height}_{T,r} : V(T) \to \mathbb{N}$ maps $v \in \text{Leaf}(T,r)$ to 0 and $v \in \text{Int}(T,r)$ to $1 + \max\{\text{height}_{T,r}(x) \mid x \in \text{Ch}_{T,r}(v)\}$. The *height* of (T,r) is $\text{height}_{T,r}(r)$. Note that the height function is decreased by one here compared to the usual definition of the height.

We use (T_u^r, u) to denote the rooted tree where $T_u^r = T[\text{Desc}_{T,r}(u)]$ and we call (T_u^r, u) *subtree of* (T,r) *rooted at* u. To simplify notation and when the root r is clear from the context, we use T_u instead of T_u^r.

A *rooted forest* is a pair (F,R) where F is a forest and R is a set of roots such that each tree in F has exactly one root in R. All notations above naturally extend to forests.

Elimination trees. We now define *elimination trees*, that can be used to define alternatively graphs of bounded elimination distance. Let $\mathcal{F}$ be a non-empty finite collection of non-empty graphs. An $\mathcal{F}$-*elimination tree* of a connected graph G is a triple (T,χ,r) where (T,r) is a rooted tree and $\chi : V(T) \to 2^{V(G)}$ such that:

— for each $t \in \text{Int}(T,r)$, $|\chi(t)| = 1$,

— $(\chi(t))_{t \in V(T)}$ is a partition of $V(G)$,

— for each $uv \in E(G)$, if $u \in \chi(t_1)$ and $v \in \chi(t_2)$, then $t_1 \in \text{Anc}_{T,r}(t_2) \cup \text{Desc}_{T,r}(t_2)$,

— for each $t \in \text{Leaf}(T,r)$, $G[\chi(t)] \in \text{exc}(\mathcal{F})$, and

— for each $t \in V(T)$, $G[\chi(T_t)]$ is connected.

The *height* of (T,χ,r) is the height of (T,r). It is straightforward to see that the minimum height of an $\mathcal{F}$-elimination tree of a connected graph G is $\text{ed}_{\text{exc}(\mathcal{F})}(G)$. Note that $\chi(\text{Int}(T,r))$ is a k-elimination set of G for $\text{exc}(\mathcal{F})$ and that, if $\mathcal{F}$ is trivial, then, for each $t \in \text{Leaf}(T,r)$, $\chi(t) = \emptyset$. Observe also that for every $u \in \text{Int}(T,r)$ with at least two children x and y, any path between $\chi(T_x)$ and $\chi(T_y)$ intersects $\chi(uTr)$.

An $\mathcal{F}$-*elimination forest* of a graph G is a triple (F,χ,R), such that, if $\text{cc}(G) = \{G_1, ..., G_l\}$, then F is the disjoint union of the trees $T_1, \ldots, T_l$ and $R = \{r_1, ..., r_l\}$ where $(T_i, \chi|_{V(T_i)}, r_i)$ is an $\mathcal{F}$-elimination tree of G_i for $i \in [l]$.

The following simple lemma is based on the fact that, given an $\mathcal{F}$-elimination tree (T,χ,r) of a graph G, for every non-leaf node u of T, $\chi(uTr)$ separates the vertex sets $\chi(T_x)$ and $\chi(T_y)$, where x and y are distinct children of v in (T,r).

LEMMA 4.1. *Let $\mathcal{F}$ be a finite collection of graphs. Let G be a graph and let H be a connected subgraph of G. Let (F,χ,R) be an $\mathcal{F}$-elimination forest of G. Then the least common ancestor of $\chi^{-1}(V(H))$ exists and belongs to $\chi^{-1}(V(H))$.*

PROOF. Let $K := \chi^{-1}(V(H))$. Since H is connected, K is a subset of a tree in F, and therefore the least common ancestor of K is defined. Let u be the least common ancestor of K. Let $r \in R$ be the root of the tree containing u. Let $x, y \in V(H)$ such that the least common ancestor of $\chi^{-1}(x)$ and $\chi^{-1}(y)$ is u. Since H is connected, there is a path P in H between x and y. By the

third property of elimination trees, $\{u\} \cup \text{Anc}_{F,R}(u)$ intersects $\chi^{-1}(V(P))$, and so $\{u\} \cup \text{Anc}_{F,R}(u)$ intersects K. Since u is the least common ancestor of K, $u \in K$. ∎

We now present a lemma to justify that the graphs with bounded elimination distance to $\text{exc}(\mathcal{F})$ are minor-free. Intuitively, the proof of this lemma is based on the fact that, due to Lemma 4.1, the size of the largest clique minor that can "fit" inside an elimination tree is equal to the height of the elimination tree.

LEMMA 4.2. *Let $\mathcal{F}$ be a finite collection of graphs. Let G be a graph and $k \in \mathbb{N}$ such that $\text{ed}_{\text{exc}\mathcal{F}}(G) \leq k$. Then $K_{s_{\mathcal{F}}+k}$ is not a minor of G.*

PROOF. Let (F, χ, R) be an $\mathcal{F}$-elimination forest of G of height at most k. Suppose towards a contradiction that there is a model of $K_{s_{\mathcal{F}}+k}$ in G. Let $x_1, ..., x_{s_{\mathcal{F}}+k}$ be the vertices of $K_{s_{\mathcal{F}}+k}$ and for every $i \in [s_{\mathcal{F}} + k]$, let V_i be the model of x_i in G. Let G' be the graph obtained by contracting, for each $i \in [s_{\mathcal{F}} + k]$, the edges in each V_i. Let $v_i, i \in [s_{\mathcal{F}} + k]$ be the resulting vertices after the contraction of each V_i. Thus, the graph $G'[\{v_1, ..., v_{s_{\mathcal{F}}+k}\}]$ is isomorphic to $K_{s_{\mathcal{F}}+k}$.

Let (F', χ', R) be obtained from (F, χ, R) as follows. For every $i \in [s_{\mathcal{F}} + k]$, let u_i be the least common ancestor of $K_i := \chi^{-1}(V_i)$ in (F, R). Due to Lemma 4.1, $u_i \in K_i$. The forest F' is obtained after removing each node $v \in (K_i \setminus u_i) \cap (\text{Int}(F, R) \setminus R)$ from $V(F)$ and adding an edge between $\text{Par}_{F,R}(v)$ and each node in $\text{Ch}_{F,R}(v)$. The function χ' is defined as $\chi'(v) := \chi(v)$ if $v \in \text{Int}(F', R)$ and $\chi'(v) := \chi(v) \setminus (V_i \setminus u_i)$ if $v \in \text{Leaf}(F', R)$. In the latter case, if $G'[\chi'(v)]$ is not connected, then we update F' by replacing v by $|\text{cc}(G'[\chi'(v)])|$ nodes, each one associated with a connected component of $G'[\chi'(v)]$. Observe that (F', χ', R) is an $\mathcal{F}$-elimination forest of G' of height at most k and that we can assume that for every $i \in [s_{\mathcal{F}} + k]$, $v_i \in \chi'(u_i)$.

Since the vertices in $\{v_1, ..., v_{s_{\mathcal{F}}+k}\}$ are pairwise connected by an edge in G', the third property of elimination trees implies that there is $u \in \text{Leaf}(F', R)$ and $r \in R$ such that $\{v_1, ..., v_{s_{\mathcal{F}}+k}\} \subseteq \chi'(uF'r)$. Let $u' := \text{Par}_{F',R}(u)$. For every $t \in u'F'r$, $t \in \text{Int}(F', R)$, so $|\chi'(t)| = 1$, and therefore, $|\chi'(u'Fr)| \leq k$. Thus, $|\chi'(u) \cap V(H)| \geq s_{\mathcal{F}}$. Therefore, $K_{s_{\mathcal{F}}}$ is a minor of $G'[\chi'(u)]$. This contradicts the fact that $G'[\chi'(u)] \in \text{exc}(\mathcal{F})$, so $K_{s_{\mathcal{F}}+k}$ is not a minor of G. ∎

4.2   Tree decompositions

The notions of tree decompositions and treewidth are used throughout the paper.

Treewidth. A *tree decomposition* of a graph G is a pair (T, β) where T is a tree and $\beta : V(\mathsf{T}) \to 2^{V(G)}$ such that

— $\bigcup_{t \in V(\mathsf{T})} \beta(t) = V(G)$,

— for every $e \in E(G)$, there is a $t \in V(\mathsf{T})$ such that $\beta(t)$ contains both endpoints of e, and

— for every $v \in V(G)$, the subgraph of T induced by $\{t \in V(\mathsf{T}) \mid v \in \beta(t)\}$ is connected.

The *width* of (T, β) is equal to $\max\left\{ |\beta(t)| - 1 \mid t \in V(\mathsf{T}) \right\}$ and the *treewidth* of G, denoted by $\mathrm{tw}(G)$, is the minimum width over all tree decompositions of G. A *rooted tree decomposition* is a triple (T, β, r) where (T, β) is a tree decomposition and (T, r) is a rooted tree. For every $q \in V(\mathsf{T})$, we set $G_q^{(\mathsf{T},\beta,r)} = G[\beta(\mathsf{T}_q)]$. We may write G_q instead of $G_q^{(\mathsf{T},\beta,r)}$ when there is no ambiguity about (T, β, r).

To compute a tree decomposition of a graph of bounded treewidth, we use the recent single-exponential 2-approximation algorithm for treewidth of Korhonen [50].

PROPOSITION 4.3 ([50]). *There is an algorithm that, given an graph G and an integer k, outputs either a report that $\mathrm{tw}(G) > k$, or a tree decomposition of G of width at most $2k + 1$ with $O(n)$ nodes. Moreover, this algorithm runs in time $2^{O(k)} \cdot n$.*

The relation between the treedepth and the treewidth of a graph proved by Bodlaender, Gilbert, Kloks, and Hafsteinsson in [15] will be used in Section 7 in order to obtain an XP-algorithm for $\mathcal{F}$-M-ELIMINATION DISTANCE parameterized by treewidth.

PROPOSITION 4.4 ([15]). *Let G be a graph with n vertices. Then $\mathrm{tw}(G) \le \mathrm{td}(G) \le \mathrm{tw}(G) \cdot \log n$.*

The following result has been proved by Adler, Dorn, Fomin, Sau, and Thilikos in [1]. We use it in Section 8 and, in particular, in the algorithm `Find-Wall-Ed` of Proposition 3.1, to detect a wall in a graph of bounded treewidth.

PROPOSITION 4.5 ([1]). *There is an algorithm that, given a graph G on m edges, a graph H on h edges without isolated vertices, and a tree decomposition of G of width at most k, outputs, if it exists, a minor of G isomorphic to H. Moreover, this algorithm runs in time $2^{O(k \log k)} \cdot h^{O(k)} \cdot 2^{O(h)} \cdot m$.*

We also show that given a graph G and an integer k, the removal of a k-elimination set from G does not decrease the treewidth of G more than k.

LEMMA 4.6. *Let $\mathcal{F}$ be a finite collection of graphs. Let c, k be two integers and let G be a graph such that $\mathrm{tw}(G) \ge c$. Let S be a k-elimination set of G for $\mathrm{exc}(\mathcal{F})$. Then $\mathrm{tw}(G \setminus S) \ge c - k$.*

PROOF. Suppose first that G is connected. Let (T, χ, r) be an $\mathcal{F}$-elimination tree of G with $\chi(\mathrm{Int}(T, r)) = S$. Let $v_1, \ldots, v_l$ be the leaves of (T, r), whose label is given by a depth-first search order starting from r. Let $C_i := G[\chi(v_i)]$ for $i \in [l]$, and note that $\mathrm{tw}(G \setminus S) = \max_{i \in [l]} \mathrm{tw}(C_i)$. Suppose for contradiction that $\mathrm{tw}(G \setminus S) < c - k$, and we proceed to show that $\mathrm{tw}(G) < c$, contradicting our hypothesis. Let (T_i, β_i) be an optimal tree decomposition of C_i of width w_i and let P_i be the path from the parent of v_i to r in T, for $i \in [l]$. Let $w := \max_{i \in [l]} w_i$, so we have that $w \le c - k - 1$. We construct a tree decomposition $(\mathcal{T}, \beta)$ of G, starting from the tree decompositions (T_i, β_i), as follows. Create a path with nodes $x_1, \ldots, x_l$ such that for $i \in [l]$,

$\beta(x_i) = V(P_i)$. Then for $i \in [l]$, add an edge between x_i and a node of T_i. For each $x \in V(\mathsf{T}_i)$, we set $\beta(x) := \beta_i(x) \cup V(P_i)$. Since the height of (T, r) is at most k, P_i has size at most k for $i \in [l]$, so $(\mathcal{T}, \beta)$ has width at most $w + k \leq c - 1$, a contradiction.

If G is not connected, we can apply the above proof to each of its connected components. ∎

To describe our dynamic programming algorithm presented in Section 7, we need a particular type of tree decompositions, namely nice tree decompositions.

Nice tree decompositions. A *nice tree decomposition* of a graph G is a rooted tree decomposition $(\mathsf{T}, \beta, \mathsf{r})$ such that:

— every node has either zero, one or two children,
— if x is a leaf of T, then $\beta(x)$ is a singleton (x is a *leaf node*),
— if x is a node of T with a single child y, then $|\beta(x) \setminus \beta(y)| = 1$ (x is an *introduce node*) or $|\beta(y) \setminus \beta(x)| = 1$ (x is a *forget node*), and
— if x is a node with two children x_1 and x_2, then $\beta(x) = \beta(x_1) = \beta(x_2)$ (x is a *join node*).

To find a nice tree decomposition from a given a tree decomposition, we use the following well-known result proved, for instance, in [7].

PROPOSITION 4.7 ([7]). *Given a graph G with n vertices and a tree decomposition (T, β) of G of width w, there is an algorithm that computes a nice tree decomposition of G of width w with at most $O(w \cdot n)$ nodes in time $O(w^2 \cdot (n + |V(\mathsf{T})|))$.*

4.3   Boundaried graphs and representatives

Boundaried graphs are extensively used in Section 7 and Section 10. We present here some useful definitions and results.

Boundaried graphs. Let $t \in \mathbb{N}$. A *t-boundaried graph* is a triple $\mathbf{G} = (G, B, \rho)$ where G is a graph, $B \subseteq V(G)$, $|B| = t$, and $\rho : B \rightarrow [t]$ is a bijection. We say that two t-boundaried graphs $\mathbf{G}_1 = (G_1, B_1, \rho_1)$ and $\mathbf{G}_2 = (G_2, B_2, \rho_2)$ are *isomorphic* if there is an isomorphism from G_1 to G_2 that extends the bijection $\rho_2^{-1} \circ \rho_1$. The triple (G, B, ρ) is a *boundaried graph* if it is a t-boundaried graph for some $t \in \mathbb{N}$. We denote by $\mathcal{B}^t$ the set of all (pairwise non-isomorphic) t-boundaried graphs. We also set $\mathcal{B} = \bigcup_{t \in \mathbb{N}} \mathcal{B}^t$.

Minors of boundaried graphs. We say that a t-boundaried graph $G_1 = (G_1, B_1, \rho_1)$ is a minor of a t-boundaried graph $G_2 = (G_2, B_2, \rho_2)$, denoted by $G_1 \preceq G_2$, if there is a sequence of removals of non-boundary vertices, edge removals, and edge contractions in G_2, not allowing contractions of edges with both endpoints in B_2, that transforms G_2 to a boundaried graph that is isomorphic to G_1 (during edge contractions, boundary vertices prevail). Note that this extends the usual definition of minors in graphs without boundary.

Equivalent boundaried graphs and representatives. We say that two boundaried graphs $\mathbf{G}_1 = (G_1, B_1, \rho_1)$ and $\mathbf{G}_2 = (G_2, B_2, \rho_2)$ are *compatible* if $\rho_2^{-1} \circ \rho_1$ is an isomorphism from $G_1[B_1]$ to $G_2[B_2]$. Given two compatible boundaried graphs $\mathbf{G}_1 = (G_1, B_1, \rho_1)$ and $\mathbf{G}_2 = (G_2, B_2, \rho_2)$, we define $\mathbf{G}_1 \oplus \mathbf{G}_2$ as the graph obtained if we take the disjoint union of G_1 and G_2 and, for every $i \in [|B_1|]$, we identify vertices $\rho_1^{-1}(i)$ and $\rho_2^{-1}(i)$. We also define $\mathbf{G}_1 \widetilde{\oplus} \mathbf{G}_2$ as the *boundaried* graph $(\mathbf{G}_1 \oplus \mathbf{G}_2, B_1, \rho_1)$. Given $h \in \mathbb{N}$, we say that two boundaried graphs $\mathbf{G}_1$ and $\mathbf{G}_2$ are *h-equivalent*, denoted by $\mathbf{G}_1 \equiv_h \mathbf{G}_2$, if they are compatible and, for every graph H with detail at most h and every boundaried graph $\mathbf{F}$ compatible with $\mathbf{G}_1$ (hence, with $\mathbf{G}_2$ as well), it holds that

$$H \leq \mathbf{F} \oplus \mathbf{G}_1 \iff H \leq \mathbf{F} \oplus \mathbf{G}_2.$$

Note that $\equiv_h$ is an equivalence relation on $\mathcal{B}$. A minimum-sized (in terms of number of vertices) element of an equivalent class of $\equiv_h$ is called *representative* of $\equiv_h$. For $t \in \mathbb{N}$, a *set of t-representatives* for $\equiv_h$, denoted by $\mathcal{R}_h^t$, is a collection containing a minimum-sized representative for each equivalence class of $\equiv_h$ restricted to $\mathcal{B}^t$.

The following results were proved by Baste, Sau, and Thilikos [9] and give a bound on the size of a representative and on the number of representatives for this equivalence relation, respectively.

PROPOSITION 4.8 ([9]). *For every $t \in \mathbb{N}$, $q, h \in \mathbb{N}_{\geq 1}$, and $G = (G, B, \rho) \in \mathcal{R}_h^t$, if G is does not contain K_q as a minor, then $|V(G)| = O_{q,h}(t)$.*

PROPOSITION 4.9 ([9]). *For every $t \in \mathbb{N}_{\geq 1}$, $|\mathcal{R}_h^t| = 2^{O_h(t \log t)}$.*

Moreover, given a boundaried graph of bounded size, the following lemma gives an algorithm to find its representative. While this is might be considered folklore, we include here its proof for the sake of completeness.

LEMMA 4.10. *Given a finite collection of graphs $\mathcal{F}$, $h, t, k \in \mathbb{N}$, the set $\mathcal{R}$ of representatives in $\mathcal{R}_h^t$ whose underlying graphs are $\mathcal{F}$-minor-free, and a t-boundaried graph $\mathbf{G}$ with k vertices whose underlying graph is $\mathcal{F}$-minor-free, there is an algorithm that outputs the representative of $\mathbf{G}$ in $\mathcal{R}$ in time $2^{O_{\ell_{\mathcal{F}},h}(t \log t + \log(k+t))}$.*

PROOF. Let $\mathcal{H}$ be the set of graphs with detail at most h. For a t-boundaried graph $\mathbf{G}$ of size k whose underlying graph is $\mathcal{F}$-minor-free, we define the matrix $M_{\mathbf{G}}$, whose rows are the representatives in $\mathcal{R}$ and whose columns are the graphs of $\mathcal{H}$, such that for $\mathbf{R} \in \mathcal{R}$ and $H \in \mathcal{H}$, we have $M_{\mathbf{G}}(\mathbf{R}, H) = 1$ if $\mathbf{G}$ and $\mathbf{R}$ are compatible and $H \leq \mathbf{G} \widetilde{\oplus} \mathbf{R}$, and $M_{\mathbf{G}}(\mathbf{R}, H) = 0$ otherwise. Observe that $\mathbf{R} \in \mathcal{R}$ is the representative of $\mathbf{G}$ if and only if $M_{\mathbf{R}} = M_{\mathbf{G}}$. According to Proposition 4.9, $M_{\mathbf{G}}$ has size $2^{O_h(t \log t)} \cdot O_h(1) = 2^{O_h(t \log t)}$.

For all $\mathbf{R} \in \mathcal{R}$, we compute $M_{\mathbf{R}}$. Every representative in $\mathcal{R}$ has size at most $O_{s_{\mathcal{F}},h}(t)$ by Proposition 4.8, so when two representatives $\mathbf{R}$ and $\mathbf{R}'$ are compatible, $\mathbf{R} \oplus \mathbf{R}'$ has size $O_{\ell_{\mathcal{F}},h}(t)$

as well. From [43], we know that checking if a graph $H \in \mathcal{H}$ is a minor of $\mathbf{R} \oplus \mathbf{R}'$ can be done in time $O_{\ell_{\mathcal{F}},h}(t^2)$. Therefore, we can compute $M_{\mathbf{R}}$ in time $2^{O_{h,s_{\mathcal{F}}}(t \log t)}$.

Let $\mathbf{G}$ be a t-boundaried graph of size k whose underlying graph is $\mathcal{F}$-minor-free. For $\mathbf{R} \in \mathcal{R}$ compatible with $\mathbf{G}$, $\mathbf{G} \oplus \mathbf{R}$ has size $O_{\ell_{\mathcal{F}},h}(k + t)$, so checking if $H \in \mathcal{H}$ is a minor of $\mathbf{G} \oplus \mathbf{R}$ can be done in time $O_{\ell_{\mathcal{F}},h}((k+t)^2)$. Thus we can compute $M_{\mathbf{G}}$ in time $2^{O_h(t \log t)} \cdot O_{\ell_{\mathcal{F}},h}((k+t)^2) = 2^{O_{\ell_{\mathcal{F}},h}(t \log t + \log(k+t))}$.

Finally, we just need to find $\mathbf{R} \in \mathcal{R}$ such that $M_{\mathbf{R}} = M_{\mathbf{G}}$, which can be done in time $2^{O_h(t \log t)}$. Thus, we can find the representative of $\mathbf{G}$ in time $2^{O_{\ell_{\mathcal{F}},h}(t \log t + \log(k+t))}$.   ∎

5. Even more definitions: Flat walls

In this section we deal with flat walls using the framework of [68]. More precisely, in Subsection 5.1, we introduce walls and several notions concerning them. In Subsection 5.2 we provide the definitions of a rendition and a painting. Using the above notions, in Subsection 5.3 we define flat walls and provide some results about them, including the Flat Wall Theorem (namely, the version proved in [44]) and its algorithmic version restated in the "more accurate" framework of [68]. In Subsection 5.4 and Subsection 5.5, we define canonical partitions and the notion of bidimensionality and give some combinatorial results that allow us to use branching in the algorithm in Section 6. Finally in Subsection 5.6 we present homogeneous walls and give some results to find an irrelevant vertex. We note that most of the definitions of this section can also be found in [68, 69, 70] with more details and illustrations.

5.1 Walls and subwalls

We start with some basic definitions about walls.

Walls. Let $k, r \in \mathbb{N}$. The $(k \times r)$-*grid* is the graph whose vertex set is $[k] \times [r]$ and two vertices (i, j) and (i', j') are adjacent if and only if $|i - i'| + |j - j'| = 1$. An *elementary r-wall*, for some odd integer $r \geq 3$, is the graph obtained from a $(2r \times r)$-grid with vertices $(x, y) \in [2r] \times [r]$, after the removal of the "vertical" edges $\{(x, y), (x, y + 1)\}$ for odd $x + y$, and then the removal of all vertices of degree one. Notice that, as $r \geq 3$, an elementary r-wall is a planar graph that has a unique (up to topological isomorphism) embedding in the plane $\mathbb{R}^2$ such that all its finite faces are incident to exactly six edges. The *perimeter* of an elementary r-wall is the cycle bounding its infinite face, while the cycles bounding its finite faces are called *bricks*. Also, the vertices in the perimeter of an elementary r-wall that have degree two are called *pegs*, while the vertices $(1, 1), (2, r), (2r - 1, 1), (2r, r)$ are called *corners* (notice that the corners are also pegs).

An *r-wall* is any graph W obtained from an elementary r-wall $\bar{W}$ after subdividing edges (see Figure 2). A graph W is a *wall* if it is an r-wall for some odd $r \geq 3$ and we refer to r as the
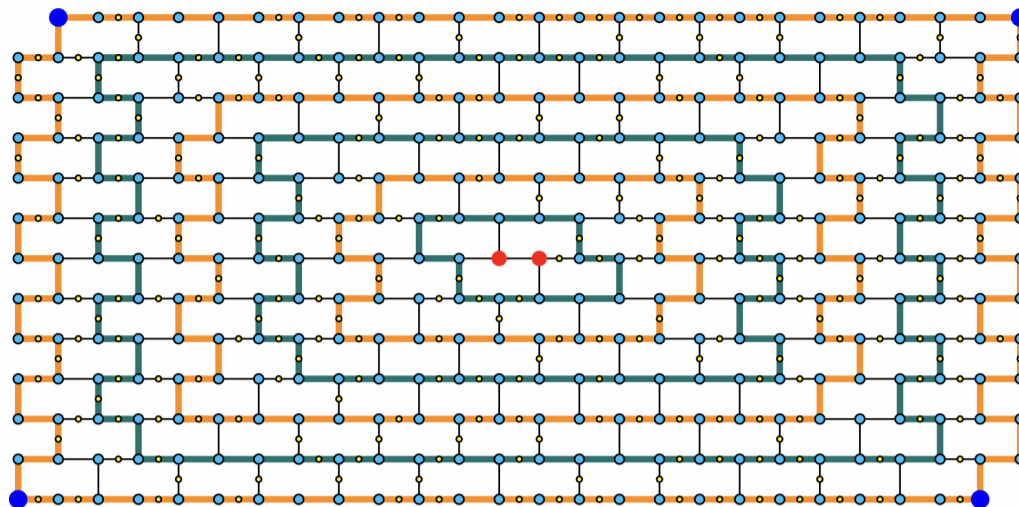
Figure 2. A 13-wall and its six layers, depicted in alternating orange and green. The central vertices of the wall are depicted in red and the corners are depicted in blue.

height of W. Given a graph G, a *wall of G* is a subgraph of G that is a wall. We insist that, for every r-wall, the number r is always odd.

We call the vertices of degree three of a wall W *3-branch vertices*. A cycle of W is a *brick* (resp. the *perimeter*) of W if its 3-branch vertices are the vertices of a brick (resp. the perimeter) of $\bar{W}$. We denote by $C(W)$ the set of all cycles of W. We use $D(W)$ in order to denote the perimeter of the wall W. A brick of W is *internal* if it is disjoint from $D(W)$.

Subwalls. Given an elementary r-wall $\bar{W}$, some odd $i \in \{1, 3, \ldots, 2r-1\}$, and $i' = (i+1)/2$, the *i'-th vertical path* of $\bar{W}$ is the one whose vertices, in order of appearance, are $(i, 1), (i, 2), (i+1, 2), (i+1, 3), (i, 3), (i, 4), (i+1, 4), (i+1, 5), (i, 5), \ldots, (i, r-2), (i, r-1), (i+1, r-1), (i+1, r)$. Also, given some $j \in [2, r-1]$ the *j-th horizontal path* of $\bar{W}$ is the one whose vertices, in order of appearance, are $(1, j), (2, j), \ldots, (2r, j)$.

A *vertical* (resp. *horizontal*) path of an r-wall W is one that is a subdivision of a vertical (resp. horizontal) path of $\bar{W}$. Notice that the perimeter of an r-wall W is uniquely defined regardless of the choice of the elementary r-wall $\bar{W}$. A *subwall* of W is any subgraph W' of W that is an r'-wall, with $r' \leq r$, and such the vertical (resp. horizontal) paths of W' are subpaths of the vertical (resp. horizontal) paths of W.

Layers. The *layers* of an r-wall W are recursively defined as follows. The first layer of W is its perimeter. For $i = 2, \ldots, (r-1)/2$, the i-th layer of W is the $(i-1)$-th layer of the subwall W' obtained from W after removing from W its perimeter and removing recursively all occurring vertices of degree one. We refer to the $(r-1)/2$-th layer as the *inner layer* of W. The *central vertices* of an r-wall are its two 3-branch vertices that do not belong to any of its layers and that

are connected by a path of W that does not intersect any layer. See Figure 2 for an illustration of the notions defined above.

Central walls. Given an r-wall W and an odd $q \in \mathbb{N}_{\geq 3}$ where $q \leq r$, we define the *central q-subwall* of W, denoted by $W^{(q)}$, to be the q-wall obtained from W after removing its first $(r - q)/2$ layers and all occurring vertices of degree one.

Tilts. The *interior* of a wall W is the graph obtained from W if we remove from it all edges of $D(W)$ and all vertices of $D(W)$ that have degree two in W. Given two walls W and $\tilde{W}$ of a graph G, we say that $\tilde{W}$ is a *tilt* of W if $\tilde{W}$ and W have identical interiors.

Minor models grasped by walls. Let G be a graph and W be an r-wall in G. Let $P_1, ..., P_r$ be the horizontal paths and $Q_1, ..., Q_r$ be the vertical paths of W. Let $t \geq 1$ be an integer and $v_1, ..., v_t$ be the vertices in K_t. A model of a K_t-minor in G is *grasped* by W if, for every model X_i of v_i, there exist distinct indices $i_1, ..., i_t \in [r]$ and distinct indices $j_1, ..., j_t \in [r]$ such that $V(P_{i_l}) \cap V(Q_{j_l}) \subseteq X_l$ for all $l \in [t]$.

We present the following result of Kawarabayashi and Kobayashi [42], which provides a linear relation between the treewidth and the height of a largest wall in a minor-free graph.

PROPOSITION 5.1 ([42]). *There is a function $f_{12} : \mathbb{N} \to \mathbb{N}$ such that, for every $t, r \in \mathbb{N}$ and every graph G that does not contain K_t as a minor, if $\mathrm{tw}(G) \geq f_{12}(t) \cdot r$, then G contains an r-wall as a subgraph. In particular, one may choose $f_{12}(t) = 2^{O(t^2 \cdot \log t)}$.*

5.2   Paintings and renditions

In this subsection we present the notions of renditions and paintings, originating in the work of Robertson and Seymour [63]. The definitions presented here were introduced in [44] (see also [9, 68]).

Paintings. A *closed* (resp. *open*) *disk* is a set homeomorphic to the set $\{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$ (resp. $\{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 < 1\}$). Let Δ be a closed disk. Given a subset X of Δ, we denote its closure by $\bar{X}$ and its boundary by $\mathrm{bd}(X)$. A Δ-*painting* is a pair $\Gamma = (U, N)$ where
— N is a finite set of points of Δ,
— $N \subseteq U \subseteq \Delta$, and
— $U \setminus N$ has finitely many arcwise-connected components, called *cells*, where for every cell c,
    — the closure $\bar{c}$ of c is a closed disk and
    — $|\tilde{c}| \leq 3$, where $\tilde{c} := \mathrm{bd}(c) \cap N$.

We use the notation $U(\Gamma) := U$, $N(\Gamma) := N$ and denote the set of cells of Γ by $C(\Gamma)$. For convenience, we may assume that each cell of Γ is an open disk of Δ.

Notice that, given a Δ-painting Γ, the pair $(N(\Gamma), \{\tilde{c} \mid c \in C(\Gamma)\})$ is a hypergraph whose hyperedges have cardinality at most three and Γ can be seen as a plane embedding of this hypergraph in Δ.

Renditions. Let G be a graph and let Ω be a cyclic permutation of a subset of $V(G)$ that we denote by $V(\Omega)$. By an Ω-*rendition* of G we mean a triple (Γ, σ, π), where

(a) Γ is a Δ-painting for some closed disk Δ,

(b) $\pi : N(\Gamma) \to V(G)$ is an injection, and

(c) σ assigns to each cell $c \in C(\Gamma)$ a subgraph $\sigma(c)$ of G, such that

   (1) $G = \bigcup_{c \in C(\Gamma)} \sigma(c)$,

   (2) for distinct $c, c' \in C(\Gamma)$, $\sigma(c)$ and $\sigma(c')$ are edge-disjoint,

   (3) for every cell $c \in C(\Gamma)$, $\pi(\tilde{c}) \subseteq V(\sigma(c))$,

   (4) for every cell $c \in C(\Gamma)$, $V(\sigma(c)) \cap \bigcup_{c' \in C(\Gamma) \setminus \{c\}} V(\sigma(c')) \subseteq \pi(\tilde{c})$, and

   (5) $\pi(N(\Gamma) \cap \mathsf{bd}(\Delta)) = V(\Omega)$, such that the points in $N(\Gamma) \cap \mathsf{bd}(\Delta)$ appear in $\mathsf{bd}(\Delta)$ in the same ordering as their images, via π, in Ω.

5.3   Flatness pairs

In this subsection we define the notion of a flat wall. The definitions given here are originating from [68]. We refer the reader to that paper for a more detailed exposition of these definitions and the reasons for which they were introduced. We use the more accurate framework of [68] concerning flat walls, instead of that of [44], in order to be able to use tools that are developed in [68, 69, 70].

Flat walls. Let G be a graph and let W be an r-wall of G, for some odd integer $r \geq 3$. We say that a pair $(P, C) \subseteq V(D(W)) \times V(D(W))$ is a *choice of pegs and corners for W* if W is a subdivision of an elementary r-wall $\overline{W}$ where P and C are the pegs and the corners of $\overline{W}$, respectively (clearly, $C \subseteq P$). To get more intuition, notice that a wall W can occur in several ways from the elementary wall $\overline{W}$, depending on the way the vertices in the perimeter of $\overline{W}$ are subdivided. Each of them gives a different selection (P, C) of pegs and corners of W.

We say that W is a *flat r-wall* of G if there is a separation (X, Y) of G and a choice (P, C) of pegs and corners for W such that:

— $V(W) \subseteq Y$,

— $P \subseteq X \cap Y \subseteq V(D(W))$, and

— if Ω is the cyclic ordering of the vertices $X \cap Y$ as they appear in $D(W)$, then there exists an Ω-rendition (Γ, σ, π) of $G[Y]$.

We say that W is a *flat wall* of G if it is a flat r-wall for some odd integer $r \geq 3$.

Flatness pairs. Given the above, we say that the choice of the 7-tuple $\mathfrak{R} = (X, Y, P, C, \Gamma, \sigma, \pi)$ *certifies that W is a flat wall of G*. We call the pair $(W, \mathfrak{R})$ a *flatness pair* of G and define the *height* of the pair $(W, \mathfrak{R})$ to be the height of W. We use the term *cell of* $\mathfrak{R}$ in order to refer to the cells of Γ.

We call the graph $G[Y]$ the $\mathfrak{R}$-*compass* of W in G, denoted by $\mathsf{Compass}_{\mathfrak{R}}(W)$. We can assume that $\mathsf{Compass}_{\mathfrak{R}}(W)$ is connected, updating $\mathfrak{R}$ by removing from Y the vertices of all the connected components of $\mathsf{Compass}_{\mathfrak{R}}(W)$ except for the one that contains W and including them in X (Γ, σ, π can also be easily modified according to the removal of the aforementioned vertices from Y). We define the *flaps* of the wall W in $\mathfrak{R}$ as $\mathsf{Flaps}_{\mathfrak{R}}(W) := \{\sigma(c) \mid c \in C(\Gamma)\}$. Given a flap $F \in \mathsf{Flaps}_{\mathfrak{R}}(W)$, we define its *base* as $\partial F := V(F) \cap \pi(N(\Gamma))$. A cell c of $\mathfrak{R}$ is *untidy* if $\pi(\tilde{c})$ contains a vertex x of W such that two of the edges of W that are incident to x are edges of $\sigma(c)$. Notice that if c is untidy then $|\tilde{c}| = 3$. A cell c of $\mathfrak{R}$ is *tidy* if it is not untidy.

Cell classification. Given a cycle C of $\mathsf{Compass}_{\mathfrak{R}}(W)$, we say that C is $\mathfrak{R}$-*normal* if it is *not* a subgraph of a flap $F \in \mathsf{Flaps}_{\mathfrak{R}}(W)$. Given an $\mathfrak{R}$-normal cycle C of $\mathsf{Compass}_{\mathfrak{R}}(W)$, we call a cell c of $\mathfrak{R}$ C-*perimetric* if $\sigma(c)$ contains some edge of C. Notice that if c is C-perimetric, then $\pi(\tilde{c})$ contains two points $p, q \in N(\Gamma)$ such that $\pi(p)$ and $\pi(q)$ are vertices of C where one, say P_c^{in}, of the two $(\pi(p), \pi(q))$-subpaths of C is a subgraph of $\sigma(c)$ and the other, denoted by P_c^{out}, $(\pi(p), \pi(q))$-subpath contains at most one internal vertex of $\sigma(c)$, which should be the (unique) vertex z in $\partial\sigma(c) \setminus \{\pi(p), \pi(q)\}$. We pick a (p, q)-arc A_c in $\hat{c} := c \cup \tilde{c}$ such that $\pi^{-1}(z) \in A_c$ if and only if P_c^{in} contains the vertex z as an internal vertex.

We consider the circle $K_C = \bigcup\{A_c \mid c \text{ is a } C\text{-perimetric cell of } \mathfrak{R}\}$ and we denote by Δ_C the closed disk bounded by K_C that is contained in Δ. A cell c of $\mathfrak{R}$ is called C-*internal* if $c \subseteq \Delta_C$ and is called C-*external* if $\Delta_C \cap c = \emptyset$. Notice that the cells of $\mathfrak{R}$ are partitioned into C-internal, C-perimetric, and C-external cells.

Let c be a tidy C-perimetric cell of $\mathfrak{R}$ where $|\tilde{c}| = 3$. Notice that $c \setminus A_c$ has two arcwise-connected components and one of them is an open disk D_c that is a subset of Δ_C. If the closure $\overline{D}_c$ of D_c contains only two points of $\tilde{c}$ then we call the cell c C-*marginal*.

Influence. For every $\mathfrak{R}$-normal cycle C of $\mathsf{Compass}_{\mathfrak{R}}(W)$ we define the set $\mathsf{influence}_{\mathfrak{R}}(C) = \{\sigma(c) \mid c \text{ is a cell of } \mathfrak{R} \text{ that is not } C\text{-external}\}$.

A wall W' of $\mathsf{Compass}_{\mathfrak{R}}(W)$ is $\mathfrak{R}$-*normal* if $D(W')$ is $\mathfrak{R}$-normal. Notice that every wall of W (and hence every subwall of W) is an $\mathfrak{R}$-normal wall of $\mathsf{Compass}_{\mathfrak{R}}(W)$. We denote by $\mathcal{S}_{\mathfrak{R}}(W)$ the set of all $\mathfrak{R}$-normal walls of $\mathsf{Compass}_{\mathfrak{R}}(W)$. Given a wall $W' \in \mathcal{S}_{\mathfrak{R}}(W)$ and a cell c of $\mathfrak{R}$, we say that c is W'-*perimetric/internal/external/marginal* if c is $D(W')$-perimetric/internal/external/marginal, respectively. We also use $K_{W'}, \Delta_{W'}, \mathsf{influence}_{\mathfrak{R}}(W')$ as shortcuts for $K_{D(W')}, \Delta_{D(W')}$, and $\mathsf{influence}_{\mathfrak{R}}(D(W'))$, respectively.

Regular flatness pairs. We call a flatness pair $(W, \mathfrak{R})$ of a graph G *regular* if none of its cells is W-external, W-marginal, or untidy.

Tilts of flatness pairs. Let $(W, \mathfrak{R})$ and $(\tilde{W}', \tilde{\mathfrak{R}}')$ be two flatness pairs of a graph G and let $W' \in \mathcal{S}_{\mathfrak{R}}(W)$. We assume that $\mathfrak{R} = (X, Y, P, C, \Gamma, \sigma, \pi)$ and $\tilde{\mathfrak{R}}' = (X', Y', P', C', \Gamma', \sigma', \pi')$. We say that $(\tilde{W}', \tilde{\mathfrak{R}}')$ is a W'-*tilt* of $(W, \mathfrak{R})$ if

— $\tilde{\mathfrak{R}}'$ does not have $\tilde{W}'$-external cells,

— $\tilde{W}'$ is a tilt of W',

— the set of $\tilde{W}'$-internal cells of $\tilde{\mathfrak{R}}'$ is the same as the set of W'-internal cells of $\mathfrak{R}$ and their images via σ' and σ are also the same,

— Compass$_{\tilde{\mathfrak{R}}'}(\tilde{W}')$ is a subgraph of $\bigcup$influence$_{\mathfrak{R}}(W')$, and

— if c is a cell in $C(\Gamma') \setminus C(\Gamma)$, then $|\tilde{c}| \leq 2$.

The next observation follows from the third item above and the fact that the cells corresponding to flaps containing a central vertex of W' are all internal (recall that the height of a wall is always at least three).

OBSERVATION 5.2. *Let $(W, \mathfrak{R})$ be a flatness pair of a graph G and $W' \in \mathcal{S}_{\mathfrak{R}}(W)$. For every W'-tilt $(\tilde{W}', \tilde{\mathfrak{R}}')$ of $(W, \mathfrak{R})$, the central vertices of W' belong to the vertex set of* Compass$_{\tilde{\mathfrak{R}}'}(\tilde{W}')$.

Also, given a regular flatness pair $(W, \mathfrak{R})$ of a graph G and a $W' \in \mathcal{S}_{\mathfrak{R}}(W)$, for every W'-tilt $(\tilde{W}', \tilde{\mathfrak{R}}')$ of $(W, \mathfrak{R})$, by definition none of its cells is $\tilde{W}'$-external, $\tilde{W}'$-marginal, or untidy – thus, $(\tilde{W}', \tilde{\mathfrak{R}}')$ is regular. Therefore, regularity of a flatness pair is a property that its tilts "inherit".

OBSERVATION 5.3. *If $(W, \mathfrak{R})$ is a regular flatness pair, then for every $W' \in \mathcal{S}_{\mathfrak{R}}(W)$, every W'-tilt of $(W, \mathfrak{R})$ is also regular.*

Furthermore, we need the following propositions, that are the main results of [68].

PROPOSITION 5.4 ([68]). *There exists an algorithm that, given a graph G, a flatness pair $(W, \mathfrak{R})$ of G, and a wall $W' \in \mathcal{S}_{\mathfrak{R}}(W)$, outputs a W'-tilt of $(W, \mathfrak{R})$ in time $O(n + m)$.*

PROPOSITION 5.5 ([68]). *Let G be a graph and $(W, \mathfrak{R})$ be a flatness pair of G. There is a regular flatness pair $(W^*, \mathfrak{R}^*)$ of G, with the same height as $(W, \mathfrak{R})$, such that* Compass$_{\mathfrak{R}^*}(W^*) \subseteq$ Compass$_{\mathfrak{R}}(W)$.

5.4   Canonical partitions

In this subsection, we define the notion of canonical partition of a graph with respect to some wall. This refers to a partition of the vertex set of a graph in bags that follow the structure of a wall subgraph of the given graph. For this reason, we start by defining the canonical partition of a wall, as a "canonical" way to partition the vertices of the wall in connected subsets that preserve the grid-like structure of the wall.

Canonical partition of a wall. Let $r \geq 3$ be an odd integer. Let W be an r-wall and let $P_1, \ldots, P_r$ (resp. $L_1, \ldots, L_r$) be its vertical (resp. horizontal) paths. For every even (resp. odd) $i \in [2, r-1]$ and every $j \in [2, r-1]$, we define $A^{(i,j)}$ to be the subpath of P_i that starts from a vertex of $P_i \cap L_j$ and finishes at a neighbor of a vertex in L_{j+1} (resp. L_{j-1}), such that $P_i \cap L_j \subseteq A^{(i,j)}$ and $A^{(i,j)}$ does not intersect L_{j+1} (resp. L_{j-1}). Similarly, for every $i, j \in [2, r-1]$, we define $B^{(i,j)}$ to be the subpath of L_j that starts from a vertex of $P_i \cap L_j$ and finishes at a neighbor of a vertex in P_{i-1}, such that $P_i \cap L_j \subseteq B^{(i,j)}$ and $B^{(i,j)}$ does not intersect P_{i-1}.
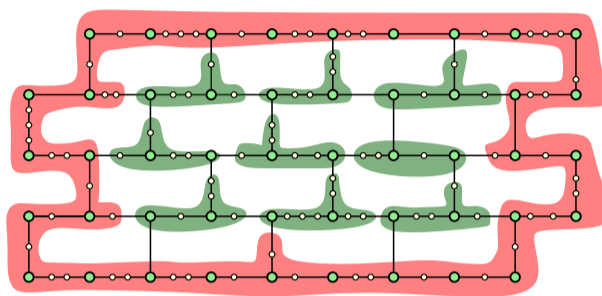

Figure 3. A 5-wall and its canonical partition Q. The red bag is the external bag Q_{ext}.

For every $i, j \in [2, r-1]$, we denote by $Q^{(i,j)}$ the graph $A^{(i,j)} \cup B^{(i,j)}$ and by Q_{ext} the graph $W \setminus \bigcup_{i,j \in [2,r-1]} V(Q_{i,j})$. Now consider the collection $Q = \{Q_{\text{ext}}\} \cup \{Q_{i,j} \mid i, j \in [2, r-1]\}$ and observe that the graphs in Q are connected subgraphs of W and their vertex sets form a partition of $V(W)$. We call Q the *canonical partition* of W. Also, we call every $Q_{i,j}, i, j \in [2, r-1]$ an *internal bag* of Q, while we refer to Q_{ext} as the *external bag* of Q. See Figure 3 for an illustration of the notions defined above. For every $i \in [(r-1)/2]$, we say that a set $Q \in Q$ is an *i-internal bag of Q* if $V(Q)$ does not contain any vertex of the first i layers of W. Notice that the 1-internal bags of Q are the internal bags of Q.

Canonical partition of a graph with respect to a wall. Let W be a wall of a graph G. Consider the canonical partition Q of W. The *enhancement* of the canonical partition Q on G is the following operation. We set $\tilde{Q} := Q$ and, as long as there is a vertex $x \in G \setminus V(\bigcup \tilde{Q})$ that is adjacent to a vertex of a graph $Q \in \tilde{Q}$, we update $\tilde{Q} := \tilde{Q} \setminus \{Q\} \cup \{\tilde{Q}\}$, where $\tilde{Q} = G[\{x\} \cup V(Q)]$. We call the $\tilde{Q} \in \tilde{Q}$ that contains Q_{ext} as a subgraph the *external bag* of $\tilde{Q}$, and we denote it by $\tilde{Q}_{\text{ext}}$, while we call *internal bags* of $\tilde{Q}$ all graphs in $\tilde{Q} \setminus \{\tilde{Q}_{\text{ext}}\}$. Moreover, we enhance $\tilde{Q}$ by adding all vertices of $G \setminus \bigcup_{\tilde{Q} \in \tilde{Q}} V(\tilde{Q})$ in its external bag, i.e., by updating $\tilde{Q}_{\text{ext}} := G[V(\tilde{Q}_{\text{ext}}) \cup V(G) \setminus \bigcup_{\tilde{Q} \in \tilde{Q}} V(\tilde{Q})]$.

We call such a partition $\tilde{Q}$ a *W-canonical partition of G*. Notice that a W-canonical partition of G is not unique, since the sets in Q can be "expanded" arbitrarily when introducing vertex x.

Let W be an r-wall of a graph G, for some odd integer $r \geq 3$ and let $\tilde{Q}$ be a W-canonical partition of G. For every $i \in [(r-1)/2]$, we say that a set $Q \in \tilde{Q}$ is an *i-internal bag of $\tilde{Q}$* if it contains an i-internal bag of Q as a subgraph.

We stress that given a graph G and an r-wall W of G for some odd integer $r \geq 3$, a W-canonical partition $\tilde{Q}$ of G can have internal bags that are adjacent to vertices of arbitrarily many other bags. However, if W is a flat wall of G certified by some 7-tuple $\mathfrak{R}$, the "flat structure" of the $\mathfrak{R}$-compass of W implies that every bag can be adjacent to only bags that contain vertices of the same brick of W.

The next result is also proved in [69] and intuitively states that, given a flatness pair $(W, \mathfrak{R})$ of "big enough" height and a W-canonical partition $\tilde{Q}$ of G, we can find a "packing" of subwalls of W that are inside some central part of W and such that the vertex set of every internal bag of $\tilde{Q}$ intersects the vertices of the flaps in the influence of at most one of these walls. We will use this result in the case where the set A' of Proposition 3.7 is "small", i.e., there are only "few" vertices in A that have "big enough" degree with respect to the central part of the canonical partition, and therefore Proposition 3.7 cannot justify branching. Following the latter condition and Proposition 5.6, we will be able to find a flatness pair with "few" apices so as to build irrelevant vertex arguments inside its compass. In [69], $(W, \mathfrak{R})$-*canonical partition* is used to denote a W-canonical partition of G, where $(W, \mathfrak{R})$ is a flatness pair of G. However, in this subsection we provide a more general definition that does not take into account the flatness of W.

PROPOSITION 5.6 ([69]). *There exists a function $f_{13} : \mathbb{N}^3 \to \mathbb{N}$ such that if $p, l \in \mathbb{N}_{\geq 1}$, $r \in \mathbb{N}_{\geq 3}$ is an odd integer, G is a graph, $(W, \mathfrak{R})$ is a flatness pair of G of height at least $f_{13}(l, r, p)$, and $\tilde{Q}$ is a W-canonical partition of G, then there is a collection $\mathcal{W} = \{W^1, \ldots, W^l\}$ of r-subwalls of W such that*

— *for every $i \in [l]$, $\bigcup \mathsf{influence}_{\mathfrak{R}}(W^i)$ is a subgraph of $\bigcup\{Q \mid Q$ is a p-internal bag of $\tilde{Q}\}$ and*
— *for every $i, j \in [l]$, with $i \neq j$, there is no internal bag of $\tilde{Q}$ that contains vertices of both $V(\bigcup \mathsf{influence}_{\mathfrak{R}}(W^i))$ and $V(\bigcup \mathsf{influence}_{\mathfrak{R}}(W^j))$.*

Moreover, $f_{13}(l, r, p) = O(\sqrt{l} \cdot r + p)$ and $\mathcal{W}$ can be constructed in time $O(n + m)$.

5.5   Bidimensionality of sets

In this subsection, we present the notion of bidimensionality of a set with respect to a wall of a graph. This notion intuitively expresses the "spread" of a set X in a W-canonical partition of G. The crucial idea is that a set X of small bidimensionality cannot "destroy" a (flat) wall too much.

Bidimensionality. Let W be a wall of a graph G, $\tilde{Q}$ be a W-canonical partition of G, and $X \subseteq V(G)$. The *bidimensionality* of X in G with respect to $\tilde{Q}$, denoted by $\mathsf{bid}_{\tilde{Q}}(X)$, is the number of internal bags of $\tilde{Q}$ intersected by X. The *bidimensionality* of X in G with respect to W, denoted by $\mathsf{bid}_{G,W}(X)$, is the maximum bidimensionality of X with respect to a W-canonical partition of G.

Observation 5.7 and Lemma 5.9 provide sets $X \subseteq V(G)$ that can be used to apply Proposition 3.7 to $\mathcal{F}$-M-DELETION and $\mathcal{F}$-M-ELIMINATION DISTANCE, respectively.

Every set $S \subseteq V(G)$ of size at most k clearly has bidimensionality at most k.

OBSERVATION 5.7. *Let $\mathcal{F}$ be a finite collection of graphs, let G be a graph, let $k \in \mathbb{N}$, and let W be a wall of G. Then for every k-apex set S of G for $\mathrm{exc}(\mathcal{F})$ it holds that $\mathrm{bid}_{G,W}(S) \leq k$.*

Moreover, given a k-elimination set S, we can find a set $X \supseteq S$ of bidimensionality at most $k(k+1)/2$ such that $G \setminus X \in \mathrm{exc}(\mathcal{F})$. To prove this, we first prove the following result, which intuitively states that a k-elimination set can intersect at most k horizontal and vertical paths of a wall.

LEMMA 5.8. *Let $\mathcal{F}$ be a finite collection of graphs. Let G be a graph, let $k \in \mathbb{N}$, let r, h be odd integers with $r \geq h + k$, let W be an r-wall of G, and let S be a k-elimination set of G for $\mathrm{exc}(\mathcal{F})$. Then there is an h-subwall W' of W with $V(W') \cap S = \emptyset$.*

PROOF. Since S is a k-elimination set of G for $\mathrm{exc}(\mathcal{F})$, there is an $\mathcal{F}$-elimination forest (F, χ, R) of G of height k such that $\chi(\mathrm{Int}(F, R)) = S$.

We set $W_0 := W$. For $i \in [k]$, we proceed to construct an r_i-subwall W_i of the r_{i-1}-wall W_{i-1} with $r_i \geq r - i$ such that for every $i \in [k]$ there is a node z_i of F such that (F_{z_i}, z_i) has height at most $k - i$ and $V(W_i) \subseteq \chi(V(F_{z_i}))$. This will imply the existence of a wall of size at least $r - k$ whose vertex set will be a subset of $\chi(V(F_z))$, where $z \in \mathrm{Leaf}(F, R)$.

Let $S_i := V(W_{i-1}) \cap S$. If $S_i = \emptyset$, we set $W_j := W_{i-1}$ for $j \in [i, k]$. Otherwise, let u_i be the least common ancestor of $\chi^{-1}(S_i)$ in (F, R). According to Lemma 4.1, since W_{i-1} is connected, u_i exists and belongs to $\chi^{-1}(S_i)$.

We obtain an $(r_{i-1} - 1)$-subwall W_i of W_{i-1} that does not contain $\chi(u_i)$ by taking the wall containing all horizontal and vertical paths of W_{i-1} aside from the ones intersecting u_i and we set $r_i := r_{i-1} - 1$ (to simplify the argument, here we call the resulting graph a wall even when the height is even). Note that since W_i is a subgraph of $G[\chi(V(F_{u_i}))]$ that is connected, there is a $z_i \in \mathrm{Ch}_{F_{u_i}, u_i}(u_i)$ such that $V(W_i) \subseteq V(F_{z_i})$. Notice that $(F_{u_i}, \chi|_{V(F_{u_i})}, u_i)$ is an $\mathcal{F}$-elimination forest of $G[\chi(F_{u_i})]$ of height at most $k - i$.

Observe that z_k should be a leaf of (F, R) and therefore, since we have that $V(W_k) \subseteq V(F_{z_k}) \subseteq V(G) \setminus S$, W_k is a wall of $G \setminus S$ of height at least $r - k \geq h$. ∎

Now we prove our result regarding the bidimensionality of k-elimination sets.

LEMMA 5.9. *Let $\mathcal{F}$ be a finite collection of graphs. Let G be a graph, let $A \subseteq V(G)$, let $k \in \mathbb{N}$, let $r \geq 2k + 3$ be an odd integer, let $(W, \mathfrak{R})$ be a flatness pair of $G \setminus A$, and let S be a k-elimination set of G for $\mathrm{exc}(\mathcal{F})$. There is a set $X \supseteq S$ such that $G \setminus X \in \mathrm{exc}(\mathcal{F})$ and $\mathrm{bid}_{G \setminus A, W}(X) \leq k(k+1)/2$.*

PROOF. Let $p = \mathsf{odd}(r - k)$. Let W' be a p-subwall of W that is a wall of $G \setminus S$, which exists due to Lemma 5.8. Let C be the connected component of $G \setminus S$ that contains W'. Since $C \in \mathsf{exc}(\mathcal{F})$, $K_{s_{\mathcal{F}}}$ is not a minor of C. Moreover, since S is a k-elimination set of G for $\mathsf{exc}(\mathcal{F})$, there is a set $P \subseteq S$ of size at most k such that $(L, R) := (V(G) \setminus V(C), V(C) \cup P)$ is a separation of G with $L \cap R = P$.

Let us show that $\mathsf{bid}_{G \setminus A, W}(V(G) \setminus V(C)) \le k(k + 1)/2$. Let $\tilde{Q}$ be a W-canonical partition of $G \setminus A$. Let l be the number of internal bags of $\tilde{Q}$ intersected by P and note that $l \le k$.

Let G' be the graph obtained from G after contracting each bag of $\tilde{Q}$ to a vertex. It is easy to observe that G' is isomorphic to a planar supergraph of an h-grid H, where $h = r - 2$, together with an additional vertex that is adjacent to every vertex of the perimeter of H.

We let $[h]^2$ be the vertex set of H, where (i, j) and (i', j') are adjacent if and only if $|i - i'| + |j - j'| = 1$. We will show that there is a separation (L', R') of H of order at most l that maximizes $\min\{|L'|, |R'|\}$. Let $A = L' \cap R'$. We suppose without loss of generality that $|L'| \le |R'|$. Notice that $l < h$. We take $A := \{(i, j) \in [h]^2 \mid i + j = l\}$, i.e., L' is the set of pairs of indices in the triangle bounded by $(0, 0)$, $(0, l)$, and $(l, 0)$. Thus, $|L'| = l(l + 1)/2$. It is easy to verify that this maximizes $|L'|$.

Therefore, since the vertices of H are the internal bags of $\tilde{Q}$ and P intersects l internal bags, it implies that one of L and R intersects at most $l(l + 1)/2 \le k(k + 1)/2$ internal bags of $\tilde{Q}$. Recall that W' is a wall of C of height p. It is easy to verify that an elementary x-wall W^* has $2x^2 - 2$ vertices with $8x - 10$ vertices in the perimeter. Hence, it has $2(x - 2)^2$ vertices not in the perimeter, and therefore the canonical partition of W^* has $(x - 2)^2$ internal bags. Thus, the canonical partition of W' has $(p - 2)^2$ internal bags. Observe that each such a bag is contained in an internal bag of $\tilde{Q}$ and therefore $V(C)$ intersects at least $(p - 2)^2$ internal bags of $\tilde{Q}$. Since $(p - 2)^2 \ge (r - k - 2)^2 \ge (k + 1)^2 > k(k + 1)/2$, it holds that $\mathsf{bid}_{G \setminus A, W}(V(C) \cup P) > k(k + 1)/2$. Therefore, $\mathsf{bid}_{G \setminus A, W}(V(G) \setminus V(C)) \le k(k + 1)/2$. ∎

5.6   Homogeneous walls

In this subsection, we define homogeneous flat walls. Intuitively, homogeneous flat walls are flat walls that allow the routing of the same set of (topological) minors in the augmented flaps (i.e., the flaps together with the apex set) "cropped" by each one of their bricks. Such a flat wall can be detected in a big enough flat wall (Proposition 3.5) and this "homogeneity" property implies that some central part of a big enough homogeneous wall can be declared irrelevant (Proposition 3.6). The results presented in this subsection are from [69, 70].

Folios. We say that (M, T) is a tm-*pair* if M is a graph, $T \subseteq V(M)$, and all vertices in $V(M) \setminus T$ have degree two. We denote by $\mathsf{diss}(M, T)$ the graph obtained from M by dissolving all vertices in $V(M) \setminus T$. A tm-*pair* of a graph G is a tm-pair (M, T) where M is a subgraph of G. We call the vertices in T *branch* vertices of (M, T). We need to deal with topological minors for

the notion of homogeneity defined below, on which the statement of [9, Theorem 5.2] relies. If $\mathbf{M} = (M, B, \rho) \in \mathcal{B}$ and $T \subseteq V(M)$ with $B \subseteq T$, we call $(\mathbf{M}, T)$ a btm-*pair* and we define $\mathrm{diss}(\mathbf{M}, T) = (\mathrm{diss}(M, T), B, \rho)$. Note that we do not permit dissolution of boundary vertices, as we consider all of them to be branch vertices. If $\mathbf{G} = (G, B, \rho)$ is a boundaried graph and (M, T) is a tm-pair of G where $B \subseteq T$, then we say that $(\mathbf{M}, T)$, where $\mathbf{M} = (M, B, \rho)$, is a btm-*pair* of $\mathbf{G} = (G, B, \rho)$. Let $\mathbf{G}_1, \mathbf{G}_2$ be two boundaried graphs. We say that $\mathbf{G}_1$ is a *topological minor* of $\mathbf{G}_2$, denoted by $\mathbf{G}_1 \preceq_{\mathrm{tm}} \mathbf{G}_2$, if $\mathbf{G}_2$ has a btm-pair $(\mathbf{M}, T)$ such that $\mathrm{diss}(\mathbf{M}, T)$ is isomorphic to $\mathbf{G}_1$. Given a $\mathbf{G} \in \mathcal{B}$ and a positive integer ℓ, we define the ℓ-*folio* of $\mathbf{G}$ as

$$\ell\text{-folio}(\mathbf{G}) = \{\mathbf{G}' \in \mathcal{B} \mid \mathbf{G}' \preceq_{\mathrm{tm}} \mathbf{G} \text{ and } \mathbf{G}' \text{ has detail at most } \ell\}.$$

Augmented flaps. Let G be a graph, A be a subset of $V(G)$ of size a, and $(W, \mathfrak{R})$ be a flatness pair of $G \setminus A$. For each flap $F \in \mathsf{Flaps}_{\mathfrak{R}}(W)$ we consider a labeling $\ell_F : \partial F \to \{1, 2, 3\}$ such that the set of labels assigned by ℓ_F to ∂F is one of $\{1\}, \{1, 2\}, \{1, 2, 3\}$. Also, let $\tilde{a} \in [a]$. For every set $\tilde{A} \in \binom{A}{\tilde{a}}$, we consider a bijection $\rho_{\tilde{A}} : \tilde{A} \to [\tilde{a}]$. The labelings in $\mathcal{L} = \{\ell_F \mid F \in \mathsf{Flaps}_{\mathfrak{R}}(W)\}$ and the labelings in $\{\rho_{\tilde{A}} \mid \tilde{A} \in \binom{A}{\tilde{a}}\}$ will be useful for defining a set of boundaried graphs that we will call augmented flaps. We first need some more definitions.

　　　Given a flap $F \in \mathsf{Flaps}_{\mathfrak{R}}(W)$, we define an ordering $\Omega(F) = (x_1, \ldots, x_q)$, with $q \leq 3$, of the vertices of ∂F so that

—　$(x_1, \ldots, x_q)$ is a counter-clockwise cyclic ordering of the vertices of ∂F as they appear in the corresponding cell of $C(\Gamma)$. Notice that this cyclic ordering is significant only when $|\partial F| = 3$, in the sense that (x_1, x_2, x_3) remains invariant under shifting, i.e., (x_1, x_2, x_3) is the same as (x_2, x_3, x_1) but not under inversion, i.e., (x_1, x_2, x_3) is not the same as (x_3, x_2, x_1), and

—　for $i \in [q]$, $\ell_F(x_i) = i$.

Notice that the second condition is necessary for completing the definition of the ordering $\Omega(F)$, and this is the reason why we set up the labelings in $\mathcal{L}$.

　　　For each set $\tilde{A} \in \binom{A}{\tilde{a}}$ and each $F \in \mathsf{Flaps}_{\mathfrak{R}}(W)$ with $t_F := |\partial F|$, we fix $\rho_F : \partial F \to [\tilde{a}+1, \tilde{a}+t_F]$ such that $(\rho_F^{-1}(\tilde{a}+1), \ldots, \rho_F^{-1}(\tilde{a}+t_F)) = \Omega(F)$. Also, we define the boundaried graph

$$\mathbf{F}^{\tilde{A}} := (G[\tilde{A} \cup F], \tilde{A} \cup \partial F, \rho_{\tilde{A}} \cup \rho_F)$$

and we denote by $F^{\tilde{A}}$ the underlying graph of $\mathbf{F}^{\tilde{A}}$. We call $\mathbf{F}^{\tilde{A}}$ an $\tilde{A}$-*augmented flap* of the flatness pair $(W, \mathfrak{R})$ of $G \setminus A$ in G.

Palettes and homogeneity. For each $\mathfrak{R}$-normal cycle C of $\mathsf{Compass}_{\mathfrak{R}}(W)$ and each set $\tilde{A} \subseteq A$, we define $(\tilde{A}, \ell)\text{-palette}(C) = \{\ell\text{-folio}(\mathbf{F}^{\tilde{A}}) \mid F \in \mathsf{influence}_{\mathfrak{R}}(C)\}$. Given a set $\tilde{A} \subseteq A$, we say that the flatness pair $(W, \mathfrak{R})$ of $G \setminus A$ is ℓ-*homogeneous with respect to* $\tilde{A}$ if every *internal* brick of W has the *same* $(\tilde{A}, \ell)$-palette (seen as a cycle of $\mathsf{Compass}_{\mathfrak{R}}(W)$). Also, given a collection

$\mathcal{S} \subseteq 2^A$, we say that the flatness pair $(W, \mathfrak{R})$ of $G \setminus A$ is ℓ-*homogeneous with respect to* $\mathcal{S}$ if it is ℓ-homogeneous with respect to every $\tilde{A} \in \mathcal{S}$.

6.   Vertex deletion to a minor-closed graph class

In this section, we prove our main result for the $\mathcal{F}$-M-DELETION problem. The following theorem is a restatement of Theorem 1.2 using the reformulation introduced in Subsection 2.2.

THEOREM 6.1. *For every finite collection of graphs $\mathcal{F}$, there exists an algorithm that, given a graph G and a non-negative integer k, runs in time $2^{k^{O_{\ell_{\mathcal{F}}}(1)}} \cdot n^2$ and either outputs a k-apex set of G for* $\mathrm{exc}(\mathcal{F})$ *or reports that such a set does not exist.*

This algorithm is a generalization of the algorithm in the apex-minor free case of [70]. In order to give an algorithm without the apex-minor restriction, we enhance the techniques of [70] with some new tricks. We present this algorithm in this paper as a stepping stone to present the algorithms for elimination distance in Section 8 and Section 9, since the techniques used are similar (however not the same).

6.1   Description of the algorithm for $\mathcal{F}$-M-DELETION

Our algorithm for $\mathcal{F}$-M-DELETION has three steps. In Step 1, either we can easily conclude with a positive or a negative answer or we find a big wall. If we can find a large flat wall of bounded treewidth inside this wall, then we go to Step 2 and find an irrelevant vertex. Otherwise, we proceed to Step 3 where, by using flow techniques, we find a set of vertices that intersects every solution, and we branch on this set or we report a negative answer. The correctness of the algorithm is not trivial and will be justified in Subsection 6.2. While the general scheme of the algorithm is similar to the algorithm in [70] for $\mathcal{F}$-M-DELETION in the case where $\mathcal{F}$ contains an apex-graph, here, in order to obtain a quadratic algorithm for general $\mathcal{F}$, we employ additional novel tricks so as to deal with the possible existence of many apices in all graphs in $\mathcal{F}$.

We define the following constants.

$$a = f_3(s_{\mathcal{F}} + a_{\mathcal{F}} - 1), \qquad\qquad b = f_3(s_{\mathcal{F}}),$$
$$q = f_{10}(a_{\mathcal{F}}, s_{\mathcal{F}}, k), \qquad\qquad p = f_{11}(a_{\mathcal{F}}, s_{\mathcal{F}}, k),$$
$$l = (q - 1) \cdot (k + b), \qquad\qquad r_6 = f_7(a + b, \ell_{\mathcal{F}}, 3, k)$$
$$d = f_8(a + b, \ell_{\mathcal{F}}) \qquad\qquad r_5 = f_5(r_6, a + b, a + b, d),$$
$$t = f_4(s_{\mathcal{F}}) \cdot r_5, \qquad\qquad r_4 = \mathrm{odd}(t + 3),$$
$$r_3 = f_{13}(a_{\mathcal{F}}, r_4, 1), \qquad\qquad r_2 = \mathrm{odd}(2 + f_2(s_{\mathcal{F}} + a_{\mathcal{F}} - 1) \cdot r_3),$$
$$r_2' = \mathrm{odd}(\max\{f_9(a_{\mathcal{F}}, s_{\mathcal{F}}, k), f_{13}(l + 1, r_2, p)\}), \qquad\qquad r_1 = \mathrm{odd}(f_2(s_{\mathcal{F}}) \cdot r_2' + k).$$

Note that $r_6 = O_{\ell_{\mathcal{F}}}(k), r_5, r_4, r_3, r_2, t = O_{\ell_{\mathcal{F}}}(k^c)$ and $r_2', r_1 = O_{\ell_{\mathcal{F}}}(k^{c+2})$ where $c = f_6(a+b, a+b, d)$. Recall from Subsection 2.2 that we assume that G has $O_{s_{\mathcal{F}}}(k\sqrt{\log k} \cdot n)$ edges.

Step 1. Run the algorithm `Find-Wall` from Proposition 3.1 with input (G, r_1, k) and, in time $2^{O_{\ell_{\mathcal{F}}}(r_1^2 + (k+r_1)\log(k+r_1))} \cdot n = 2^{O_{\ell_{\mathcal{F}}}(k^{2(c+2)})} \cdot n$,

— either report a no-instance, or

— conclude that $\mathrm{tw}(G) \le f_1(s_{\mathcal{F}}) \cdot r_1 + k$ and solve $\mathcal{F}$-M-DELETION in time $2^{O_{\ell_{\mathcal{F}}}((r_1+k)\log(r_1+k))} \cdot n = 2^{O_{\ell_{\mathcal{F}}}(k^{c+2} \cdot \log k)} \cdot n$ using the algorithm of Proposition 3.2, or

— obtain an r_1-wall W_1 of G.

If the output of Proposition 3.1 is an r_1-wall W_1, consider all the $\binom{r_1}{r_2}^2 = 2^{O_{\ell_{\mathcal{F}}}(k^c \log k)}$ r_2-subwalls of W_1. For each one of them, say W_2, let W_2^* be the central $(r_2 - 2)$-subwall of W_2 and let D_{W_2} be the graph obtained from G after removing the perimeter of W_2 and taking the connected component containing W_2^*. Run the algorithm `Grasped-or-Flat` of Proposition 3.3 with input $(D_{W_2}, r_3, s_{\mathcal{F}} + a_{\mathcal{F}} - 1, W_2^*)$. This can be done in time $O_{s_{\mathcal{F}}}(k\sqrt{\log k} \cdot n)$.

If for some of these subwalls the result is a set $A \subseteq V(D_{W_2})$ with $|A| \le a$ and a flatness pair $(W_3, \mathfrak{R}_3)$ of $D_{W_2} \setminus A$ of height r_3 then, as in Proposition 5.6, compute a W_3-canonical partition $\tilde{Q}$ of $D_{W_2} \setminus A$ and a collection $\mathcal{W} = \{W^1, ..., W^{a_{\mathcal{F}}}\}$ of r_4-subwalls of W_3 such that for every $i \in [a_{\mathcal{F}}]$, $\bigcup \mathrm{influence}_{\mathfrak{R}_3}(W^i)$ is a subgraph of $\bigcup\{Q \mid Q$ is a p-internal bag of $\tilde{Q}\}$ and for every $i, j \in [a_{\mathcal{F}}]$, with $i \ne j$, there is no internal bag of $\tilde{Q}$ that contains vertices of both $V(\bigcup \mathrm{influence}_{\mathfrak{R}_3}(W^i))$ and $V(\bigcup \mathrm{influence}_{\mathfrak{R}_3}(W^j))$. This can be done in time $O_{s_{\mathcal{F}}}(k\sqrt{\log k} \cdot n)$.

For $i \in [a_{\mathcal{F}}]$, let W^{i*} be the central $(r_4 - 2)$-subwall of W^i and let D_{W^i} be the graph obtained from D_{W_2} after removing A and the perimeter of W^i and taking the connected component containing W^{i*}. Run the algorithm `Clique-or-twFlat` of Proposition 3.4 with input $(D_{W^i}, r_5, s_{\mathcal{F}})$. This takes time $2^{O_{\ell_{\mathcal{F}}}(r_5^2)} \cdot n = 2^{O_{\ell_{\mathcal{F}}}(k^{2c})} \cdot n$. If for one of these subwalls the result is a set A' of size at most b and a regular flatness pair $(W_5, \mathfrak{R}_5)$ of $D_{W^i} \setminus A'$ of height r_5 whose $\mathfrak{R}_5$-compass has treewidth at most t, then we proceed to Step 2.

If, for every flatness pair $(W_3, \mathfrak{R}_3)$ and for every $i \in [a_{\mathcal{F}}]$, the result is a report that $K_{s_{\mathcal{F}}}$ is a minor of D_{W^i}, then we proceed to Step 3.

Step 2 (irrelevant vertex case). We now obtain a 7-tuple $\mathfrak{R}_5'$ by adding all vertices of $G \setminus V(\mathrm{Compass}_{\mathfrak{R}_5}(W_5))$ to the set in the first coordinate of $\mathfrak{R}_5$, such that $(W_5, \mathfrak{R}_5')$ is a regular flatness pair of $G \setminus (A \cup A')$ whose $\mathfrak{R}_5'$-compass has treewidth at most t. We apply the algorithm `Homogeneous` of Proposition 3.5 with input $(r_6, a+b, a+b, d, t, G, A \cup A', W_5, \mathfrak{R}_5')$, which outputs, in time $2^{O_{\ell_{\mathcal{F}}}(t \log t + k \log k)} \cdot n = 2^{O_{\ell_{\mathcal{F}}}(k^c \log k)} \cdot n$, a flatness pair $(W_6, \mathfrak{R}_6)$ of $G \setminus (A \cup A')$ of height r_6 that is d-homogeneous with respect to $2^{A \cup A'}$ and is a W^*-tilt of $(W_5, \mathfrak{R}_5')$ for some subwall W^* of W_5. We apply the algorithm `Find-Irrelevant-Vertex` of Proposition 3.6 with input $(k, a+b, G, A \cup A', W_6, \mathfrak{R}_6)$, which outputs, in time $O(n+m) = O_{\ell_{\mathcal{F}}}(k\sqrt{\log k} \cdot n)$, a vertex v such

that (G, k) and $(G \setminus v, k)$ are equivalent instances of $\mathcal{F}$-M-DELETION. Then the algorithm runs recursively on the equivalent instance $(G \setminus v, k)$.

Step 3 (branching case). Consider all the r_2'-subwalls of W_1, which are at most $\binom{r_1}{r_2'}^2 = 2^{O_{\ell_{\mathcal{F}}}(k^{c+2}\log k)}$ many, and for each of them, say W_2', compute its canonical partition Q. Then, contract each bag Q of Q to a single vertex v_Q, and add a new vertex v_{all} and make it adjacent to all v_Q's. In the resulting graph G', for every vertex y of $G \setminus V(W_2')$, check, using a flow augmentation algorithm [20], whether there are q internally vertex-disjoint paths from v_{all} to y in time $O(q \cdot m) = O_{\ell_{\mathcal{F}}}(k^4\sqrt{\log k} \cdot n)$. Let $\tilde{A}$ be the set of all such y's.

<If $|\tilde{A}| < a_{\mathcal{F}}$, then report a no-instance.

If $a_{\mathcal{F}} \le |\tilde{A}| \le k + b$, then consider all the $\binom{|\tilde{A}|}{a_{\mathcal{F}}} = 2^{O_{\ell_{\mathcal{F}}}(\log k)}$ subsets of $\tilde{A}$ of size $a_{\mathcal{F}}$. For each one of them, say A^*, construct $\tilde{Q}$ by enhancing Q on $G \setminus A^*$. Then, we distinguish two cases depending on whether for every A^* all its vertices are adjacent to vertices of q p-internal bags of $\tilde{Q}$.

If each vertex of A^* is adjacent to vertices of q p-internal bags of $\tilde{Q}$, then (due to Proposition 3.7) A^* should intersect every solution of $\mathcal{F}$-M-DELETION for the instance (G, k). Therefore, the algorithm runs recursively on each instance $(G \setminus y, k - 1)$ for $y \in A^*$. If one of them is a yes-instance with $(k - 1)$-apex set S of $G \setminus y$, then (G, k) is a yes-instance with k-apex set $S \cup \{y\}$ of G. If all of them are no-instances, then report a no-instance. This concludes the case where each vertex of A^* is adjacent to vertices of q p-internal bags of $\tilde{Q}$.

If for every subset A^* of $\tilde{A}$ of size $a_{\mathcal{F}}$, there is a vertex of A^* that is not adjacent to vertices of q p-internal bags of the given $\tilde{Q}$, then report a no-instance. This concludes the case that $a_{\mathcal{F}} \le |\tilde{A}| \le k + b$.

If for every wall, $|\tilde{A}| > k + b$, then report that (G, k) is a no-instance of $\mathcal{F}$-M-DELETION.

Notice that Step 3, when applied, takes time $2^{O_{\ell_{\mathcal{F}}}(k^{c+2}\log k)} \cdot n^2$, because we apply the flow algorithms to each of the $2^{O_{\ell_{\mathcal{F}}}(k^{c+2}\log k)}$ r_2'-subwalls and for each vertex of G. However, the search tree created by the branching technique has at most $a_{\mathcal{F}}$ branches and depth at most k. So Step 3 cannot be applied more than $a_{\mathcal{F}}^k$ times during the course of the algorithm. Since Step 1 runs in time $2^{O_{\ell_{\mathcal{F}}}(k^{2(c+2)})} \cdot n$, Step 2 runs in time $2^{O_{\ell_{\mathcal{F}}}(k^{2c})} \cdot n$, and both may be applied at most n times, the claimed time complexity follows: the algorithm runs in time $2^{O_{\ell_{\mathcal{F}}}(k^{2(c+2)})} \cdot n^2$.

6.2   Correctness of the algorithm

Suppose first that (G, k) is a yes-instance and let S be a k-apex set of G. The application of the algorithm Find-Wall of Proposition 3.1 with input (G, r_1, k) either returns a report that $\text{tw}(G) \le f_1(s_{\mathcal{F}}) \cdot r_1 + k$ or returns an r_1-wall. In the first case, i.e., if $\text{tw}(G) \le f_1(s_{\mathcal{F}}) \cdot r_1 + k$, the application of the algorithm of Proposition 3.2 correctly outputs a k-apex set of G. We will focus on the latter case, i.e., where the algorithm Find-Wall returns an r_1-wall of G, say W_1.

Since $r_1 \geq f_2(s_{\mathcal{F}}) \cdot r_2' + k$, there is an $(f_2(s_{\mathcal{F}}) \cdot r_2')$-subwall of W_1, say W_1^*, that does not contain vertices of S. Since $G \setminus S$ does not contain $K_{s_{\mathcal{F}}}$ as a minor, there is no model of $K_{s_{\mathcal{F}}}$ grasped by W_1^* and therefore, due to Proposition 3.3 with input $(G \setminus S, r_2', s_{\mathcal{F}}, W_1^*)$, we know that there is a set $B \subseteq V(G \setminus S)$, with $|B| \leq b$, and a flatness pair $(W_2', \mathfrak{R}_2')$ of $G \setminus (S \cup B)$ of height r_2' such that W_2' is a W''-tilt of some subwall W'' of W_1^*.

Let Q be the canonical partition of W_2'. Let G' be the graph obtained by contracting each bag Q of Q to a single vertex v_Q, and adding a new vertex v_{all} and making it adjacent to all v_Q's. Let $\tilde{A}$ be the set of vertices y of $G \setminus V(W_2')$ such that there are q internally vertex-disjoint paths from v_{all} to y in G'. We claim that $\tilde{A} \subseteq S \cup B$. To show this, we first prove that, for every $y \notin S \cup B$, the maximum number of internally vertex-disjoint paths from v_{all} to y in G' is $k + b + 4$. Indeed, if y is a vertex in the $\mathfrak{R}_2'$-compass of W_2', there are at most $k + b$ such paths that intersect the set $S \cup B$ and at most four paths that do not intersect $S \cup B$ (in the graph $G' \setminus (S \cup B)$) due to the fact that $(W_2', \mathfrak{R}_2')$ is a flatness pair of $G \setminus (S \cup B)$. If y is not a vertex in the $\mathfrak{R}_2'$-compass of W_2', then, since by the definition of flatness pairs the perimeter of W_2' together with the set $S \cup B$ separate y from the $\mathfrak{R}_2'$-compass of W_2', every collection of internally vertex-disjoint paths from v_{all} to y in G' should intersect the set $\{v_{Q_{\text{ext}}}\} \cup S \cup B$, where Q_{ext} is the external bag of Q. Therefore, in both cases, if $y \notin S \cup B$, the maximum number of internally vertex-disjoint paths from v_{all} to y in G' is $k + b + 4$. Since $k + b + 4 < q$, we have that $y \notin \tilde{A}$. Hence, $\tilde{A} \subseteq S \cup B$ and therefore $|\tilde{A}| \leq k + b$. Hence, if (G, k) is a yes-instance we cannot have that $|\tilde{A}| > k + b$, so the algorithm correctly reports a no-instance at the end of Step 3.

Let $\tilde{Q}$ be a W_2'-canonical partition of $G \setminus (S \cup B)$ obtained by enhancing Q on $G \setminus (S \cup B)$. Let $\tilde{A}'$ be the set of vertices in $S \cup B$ that are adjacent to vertices of at least q p-internal bags of $\tilde{Q}$ (recall that $\tilde{A}$ is the set of vertices in $S \cup B$ that are adjacent to vertices of at least q internal bags of $\tilde{Q}$). Note that $\tilde{A}' \subseteq \tilde{A}$ and therefore $|\tilde{A}'| \leq |\tilde{A}|$.

If $|\tilde{A}'| < a_{\mathcal{F}}$, then at most $a_{\mathcal{F}} - 1$ vertices of $S \cup B$ are adjacent to vertices of at least q p-internal bags of $\tilde{Q}$. This means that the p-internal bags of $\tilde{Q}$ that contain vertices adjacent to some vertex of $(S \cup B) \setminus \tilde{A}'$ are at most $(q - 1) \cdot (k + b) = l$.

Consider a family $\mathcal{W} = \{W^1, \ldots, W^{l+1}\}$ of $l + 1$ r_2-subwalls of W_2' such that for every $i \in [l+1]$, $\bigcup \text{influence}_{\mathfrak{R}_2'}(W^i)$ is a subgraph of $\bigcup \{Q \mid Q \text{ is a } p\text{-internal bag of } \tilde{Q}\}$ and for every $i, j \in [l+1]$, with $i \neq j$, there is no internal bag of $\tilde{Q}$ that contains vertices of both $V(\bigcup \text{influence}_{\mathfrak{R}_2'}(W^i))$ and $V(\bigcup \text{influence}_{\mathfrak{R}_2'}(W^j))$. The existence of $\mathcal{W}$ follows from Proposition 5.6 and the fact that $r_2' \geq f_{13}(l + 1, r_2, p)$.

The fact that the p-internal bags of $\tilde{Q}$ that contain vertices adjacent to some vertex of $(S \cup B) \setminus \tilde{A}'$ are at most l implies that there exists an $i \in [l+1]$ such that no vertex of $V(\bigcup \text{influence}_{\mathfrak{R}_2'}(W^i))$ is adjacent, in G, to a vertex in $(S \cup B) \setminus \tilde{A}'$. Let $W_2 := W^i$, let W_2^* be the central $(r_2 - 2)$-subwall of W_2, and let D_{W_2} be the graph obtained from G by removing the perimeter of W_2 and taking the connected component that contains W_2^*. Since no vertex of $V(\bigcup \text{influence}_{\mathfrak{R}_2'}(W^i))$ is adjacent,

in G, to a vertex in $(S \cup B) \setminus \tilde{A}'$, any path in D_{W_2} going from a vertex of W_2^* to a vertex in S must intersect a vertex of $\tilde{A}'$. Thus, there is no model of $K_{s_{\mathcal{F}}+a_{\mathcal{F}}-1}$ grasped by W_2^* in D_{W_2}, because otherwise, $K_{s_{\mathcal{F}}}$ would be a minor of $G \setminus S$. So, by applying the algorithm `Grasped-or-Flat` of Proposition 3.3 with input $(D_{W_2}, r_3, s_{\mathcal{F}} + a_{\mathcal{F}} - 1, W_2^*)$, since $r_2 - 2 \geq f_2(s_{\mathcal{F}} + a_{\mathcal{F}} - 1) \cdot r_3$, we should find a set $A \subseteq V(D_{W_2})$ with $|A| \leq a$ and a flatness pair $(W_3, \mathfrak{R}_3)$ of $D_{W_2} \setminus A$ of height r_3, such that W_3 is a tilt of some subwall $\tilde{W}_3$ of W_2.

Let $\tilde{Q}'$ be a W_3-canonical partition of $D_{W_2} \setminus A$. Let $\mathcal{W}' = \{W^1, ..., W^{a_{\mathcal{F}}}\}$ be a collection of r_4-subwalls of W_3 such that for every $i \in [a_{\mathcal{F}}]$, $\bigcup \text{influence}_{\mathfrak{R}_3}(W^i)$ is a subgraph of $\bigcup \{Q \mid Q$ is an internal bag of $\tilde{Q}'\}$ and for every $i, j \in [a_{\mathcal{F}}]$, with $i \neq j$, there is no internal bag of $\tilde{Q}'$ that contains vertices of both $V(\bigcup \text{influence}_{\mathfrak{R}_3}(W^i))$ and $V(\bigcup \text{influence}_{\mathfrak{R}_3}(W^j))$. Since $|\tilde{A}'| < a_{\mathcal{F}}$, there is an $i \in [a_{\mathcal{F}}]$ such that $V(\bigcup \text{influence}_{\mathfrak{R}_3}(W^i))$ does not intersect $\tilde{A}'$. The existence of $\mathcal{W}'$ follows from Proposition 5.6 and the fact that $r_3 \geq f_{13}(a_{\mathcal{F}}, r_4, 1)$.

Let $W_4 := W^i$. Let W_4^* be the central (r_4-2)-subwall of W_4 and let D_{W_4} be the graph obtained from D_{W_2} after removing A and the perimeter of W_4 and taking the connected component containing W_4^*. Observe that any path between a vertex of S and a vertex of $V(\bigcup \text{influence}_{\mathfrak{R}_3}(W_4))$ in D_{W_2} intersects $\tilde{A}'$. Since $\tilde{A}'$ does not intersect $V(\bigcup \text{influence}_{\mathfrak{R}_3}(W_4))$, it implies that $\tilde{A}'$ does not intersect D_{W_4}, and thus $S \cap D_{W_4} = \emptyset$. Therefore, D_{W_4} is a minor of $G \setminus S$ and $K_{s_{\mathcal{F}}}$ is not a minor of D_{W_4}. Moreover, W_4^* is a wall of D_{W_4} of height $r_4 - 2 \geq t+1$, so $\text{tw}(D_{W_4}) > t = f_4(s_{\mathcal{F}}) \cdot r_5$. Therefore, by applying the algorithm `Clique-or-twFlat` of Proposition 3.4 with input $(D_{W_4}, r_5, s_{\mathcal{F}})$, we should obtain a set A' of size at most b and a regular flatness pair $(W_5, \mathfrak{R}_5)$ of $D_{W_4} \setminus A'$ of height r_5 whose $\mathfrak{R}_5$-compass has treewidth at most t. All this is checked in Step 1, and thus, the algorithm should run Step 2.

If $|\tilde{A}'| \geq a_{\mathcal{F}}$, then, due to Proposition 3.7 and the fact that $r_2' \geq f_9(a_{\mathcal{F}}, s_{\mathcal{F}}, k)$, for any set $X \subseteq V(G)$ such that $\text{bid}_{G \setminus (S \cup B), W_2'}(X) \leq k$ and such that $G \setminus X \in \text{exc}(\mathcal{F})$, it holds that $X \cap \tilde{A}' \neq \emptyset$. In particular, for any k-apex set S', $S' \cap \tilde{A}' \neq \emptyset$ due to Observation 5.7. Thus, there is a vertex $y \in \tilde{A}'$ such that $(G \setminus y, k - 1)$ is a yes-instance. Hence, if the algorithm runs Step 3, it finds a vertex $y \in \tilde{A}'$ such that $(G \setminus y, k - 1)$ is a yes-instance.

Note that the enhancement $\tilde{Q}$ of the canonical partition Q is not unique. In particular, $\tilde{A}'$ depends on $\tilde{Q}$. However, as long as there is such a $\tilde{Q}$ such that $|\tilde{A}'| < a_{\mathcal{F}}$, the algorithm finds the wanted flatness pair $(W_4, \mathfrak{R}_4)$ in Step 1 and then runs Step 2. Hence, if (G, k) is a yes-instance, the algorithm runs Step 3 only if for all such $\tilde{A}'$, $|\tilde{A}'| \geq a_{\mathcal{F}}$. Note that, since $|\tilde{A}| \geq |\tilde{A}'|$, in this case we have that, for all such $\tilde{A}'$, $|\tilde{A}| \geq a_{\mathcal{F}}$. This justifies the arbitrary canonical partition enhancement in Step 3 and the fact that, if $|\tilde{A}| < a_{\mathcal{F}}$ in Step 3, then the algorithm reports a no-instance.

Let us now show the correctness of Step 2, and for this we do not suppose anymore that (G, k) is a yes-instance since the argument is the same for both types of instances. Suppose that the algorithm finds in Step 1 a set A' of size at most b and a regular flatness pair $(W_5, \mathfrak{R}_5)$ of

$D_{W_4} \setminus A'$ of height r_5 whose $\mathfrak{R}_5$-compass has treewidth at most t. We obtain a 7-tuple $\mathfrak{R}_5'$ by adding all vertices of $G \setminus V(\text{Compass}_{\mathfrak{R}_5}(W_5))$ to the set in the first coordinate of $\mathfrak{R}_5$. Since $(W_5, \mathfrak{R}_5)$ is a regular flatness pair of $D_{W_4} \setminus A'$ and since the vertices added in $\mathfrak{R}_5'$ are either in A, or adjacent at most to the perimeter of W_4, then $(W_5, \mathfrak{R}_5')$ is a regular flatness pair of $G \setminus (A \cup A')$. Since $\text{Compass}_{\mathfrak{R}_5}(W_5) = \text{Compass}_{\mathfrak{R}_5'}(W_5)$, $\text{Compass}_{\mathfrak{R}_5'}(W_5)$ has treewidth at most t. Thus, if we apply the algorithm Homogeneous of Proposition 3.5 with input $(r_6, a + b, a + b, d, t, G, A \cup A', W_5, \mathfrak{R}_5')$ we obtain a flatness pair $(W_6, \mathfrak{R}_6)$ of $G \setminus (A \cup A')$ of height r_6 that is d-homogeneous with respect to $2^{A \cup A'}$ and is a W^*-tilt of $(W_5, \mathfrak{R}_5')$ for some subwall W^* of W_5. Due to Observation 5.3, we know that $(W_6, \mathfrak{R}_6)$ is regular. Since $|A \cup A'| \leq a + b$, for any set $X \subseteq V(G)$, $|A \setminus X| \leq a + b$. Since $G \setminus S \in \text{exc}(\mathcal{F})$ and Observation 5.7 implies that $\text{bid}_{G \setminus (A \cup A'), W_6}(S) \leq k$, by applying the algorithm Find-Irrelevant Vertex of Proposition 3.6 with input $(k, a + b, G, A \cup A', W_6, \mathfrak{R}_6)$, we obtain a vertex v such that $G \setminus S \in \text{exc}(\mathcal{F})$ if and only if $G \setminus (S \setminus v) \in \text{exc}(\mathcal{F})$. It follows that (G, k) and $(G \setminus v, k)$ are indeed equivalent instances of $\mathcal{F}$-M-DELETION.

We now suppose that (G, k) is a no-instance. In the beginning of Step 1, the algorithm either reports a no-instance or finds a wall. In the latter case, the algorithm either goes to Step 2 or Step 3. If it runs Step 2, the previous paragraph justifies that the algorithm finds a vertex v such that $(G \setminus v, k)$ is a no-instance. If the algorithm runs Step 3, then it either reports a no-instance or recursively runs on instances $(G \setminus y, k - 1)$. If $(G \setminus y, k - 1)$ is yes-instance, then so is (G, k). Thus, $(G \setminus y, k - 1)$ is a no-instance for every considered vertex y and the algorithm always reports a no-instance. Hence, Theorem 6.1 follows.

7. Solving $\mathcal{F}$-M-ELIMINATION DISTANCE **on tree decompositions**

In the rest of the paper (except for Section 11) we focus on $\mathcal{F}$-M-ELIMINATION DISTANCE. In order to design an algorithm for this problem and prove Theorem 1.2 and Theorem 1.3 , we follow the same scheme as for $\mathcal{F}$-M-DELETION. The first step of this strategy is to present a dynamic programming algorithm that will allow us to solve $\mathcal{F}$-M-ELIMINATION DISTANCE for instances of bounded treewidth in FPT-time, namely Theorem 1.4. This is the analogue of for $\mathcal{F}$-M-ELIMINATION DISTANCE of the corresponding result for $\mathcal{F}$-M-DELETION, namely Proposition 3.2. The following theorem is a reformulation of Theorem 1.4.

THEOREM 7.1. *For every finite collection of graphs $\mathcal{F}$, there exists an algorithm that, given a graph G of treewidth at most tw and a non-negative integer k, decides whether $\text{ed}_{\text{exc}(\mathcal{F})}(G) \leq k$ in time $2^{O_{\ell_{\mathcal{F}}}(\text{tw} \cdot k + \text{tw} \log \text{tw})} \cdot n$.*

According to Proposition 4.4, $\text{td}(G) \leq \text{tw}(G) \cdot \log n$ for any graph G. Since $\text{ed}_{\text{exc}(\mathcal{F})}(G) \leq \text{td}(G) \leq \text{tw}(G) \cdot \log n$, Theorem 7.1 implies the existence of an XP-algorithm for $\mathcal{F}$-M-ELIMINATION DISTANCE parameterized by treewidth.

COROLLARY 7.2. *For every finite collection of graphs $\mathcal{F}$, there exists an algorithm that, given a graph G of treewidth at most* tw, *computes* $\mathrm{ed}_{\mathrm{exc}(\mathcal{F})}(G)$ *in time* $n^{O_{\ell_{\mathcal{F}}}(\mathrm{tw}^2)}$.

According to Proposition 4.4 again, $\mathrm{tw}(G) \leq \mathrm{td}(G)$ for any graph G. Since we moreover have $\mathrm{ed}_{\mathrm{exc}(\mathcal{F})}(G) \leq \mathrm{td}(G)$, Theorem 7.1 implies the existence of an FPT-algorithm for $\mathcal{F}$-M-ELIMINATION DISTANCE parameterized by treedepth.

COROLLARY 7.3. *For every finite collection of graphs $\mathcal{F}$, there exists an algorithm that, given a graph G of treedepth at most* td, *computes* $\mathrm{ed}_{\mathrm{exc}(\mathcal{F})}(G)$ *in time* $2^{O_{\ell_{\mathcal{F}}}(\mathrm{td}^2)} \cdot n$.

Our algorithm takes inspiration from the dynamic programming algorithm of Reidl, Rossmanith, Villaamil, and Sikdar [61] for treedepth.

PROPOSITION 7.4 ([61]). *Given a graph G, a tree decomposition of G of width w, and an integer k, there is an algorithm that decides whether* $\mathrm{td}(G) \leq k$ *in time* $2^{O(k \cdot w)} \cdot n$.

If $\mathcal{F} = \{K_1\}$, elimination distance reduces to treedepth. Recall that $\{K_1\}$-M-ELIMINATION DISTANCE is the problem asking whether $\mathrm{td}(G) \leq k$, which admits an algorithm in time $2^{O(k \cdot w)} \cdot n$ because of Proposition 7.4. Therefore, we may assume throughout this section that $\mathcal{F}$ is non-trivial in order to have the useful property that a graph with a single vertex belongs to $\mathcal{G} = \mathrm{exc}(\mathcal{F})$. This will simplify the algorithm. Moreover, the elimination distance to $\mathrm{exc}(\mathcal{F})$ of a disconnected graph is the maximum of the elimination distance of its connected components, and therefore, we may assume that the considered graphs and boundaried graphs are connected.

Compared to the approach of [61], in order to deal with a general minor-closed graph class $\mathcal{G}$, we use the framework introduced in [9] based on the notion of representatives of an appropriately defined equivalence relation on boundaried graphs. Intuitively, since the "leaves" of the desired elimination tree are graphs in the minor-closed family $\mathcal{G}$, it will be possible to encode those graphs via their corresponding representatives. The fact that the boundary size that we need to consider is bounded follows from the description of the algorithm and its analysis.

In order to describe our dynamic programming algorithm, we have to describe its corresponding tables, encode "partial elimination sets", and show how to calculate this information using a nice tree decomposition of the input graph. For this reason, in Subsection 7.1 we start by giving some additional notations on functions on sets and in Subsection 7.2 we define annotated trees. Annotated trees are labeled rooted trees that come together with a boundaried graph such that the annotated nodes of the tree are mapped to the vertices of the boundaried graph with the same label. This notion is used in Subsection 7.3 in order to define the *characteristic* of a boundaried graph, which intuitively encodes how partial elimination trees can be present inside the boundaried graph. Forget, introduce, and join procedures that shall be used in the dynamic program on nice tree decompositions are presented in Subsection 7.4. In Subsection 7.5,

we present the dynamic program and prove its correctness. We conclude this section with Subsection 7.6, where we show that boundaried graphs with the same characteristic can be exchanged, i.e., give graphs of the same elimination distance to $\mathcal{F}$ when "glued" to the same boundaried graph. This latter result will also be used in Section 10.

7.1   Some additional notation

We denote by $\mathsf{Im}(f)$ the image of a function f and by $\mathsf{Ker}(f)$ its kernel, i.e., the elements whose image by f is 0. Given two sets A and B, two subsets $A' \subseteq A$ and $B' \subseteq B$, a function $f : A' \to B'$, $a \in A$, and $b \in B$, $f \oplus [a \mapsto b]$ is the function that maps a to b and every $a' \in A' \setminus \{a\}$ to $f(a')$. If $f : A \to B$, we denote by $f|_{A'}$ the restriction of f to A'. When f is a bijection, $f|_{A'} : A' \to \mathsf{Im}(A')$ is seen as a bijection. $(i \leftrightarrow j)$ denotes the transposition of i and j, for i, j in some set $I \subseteq \mathbb{N}$.

7.2   Annotated trees

We proceed to define annotated trees, which we will use to codify the tables of our dynamic program.

Annotated trees. An *annotated tree* is a tuple $\hat{T} = (T, r, h, \mathbf{R}, f)$, where (T, r) is a rooted tree, $h : V(T) \to \mathbb{N}$, $\mathbf{R} = (R, B, \phi)$ is a boundaried graph, and $f : [|B|] \to V(T)$. See Figure 4 for an illustration of an annotated tree. We stress that different integers in $[|B|]$ can be mapped, via f, to the same node of T. The *trivial annotated tree*, denoted by $\hat{\mathbb{1}}$, is $(T, r, h, \mathbb{1}, f)$ where T is the rooted tree with a single node r, h is the constant function 0, $\mathbb{1}$ is the boundaried graph with one single vertex that is also part of the boundary, and f maps 1 to r. The *height* of an annotated tree is $h(r)$. Given an annotated tree $\hat{T} = (T, r, h, (R, B, \phi), f)$, we refer to (T, r) as its *rooted tree*. Given an annotated tree $\hat{T} = (T, r, h, (R, B, \phi), f)$ and a permutation σ of $[|B|]$, we use $\sigma(\hat{T})$ to denote $(T, r, h, (R, B, \sigma \circ \phi), f \circ \sigma^{-1})$.

 We also define the following operations on annotated trees, which will be used to combine the tables of the dynamic programming algorithm. The first one is inspired by a similar operation introduced in [61].

Crop operation. Given an annotated tree $\hat{T} = (T, r, h, (R, B, \phi), f)$, the *crop operation*, denoted by $\mathsf{crop}(\hat{T})$, outputs the annotated tree obtained from $\hat{T}$ by iteratively removing the leaves of T that are not in $\mathsf{Im}(f)$. Given a set $\mathcal{A}$ of annotated trees, $\mathsf{crop}(\mathcal{A}) := \bigcup_{\hat{T} \in \mathcal{A}} \mathsf{crop}(\hat{T})$.

Representation operation. Given an annotated tree $\hat{T} = (T, r, h, (R, B, \phi), f)$, the *representation operation*, denoted by $\mathsf{rep}(\hat{T})$, outputs the annotated tree $\hat{T}' = (T, r, h, (R', B, \phi), f)$ constructed as follows. For each $v \in \mathsf{Im}(f)$, let $B_v := \phi^{-1} \circ f^{-1}(v)$, let $\sigma_v : f^{-1}(v) \to [|B_v|]$ be a bijective function, and let R_v be the union of the connected components of R containing B_v. If there is a node $v \in \mathsf{Im}(f)$ such that $R_v \notin \mathsf{exc}(\mathcal{F})$, then $R' := K_{s_{\mathcal{F}}}$. Otherwise, for each $v \in \mathsf{Im}(f)$,
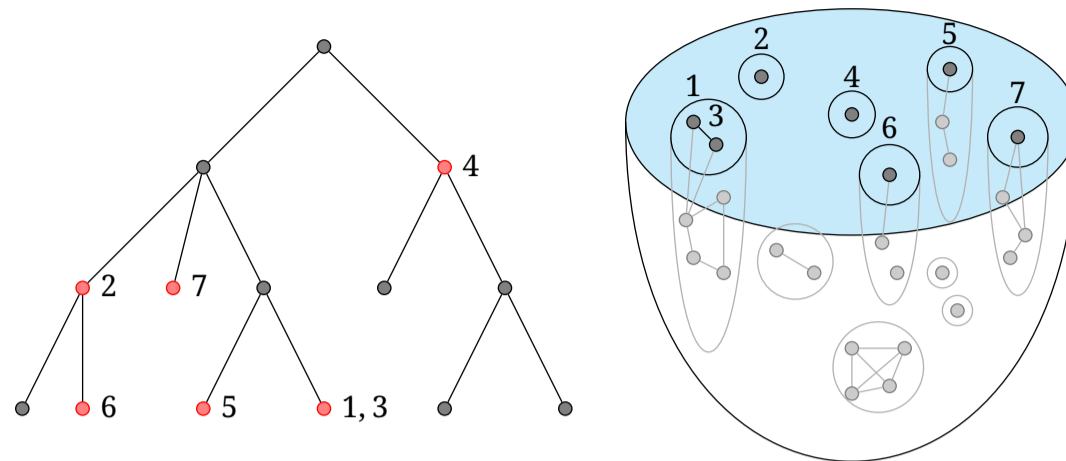
Figure 4. An annotated tree made of a rooted tree (left) and a boundaried graph (right). The numbers in the left figure correspond to the pre-images of f for the nodes of $V(T)$ and the numbers on the right figure correspond to the images of ϕ. The function h that gives a value to each node of the tree is not represented.

let $(R'_v, B_v, \sigma_v \circ \phi|_{B_v}) \in \mathcal{R}^{|B_v|}_{\ell_{\mathcal{F}}}$ be the representative of $(R_v, B_v, \sigma_v \circ \phi|_{B_v})$ for the equivalence relation $\equiv_{\ell_{\mathcal{F}}}$. Then $R' = \bigcup_{v \in \mathrm{Im}(f)} R'_v$. Intuitively, if there is a node $v \in \mathrm{Im}(f)$ such that $R_v \notin \mathrm{exc}(\mathcal{F})$, to store this information it suffices to set $R' := K_{s_{\mathcal{F}}}$, while otherwise, we keep for each R_v (in fact, for the boundaried version of R_v) its representative.

An example of the crop and representation operation is give in Figure 5. Observe that $\mathrm{rep}(\hat{\mathbb{1}}) = \hat{\mathbb{1}}$ since this is a minimum-sized representative and since we make the assumption that $\mathcal{F}$ is non-trivial. Given a set $\mathcal{A}$ of annotated trees, $\mathrm{rep}(\mathcal{A}) := \bigcup_{\hat{T} \in \mathcal{A}} \mathrm{rep}(\hat{T})$.
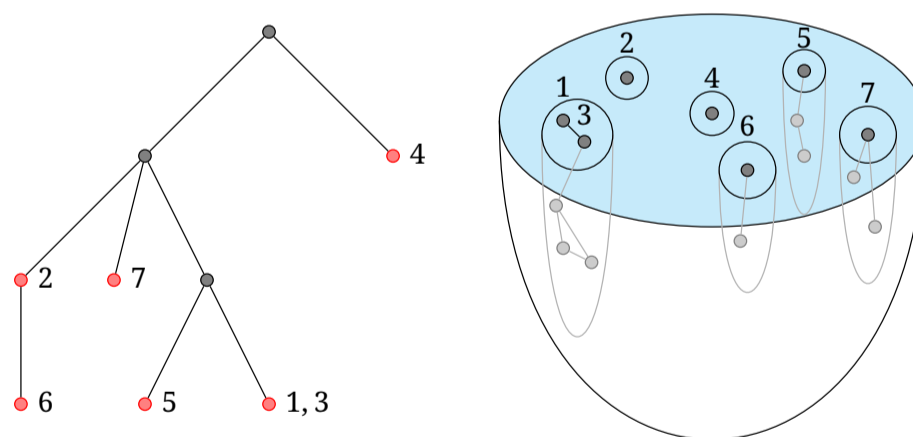
Figure 5. The crop and representation operation applied to the annotated tree of Figure 4. The unlabeled leaves of the tree are iteratively removed. A representative of each component attached to the boundary of the boundaried graph is kept.

Filter operation. Given a set $\mathcal{A}$ of annotated trees and a positive integer k, the *filter operation*, denoted by filter_k, outputs the set of annotated trees in $\mathcal{A}$ with height at most k.

Note that the crop, representation, and filter operations are commutative since they do not modify the same objects. For more simplicity, we define $\mathcal{M}_k = \mathsf{filter}_k \circ \mathsf{rep} \circ \mathsf{crop}$. We stress that $\mathcal{M}_k$ is an operation acting on *sets* of annotated trees.

7.3   Characteristic of a boundaried graph

In this subsection we define the *characteristic* of a boundaried graph that shall be computed by the dynamic program in Subsection 7.5. This characteristic will consist of a set of annotated trees with some additional properties. In order to present this definition, we first define the complete characteristic of a boundaried graph, that is a slightly more complicated way to see $\mathcal{F}$-elimination trees with some distinguished nodes.

Complete characteristic of a boundaried graph. Given a connected boundaried graph $\mathbf{G} = (G, X, \rho)$, the *complete characteristic* of $\mathbf{G}$, denoted by $\mathsf{char}^*(\mathbf{G})$, is the set of annotated trees $\hat{T} = (T, r, h, \mathbf{R}, f)$ such that

— $|\mathrm{Im}(f)| = |X|$,

— there exists a function $\chi : V(T) \to 2^{V(G)}$ such that (T, χ, r) is an $\mathcal{F}$-elimination tree of G and for $x \in X$, $x \in \chi \circ f \circ \rho(x)$,

— there exists an isomorphism σ between $\mathbf{R}$ and $(\bigcup_{v \in \mathrm{Im}(f)} G[\chi(v)], X, \rho)$, and

— h is the height function $\mathsf{height}_{T,r}$.

(χ, σ) is called the *witness pair* of $\hat{T}$ with respect to $\mathbf{G}$. Since (T, χ, r) is an $\mathcal{F}$-elimination tree, it is straightforward to see that for any boundaried graph $\mathbf{G}$ with underlying graph G, the minimum height of an annotated tree in $\mathsf{char}^*(\mathbf{G})$ is $\mathsf{ed}_{\mathsf{exc}(\mathcal{F})}(G)$.

Characteristic of a boundaried graph. Let $\mathbf{G} = (G, X, \rho)$ be a boundaried graph and k be an integer. The *characteristic* of $\mathbf{G}$, denoted by $\mathsf{char}_k(\mathbf{G})$, is the set $\mathcal{M}_k(\mathsf{char}^*(\mathbf{G}))$.

LEMMA 7.5. *Given a boundaried graph $\mathbf{G} = (G, X, \rho)$ with $X \neq \emptyset$ and an integer k, the elimination distance of G to $\mathsf{exc}(\mathcal{F})$ is the minimum height of an annotated tree in $\mathsf{char}_k(\mathbf{G})$ if $\mathsf{ed}_{\mathsf{exc}(\mathcal{F})}(G) \leq k$, and $\mathsf{char}_k(\mathbf{G}) = \emptyset$ otherwise.*

PROOF. Let $\mathbf{G} = (G, X, \rho)$ be a boundaried graph with $X \neq \emptyset$ and let $\hat{T} \in \mathsf{char}^*(\mathbf{G})$. Since $X \neq \emptyset$, the crop operation will not remove the root of the underlying tree of $\hat{T}$. Moreover, neither the crop operation nor the representation operation change the height of the nodes that stay in the tree. So the height of $\mathsf{rep} \circ \mathsf{crop}(\hat{T})$ is equal to the height of $\hat{T}$. ∎

We now prove that the size of the characteristic of a boundaried graph is upper-bounded by a function of its boundary size and k.

LEMMA 7.6. *There exists a function $f_{14} : \mathbb{N}^2 \rightarrow \mathbb{N}$ such that, given two integers k and w, if $\mathbf{G} = (G, X, \rho)$ is a boundaried graph with $|X| \leq w$, then $|\mathrm{char}_k(\mathbf{G})| \leq f_{14}(w, k)$. Moreover, $f_{14}(w, k) = 2^{O_{\ell_{\mathcal{F}}}(w \cdot k + w \log w)}$.*

PROOF. Let $\hat{T} = (T, r, h, \mathbf{R}, f)$ be an annotated tree in $\mathrm{char}_k(\mathbf{G})$. Let $l := |X|$. Let $x_1, ..., x_l$ be an ordering of the nodes in $\mathrm{Im}(f)$ such that if $x_i \in \mathrm{Anc}_{T,r}(x_j)$, then $i < j$. Without loss of generality, we suppose that $f(i) = x_i$ for $i \in [l]$ (this is true up to a permutation of $[l]$). Let f_i be the restriction of f to $[i]$, let T_i be the tree obtained from T by iteratively removing the leaves not in $\mathrm{Im}(f_i)$, and let h_i be the restriction of h to $V(T_i)$. Note that $\mathrm{Im}(h_i) \subseteq [0, k]$ and T_i is a tree with at most i leaves because the leaves of T are in $\mathrm{Im}(f_i)$ and $|\mathrm{Im}(f_i)| = i$. So T_i has at most $i \cdot (k + 1)$ nodes.

We set (T_0, h_0, f_0) to be the empty triple. Let us bound the number of triples (T_i, h_i, f_i) that can be constructed from $(T_{i-1}, h_{i-1}, f_{i-1})$ for $i \in [l]$. For $i \in [l]$, we can construct (T_i, h_i, f_i) from $(T_{i-1}, h_{i-1}, f_{i-1})$ by choosing a node of T_{i-1} (if it exists) and adding a path of length at most k with leaf x_i (all nodes in this path are new, except from x_i). We consider the function h_i that has the same values as h_{i-1} on $V(T_{i-1})$ and values in $[0, k]$ on the new path such that the value of h_i strictly increases from a leaf to the root r. Observe that the value of h_i on the new path is a subset of $2^{[k+1]}$. Therefore, the number of different triples (T, h, f) is at most

$$\prod_{i=1}^{w} i(k+1)2^{k+1} = w!(k+1)^w 2^{w(k+1)} \leq 2^{w \log w + w \log(k+1) + w(k+1)}.$$

Since $\hat{T} = (T, r, h, \mathbf{R}, f)$ is an annotated tree in $\mathrm{char}_k(\mathbf{G})$, it holds that $\mathbf{R}$ is the union of l representatives $\mathbf{R}_i \in \mathcal{R}_{\ell_{\mathcal{F}}}^{w_i}$ for $i \in [l]$ where $l = |\mathrm{Im}(f)|$, such that $\sum_{i=1}^{l} w_i = l \leq w$. By Proposition 4.9, $|\mathcal{R}_{\ell_{\mathcal{F}}}^{w_i}| = 2^{O_{\ell_{\mathcal{F}}}(w_i \log w_i)}$. So the number of ways to construct $\mathbf{R}$ is bounded by

$$\prod_{i=1}^{l} 2^{O_{\ell_{\mathcal{F}}}(w_i \log w_i)} = 2^{O_{\ell_{\mathcal{F}}}(\sum_{i=1}^{l} w_i \log w)} = 2^{O_{\ell_{\mathcal{F}}}(w \log w)}.$$

Hence, we obtain the desired result. ∎

7.4   The procedures

We define here the procedures that will be used in the dynamic programming algorithm. Given a nice tree decomposition $\mathcal{T}$ of a graph G, we want to define forget, introduce, and join procedures to obtain the characteristic of G_v for each internal node v in $\mathcal{T}$, given the characteristics of $G_{v'}$ for each child v' of v. Before defining the procedures for the characteristics, we define the procedures for the complete characteristics.

7.4.1   Forget procedure

With the forget procedure, given the characteristic of a boundaried graph, we want to compute the characteristic of the boundaried graph obtained by removing a vertex from the boundary.

Complete forget procedure. The complete forget procedure applied on the annotated tree $(T, r, h, \mathbf{R}, f)$ corresponds to removing the vertex with the largest label from the boundary of $\mathbf{R}$. More formally, given an annotated tree $\hat{T} = (T, r, h, (R, B, \phi), f)$, the *complete forget procedure*, denoted by $\mathrm{forget}^*(\hat{T})$, outputs the annotated tree $\hat{T}' = (T, r, h, (R, B', \phi|_{B'}), f|_{[|B'|]}))$, where $B' = B \setminus \phi^{-1}(|B|)$. Given a set $\mathcal{A}$ of annotated trees, $\mathrm{forget}^*(\mathcal{A}) := \bigcup_{\hat{T} \in \mathcal{A}} \mathrm{forget}^*(\hat{T})$.

LEMMA 7.7. *Let G be a graph, $(\mathsf{T}, \beta, \mathsf{r})$ be a nice tree decomposition of G, v be a forget node of T with child v' and forgotten vertex x, $\rho : \beta(v) \to [|\beta(v)|]$ be a bijection, and $\rho' := \rho \oplus [x \mapsto |\beta(v')|]$. Let $\mathcal{A} := \mathrm{char}^*(G_v, \beta(v), \rho)$ and $\mathcal{A}' := \mathrm{char}^*(G_{v'}, \beta(v'), \rho')$. Then $\mathcal{A} = \mathrm{forget}^*(\mathcal{A}')$.*

PROOF. Let $\hat{T} \in \mathrm{forget}^*(\mathcal{A}')$. There exists $\hat{T}' \in \mathcal{A}'$ with witness pair (χ', σ') such that $\mathrm{forget}^*(\hat{T}') = \hat{T}$. Let (T, r) (resp. (T', r)) be the rooted tree of $\hat{T}$ (resp. $\hat{T}'$). Note that $\mathrm{height}_{T,r} = \mathrm{height}_{T',r}$. Thus, it is easy to see that (χ', σ') also witnesses that $\hat{T} \in \mathcal{A}$. Conversely, let $\hat{T} = (T, r, h, (R, X, \phi), f) \in \mathcal{A}$ with witness pair (χ, σ). Let $u := \chi^{-1}(x)$, $w := \sigma^{-1}(x)$, and $t = |X| + 1$. Let $\hat{T}' := (T, r, h, (R, X \cup \{w\}, \phi \oplus [w \mapsto t]), f \oplus [t \mapsto u])$. Then $\mathrm{forget}^*(\hat{T}') = \hat{T}$ and the pair (χ, σ) also witnesses that $\hat{T}' \in \mathcal{A}'$, so $\hat{T} \in \mathrm{forget}^*(\mathcal{A}')$. ∎

Forget procedure. Given an annotated tree $\hat{T}$, the *forget procedure*, denoted by $\mathrm{forget}(\hat{T})$, outputs $\mathrm{rep} \circ \mathrm{crop} \circ \mathrm{forget}^*(\hat{T})$. See Figure 6 for an illustration. Note that we do not apply the filter operation since the height does not change under the forget procedure. Given a set $\mathcal{A}$ of annotated trees, $\mathrm{forget}(\mathcal{A})$ outputs $\bigcup_{\hat{T} \in \mathcal{A}} \mathrm{forget}(\hat{T})$.
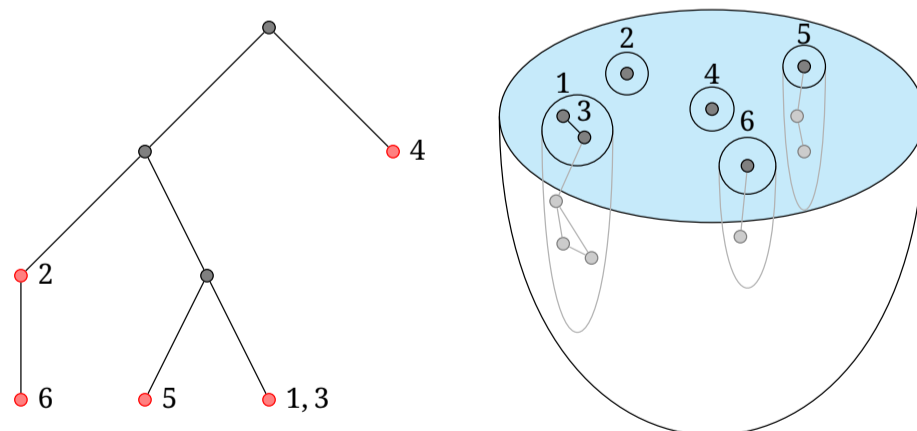

Figure 6. The forget procedure applied to the annotated tree of Figure 4.

LEMMA 7.8. *Let G be a graph, k be an integer, (T, β, r) be a nice tree decomposition of G, v be a forget node of T with child v' and forgotten vertex x, $\rho : \beta(v) \to [|\beta(v)|]$ be a bijection, and $\rho' := \rho \oplus [x \mapsto |\beta(v')|]$. Let $\mathcal{A} := \mathrm{char}_k(G_v, \beta(v), \rho)$ and $\mathcal{A}' := \mathrm{char}_k(G_{v'}, \beta(v'), \rho')$. Then $\mathcal{A} = \mathrm{forget}(\mathcal{A}')$.*

PROOF. Let $\mathcal{D} := \mathrm{char}^*(G_v, \beta(v), \rho)$ and $\mathcal{D}' := \mathrm{char}^*(G_{v'}, \beta(v'), \rho')$. According to Lemma 7.7, $\mathcal{D} = \mathrm{forget}^*(\mathcal{D}')$. The representation and the crop operation do not change the labels of an annotated tree, while the complete forget procedure only changes the labels of the input annotated tree and it does not change its height. Thus, $\mathrm{rep} \circ \mathrm{forget}^* = \mathrm{rep} \circ \mathrm{forget}^* \circ \mathrm{rep}$, $\mathrm{crop} \circ \mathrm{forget}^* = \mathrm{crop} \circ \mathrm{forget}^* \circ \mathrm{crop}$, and $\mathrm{filter}_k \circ \mathrm{forget}^* = \mathrm{forget}^* \circ \mathrm{filter}_k$. Since the three operations are commutative, it follows that $\mathcal{M}_k \circ \mathrm{forget}^* = \mathrm{rep} \circ \mathrm{crop} \circ \mathrm{forget}^* \circ \mathcal{M}_k = \mathrm{forget} \circ \mathcal{M}_k$. Hence, $\mathcal{A} = \mathcal{M}_k(\mathcal{D}) = \mathcal{M}_k \circ \mathrm{forget}^*(\mathcal{D}') = \mathrm{forget}(\mathcal{M}_k(\mathcal{D}')) = \mathrm{forget}(\mathcal{A}')$. ∎

7.4.2   Introduce procedure

With the introduce procedure, given the characteristic of a boundaried graph and a set I of labels from the boundary, we want to compute the characteristic of the boundaried graph obtained by adding a new vertex to the boundary, which is adjacent to the nodes with a label in I.

Diamond-introduce operation. Let (T, r) be a rooted tree, w be an integer, $f : [w] \to V(T)$ be a function, and I be a subset of $[w]$. $(T, r, f) \diamond_{\mathrm{intr}} I$ is defined as the set of all pairs (T', r', f') such that:

1. (T', r') is a rooted tree,
2. $V(T') = V(T) \cup \{u\}$ for some new node u,
3. $f' = f \oplus [w + 1 \mapsto u]$,
4. if $v_1 \in V(T)$ and $v_2 \in \mathrm{Anc}_{T,r}(v_1)$, then $v_2 \in \mathrm{Anc}_{T',r'}(v_1)$,
5. if $v \in f(I)$, then $v \in \mathrm{Anc}_{T,r}(u) \cup \mathrm{Desc}_{T,r}(u)$, and
6. $T'_u \cap f(I) \neq \emptyset$, or $u \in \mathrm{Leaf}(T', r')$ and $\mathrm{Par}_{T',r'}(u) \in f(I)$.

This operation corresponds to introducing a new node u in T so that u has ancestor-descendant relations with the nodes labeled by a label in I. The last item, which states that u either has a descendant in $f(I)$ or is a leaf and its parent belongs to $f(I)$, is a property needed to ensure connectivity and allows the application of the crop operation in Lemma 7.9 and Lemma 7.10.

Let (T, r) be a rooted tree, K be a subset of $V(T)$, and $h : K \to \mathbb{N}$. We define the function $\mathrm{update}_{T,r,K}(h) : V(T) \to \mathbb{N}$, that maps every $v \in V(T)$ to the integer

$$\mathrm{update}_{T,r,K}(h)(v) = \max\{h(v), 1 + \max_{c \in \mathrm{Ch}(v)} \{\mathrm{update}_{T,r,K}(h)(c)\}\},$$

where we suppose that $h(v) = 0$ if $v \notin K$ and $\mathrm{update}_{T,r,K}(h)(c) = -1$ if v is childless. Let (T', r') be a rooted tree with $V(T') = K$ and such that the ancestor-descendant relationship between the nodes of K is the same in (T, r) and (T', r'). Then we can observe that $\mathrm{update}_{T,r,K}(\mathrm{height}_{T',r'}) = \mathrm{height}_{T,r}$.

Complete introduce procedure. Let $\hat{T} = (T, r, h, (R, X, \phi), f)$ be an annotated tree and let I be a set of labels in $[|X|]$. The complete introduce procedure corresponds to adding a new vertex v to the boundary X of the boundaried graph (R, X, ϕ) of an annotated tree $(T, r, h, (R, X, \phi), f)$, such that v is adjacent to the nodes with a label in I and it is either mapped, via $f \circ \phi$, to an already existing leaf of T (item (a) below) or to a new node of T (item (b) below).

More formally, given an annotated tree $\hat{T} = (T, r, h, (R, X, \phi), f)$, a set $I \subseteq [|X|]$ of labels, the *complete introduce procedure*, denoted by $\mathrm{intr}^*(\hat{T}, I)$, outputs a set $\mathcal{A}$ of annotated trees constructed as follows. For each $(T', r', f') \in (T, r, f) \diamond_{\mathrm{intr}} I$, let $w = |\mathrm{Im}(f')|$, $u := f'(w)$, and $u' := \mathrm{Par}_{T',r'}(u)$ (or $u' := u$ if u is the root). We add a new vertex v to R and we set $X' := X \cup \{v\}$ and $\phi' := \phi \oplus [v \mapsto w]$. Let $(R_{u'}, X_{u'}, \phi_{u'})$ be the part of the representative R corresponding to u', as defined for the representation operation (i.e., $R_{u'}$ is the union of the connected components of R containing $\phi^{-1} \circ f^{-1}(u')$, $X_{u'} = \phi^{-1} \circ f^{-1}(u')$, $\phi_{u'} = \sigma_{u'} \circ \phi|_{X_{u'}}$ and $\sigma_{u'} : f^{-1}(u') \to [|X_{u'}|]$ is a bijective function).

(a) *If $h(u') = 0$:* We set $f'' := f \oplus [w \mapsto u']$ and $R' := (V(R) \cup \{v\}, E(R) \cup E(\{v\}, \phi^{-1}(I) \cap X_{u'}))$ and we add $(T, r, h, (R', X', \phi'), f'')$ to $\mathcal{A}$ if the connected component of R' containing v belongs to $\mathrm{exc}(\mathcal{F})$.

(b) *If $u' \notin \mathrm{Im}(f)$, or if $u' \in \mathrm{Im}(f)$ and $|V(R_{u'})| = 1$:* We set $R' := (V(R) \cup \{v\}, E(R))$ and $h' := \mathrm{update}_{T',r',V(T)}(h)$ and we add $(T', r', h', (R', X,' \phi'), f')$ to $\mathcal{A}$.

Note that this is not a dichotomy: if both criteria are fulfilled, then we apply both cases. Given a set $\mathcal{A}$ of annotated trees, we define $\mathrm{intr}^*(\mathcal{A}, I) := \bigcup_{\hat{T} \in \mathcal{A}} \mathrm{intr}^*(\hat{T}, I)$.

LEMMA 7.9. *Let G be a graph, (T, β, r) be a nice tree decomposition of G, v be an introduce node of T with child v' and introduced vertex x, $\rho' : \beta(v) \to [|\beta(v)|]$ be a bijection, $\rho := \rho' \oplus [x \mapsto |\beta(v)|]$, and $I := \rho(N_{G_v}(x))$. Let $\mathcal{A} := \mathrm{char}^*(G_v, \beta(v), \rho)$ and $\mathcal{A}' := \mathrm{char}^*(G_{v'}, \beta(v'), \rho')$. Then $\mathcal{A} = \mathrm{intr}^*(\mathcal{A}', I)$.*

PROOF. Let $\hat{T} = (T, r, h, (R, X, \phi), f) \in \mathrm{intr}^*(\mathcal{A}', I)$. There is $\hat{T}' = (T', r', h', \mathbf{R}', f') \in \mathcal{A}'$ with witness pair (χ', σ') such that $\hat{T} \in \mathrm{intr}^*(\hat{T}', I)$. Let $\chi := \chi' \oplus [u \mapsto x]$ and $\sigma = \sigma' \oplus [w \to x]$ where $u := f(|X|)$ and $w := \phi^{-1}(|X|)$.

Suppose that we are in case (a). Thus, $h = h' = \mathrm{height}_{T',r'} = \mathrm{height}_{T,r}$. By the construction of $\mathbf{R}$ from $\mathbf{R}'$, since x is only adjacent in G_v to the nodes with a label in I, σ is an isomorphism between $\mathbf{R}$ and $(\bigcup_{w \in \mathrm{Im}(f)} G_v[\chi(w)], \beta(v), \rho)$. Moreover, $T = T'$ so (T, χ, r) is an $\mathcal{F}$-elimination tree of G_v. Hence, (χ, σ) witnesses that $\hat{T} \in \mathcal{A}$.

Suppose now that we are in case (b). Notice that in this case

$$h = \mathsf{update}_{T,r,V(T')}(h') = \mathsf{update}_{T,r,V(T')}(\mathsf{height}_{T',r'}) = \mathsf{height}_{T,r}.$$

It is easy to see that σ is an isomorphism between $\mathbf{R}$ and $(\bigcup_{w \in \mathrm{Im}(f)} G_v[\chi(w)], \beta(v), \rho)$. More-over, as $\hat{T} \in \mathsf{intr}^*(\hat{T}', I)$, it holds that $(T, r, f) \in (T', r', f') \diamond_{\mathsf{intr}} I$ and therefore (T, r) keeps the ancestor-descendant relations from (T', r') (item 4 of the $\diamond_{\mathsf{intr}}$ operation), while adding ancestor-descendant relations between the new node of T and the nodes labeled by I (item 5), and guaranteeing the connectivity of $G_v[\chi(T_w)]$ for each node $w \in V(T)$ (item 6). Hence, (χ, σ) witnesses that $\hat{T} \in \mathcal{A}$.

Conversely, let $\hat{T} = (T, r, h, \mathbf{R}, f) \in \mathcal{A}$ with witness pair (χ, σ). If $\chi(\chi^{-1}(x)) = \{x\}$, then let T' be the tree obtained from T by removing $u := \chi^{-1}(x)$ and adding edges between the parent of u and the children of u, if any. Otherwise, set $T' := T$. Let h' be the height function of T' and r' be the root of T'. Let (R', X', ϕ') be the boundaried graph obtained from $\mathbf{R}$ by removing $\sigma^{-1}(x)$. Let $f' := f|_{X'}$. Then the functions obtained from χ and σ after restricting their image to $V(G) \setminus x$ witness that $\hat{T}' = (T', r', h', (R', X', \phi'), f') \in \mathcal{A}'$. To show that $\hat{T} \in \mathsf{intr}^*(\hat{T}', I)$, the only non-trivial part is to prove that either $T_u \cap f(I) \neq \emptyset$, or $u \in \mathsf{Leaf}(T, r)$ and $\mathsf{Par}_{T,r}(u) \in f(I)$ (item 6 of the $\diamond_{\mathsf{intr}}$ operation). Suppose that $T_u \cap f(I) = \emptyset$. We know that x is exactly adjacent to $\rho^{-1}(I)$ in G_v since it is an introduce vertex. Moreover, (T, χ, r) is an $\mathcal{F}$-elimination tree of G_v, so $G_v[\chi(T_w)]$ is connected for all $w \in V(T)$. In particular, $G_v[\chi(T_u)]$ is connected, but $x \in G_v[\chi(T_u)]$ and $\rho^{-1}(I) \notin G_v[\chi(T_u)]$ because $T_u \cap f(I) = \emptyset$. Thus, we must have $G_v[\chi(T_u)] = \{x\}$, and so $u \in \mathsf{Leaf}(T, r)$. Since $G_v[\chi(T_{\mathsf{Par}_{T,r}(u)})]$ is also connected, it implies that $\chi(\mathsf{Par}_{T,r}(u))$ and x are connected, so $\mathsf{Par}_{T,r}(u) \in f(I)$. Hence, $\hat{T} \in \mathsf{intr}^*(\mathcal{A}', I)$. ∎

Introduce procedure. Given an annotated tree $\hat{T}$, a set I of labels, and a positive integer k, the *introduce procedure*, denoted by $\mathsf{intr}_k(\hat{T}, I)$, outputs $\mathsf{filter}_k \circ \mathsf{rep} \circ \mathsf{intr}^*(\hat{T}, I)$. Examples of the introduce procedure can be found in Figure 7. Note that we do not apply the crop operation, since the complete introduction procedure applied to a cropped annotated tree outputs a cropped annotated tree. Given a set $\mathcal{A}$ of annotated trees, $\mathsf{intr}_k(\mathcal{A}, I)$ outputs $\bigcup_{\hat{T} \in \mathcal{A}} \mathsf{intr}_k(\hat{T}, I)$.

LEMMA 7.10. *Let G be a graph, $(\mathsf{T}, \beta, \mathsf{r})$ be a nice tree decomposition of G, v be an introduce node of T with child v' and introduced vertex x, $\rho' : \beta(v) \to [|\beta(v)|]$ be a bijection, $\rho := \rho' \oplus [x \mapsto |\beta(v)|]$, and $I := \rho(N_{G_v}(x))$. Let k be an integer, $\mathcal{A} := \mathsf{char}_k(G_v, \beta(v), \rho)$, and $\mathcal{A}' := \mathsf{char}_k(G_{v'}, \beta(v'), \rho')$. Then $\mathcal{A} = \mathsf{intr}_k(\mathcal{A}', I)$.*

PROOF. Let $\mathcal{D} := \mathsf{char}^*(G_v, \beta(v), \rho)$ and $\mathcal{D}' := \mathsf{char}^*(G_{v'}, \beta(v'), \rho')$. According to Lemma 7.9, $\mathcal{D} = \mathsf{intr}^*(\mathcal{D}', I)$.

The crop operation acts on the tree T of an annotated tree $(T, r, h, \mathbf{R}, f)$ to remove the leaves that are not in $\mathsf{Im}(f)$. By item 6 of the $\diamond_{\mathsf{intr}}$ operation, the complete introduce procedure may only add a new node in T above a node in $\mathsf{Im}(f)$, or add a leaf to T with a parent node in $\mathsf{Im}(f)$.
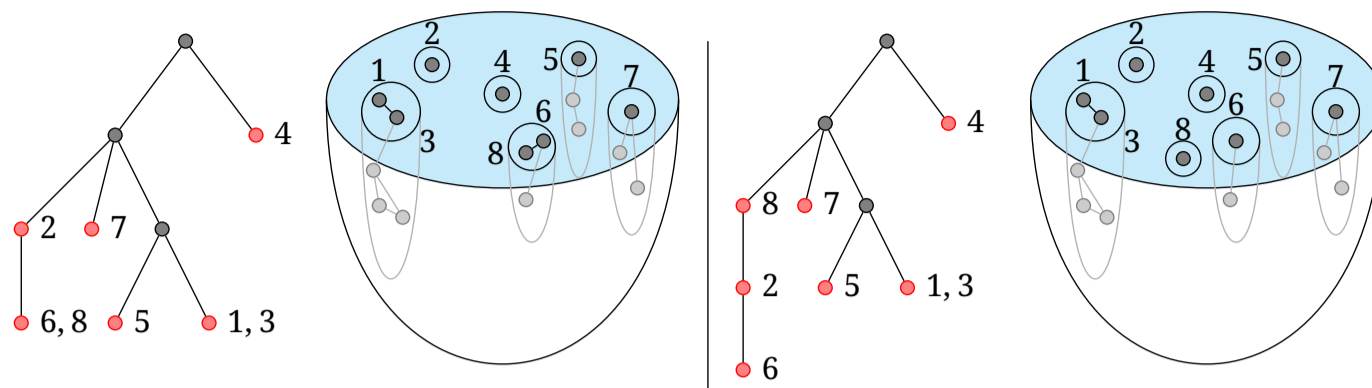
Figure 7. Two annotated trees obtained from the introduce procedure applied to the annotated tree of Figure 4 with $I = \{2, 6\}$.

Hence, the new node may only be added to the cropped tree. Since the new vertex is also in the boundary, it implies that, given a set C of annotated trees, $\mathsf{crop} \circ \mathsf{intr}^*(C, I) = \mathsf{intr}^*(\mathsf{crop}(C), I)$.

The representation operation acts on $\mathbf{R}$ to replace it by a boundaried graph whose connected components are representatives of the connected components of $\mathbf{R}$. The complete introduce procedure adds a new vertex in the boundary of $\mathbf{R}$ that is adjacent to boundary vertices only. Hence, $\mathsf{rep} \circ \mathsf{intr}^*(C, I) = \mathsf{rep} \circ \mathsf{intr}^*(\mathsf{rep}(C), I)$.

Moreover, the complete introduce procedure can only increase the height of an annotated tree. Therefore, $\mathsf{filter}_k \circ \mathsf{intr}^*(C, I) = \mathsf{filter}_k \circ \mathsf{intr}^*(\mathsf{filter}_k(C), I)$.

Since the three operations are commutative, we have that $\mathcal{M}_k \circ \mathsf{intr}^*(C, I) = \mathsf{filter}_k \circ \mathsf{rep} \circ \mathsf{intr}^*(\mathcal{M}_k(C), I) = \mathsf{intr}_k(\mathcal{M}_k(C), I)$. Therefore, $\mathcal{A} = \mathcal{M}_k(\mathcal{D}) = \mathcal{M}_k \circ \mathsf{intr}^*(\mathcal{D}', I) = \mathsf{intr}_k(\mathcal{M}_k(\mathcal{D}'), I) = \mathsf{intr}_k(\mathcal{A}', I)$. ∎

For the introduce procedure, we finally prove that it can generate a bounded number of annotated trees.

LEMMA 7.11. *Let w and k be two positive integers, let $\mathbf{G}$ be a w-boundaried graph, let $I \subseteq [w]$, and let $\hat{T} \in \mathsf{char}_k(\mathbf{G})$. Then $|\mathsf{intr}_k(\hat{T}, I)| = O(w \cdot k)$.*

PROOF. Let $\hat{T} = (T, r, h, \mathbf{R}, f)$. Let us show that $|(T, r, f) \diamond_{\mathsf{intr}} I| = O(w \cdot k)$. Since $\mathsf{Leaf}(T, r) \subseteq \mathsf{Im}(f)$, T has at most w leaves. Thus, T has at most $w \cdot (k + 1)$ nodes. The $\diamond_{\mathsf{intr}}$ operation consists in adding a new node to T, either as the new parent of a node, or as a new leaf of T. Therefore, $|(T, r, f) \diamond_{\mathsf{intr}} I| \leq 2w \cdot (k + 1)$.

Let $(T', r', f') \in (T, r, f) \diamond_{\mathsf{intr}} I$. In the introduction procedure, we obtain at most two annotated trees from (T', r', f'). Therefore, $|\mathsf{intr}_k(\hat{T}, I)| \leq 4w \cdot (k + 1)$. ∎

7.4.3   Join procedure

With the join procedure, given the characteristic of two boundaried graphs $\mathbf{G}_1$ and $\mathbf{G}_2$ that are compatible, we want to compute the characteristic of $\mathbf{G}_1 \oplus \mathbf{G}_2$.

Diamond-join operation. Let (T_1, r_1) and (T_2, r_2) be two rooted trees, w be an integer, and $f_1 : [w] \to V(T_1)$ and $f_2 : [w] \to V(T_2)$ be two functions such that $\{f_1^{-1}(u) \mid u \in \mathrm{Im}(f_1)\} = \{f_2^{-1}(u) \mid u \in \mathrm{Im}(f_2)\}$. Thanks to this equality, we can identify f_1 with f_2 and say that $V(T_1) \cap V(T_2) = \mathrm{Im}(f_1)$. We define $(T_1, r_1, f_1) \diamond_{\mathrm{join}} (T_2, r_2, f_2)$ as the set of all pairs (T, r, f) such that:

1. $f = f_1 = f_2$,
2. (T, r) is a rooted tree,
3. $V(T) = V(T_1) \cup V(T_2)$,
4. for $i \in \{1, 2\}$, if $u \in V(T_i)$ and $v \in \mathrm{Anc}_{T_i, r_i}(u)$, then $v \in \mathrm{Anc}_{T,r}(u)$, and
5. for every $v \in \mathrm{Leaf}(T, r)$ and every $w \in \mathrm{Anc}_{T,r}(v)$, if $V(vTw) \cap \mathrm{Im}(f) = \emptyset$, then there is an $i \in \{1, 2\}$ such that $vTw = vT_iw$.

The last item, which states that a branch of T that does not intersect $\mathrm{Im}(f)$ either belongs to T_1 or T_2, is a property needed to ensure connectivity and allows the application of the crop operation in Lemma 7.12 and Lemma 7.13.

Let (T, r) be a rooted tree, and for $i \in \{1, 2\}$, let $K_i \subseteq V(T)$ and $h_i : K_i \to \mathbb{N}$. $\mathrm{update}_{T,r,K_1,K_2}(h_1, h_2)$ is the function that maps $v \in V(T)$ to the maximum of

$$h_1(v), h_2(v), \text{ and } \max_{c \in \mathrm{Ch}(v)}\{1 + \mathrm{update}_{T,r,K_1,K_2}(h_1, h_2)(c)\},$$

when they are defined. For $i \in \{1, 2\}$, let (T_i, r_i) be a rooted tree with $V(T_i) = K_i$ and such that the ancestor-descendant relationship between the nodes of K_i is the same in T and T_i. Then we can observe that $\mathrm{update}_{T,r,K_1,K_2}(\mathrm{height}_{T_1,r_1}, \mathrm{height}_{T_2,r_2}) = \mathrm{height}_{T,r}$.

Complete join procedure. The join procedure corresponds to "merging" two annotated trees whose intersection is exactly their boundary. More formally, given two annotated trees $\hat{T}_1 = (T_1, r_1, h_1, \mathbf{R}_1, f_1)$ and $\hat{T}_2 = (T_2, r_2, h_2, \mathbf{R}_2, f_2)$, the *complete join procedure*, denoted by $\mathrm{join}(\hat{T}_1, \hat{T}_2)$, outputs a set $\mathcal{A}$ of annotated trees constructed as follows. Initially, $\mathcal{A}$ is empty. If $\mathbf{R}_1$ and $\mathbf{R}_2$ are compatible (i.e., such that $\{f_1^{-1}(u) \mid u \in \mathrm{Im}(f_1)\} = \{f_2^{-1}(u) \mid u \in \mathrm{Im}(f_2)\}$) and such that $\mathrm{Ker}(h_1 \circ f_1) = \mathrm{Ker}(h_2 \circ f_2)$, then for $(T, r, f) \in (T_1, r_1, f_1) \diamond_{\mathrm{join}} (T_2, r_2, f_2)$, let $h := \mathrm{update}_{T,r,V(T_1),V(T_2)}(h_1, h_2)$. Let $(R, X, \rho) := \mathbf{R}_1 \oplus \mathbf{R}_2$. If each connected component of R belongs to $\mathrm{exc}(\mathcal{F})$, then we add $(T, r, h, \mathbf{R}_1 \oplus \mathbf{R}_2, f)$ to $\mathcal{A}$. Given two sets $\mathcal{A}_1$ and $\mathcal{A}_2$ of annotated trees, we set $\mathrm{join}^*(\mathcal{A}_1, \mathcal{A}_2) := \bigcup_{\hat{T}_1 \in \mathcal{A}_1, \hat{T}_2 \in \mathcal{A}_2} \mathrm{join}^*(\hat{T}_1, \hat{T}_2)$.

LEMMA 7.12. *Let G be a graph, (T, β, r) be a nice tree decomposition of G, v be a join node of T with children v_1 and v_2, and $\rho : \beta(v) \to [|\beta(v)|]$ be a bijection. Let $\mathcal{A} := \mathrm{char}^*(G_v, \beta(v), \rho)$, $\mathcal{A}_1 := \mathrm{char}^*(G_{v_1}, \beta(v_1), \rho)$, and $\mathcal{A}_2 := \mathrm{char}^*(G_{v_2}, \beta(v_2), \rho)$. Then $\mathcal{A} = \mathrm{join}^*(\mathcal{A}_1, \mathcal{A}_2)$.*

PROOF. Let $\hat{T} = (T, r, h, (R, X, \phi), f) \in \mathsf{join}^*(\mathcal{A}_1, \mathcal{A}_2)$. There is $\hat{T}_1 = (T_1, r_1, h_1, \mathbf{R}_1, f_1) \in \mathcal{A}_1$ and $\hat{T}_2 = (T_2, r_2, h_2, \mathbf{R}_2, f_2) \in \mathcal{A}_2$ such that $\hat{T} \in \mathsf{join}^*(\hat{T}_1, \hat{T}_2)$ with witness pair (χ_1, σ_1) and (χ_2, σ_2), respectively, such that $\chi_{1|\mathsf{Im}(f)} = \chi_{2|\mathsf{Im}(f)}$ and $\sigma_{1|X} = \sigma_{2|X}$. Note that

$$h := \mathsf{update}_{T,r,V(T_1),V(T_2)}(h_1, h_2) = \mathsf{update}_{T,r,V(T_1),V(T_2)}(\mathsf{height}_{T_1,r_1}, \mathsf{height}_{T_2,r_2}) = \mathsf{height}_{T,r}.$$

Let $(\chi, \sigma) = (\chi_1 \cup \chi_2, \sigma_1 \cup \sigma_2)$.

It is easy to see that σ is an isomorphism between $\mathbf{R}$ and $(\bigcup_{w \in \mathsf{Im}(f)} G[\chi(w)], \beta(v), \rho)$. If $uv \in E(G_v)$, since $G_{v_1} \setminus \beta(v)$ and $G_{v_2} \setminus \beta(v)$ are not connected, then there is $i \in \{1, 2\}$ such that $\chi_i(u) \in \mathsf{Anc}_{T,r}(\chi_i(v)) \cup \mathsf{Desc}_{T,r}(\chi_i(v))$ holds, so $\chi(u) \in \mathsf{Anc}_{T,r}(\chi(v)) \cup \mathsf{Desc}_{T,r}(\chi(v))$ due to item 4 of the $\diamond_{\mathsf{join}}$ operation.

Moreover, item 5 of the $\diamond_{\mathsf{join}}$ operation ensures the connectivity of $G_v[\chi(T_w)]$ for each $w \in V(T)$. Indeed, let $i \in \{1, 2\}$ be such that $w \in V(T_i)$. Suppose towards a contradiction that $G_v[\chi(T_w)]$ is not connected. Note that it implies that $w \notin \mathsf{Im}(f)$, because $\mathsf{Im}(f) \subseteq V(T_1) \cup V(T_2)$ and (T_1, χ_1, r_1) and (T_2, χ_2, r_2) are $\mathcal{F}$-elimination trees of G_{v_1} and G_{v_2}, respectively, so $G_{v_1}[\chi_1((T_1)_w)]$ and $G_{v_2}[\chi_2((T_2)_w)]$ are connected and therefore $G_v[\chi(T_w)]$ would be connected. We assume that w is a minimal node such that $G_v[\chi(T_w)]$ is not connected, i.e., for every $u \in V(T_w) \setminus \{w\}$, $G_v[\chi(T_u)]$ is connected. Hence, there is $u \in \mathsf{Ch}_{T,r}(w)$ such that $\chi(w)$ is not connected to $G_v[\chi(T_u)]$. So $V(T_u) \cup V(T_i) = \emptyset$, since otherwise, the connectivity of $G_{v_i}[\chi_i((T_i)_w)]$ would imply the connectivity of $\chi(w)$ with $G_v[\chi(T_u)]$. Thus, there is an $x \in \mathsf{Leaf}(T, r)$ such that $u \in \mathsf{Anc}_{T,r}(x)$, and therefore $w \in \mathsf{Anc}_{T,r}(x)$, and $V(xTw) \cap \mathsf{Im}(f) = \emptyset$. So, according to item 5 of the $\diamond_{\mathsf{join}}$ operation, there is $i \in \{1, 2\}$ such that $xTw = xT_iw$. This contradicts the fact that $w \in V(T_i) \setminus \mathsf{Im}(f)$ and $V(T_u) \subseteq V(T_j) \setminus \mathsf{Im}(f)$ where $\{i, j\} = \{1, 2\}$. Thus, (T, χ, r) is an $\mathcal{F}$-elimination tree of G_v. Therefore, (χ, σ) witnesses that $\hat{T} \in \mathcal{A}$.

Conversely, let $\hat{T} \in \mathcal{A}$ with witness pair (χ, σ). Let (χ_1, σ_1) and (χ_2, σ_2) be the co-restrictions of (χ, σ) to G_{v_1} and G_{v_2}, respectively.

Let T_1 be the tree obtained from T by removing the nodes not in $\mathsf{Im}(\chi_1)$ and adding edges between the parent and children of each removed node. Let r_1 be the root of T_1 and $h_1 := \mathsf{height}_{T_1,r_1}$. Let $\mathbf{R}_1$ be obtained from $\mathbf{R}$ by removing the vertices not in $\mathsf{Im}(\sigma_1)$. Then it is easy to see that $\hat{T}_1 = (T_1, r_1, h_1, \mathbf{R}_1, f)$ belongs to B_1 with witness pair (χ_1, σ_1). We construct similarly $\hat{T}_2 = (T_2, r_2, h_2, \mathbf{R}_2, f) \in B_2$ with witness pair (χ_2, σ_2).

Moreover, we claim that $\hat{T} \in \mathsf{join}^*(\hat{T}_1, \hat{T}_2)$. To prove this claim, the less trivial part is to show that T respects item 5 of the $\diamond_{\mathsf{join}}$ operation. Let $x \in \mathsf{Leaf}(T, r)$ and $w \in \mathsf{Anc}_{T,r}(x)$ such that $V(xTw) \cap \mathsf{Im}(f) = \emptyset$. Suppose towards a contradiction that $V(xTw) \cap V(T_1) \neq \emptyset$ and $V(xTw) \cap V(T_2) \neq \emptyset$. Thus, there exist $u_1 \in V(xTw) \cap V(T_1) \setminus \mathsf{Im}(f)$ and $u_2 \in V(xTw) \cap V(T_2) \setminus \mathsf{Im}(f)$. Without loss of generality, suppose that $x \in V(T_1)$. We take such a node u_2 that is closest to x and u_1 to be the node just before u_2 in the path from x to u_2. Since $V(xTu_1) \subseteq V(T_1) \setminus \mathsf{Im}(f)$ and $u_2 \in V(T_2) \setminus \mathsf{Im}(f)$, $\chi(u_2)$ and $\chi(xTu_1)$ are not connected. This contradicts the fact that $G_v[\chi(T_{u_2})]$ is connected. Therefore, $\hat{T} \in \mathsf{join}^*(\hat{T}_1, \hat{T}_2)$, which implies that $\hat{T} \in \mathsf{join}^*(\mathcal{A}_1, \mathcal{A}_2)$. $\blacksquare$

Join procedure. Given two annotated trees $\hat{T}_1$ and $\hat{T}_2$, the *join procedure*, denoted by $\mathrm{join}(\hat{T}_1, \hat{T}_2)$, outputs $\mathrm{filter}_k \circ \mathrm{rep} \circ \mathrm{join}^*(\hat{T}_1, \hat{T}_2)$. See Figure 8 for an example. Note that we do not apply the crop operation since joining two cropped annotated trees gives a cropped annotated tree. Given two sets $\mathcal{A}_1$ and $\mathcal{A}_2$ of annotated trees, $\mathrm{join}(\mathcal{A}_1, \mathcal{A}_2)$ outputs $\bigcup_{\hat{T}_1 \in \mathcal{A}_1, \hat{T}_2 \in \mathcal{A}_2} \mathrm{join}(\hat{T}_1, \hat{T}_2)$.
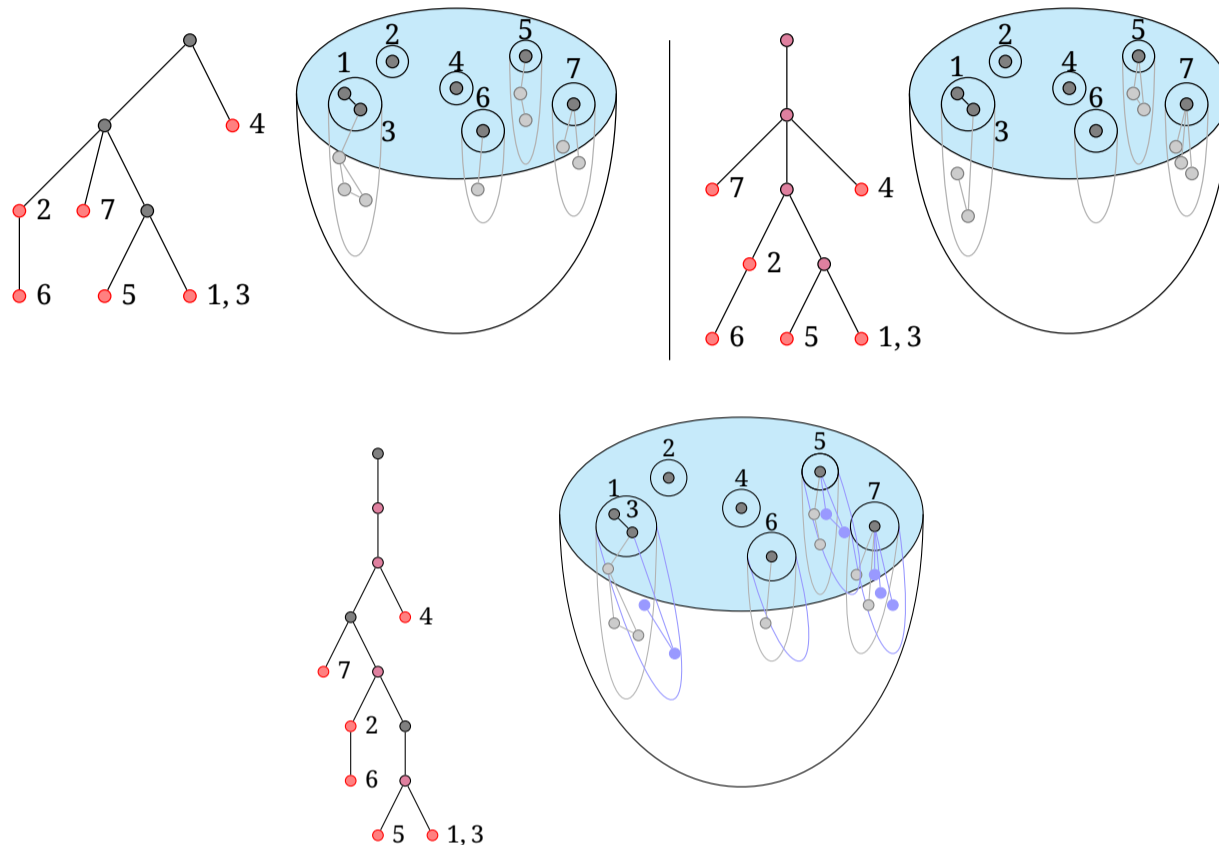

Figure 8. An annotated tree (below) obtained from the join procedure applied to the two annotated trees above.

LEMMA 7.13. *Let G be a graph, $(\mathsf{T}, \beta, \mathsf{r})$ be a nice tree decomposition of G, v be a join node of T with children v_1 and v_2, and $\rho : \beta(v) \to [|\beta(v)|]$ be a bijection. Let k be an integer, $\mathcal{A} := \mathrm{char}_k(G_v, \beta(v), \rho)$, $\mathcal{A}_1 := \mathrm{char}_k(G_{v_1}, \beta(v_1), \rho)$, and $\mathcal{A}_2 := \mathrm{char}_k(G_{v_2}, \beta(v_2), \rho)$. Then $\mathcal{A} = \mathrm{join}_k(\mathcal{A}_1, \mathcal{A}_2)$.*

PROOF. Let $\mathcal{B} := \mathrm{char}^*(G_v, \beta(v), \rho)$, $\mathcal{B}_1 := \mathrm{char}^*(G_{v_1}, \beta(v_1), \rho)$, and $\mathcal{B}_2 := \mathrm{char}^*(G_{v_2}, \beta(v_2), \rho)$. According to Lemma 7.12, $\mathcal{B} = \mathrm{join}^*(\mathcal{B}_1, \mathcal{B}_2)$.

Let $\hat{T} = (T, r, h, \mathbf{R}, f) \in \mathcal{B}$, $\hat{T}_1 = (T_1, r_1, h_1, \mathbf{R}_1, f_1) \in \mathcal{B}_1$, and $\hat{T}_2 = (T_2, r_2, h_2, \mathbf{R}_2, f_2) \in \mathcal{B}_2$, such that $\hat{T} \in \mathrm{join}^*(\hat{T}_1, \hat{T}_2)$. The complete join procedure joins $\mathbf{R}_1$ and $\mathbf{R}_2$ to obtain $\mathbf{R}$, without modifying their boundary nor deleting vertices or edges, so given two sets of annotated trees C_1 and C_2, $\mathrm{rep} \circ \mathrm{join}^*(C_1, C_2) = \mathrm{rep} \circ \mathrm{join}^*(\mathrm{rep}(C_1), \mathrm{rep}(C_2))$.

The procedure can only increase the height of annotated trees, so $\mathrm{filter}_k \circ \mathrm{join}^*(C_1, C_2) = \mathrm{filter}_k \circ \mathrm{join}^*(\mathrm{filter}_k(C_1), \mathrm{filter}_k(C_2))$.

Moreover, item 6 of the $\diamond_{\mathsf{join}}$ operation implies that, if for $w \in V(T)$, $V(T_w) \cap \mathrm{Im}(f) = \emptyset$, then there is $i \in \{1, 2\}$ such that $T_w = (T_i)_w$. Therefore, each cropped subtree of T is exactly a cropped subtree of T_1 or a cropped subtree of T_2. Thus, $\mathsf{crop} \circ \mathsf{join}^*(C_1, C_2) = \mathsf{join}^*(\mathsf{crop}(C_1), \mathsf{crop}(C_2))$.

Since these three operations are commutative, we have $\mathcal{A} = \mathcal{M}_k(\mathcal{B}) = \mathsf{filter}_k \circ \mathsf{rep} \circ \mathsf{join}^*(\mathcal{M}_k(\mathcal{B}_1), \mathcal{M}_k(\mathcal{B}_2)) = \mathsf{join}_k(\mathcal{A}_1, \mathcal{A}_2)$. ∎

For the join procedure, we finally prove that it can generate a bounded number of annotated trees.

LEMMA 7.14. *Let w and k be two positive integers, let $\mathbf{G}_1$ and $\mathbf{G}_2$ be two compatible w-boundaried graphs, let $\hat{T}_1 \in \mathsf{char}_k(\mathbf{G}_1)$, and let $\hat{T}_2 \in \mathsf{char}_k(\mathbf{G}_2)$. Then $|\mathsf{join}_k(\hat{T}_1, \hat{T}_2)| = 2^{O(w \cdot k)}$.*

PROOF. Let $\hat{T}_1 = (T_1, r_1, h_1, \mathbf{R}_1, f_1)$ and $\hat{T}_2 = (T_2, r_2, h_2, \mathbf{R}_2, f_2)$. We will first show that

$$|(T_1, r_1, f_1) \diamond_{\mathsf{join}} (T_2, r_2, f_2)| = 2^{O(w \cdot k)}.$$

Let $(T, r, f) \in (T_1, r_1, f_1) \diamond_{\mathsf{join}} (T_2, r_2, f_2)$. Notice that (T, r) has at most w leaves and height at most k. Each leaf of (T, r) is in $\mathrm{Im}(f) = \mathrm{Im}(f_1) = \mathrm{Im}(f_2)$, so it corresponds to both a leaf of (T_1, r_1) and a leaf of (T_2, r_2). Also note that T is obtained by choosing, for each path from a leaf v to r, a subset that corresponds to the path vT_1r (the rest is the path vT_2r). There are at most w such paths from a leaf to r, and each of them has length at most $k + 1$. So $|(T_1, r_1, f_1) \diamond_{\mathsf{join}} (T_2, r_2, f_2)| \leq 2^{w(k+1)}$. Then, to construct an annotated tree $(T, r, h, \mathbf{R}, f)$, the function h and the boundaried graph $\mathbf{R}$ are totally determined by T, h_1, h_2, $\mathbf{R}_1$, and $\mathbf{R}_2$. So we obtain the desired result. ∎

7.5 The algorithm

We finally present a recursive algorithm (Algorithm 1) that computes the elimination distance to $\mathsf{exc}(\mathcal{F})$ of a graph of bounded treewidth and proves Theorem 7.1. More precisely, given a boundary graph $\mathbf{G}$, a nice tree decomposition, and an integer k, the algorithm outputs $\mathsf{char}_k(\mathbf{G})$.

Note that using backtracking in Algorithm 1, we can easily construct an annotated tree of minimum height in $\mathsf{char}^*(\mathbf{G})$ as well as its witness pair. In other words, given a connected graph G such that $\mathsf{ed}_{\mathsf{exc}(\mathcal{F})}(G) \leq k$, we can construct an $\mathcal{F}$-elimination tree of G of height $\mathsf{ed}_{\mathsf{exc}(\mathcal{F})}(G)$ using Algorithm 1.

LEMMA 7.15. *Given a connected graph G, an integer k, a nice tree decomposition $\mathcal{T} = (\mathsf{T}, \beta, \mathsf{r})$ of G of width w, a bijection $\rho : \beta(\mathsf{r}) \to [|\beta(\mathsf{r})|]$, and $v \in V(\mathsf{T})$, $\mathsf{recEd}((G, \beta(v), \rho), \mathcal{T}, k, v)$ outputs $\mathsf{char}_k(G_v, \beta(v), \rho)$. Moreover, $\mathsf{recEd}((G, \beta(\mathsf{r}), \rho), \mathcal{T}, k, \mathsf{r})$ outputs $\mathsf{char}_k(G, \beta(\mathsf{r}), \rho)$ in time $2^{O_{\ell_{\mathcal{F}}}(w \cdot k + w \log w)} \cdot n$.*

PROOF. We prove that for every $v \in V(\mathsf{T})$, by induction on the height of T_v, the algorithm recEd of Algorithm 1 with input $((G, \beta(v), \rho), \mathcal{T}, k, v)$ returns $\mathsf{char}_k(G_v, \beta(v), \rho)$. Indeed, if v is a leaf, then $\mathsf{char}_k(G_v, \beta(v), \rho) = \{\hat{\mathbb{1}}\} = \mathsf{recEd}((G, \beta(v), \rho), \mathcal{T}, k, v)$. Otherwise, v is either a forget

Input: A connected boundaried graph $\mathbf{G} = (G, X, \rho)$, a nice tree
decomposition $\mathcal{T} = (T, \beta, r)$ of G, an integer k, and a node
$v \in V(T)$ such that $X = \beta(v)$.

Output: The characteristic $\text{char}_k(\mathbf{G})$ of $\mathbf{G}$.

1:    $\mathcal{A} \leftarrow \emptyset$

2:    $w \leftarrow |\beta(v)|$

3:    **if** v is a leaf **then**

4:        $\mathcal{A} \leftarrow \{\mathbb{1}\}$

5:    **else if** v is a forget node with child v' and forgotten vertex x
then

6:        $\rho' \leftarrow \rho \oplus [x \mapsto w + 1]$

7:        $\mathcal{A}' \leftarrow \text{recEd}((G, \beta(v'), \rho'), \mathcal{T}, k, v')$

8:        $\mathcal{A} \leftarrow \text{forget}(\mathcal{A}')$

9:    **else if** v is an introduce node with child v' and introduced vertex
x **then**

10:        $\tau \leftarrow (x \leftrightarrow \rho^{-1}(w))$

11:        $\rho' \leftarrow \rho \circ \tau$

12:        $\mathcal{A}' \leftarrow \text{recEd}((G, \beta(v'), \rho'|_{\beta(v')}), \mathcal{T}, k, v')$

13:        $N \leftarrow N_{G[\beta(v)]}(x)$

14:        $\mathcal{A} \leftarrow \tau(\text{intr}_k(\mathcal{A}', \rho'(N)))$

15:    **else if** v is a join node with children v_1 and v_2 **then**

16:        $\mathcal{A}_1 \leftarrow \text{recEd}((G, \beta(v), \rho), \mathcal{T}, k, v_1)$

17:        $\mathcal{A}_2 \leftarrow \text{recEd}((G, \beta(v), \rho), \mathcal{T}, k, v_2)$

18:        $\mathcal{A} \leftarrow \text{join}_k(\mathcal{A}_1, \mathcal{A}_2)$

19:    **return** $\mathcal{A}$

Algorithm 1. $\text{recEd}(\mathbf{G}, \mathcal{T}, k, v)$

node, or an introduce node, or a join node, and the correctness of the algorithm is implied from the induction hypothesis and Lemma 7.8, Lemma 7.10, and Lemma 7.13, respectively. Finally, since $G = G_r$, we have $\text{recEd}((G, \beta(r), \rho), \mathcal{T}, k, r) = \text{char}_k(G, \beta(r), \rho)$.

We now analyze the running time. A nice tree decomposition of width w constructed by Proposition 4.7 has $O(w \cdot n)$ bags, hence the linear dependence follows.

Let us first analyze the join procedure. During this procedure, we recursively obtain two characteristics of size $2^{O_{\ell_{\mathcal{F}}}(w \cdot k + w \log w)}$ according to Lemma 7.6. Each pair of annotated trees can be joined in $2^{O(w \cdot k)}$ ways, according to Lemma 7.14. Let $\mathbf{R}_1$ and $\mathbf{R}_2$ be the boundaried

graphs of such a pair of annotated trees. There is an integer $z \leq w$ such that they belong to $\mathcal{R}^z_{\ell_{\mathcal{F}}}$. By Proposition 4.8, $\mathbf{R}_1$ and $\mathbf{R}_2$ have size $O_{\ell_{\mathcal{F}}}(z)$. So $\mathbf{R} := \mathbf{R}_1 \oplus \mathbf{R}_2$ has size $O_{\ell_{\mathcal{F}}}(z)$ as well. The representation operation applied to $\mathbf{R}$ during the join operation for those two annotated trees finds the representative of $l \leq z$ boundaried graphs of respective boundaries of sizes $z_1, ..., z_l$ with $\sum_{i=1}^{l} z_i \leq z$ and with $O_{\ell_{\mathcal{F}}}(z_i)$ vertices in the underlying graph due to Proposition 4.8. So by Lemma 4.10, the representation operation in the join procedure takes time $\sum_{i=1}^{l} 2^{O_{\ell_{\mathcal{F}}}(w_i \log w_i)} = 2^{O_{\ell_{\mathcal{F}}}(w \log w)}$. Checking that this is an annotated tree, that its height is at most d, and that we did not already create it also takes time $2^{O_{\ell_{\mathcal{F}}}(w \cdot k + w \log w)}$. Hence, the total running time of the join procedure is $2^{O_{\ell_{\mathcal{F}}}(w \cdot k + w \log w)}$.

It is easy to see that the forget procedure applied to an annotated tree creates at most one annotated tree. Moreover, the introduce procedure applied to an annotated tree creates $O(w \cdot k)$ annotated trees according to Lemma 7.11. Similarly, the representation, crop, and filter operations in these procedures take time $2^{O_{\ell_{\mathcal{F}}}(w \cdot k + w \log w)}$. So the lemma follows. ∎

We can finally prove Theorem 7.1.

PROOF OF THEOREM 7.1. If $\mathcal{F} = \{K_1\}$, the algorithm of Proposition 7.4 outputs the desired result in time $2^{O(\text{tw} \cdot k)} \cdot n$. So let us assume that $\mathcal{F}$ is non-trivial.

Suppose first that G is connected. By Proposition 4.3 and Proposition 4.7, we can obtain a nice tree decomposition $\mathcal{T} = (\mathsf{T}, \beta, r)$ of width $2\text{tw} + 1$ in time $2^{O(\text{tw})} \cdot n$. Let $\rho : \beta(\mathsf{r}) \to [|\beta(\mathsf{r})|]$ be an arbitrary ordering on the vertices of $\beta(\mathsf{r})$. We apply recEd with input $((G, \beta(\mathsf{r}), \rho), \mathcal{T}, k, \mathsf{r})$. By Lemma 7.15, this gives the desired result in time $2^{O_{\ell_{\mathcal{F}}}(\text{tw}(k + \log \text{tw}))} \cdot n$.

If G is not connected, we apply the same procedure on each connected component. The running time is the same as in the above case. ∎

7.6   Exchangeability of boundaried graphs with the same characteristic

We give here a simple technical lemma on characteristics that will be used in Section 10. We show that boundaried graphs with the same characteristic can be exchanged, i.e., give graphs of the same elimination distance to $\mathcal{F}$ when "glued" to the same boundaried graph.

Given a positive integer k and a (possibly disconnected) boundaried graph $\mathbf{G}$, we define $\text{char}_k(\mathbf{G})$ as $(\text{char}_k(\mathbf{C}))_{\mathbf{C} \in \text{cc}(\mathbf{G})}$. Note that we still have $|\text{char}_k(\mathbf{G})| = 2^{O_{\ell_{\mathcal{F}}}(w \cdot k + w \log w)}$. Therefore, we can extend f_{14} so that $|\text{char}_k(\mathbf{G})| \leq f_{14}(w, k)$ with $f_{14}(w, k) = 2^{O_{\ell_{\mathcal{F}}}(w \cdot k + w \log w)}$ for any boundaried graph $\mathbf{G}$.

LEMMA 7.16. *Let $\mathbf{G}$, $\mathbf{G}'$, and $\mathbf{G}''$ be three compatible boundaried graphs and let k be an integer such that $\text{ed}_{\text{exc}(\mathcal{F})}(\mathbf{G} \oplus \mathbf{G}'') \leq k$ and $\text{char}_k(\mathbf{G}) = \text{char}_k(\mathbf{G}')$. Then $\text{ed}_{\text{exc}(\mathcal{F})}(\mathbf{G} \oplus \mathbf{G}'') = \text{ed}_{\text{exc}(\mathcal{F})}(\mathbf{G}' \oplus \mathbf{G}'')$.*

PROOF. We suppose without loss of generality that $\mathbf{G} \oplus \mathbf{G}''$, and therefore $\mathbf{G}' \oplus \mathbf{G}''$ as well, is connected. Indeed, if this is not the case, we may apply the following proof to each one of the connected components separately.

Let $cc(\mathbf{G}) = \{\mathbf{C}_1, ..., \mathbf{C}_l\}$ and $cc(\mathbf{G}') = \{\mathbf{C}'_1, ..., \mathbf{C}'_l\}$, such that $\mathsf{char}_k(\mathbf{C}_i) = \mathsf{char}_k(\mathbf{C}'_i)$ for $i \in [l]$. Let $i \in [l]$. We write $\mathbf{C}_i = (C_i, B_i, \rho_i)$. Let $\mathcal{T}_i = (\mathsf{T}_i, \beta_i, \mathsf{r}_i)$ be a nice tree decomposition of $\mathbf{C}_i$, i.e., such that $\beta_i(\mathsf{r}_i) = B_i$. Since the B_i's are pairwise disjoint, there is a rooted tree decomposition $\mathcal{T}^* = (\mathsf{T}^*, \beta^*, \mathsf{r})$ of G'', where G'' is the underlying graph of $\mathbf{G}''$, such that, for $i \in [l]$, there is $v_i \in \mathsf{Leaf}(\mathsf{T}^*, \mathsf{r})$ with $\beta^*(v_i) = B_i$. Let $\mathcal{T} = (\mathsf{T}, \beta, \mathsf{r})$ be the tree decomposition obtained from $\mathcal{T}^*$ and the $\mathcal{T}_i$'s by identifying v_i with r_i for $i \in [l]$ and adding nodes in $\mathcal{T}^*$ using Proposition 4.7, so that $\mathcal{T}$ is a nice tree decomposition of $\mathbf{G} \oplus \mathbf{G}''$.

Let $\mathcal{T}' = (\mathsf{T}', \beta', \mathsf{r})$ be a nice tree decomposition of the graph $\mathbf{G}' \oplus \mathbf{G}''$ obtained from $\mathcal{T}$ by replacing $\mathcal{T}_i$ by a nice tree decomposition $\mathcal{T}'_i = (\mathsf{T}'_i, \beta'_i, \mathsf{r}'_i)$ of $\mathbf{C}'_i$ for $i \in [l]$. Observe that, for every $i \in [l]$, according to Lemma 7.15, $\mathsf{recEd}((\mathbf{G} \oplus \mathbf{G}'', \beta(v_i), \rho_{v_i}), \mathcal{T}, k, v_i) = \mathsf{char}_k(\mathbf{G} \oplus \mathbf{G}'', \beta(v_i), \rho) = \mathsf{char}_k(\mathbf{C}_i)$. Similarly, $\mathsf{recEd}((\mathbf{G}' \oplus \mathbf{G}'', \beta'(v_i), \rho_{v_i}), \mathcal{T}', k, v_i) = \mathsf{char}_k(\mathbf{C}'_i)$. Thus, Algorithm 1 applied with input $((\mathbf{G} \oplus \mathbf{G}'', \beta(v_i), \rho_{v_i}), \mathcal{T}, k, v_i)$ and $((\mathbf{G}' \oplus \mathbf{G}'', \beta'(v_i), \rho_{v_i}), \mathcal{T}', k, v_i)$ outputs the same result.

We next set $U := V(\mathsf{T} \setminus \bigcup_{i \in [l]} V(\mathsf{T}_i))$. Note that $U = V(\mathsf{T}' \setminus \bigcup_{i \in [l]} V(\mathsf{T}'_i))$. Therefore, in each node u of U, Algorithm 1 applied with input $((\mathbf{G} \oplus \mathbf{G}'', \beta(u), \rho_u), \mathcal{T}, k, u)$ and $((\mathbf{G}' \oplus \mathbf{G}'', \beta'(u), \rho_u), \mathcal{T}', k, u)$ outputs the same result. Thus, Algorithm 1 applied with input $((\mathbf{G} \oplus \mathbf{G}'', \beta(\mathsf{r}), \rho_\mathsf{r}), \mathcal{T}, k, \mathsf{r})$ and $((\mathbf{G}' \oplus \mathbf{G}'', \beta'(\mathsf{r}), \rho_\mathsf{r}), \mathcal{T}', k, \mathsf{r})$ outputs the same result. So $\mathsf{char}_k(\mathbf{G} \oplus \mathbf{G}'', \beta(\mathsf{r}), \rho_\mathsf{r}) = \mathsf{char}_k(\mathbf{G}' \oplus \mathbf{G}'', \beta'(\mathsf{r}), \rho_\mathsf{r})$. Therefore, according to Lemma 7.5, $\mathsf{ed}_{\mathsf{exc}(\mathcal{F})}(\mathbf{G} \oplus \mathbf{G}'') = \mathsf{ed}_{\mathsf{exc}(\mathcal{F})}(\mathbf{G}' \oplus \mathbf{G}'')$. ∎

8. Elimination distance to a minor-closed graph class

We finally present our main result for $\mathcal{F}$-M-ELIMINATION DISTANCE. The following theorem is a restatement of Theorem 1.2.

THEOREM 8.1. *For every finite collection of graphs $\mathcal{F}$, there exists an algorithm that, given a graph G and an integer k, decides whether $\mathsf{ed}_{\mathsf{exc}(\mathcal{F})}(G) \leq k$ in time $2^{2^{2^{k^{O_{\ell_{\mathcal{F}}}(1)}}}} \cdot n^2$. In the particular case when $\mathcal{F}$ contains an apex-graph, this algorithm runs in time $2^{2^{O_{\ell_{\mathcal{F}}}(k^2 \log k)}} \cdot n^2$.*

This algorithm is very similar to the one in Section 6. We use the same propositions and lemmata, except from the ones that were tailored for $\mathcal{F}$-M-DELETION and for which we need to find an analogous result for $\mathcal{F}$-M-ELIMINATION DISTANCE. In Section 7, we described an algorithm to find the elimination distance when the treewidth is bounded, which is the analogue of Proposition 3.2 for $\mathcal{F}$-M-ELIMINATION DISTANCE. In Subsection 8.1, we present an analogue of Proposition 3.1, which either reports an upper bound on the treewidth of the input graph, or finds a wall, or reports that we deal with a no-instance.

In contrast to the algorithm for $\mathcal{F}$-M-DELETION, we cannot use branching the same way because we do not have a bound in k on the number of vertices in a k-elimination set, like we had for a k-apex set. Thus, while Step 3 of Section 6 could be applied at most $a_{\mathcal{F}}^k$ times, such a Step 3 would now be applied at most $a_{\mathcal{F}}^n$ times, which is not efficient. Since we do not use branching anymore to reduce the size of the apex set of a flatness pair, when we apply Proposition 3.5, the size a of the apex set in the input depends on k. Hence, we need to find in the beginning a wall whose size dependence in k is triple-exponential. This explains the triple-exponential in k that appears in the running time of Theorem 8.1. The above algorithm is described in Subsection 8.2 and in Subsection 8.3 we present the proof of its correctness.

8.1   Quickly finding a wall

We first prove Proposition 3.1 in the case of $\mathcal{F}$-M-ELIMINATION DISTANCE. The proof is very similar to the one given in [70] for Proposition 3.1 in the case of $\mathcal{F}$-M-DELETION and is achieved using the following result from Perkovic and Reed[59]. The main difference with respect to the proof of Proposition 3.1 given in [70] is that we need to use two new ingredients tailored for $\mathcal{F}$-M-ELIMINATION DISTANCE, namely Theorem 7.1 and Lemma 4.6.

PROPOSITION 8.2 ([59]). *There exists an algorithm with the following specifications:*

Input: A graph G and $t \in \mathbb{N}$ such that $|V(G)| \geq 12t^3$.
Output: A graph G^* such that $|V(G^*)| \leq (1 - \frac{1}{16t^2}) \cdot |V(G)|$ and,
  — *either G^* is a subgraph of G such that $\mathsf{tw}(G) = \mathsf{tw}(G^*)$, or*
  — *G^* is obtained from G after contracting the edges of a matching in G.*

Moreover, the algorithm runs in time $2^{O(t)} \cdot n$.

PROOF OF PROPOSITION 3.1 IN THE CASE OF $\mathcal{F}$-M-ELIMINATION DISTANCE.
Let $c := f_{12}(s_{\mathcal{F}}) \cdot r + k$.

Suppose that $|V(G)| < 12c^3$. Run the algorithm of [8] that, in time $O(|V(G)|^{c+2}) = 2^{O_{\ell_{\mathcal{F}}}((r+k) \cdot \log(r+k))}$, checks whether $\mathsf{tw}(G) \leq c$. If this is the case, report the same and stop. If not, we aim to find an r-wall of G or conclude that we are dealing with a no-instance. First consider an arbitrary ordering $(v_1, \ldots, v_{|V(G)|})$ of the vertices of G. For each $i \in [|V(G)|]$, set G_i to be the graph induced by the vertices $v_1, \ldots, v_i$. Iteratively run the algorithm of Proposition 4.3 on G_i and c for increasing values of i. This algorithm runs in time $2^{O(c)} \cdot |V(G)| = 2^{O_{\ell_{\mathcal{F}}}(r+k)}$. Let $j \in [|V(G)|]$ be the smallest integer such that the above algorithm outputs a report that $\mathsf{tw}(G_j) > c$ (it exists since $\mathsf{tw}(G) > c$) and notice that there exists a tree decomposition (T_j, β_j) of G_j of width at most $2c + 2$, obtained by the one of G_{j-1} by adding the vertex v_j to all the bags. Thus, we can call the algorithm of Theorem 7.1 with input $(G_j, 2c + 2, k)$, which runs in time $2^{O_{\ell_{\mathcal{F}}}(c \cdot (k+\log c))} \cdot |V(G_j)| = 2^{O_{\ell_{\mathcal{F}}}((r+k) \cdot (k+\log(r+k)))}$, in order to find, if it exists, a k-elimination set S_j of G_j for $\mathsf{exc}(\mathcal{F})$.

- If such a set S_j does not exist, then safely report that (G, k) is a no-instance.
- If such a set S_j exists, then call the algorithm of Proposition 4.5 for $G_j \setminus S_j$, (and the decomposition of $G_j \setminus S_j$ obtained from (T_j, β_j) by removing the vertices of S_j from all the bags) in order to check whether it contains an elementary r-wall W as a minor. This algorithm runs in time $2^{O(c \cdot \log c)} \cdot r^{O(c)} \cdot 2^{O(r^2)} \cdot |V(G_j \setminus S_j)| = 2^{O_{\ell_{\mathcal{F}}}((r+k) \cdot \log(r+k))} \cdot r^{O_{\ell_{\mathcal{F}}}(r+k)} \cdot 2^{O(r^2)} = 2^{O_{\ell_{\mathcal{F}}}(r^2 + (r+k) \cdot \log(r+k))}$, since $|E(W)| = O(r^2)$. Since all connected components of $G_j \setminus S_j$ are in $\mathrm{exc}(\mathcal{F})$, $G_j \setminus S_j$ does not contain $K_{s_{\mathcal{F}}}$ as a minor. By Lemma 4.6, $\mathrm{tw}(G_j \setminus S_j) \geq c - k = f_{12}(s_{\mathcal{F}}) \cdot r$. So because of Proposition 5.1, the algorithm of Proposition 4.5 will output an elementary r-wall W of $G_j \setminus S_j$. We also return W as a wall of G.

Therefore, in the case where $|V(G)| < 12c^3$, we obtain one of the three possible outputs in time $2^{O_{\ell_{\mathcal{F}}}(r^2 + k^2)}$.

If $|V(G)| \geq 12c^3$, then call the algorithm of Proposition 8.2 with input (G, c), which outputs a graph G^* such that $|V(G^*)| \leq (1 - \frac{1}{16c^2}) \cdot |V(G)|$ and

— either G^* is a subgraph of G such that $\mathrm{tw}(G) = \mathrm{tw}(G^*)$, or
— G^* is obtained from G after contracting the edges of a matching in G.

In both cases, recursively call the algorithm on G^* and distinguish the following two cases.

Case 1: G^* is a subgraph of G such that $\mathrm{tw}(G) = \mathrm{tw}(G^*)$.

(a) If the recursive call on G^* reports that $\mathrm{tw}(G^*) \leq c$, then return that $\mathrm{tw}(G) \leq c$.
(b) If the recursive call on G^* outputs an r-wall W of G^*, then return W as a wall of G.
(c) If (G^*, k) is a no-instance, then report that (G, k) is also a no-instance.

Case 2: G^* is obtained from G after contacting the edges of a matching in G.

(a) If the recursive call on G^* reports that $\mathrm{tw}(G^*) \leq c$, then do the following. First notice that the fact that $\mathrm{tw}(G^*) \leq c$ implies that $\mathrm{tw}(G) \leq 2c$, since we can obtain a tree decomposition (T, β) of G from a tree decomposition (T^*, β^*) of G^*, by replacing, in every $t \in V(T^*)$, every occurrence of a vertex of G^* that is a result of an edge contraction by its endpoints in G. Thus, we can call the algorithm of Theorem 7.1 with input $(G, 2c, k)$, which runs in time $2^{O_{\ell_{\mathcal{F}}}(c(k + \log c))} \cdot n$, in order to find, if it exists, a k-elimination set S of G for $\mathrm{exc}(\mathcal{F})$. We distinguish again two cases.

  - If such a set S does not exist, then the algorithm reports that (G, k) is a no-instance.
  - If such a set S exists, then apply the algorithm of Proposition 4.3 with input $(G \setminus S, 2c)$, which runs in time $2^{O(c)} \cdot n$, and obtain a tree decomposition of $G \setminus S$ of width at most $4c + 1$. Using this decomposition, call the algorithm of Proposition 4.5 for $G \setminus S$ in order to check whether it contains an elementary r-wall W as a minor. This algorithm runs in time $2^{O(c \cdot \log c)} \cdot r^{O(c)} \cdot 2^{O(r^2)} \cdot n = 2^{O_{\ell_{\mathcal{F}}}((r+k) \cdot \log(r+k))} \cdot r^{O_{\ell_{\mathcal{F}}}(r+k)} \cdot 2^{O(r^2)} \cdot n = 2^{O_{\ell_{\mathcal{F}}}(r^2 + (r+k) \cdot \log(r+k))} \cdot n$, since $|E(G \setminus S)| = O(n)$ and $|E(W)| = O(r^2)$. If this algorithm

outputs an elementary r-wall W of $G \setminus S$, then output W. Otherwise, safely report, because of Proposition 5.1 and Lemma 4.6, that $\text{tw}(G) \leq f_{12}(s_{\mathcal{F}}) \cdot r + k = c$.

(b) If the recursive call on G^* outputs an r-wall W^* of G^*, then by uncontracting the edges of M in W^*, we can return an r-wall of G.

(c) If (G^*, k) is a no-instance, then report that (G, k) is also a no-instance.

It is easy to see that the running time of the above algorithm is given by the function

$$T(n, k, r) \ \leq \ T\left((1 - \frac{1}{16c^2}) \cdot n, k, r\right) + 2^{O_{\ell_{\mathcal{F}}}(r^2 + k^2)} \cdot n,$$

which implies that $T(n, k, r) = 2^{O_{\ell_{\mathcal{F}}}(r^2 + k^2)} \cdot n$, as claimed. ∎

8.2   Description of the algorithm for $\mathcal{F}$-M-Elimination Distance

We define the following constants.

$a = f_3(s_{\mathcal{F}} + k),$ $q = f_{10}(a_{\mathcal{F}}, s_{\mathcal{F}}, k(k + 1)/2),$

$p = f_{11}(a_{\mathcal{F}}, s_{\mathcal{F}}, k(k + 1)/2),$ $l = (q - 1) \cdot a,$

$d = f_8(a_{\mathcal{F}} - 1, \ell_{\mathcal{F}}),$ $r_4 = f_7(a_{\mathcal{F}} - 1, \ell_{\mathcal{F}}, 3, k(k + 1)/2),$

$r_3 = f_5(r_4, a_{\mathcal{F}} - 1, a, d),$ $r_2 = \text{odd}(\max\{f_{13}(l + 1, r_3, p), f_9(a_{\mathcal{F}}, s_{\mathcal{F}}, k(k + 1)/2)\}),$

$r_1 = \text{odd}(f_4(s_{\mathcal{F}} + k) \cdot r_2),$

Note that $r_4 = O_{\ell_{\mathcal{F}}}(k^2)$, $r_3 = O_{\ell_{\mathcal{F}}}(k^{2 \cdot c})$, $r_2 = O_{\ell_{\mathcal{F}}}(k^{2 \cdot c + 15})$, and $r_1 = 2^{O_{\ell_{\mathcal{F}}}(k^2 \log k + c \log k)}$, where $c = f_6(a_{\mathcal{F}} - 1, a, d) = 2^{O_{\ell_{\mathcal{F}}}(k^{24 \cdot (a_{\mathcal{F}} - 1)})}$. Recall from Subsection 2.2 that we assume that G has $O_{s_{\mathcal{F}}}(k\sqrt{\log k} \cdot n)$ edges.

Run the algorithm Find-Wall-Ed from Proposition 3.1 with input (G, r_1, k) and, in time $2^{O_{\ell_{\mathcal{F}}}(r_1^2 + k^2)} \cdot n = 2^{2^{O_{\ell_{\mathcal{F}}}(k^2 \log k + c \log k)}} \cdot n,$

— either report a no-instance, or

— conclude that $\text{tw}(G) \leq f_1(s_{\mathcal{F}}) \cdot r_1 + k$ and solve $\mathcal{F}$-M-Elimination Distance in time $2^{O_{\ell_{\mathcal{F}}}((r_1 + k)k + (r_1 + k)\log(r_1 + k))} \cdot n = 2^{2^{O_{\ell_{\mathcal{F}}}(k^2 \log k + c \log k)}} \cdot n$ using the algorithm of Theorem 7.1, or

— obtain an r_1-wall W_1 of G.

If the output of Proposition 3.1 is a wall W_1, then run the algorithm Clique-or-twFlat of Proposition 3.4 with input $(G, r_2, s_{\mathcal{F}} + k)$. This takes time $2^{2^{O_{\ell_{\mathcal{F}}}(k^2 \log k)}} \cdot r_2^3 \log r_2 \cdot n = 2^{2^{O_{\ell_{\mathcal{F}}}(k^2 \log k + c \log k)}} \cdot n$. If the result is a set A of size at most a and a regular flatness pair $(W_2, \mathfrak{R}_2)$ of $G \setminus A$ of height r_2 whose $\mathfrak{R}_2$-compass has treewidth at most r_1, then proceed, otherwise output a no-answer.

Compute a W_2-canonical partition $\tilde{Q}$ of $G \setminus A$. Compute the set B of vertices of A that are adjacent to at least q p-internal bags of $\tilde{Q}$. As in Proposition 5.6, compute a collection $\mathcal{W} = \{W^1, ..., W^{l+1}\}$ of r_3-subwalls of W_2 such that for every $i \in [l + 1]$, $\bigcup \text{influence}_{\mathfrak{R}_2}(W^i)$ is a subgraph of $\bigcup\{Q \mid Q$ is a p-internal bag of $\tilde{Q}\}$ and for every $i, j \in [l + 1]$, with $i \neq j$, there is no

internal bag of $\tilde{Q}$ that contains vertices of both $V(\bigcup \text{influence}_{\mathfrak{R}_2}(W^i))$ and $V(\bigcup \text{influence}_{\mathfrak{R}_2}(W^j))$. By the choice of l, there is an $i \in [l+1]$ such that no vertex of $\bigcup \text{influence}_{\mathfrak{R}_2}(W^i)$ is adjacent to a vertex of $A \setminus B$.

Run the algorithm from Proposition 5.4 with input $(G \setminus B, W_2, \mathfrak{R}_2, W^i)$ to obtain a W^i-tilt $(W_3, \mathfrak{R}_3)$ of $(W_2, \mathfrak{R}_2)$ in time $O_{s_{\mathcal{F}}}(k\sqrt{\log k} \cdot n)$.

As a next step, we apply the algorithm Homogeneous of Proposition 3.5 with input $(r_4, a_{\mathcal{F}} - 1, a, d, r_1, G, B, W_3, \mathfrak{R}_3)$, which, in time $2^{O(c \cdot r_4 \log r_4 + r_1 \log r_1)} \cdot (n + m) = 2^{2^{O_{\ell_{\mathcal{F}}}(k^2 \log k + c \log k)}} \cdot n$, outputs a flatness pair $(W_4, \mathfrak{R}_4)$ of $G \setminus B$ of height r_4 that is d-homogeneous with respect to $\binom{B}{<a_{\mathcal{F}}}$ and is a W^*-tilt of $(W_3, \mathfrak{R}_3)$ for some subwall W^* of W_3.

Finally, apply the algorithm Find-Irrelevant-Vertex of Proposition 3.6 with input $(k(k + 1)/2, a_{\mathcal{F}} - 1, G, B, W_4, \mathfrak{R}_4)$, which outputs, in time $O_{s_{\mathcal{F}}}(k\sqrt{\log k} \cdot n)$, an irrelevant vertex v such that (G, k) and $(G \setminus v, k)$ are equivalent instances of $\mathcal{F}$-M-ELIMINATION DISTANCE. Then the algorithm runs recursively on the equivalent instance $(G \setminus v, k)$.

Since each run takes time $2^{2^{O_{\ell_{\mathcal{F}}}(k^2 \log k + c \log k)}} \cdot n$ and there are at most n runs, the algorithm indeed runs in time $2^{2^{O_{\ell_{\mathcal{F}}}(k^2 \log k + c \log k)}} \cdot n^2$.

Note that $c = 2^{O_{\ell_{\mathcal{F}}}(k^{24 \cdot (a_{\mathcal{F}} - 1)})}$, so if $\mathcal{F}$ contains an apex-graph, i.e., if $a_{\mathcal{F}} = 1$, then $c = O_{\ell_{\mathcal{F}}}(1)$. Thus, the running time is $2^{2^{O_{\ell_{\mathcal{F}}}(k^{24 \cdot (a_{\mathcal{F}} - 1)})}} \cdot n^2$ in the general case and $2^{2^{O_{\ell_{\mathcal{F}}}(k^2 \log k)}} \cdot n^2$ in the case where $\mathcal{F}$ contains an apex-graph.

8.3 Correctness of the algorithm

Let (G, k) be a yes-instance and let S be a k-elimination set of G for $\text{exc}(\mathcal{F})$. By running Proposition 3.1 with input (G, r_1, k), the algorithm should either get a report that $\text{tw}(G) \leq f_1(s_{\mathcal{F}}) \cdot r_1 + k$ or find an r_1-wall. The correctness of the former is obvious, so we will focus on the latter.

Let W_1 be an r_1-wall of G. According to Lemma 4.2, $K_{s_{\mathcal{F}}+k}$ is not a minor of G. Moreover, since W_1 is a wall of G of height r_1, $\text{tw}(G) \geq \text{tw}(W_1) \geq r_1 \geq f_4(s_{\mathcal{F}} + k) \cdot r_2$. Hence, if the algorithm runs Clique-or-twFlat of Proposition 3.4 with input $(G, r_2, s_{\mathcal{F}} + k)$, it should obtain a set A of size at most a and a regular flatness pair $(W_2, \mathfrak{R}_2)$ of $G \setminus A$ of height r_2 whose $\mathfrak{R}_2$-compass has treewidth at most r_1.

As described in the algorithm, due to Proposition 5.6 and the fact that $r_2 \geq f_{13}(l + 1, r_3, p)$, there is an r_3-wall W^i that is a subwall of W_2 such that no vertex of $\bigcup \text{influence}_{\mathfrak{R}_2}(W^i)$ is adjacent to a vertex of $A \setminus B$, where B is the set of vertices of A adjacent to at least q p-internal bags of a W_2-canonical partition $\tilde{Q}$ of $G \setminus A$.

When the algorithm applies Proposition 5.4 with input $(G \setminus B, W_2, \mathfrak{R}_2, W^i)$, it obtains a W^i-tilt $(W_3, \mathfrak{R}_3)$ of $(W_2, \mathfrak{R}_2)$. Due to Observation 5.2 and Observation 5.3, $(W_3, \mathfrak{R}_3)$ is a regular flatness

pair of $G \setminus B$ whose $\mathfrak{R}_5$-compass has treewidth at most r_1. Thus, since $r_3 = f_5(r_4, a_{\mathcal{F}} - 1, a, d)$, the algorithm can apply Homogeneous of Proposition 3.5 with input $(r_4, a_{\mathcal{F}} - 1, a, d, r_1, G, B, W_3, \mathfrak{R}_3)$ to obtain a flatness pair $(W_4, \mathfrak{R}_4)$ of $G \setminus B$ of height r_4 that is d-homogeneous with respect to $\binom{B}{<a_{\mathcal{F}}}$ and is a W^*-tilt of $(W_3, \mathfrak{R}_3)$ for some subwall W^* of W_3. According to Observation 5.3, $(W_4, \mathfrak{R}_4)$ is regular.

Let S' be a k-elimination set of G for $\mathrm{exc}(\mathcal{F})$. Lemma 5.9 implies that there is a set $X_{S'} \supseteq S'$ such that $G \setminus X_{S'} \in \mathrm{exc}(\mathcal{F})$ and $\mathrm{bid}_{G \setminus A, W_2}(X_{S'}) \leq k(k+1)/2$.

Since $r_2 \geq f_9(a_{\mathcal{F}}, s_{\mathcal{F}}, k(k+1)/2)$, every subset of B of size $a_{\mathcal{F}}$ intersects $X_{S'}$ according to Proposition 3.7. Hence, $|B \setminus X_{S'}| \leq a_{\mathcal{F}} - 1$.

Moreover, note that $(W_3, \mathfrak{R}_3)$ is a W^i-tilt of $(W_2, \mathfrak{R}_2)$, $(W_4, \mathfrak{R}_4)$ is a W^*-tilt of $(W_3, \mathfrak{R}_3)$, and $(W_4, \mathfrak{R}_4)$ is a flatness pair of $G \setminus B$ with $B \subseteq A$. Thus, given a W_4-canonical partition Q_1 of $G \setminus B$, there is a W_2-canonical partition Q_2 of $G \setminus A$ such that each internal bag of Q_1 is an internal bag of Q_2. Thus, $\mathrm{bid}_{G \setminus B, W_4}(X_{S'}) \leq \mathrm{bid}_{G \setminus A, W_2}(X_{S'})$.

Hence, the algorithm can apply Find-Irrelevant-Vertex of Proposition 3.6 with input $(k(k+1)/2, a_{\mathcal{F}} - 1, G, B, W_4, \mathfrak{R}_4)$ and obtain a vertex v such that, for any k-elimination set S' of G for $\mathrm{exc}(\mathcal{F})$, $G \setminus X_{S'} \in \mathrm{exc}(\mathcal{F})$ if and only if $G \setminus (X_{S'} \setminus v) \in \mathrm{exc}(\mathcal{F})$. Thus, there is a k-elimination set of G for $\mathrm{exc}(\mathcal{F})$ if and only if there is a k-elimination set of $G \setminus v$ for $\mathrm{exc}(\mathcal{F})$. It follows that (G, k) and $(G \setminus v, k)$ are equivalent instances of $\mathcal{F}$-M-ELIMINATION DISTANCE.

Suppose now that (G, k) is a no-instance. Note that as long as Proposition 3.4 outputs a flatness pair $(W_2, \mathfrak{R}_2)$, what follows in the proof of correctness works even if (G, k) is a no-instance. Therefore, we will find an irrelevant vertex. Otherwise, we would have declared a no-instance beforehand. Thus, Theorem 8.1 follows.

9.   Elimination distance when excluding an apex-graph

In the case where $\mathcal{F}$ contains an apex-graph, we obtain an alternative algorithm whose complexity is single-exponential in k and cubic in n. The following theorem is a restatement of Theorem 1.3.

THEOREM 9.1. *For every finite collection of graphs $\mathcal{F}$ that contains an apex-graph, there exists an algorithm that, given a graph G and an integer k, decides whether $\mathrm{ed}_{\mathrm{exc}(\mathcal{F})}(G) \leq k$ in time $2^{k^{O_{\ell_{\mathcal{F}}(1)}}} \cdot n^3$.*

Contrary to the previous section, since $a_{\mathcal{F}} = 1$, any vertex fulfilling the criteria of Proposition 3.7 belongs to every k-elimination set S of the input graph for $\mathrm{exc}(\mathcal{F})$. In the case where $\mathcal{F}$ contains an apex-graph in [70], a Step 3 similar to the one of Section 6 can be applied, where a vertex belonging to every k-apex set can be found. Hence, after running Step 3 k times, a k-apex set is found and the algorithm stops. However, a k-elimination set may have size $\Omega(n)$,

so we may run Step 3 $\Omega(n)$ times. Since our Step 3 below runs in quadratic time, this gives the cubic dependence of this algorithm. Fortunately, since we apply Step 3, just like in Section 6 and contrary to Section 8, we manage to find a flatness pair along with an apex set whose size does not depend on k. Hence, when applying Proposition 3.5, we do not get a triple-exponential dependence on k anymore for the size of the wall we need to find originally.

In order to remember the vertices that are found to belong to every k-elimination set, since they do not decrease k like in Section 6, we need to distinguish them in the input. Hence, we actually give here an algorithm to solve a more general problem with annotations described in Subsection 9.1.

9.1   Generalization to annotated elimination distance

Contrary to the previous section, since $a_{\mathcal{F}} = 1$, when applying Proposition 3.7, we find a vertex that belongs to every k-elimination set S. Such vertices are taken into account by considering the following generalization of $\mathcal{F}$-M-ELIMINATION DISTANCE.

> ANNOTATED $\mathcal{F}$-M-ELIMINATION DISTANCE
>
> **Input:** A graph G, a set $S_0 \subseteq V(G)$, and a non-negative integer k.
>
> **Objective:** Find, if it exists, a k-elimination set S of G for the class $\text{exc}(\mathcal{F})$ such that $S_0 \subseteq S$.

S_0 is a set of annotated vertices that corresponds to the vertices identified as vertices of every k-elimination set S while running the algorithm. Clearly, $\mathcal{F}$-M-ELIMINATION DISTANCE is the particular case of ANNOTATED $\mathcal{F}$-M-ELIMINATION DISTANCE when S_0 is empty.

In the following lemma, we generalize Theorem 7.1 to its "annotated" version. More precisely, we present a simple trick to reduce the above problem to its "unannotated" version while not changing the treewidth of the input graph so much.

LEMMA 9.2. *Let $\mathcal{F}$ be a finite collection of graphs. There is an algorithm that, given a graph G, a set $S_0 \subseteq V(G)$, and two integers k and tw such that the treewidth of G is bounded by tw, decides whether (G, S_0, k) is a* yes*-instance of* ANNOTATED $\mathcal{F}$-M-ELIMINATION DISTANCE *in time* $2^{O_{\ell_{\mathcal{F}}}(\text{tw}\cdot k + \text{tw}\log \text{tw})} \cdot n$.

PROOF. Given a minor-closed graph class $\mathcal{G}$, let $C(\mathcal{G}) := \{G \mid \forall C \in \text{cc}(G), C \in \mathcal{G}\}$. Bulian and Dawar [16] showed that if $H \in \text{obs}(\mathcal{G})$ has l connected components, then each graph $\bar{H}$ obtained from H by adding $l - 1$ edges to obtain a connected graph belongs to $\text{obs}(C(\mathcal{G}))$. Thus, let $H_{\mathcal{F}} \in \text{obs}(C\text{exc}(\mathcal{F})))$ obtained in such a way. As said above, $H_{\mathcal{F}}$ is connected.

Let G be a graph of treewidth at most tw and S_0 be a subset of $V(G)$. Let G' be a graph obtained from G by gluing a graph H_v isomorphic to $H_{\mathcal{F}}$ to each vertex v of S_0, where v is identified with an arbitrarily chosen vertex of H_v.

Let us show that (G, S_0, k) is a yes-instance of ANNOTATED $\mathcal{F}$-M-ELIMINATION DISTANCE if and only if (G', k) is a yes-instance of $\mathrm{obs}(\mathcal{C}\mathrm{exc}(\mathcal{F}))$-M-ELIMINATION DISTANCE. If $S_0 = \emptyset$, the proof is trivial, so we suppose $S_0 \neq \emptyset$.

Let (F', χ', R') be a $(\mathcal{C}\mathrm{exc}(\mathcal{F}))$-elimination forest of G' of height at most k with associated k-elimination set S'. For each $v \in S_0$, the fact that $H_v \notin \mathcal{C}\mathrm{exc}(\mathcal{F}))$ implies that $V(H_v) \cap S' \neq \emptyset$. Let $y \in V(F')$ be the least common ancestor of $\chi'^{-1}(H_v)$ in F'. Since H_v is connected, according to Lemma 4.1, y exists and belongs to $\chi'^{-1}(H_v)$. Moreover, since $V(H_v) \cap S' \neq \emptyset$, $y \in \mathrm{Int}(F', R')$.

Let (F'', χ'', R'') be the $\mathcal{F}$-elimination forest of G obtained from (F', χ', R') as follows. For every $v \in S_0$ and every $t \in \chi'^{-1}(H_v \setminus v)$ if $t \in \mathrm{Int}(F', R')$, remove t from F'' and add edges between the parent and the children of t and if $t \in \mathrm{Leaf}(F', R')$, remove $H_v \setminus v$ from $\chi''(t)$. If $G[\chi''(t)]$ is not connected, then we update F'' by replacing t by $|\mathrm{cc}(G[\chi''(t)])|$ nodes, each one associated with a connected component of $G[\chi''(t)]$.

Thus, we have an $\mathcal{F}$-elimination forest of G associated with a k-elimination set S with $S_0 \subseteq S$ and with height at most k, implying that (G, S_0, k) is a yes-instance of ANNOTATED $\mathcal{F}$-M-ELIMINATION DISTANCE.

Conversely, given an $\mathcal{F}$-elimination forest (F, χ, R) of G of height at most k, and $S_0 \subseteq S :=$ $\chi(\mathrm{Int}(F, R))$, we can obtain a $\mathrm{obs}(\mathcal{C}\mathrm{exc}(\mathcal{F}))$-elimination forest (F', χ', R') of G' of height at most k by adding to each node $\chi^{-1}(v)$ for $v \in S_0 \subseteq S$ a leaf associated with $H_v \setminus v \in \mathcal{C}\mathrm{exc}(\mathcal{F}))$. The height of (F', χ', R') is indeed still at most k. Therefore, (G, S_0, k) is a yes-instance for ANNOTATED $\mathcal{F}$-M-ELIMINATION DISTANCE if and only if (G', k) is a yes-instance for $\mathrm{obs}(\mathcal{C}\mathrm{exc}(\mathcal{F}))$-M-ELIMINATION DISTANCE.

Thus, if we apply the algorithm of Theorem 7.1 to (G', k) to solve $\mathrm{obs}(\mathcal{C}\mathrm{exc}(\mathcal{F}))$-M-ELIMINATION DISTANCE, we solve ANNOTATED $\mathcal{F}$-M-ELIMINATION DISTANCE on instance (G, S_0, k) in time $2^{O_{\ell_{\mathcal{F}}}(\mathrm{tw}(G') \cdot k + \mathrm{tw}(G') \log \mathrm{tw}(G'))} \cdot |V(G')|$.

Given a tree decomposition $\mathcal{T} = (\mathsf{T}, \beta)$ of G of width t, we can obtain a tree decomposition $\mathcal{T}'$ of G' of width at most $t + |V(H_{\mathcal{F}})|$ by adding a node x_v for each $v \in S_0$ such that $\beta(v) = V(H_v)$, adjacent to a node y of T such that $v \in \beta(y)$. Thus, $\mathrm{tw}(G') \leq \mathrm{tw}(G) + |V(H_{\mathcal{F}})| = \mathrm{tw}(G) + O_{\ell_{\mathcal{F}}}(1)$. Moreover, $|V(G')| = |V(G)| + (|V(H_{\mathcal{F}})| - 1) \cdot |S_0| = O_{\ell_{\mathcal{F}}}(|V(G)|)$. Therefore, we can solve ANNOTATED $\mathcal{F}$-M-ELIMINATION DISTANCE on instance (G, S_0, k) in time $2^{O_{\ell_{\mathcal{F}}}(\mathrm{tw} \cdot k + \mathrm{tw} \log \mathrm{tw})} \cdot n$, and the lemma follows. ∎

9.2   Description of the algorithm for $\mathcal{F}$-M-ELIMINATION DISTANCE when $a_{\mathcal{F}} = 1$

We now describe the algorithm to solve ANNOTATED $\mathcal{F}$-M-ELIMINATION DISTANCE, and hence $\mathcal{F}$-M-ELIMINATION DISTANCE, when $a_{\mathcal{F}} = 1$, i.e., when $\mathcal{F}$ contains an apex-graph. Note that, similarly to this algorithm, the one from Section 8 in the general case can also very easily be generalized to its "annotated" version. We stress that the reason for the better parametric dependence of this algorithm compared to the algorithm of Theorem 1.2 is that we pursue homogeneous flat walls where homogeneity is asked for subsets of size *not* depending on k.

We define the following constants.

$$a = f_3(s_{\mathcal{F}}), \qquad\qquad q = f_{10}(1, s_{\mathcal{F}}, k(k+1)/2),$$
$$p = f_{11}(1, s_{\mathcal{F}}, k(k+1)/2), \qquad l = (q-1)\cdot(k+a),$$
$$d = f_8(a, \ell_{\mathcal{F}}) \qquad\qquad r_4 = f_7(a, \ell_{\mathcal{F}}, 3, k(k+1)/2),$$
$$r_3 = f_5(r_4, a, a, d), \qquad\qquad t = f_4(s_{\mathcal{F}})\cdot r_3,$$
$$r_2 = \mathrm{odd}(t+3), \qquad\qquad r_2' = \mathrm{odd}(\max\{f_9(1, s_{\mathcal{F}}, k(k+1)/2), f_{13}(l+1, r_2, p)\}),$$
$$r_1' = \mathrm{odd}(f_2(s_{\mathcal{F}})\cdot r_2'), \qquad r_1 = \mathrm{odd}(r_1' + k).$$

Note that $r_4 = O_{\ell_{\mathcal{F}}}(k^2), r_3, r_2 = O_{\ell_{\mathcal{F}}}(k^{2c})$, and $r_2', r_1', r_1 = O_{\ell_{\mathcal{F}}}(k^{2c+7/2})$, where $c = f_6(a, a, d) = O_{\ell_{\mathcal{F}}}(1)$. Recall from Subsection 2.2 that we assume that G has $O_{s_{\mathcal{F}}}(k\sqrt{\log k}\cdot n)$ edges.

The input of this algorithm is a graph G, a set $S_0 \subseteq V(G)$, and an integer k.

Step 1. Run the algorithm Find-Wall-Ed from Proposition 3.1 with input $(G \setminus S_0, r_1, k)$ and, in time $2^{O_{\ell_{\mathcal{F}}}(r_1^2+k^2)}\cdot n = 2^{O_{\ell_{\mathcal{F}}}(k^{4c+7})}\cdot n$,

— either report a NO-instance, or

— conclude that $\mathrm{tw}(G \setminus S_0) \le f_1(s_{\mathcal{F}})\cdot r_1 + k$ and solve ANNOTATED $\mathcal{F}$-M-ELIMINATION DISTANCE with input $(G, S_0, f_1(s_{\mathcal{F}})\cdot r_1 + 2k, k)$ in time $2^{O_{\ell_{\mathcal{F}}}((r_1+k)\cdot k+(r_1+k)\log(r_1+k))}\cdot n = 2^{O_{\ell_{\mathcal{F}}}(k^{2c+9/2})}\cdot n$ using the algorithm of Lemma 9.2, or

— obtain an r_1-wall W_1 of G.

If the output of Proposition 3.1 is a wall W_1, consider all the $\binom{r_1}{r_2}^2 = 2^{O_{\ell_{\mathcal{F}}}(k^{2c}\log k)}$ r_2-subwalls of W_1 and for each one of them, say W_2, let W_2^* be the central $(r_2 - 2)$-subwall of W_2 and let D_{W_2} be the graph obtained from $G \setminus S_0$ after removing the perimeter of W_2 and taking the connected component containing W_2^*. Run the algorithm Clique-or-twFlat of Proposition 3.4 with input $(D_{W_2}, r_3, s_{\mathcal{F}})$. This takes time $2^{O_{\ell_{\mathcal{F}}}(r_3^2)}\cdot n = 2^{O_{\ell_{\mathcal{F}}}(k^{4c})}\cdot n$. If for one of these subwalls the result is a set A of size at most a and a regular flatness pair $(W_3, \mathfrak{R}_3)$ of $D_{W_2} \setminus A$ of height r_3 whose $\mathfrak{R}_3$-compass has treewidth at most t, then we proceed to Step 2, otherwise proceed to Step 3.

Step 2. We obtain a 7-tuple $\mathfrak{R}_3'$ by adding all vertices of $G \setminus (S_0 \cup V(\mathrm{Compass}_{\mathfrak{R}_3}(W_3)))$ to the set in the first coordinate of $\mathfrak{R}_3$, such that $(W_3, \mathfrak{R}_3')$ is a regular flatness pair of $G \setminus (S_0 \cup A)$.

We first apply the algorithm Homogeneous of Proposition 3.5 with input $(r_4, a, a, d, t, G \setminus S_0, A, W_3, \mathfrak{R}_3')$, which outputs, in time $2^{O_{\ell_{\mathcal{F}}}(r_4\log r_4+t\log t)}\cdot(n+m) = 2^{O_{\ell_{\mathcal{F}}}(k^{2c}\log k)}\cdot n$ a flatness pair $(W_4, \mathfrak{R}_4)$ of $G \setminus (S_0 \cup A)$ of height r_4 that is d-homogeneous with respect to 2^A and is a W^*-tilt of $(W_3, \mathfrak{R}_3')$ for some subwall W' of W. At this point, we stress that the reason for the better parametric dependence of this algorithm compared to the previous one comes from the fact that the third input parameter a in Homogeneous does *not* depend of k. We apply the algorithm Find-Irrelevant-Vertex of Proposition 3.6 with input $(k(k+1)/2, a, G \setminus S_0, A, W_4, \mathfrak{R}_4)$, which outputs, in time $O_{s_{\mathcal{F}}}(k\sqrt{\log k}\cdot n)$, a vertex v such that (G, S_0, k) and $(G \setminus v, S_0, k)$ are equivalent

instances of ANNOTATED $\mathcal{F}$-M-ELIMINATION DISTANCE. Then the algorithm runs recursively on the equivalent instance $(G \setminus v, S_0, k)$.

Step 3. Consider all the r_2'-subwalls of W_1, which are $\binom{r_1}{r_2'}^2 = 2^{O_{\ell_{\mathcal{F}}}(k^{2c+7/2}\log k)}$ many, and for each of them, say W_2', compute its canonical partition $\mathcal{Q}$. Then, contract each bag Q of $\mathcal{Q}$ to a single vertex v_Q, remove the vertices v_Q where Q is not a p-internal bag of $\mathcal{Q}$, and add a new vertex v_{all} and make it adjacent to all remaining v_Q's. In the resulting graph G', for every vertex y of $(G \setminus S_0) \setminus V(W_2')$, check, in time $O(q \cdot m) = O_{\ell_{\mathcal{F}}}(k^7 \sqrt{\log k} \cdot n)$, using a flow augmentation algorithm [20], whether there are q internally vertex-disjoint paths from v_{all} to y. Let $\tilde{A}$ be the set of such y's.

If $\tilde{A} = \emptyset$, then report a no-instance.

If $1 \le |\tilde{A}| \le k + a$, then each vertex of $\tilde{A}$ should intersect every k-elimination set S of G for $\mathrm{exc}(\mathcal{F})$. The algorithm runs recursively on $(G, S_0 \cup \tilde{A}, k)$.

If, for every wall, $|\tilde{A}| > k + a$, then report that (G, S_0, k) is a no-instance of ANNOTATED $\mathcal{F}$-M-ELIMINATION DISTANCE.

After Step 2, the size of G decreases by one, so Step 2 can be applied at most n times. After Step 3, the size of S_0 increases by at least one, so Step 3 can also be applied at most n times. Note that, if $S_0 = V(G)$, then $\mathrm{tw}(G \setminus S_0) = 0$, so the algorithm stops. Thus, the algorithm finishes. Notice also that Step 3, when applied, takes time $2^{O_{\ell_{\mathcal{F}}}(k^{2c+7/2}\log k)} \cdot n^2$, because we apply a flow algorithm for each of the $2^{O_{\ell_{\mathcal{F}}}(k^{2c+7/2}\log k)}$ r_2'-subwalls and for each vertex of G. Since Step 1 and Step 2 run in time $2^{O_{\ell_{\mathcal{F}}}(k^{4c+7})} \cdot n$ and $2^{O_{\ell_{\mathcal{F}}}(k^{2c}\log k)} \cdot n$, respectively, and both may be applied at most n times, the claimed time complexity follows: the algorithm runs in time $2^{O_{\ell_{\mathcal{F}}}(k^{4c+7})} \cdot n^3$.

9.3   Correctness of the algorithm

Let (G, S_0, k) be a yes-instance and let S be a k-elimination set of G for $\mathrm{exc}(\mathcal{F})$ with $S_0 \subseteq S$. By running Proposition 3.1 with input $(G \setminus S_0, r_1, k)$, the algorithm should either get a report that $\mathrm{tw}(G \setminus S_0) \le f_1(s_{\mathcal{F}}) \cdot r_1 + k$ or find an r_1-wall.

If $\mathrm{tw}(G \setminus S_0) \le f_1(s_{\mathcal{F}}) \cdot r_1 + k$, then since $S_0 \subseteq S$, $\mathrm{tw}(G \setminus S) \le f_1(s_{\mathcal{F}}) \cdot r_1 + k$. Hence, according to Lemma 4.6, $\mathrm{tw}(G) \le f_1(s_{\mathcal{F}}) \cdot r_1 + 2k$.

Otherwise, let W_1 be an r_1-wall of $G \setminus S_0$. According to Lemma 5.8, since $r_1 \ge r_1' + k$, there is an r_1'-subwall W_1' of W_1 that is a subwall of $G \setminus S$. Let H be the connected component of $G \setminus S$ containing W_2. The fact that H belongs to $\mathrm{exc}(\mathcal{F})$ implies that it has no $K_{s_{\mathcal{F}}}$-minor. Therefore, by Proposition 3.3, since $r_1' \ge f_2(s_{\mathcal{F}}) \cdot r_2'$, there is a set $B \subseteq V(H)$, with $|B| \le a$, and a flatness pair $(W_2', \mathfrak{R}_2')$ of $H \setminus B$ of height r_2'.

Let $\mathcal{Q}$ be the canonical partition of W_2'. Let G' be the graph obtained after contracting every bag Q of $\mathcal{Q}$ to a single vertex v_Q, removing the vertices v_Q where Q is not a p-internal bag of $\mathcal{Q}$,

and adding a new vertex v_{all} and making it adjacent to all remaining v_Q's. Let $\tilde{A}$ be the set of vertices y of $G \setminus V(W_2')$ such that there are q internally vertex-disjoint paths from v_{all} to y in G'. Since S is a k-elimination set of G for $\text{exc}(\mathcal{F})$, there is a set $P \subseteq S$ of size at most k so that $(L, R) := (V(G) \setminus V(H), V(H) \cup P)$ is a separation of G with $P = L \cap R$.

Note that $\tilde{A} \subseteq P \cup B$. To show this, we first prove that, for every $y \notin P \cup B$, the maximum number of internally vertex-disjoint paths from v_{all} to y in G' is less than q. Indeed, if y is a vertex in $(V(G) \setminus V(H)) \setminus P$, then every path from y to a vertex of W_2' intersects P. Therefore, there are at most $k < q$ internally vertex-disjoint paths from v_{all} to such a $y \in (V(G) \setminus V(H)) \setminus P$ in G'. If $y \in V(H) \setminus B$, then we distinguish two cases. First, if y is a vertex in the $\mathfrak{R}_2'$-compass of W_2', there are at most $k + a$ such paths that intersect the set $P \cup B$ and at most four paths that do not intersect $P \cup B$ (in the graph $G' \setminus (P \cup B)$) due to the flatness of W_2'. If y is in $V(H)$ but not a vertex in the $\mathfrak{R}_2'$-compass of W_2', then, since by the definition of flatness pairs the perimeter of W_2' together with the set $P \cup B$ separate y from the $\mathfrak{R}_2'$-compass of W_2', every collection of internally vertex-disjoint paths from v_{all} to y in G' should intersect the set $\{v_{Q_{ext}}\} \cup P \cup B$, where Q_{ext} is the external bag of Q. Therefore, in all cases, if $y \notin P \cup B$, the maximum number of internally vertex-disjoint paths from v_{all} to y in G' is at most $k + a + 4 < q$. Therefore, $y \notin \tilde{A}$. Hence, $|\tilde{A}| \leq k + a$.

Let $\mathfrak{R}_2''$ be the 7-tuple obtained by adding all vertices of $((G \setminus S_0) \setminus P) \setminus H$ to the set in the first coordinate of $\mathfrak{R}_2'$. Notice that since every path between $G \setminus H$ and H intersects P, $(W_2', \mathfrak{R}_2'')$ is a flatness pair of $G \setminus (P \cup B)$.

If $\tilde{A} = \emptyset$, then let $\tilde{Q}$ be an enhancement of Q on $G \setminus (P \cup B)$. No vertex of $(P \cup B) \setminus S_0$ is adjacent to vertices of at least q p-internal bags of $\tilde{Q}$. This means that the p-internal bags of $\tilde{Q}$ that contain vertices adjacent to some vertex of $P \cup B$ are at most $(q - 1) \cdot (k + a) = l$.

Consider a family $\mathcal{W} = \{W^1, \ldots, W^{l+1}\}$ of $l+1$ r_2-subwalls of W_2' such that for every $i \in [l+1]$, $\bigcup \text{influence}_{\mathfrak{R}_2''}(W^i)$ is a subgraph of $\bigcup\{Q \mid Q$ is a p-internal bag of $\tilde{Q}\}$ and for every $i, j \in [l+1]$, with $i \neq j$, there is no internal bag of $\tilde{Q}$ that contains vertices of both $V(\bigcup \text{influence}_{\mathfrak{R}_2''}(W^i))$ and $V(\bigcup \text{influence}_{\mathfrak{R}_2''}(W^j))$. The existence of $\mathcal{W}$ follows from Proposition 5.6 and the fact that $r_2' \geq f_{13}(l + 1, r_2, p)$.

The fact that the p-internal bags of $\tilde{Q}$ that contain vertices adjacent to some vertex of $(P \cup B) \setminus S_0$ are at most l implies that there exists an $i \in [l + 1]$ such that no vertex of $V(\bigcup \text{influence}_{\mathfrak{R}_2''}(W^i))$ is adjacent, in G, to a vertex in $(P \cup B) \setminus S_0$.

Let $W_2 := W^i$, let W_2^* be the central $(r_2 - 2)$-subwall of W_2, and let D_{W_2} be the graph obtained from $G \setminus S_0$ after removing the perimeter of W_2 and taking the connected component containing W_2^*. Any path going from a vertex in H to a vertex in $G \setminus H$ intersects P. Thus, $D_{W_2} \subseteq H$ and therefore, $K_{s_{\mathcal{F}}}$ is not a minor of D_{W_2}. Moreover, W_2^* is a wall of D_{W_2} of height $r_2 - 2 \geq t + 1$, so $\text{tw}(D_{W_2}) > t = f_4(s_{\mathcal{F}}) \cdot r_3$. Therefore, since , if the algorithm runs `Clique-or-twFlat` of Proposition 3.4 with input $(D_{W_2}, r_3, s_{\mathcal{F}})$, it should obtain a set A of size at most a and a regular

flatness pair $(W_3, \mathfrak{R}_3)$ of $D_{W_2} \setminus A$ of height r_3 whose $\mathfrak{R}_3$-compass has treewidth at most t. Hence, the algorithm then runs Step 2.

If $\tilde{A} \neq \emptyset$, then recall that for every $y \in \tilde{A}$, y has q internally vertex-disjoint paths $P_1, ..., P_q$ to different p-internal bags $Q_1, ..., Q_q$ of Q in G. Hence, there is an enhancement $\tilde{Q}_y$ of Q on $G \setminus (P \cup B)$ such that P_i belongs to the bag $\tilde{Q}_i$ that extends Q_i for $i \in [q]$. Therefore, y is adjacent to vertices of at least q p-internal bags of $\tilde{Q}_y$. Let S' be a k-elimination set of G for $\mathsf{exc}(\mathcal{F})$. According to Lemma 5.9, there is a set $X_{S'} \subseteq V(G)$ such that $G \setminus X_{S'} \in \mathsf{exc}(\mathcal{F})$ and $\mathsf{bid}_{G \setminus (P \cup B), W_2'}(X_{S'}) \leq k(k+1)/2$. Therefore, $y \in X_{S'}$ due to Proposition 3.7 and the fact that $r_2' \geq f_9(1, s_\mathcal{F}, k(k+1)/2)$. Let $C_{S'} := G \setminus X_{S'}$. Recall that y is adjacent to $q > k(k+1)/2$ p-internal bags of $\tilde{Q}_y$. However, $\mathsf{bid}_{\tilde{Q}_y}(X_{S'}) \leq \mathsf{bid}_{G \setminus (P \cup B), W_2'}(X_{S'}) \leq k(k+1)/2$. Therefore, y is adjacent to $C_{S'}$, so $y \in S'$. Since for every $y \in \tilde{A}$, for every k-elimination set S', we have $y \in S'$, it implies that $\tilde{A}$ is included in every k-elimination set of G for $\mathsf{exc}(\mathcal{F})$. Hence, if the algorithm runs Step 3, it then recursively runs on the equivalent instance $(G, S_0 \cup \tilde{A}, k)$.

We do not suppose that (G, k) is a yes-instance anymore. Let us show the correctness of Step 2. Suppose that we obtained the wanted flatness pair $(W_3, \mathfrak{R}_3)$ in Step 1. We obtain a 7-tuple $\mathfrak{R}_3'$ by adding all vertices of $G \setminus (S_0 \cup V(\mathsf{Compass}_{\mathfrak{R}_3}(W_3)))$ to the set in the first coordinate of $\mathfrak{R}_3$. Since $(W_3, \mathfrak{R}_3)$ is a regular flatness pair of $D_{W^i} \setminus A$ whose $\mathfrak{R}_3$-compass has treewidth at most t and since the vertices added in $\mathfrak{R}_3'$ are only adjacent to the perimeter of W^i, it follows that $(W_3, \mathfrak{R}_3')$ is a regular flatness pair of $G \setminus (S_0 \cup A)$ whose $\mathfrak{R}_3'$-compass has treewidth at most t.

If the algorithm applies the algorithm Homogeneous of Proposition 3.5 with $(r_4, a, a, d, t, G \setminus S_0, A, W_3, \mathfrak{R}_3')$ as input, it obtains a flatness pair $(W_4, \mathfrak{R}_4)$ of $G \setminus (S_0 \cup A)$ of height r_4 that is d-homogeneous with respect to 2^A and is a W^*-tilt of $(W_3, \mathfrak{R}_3')$ for some subwall W' of W. According to Observation 5.3, $(W_4, \mathfrak{R}_4)$ is regular.

Lemma 5.9 implies that for every k-elimination set $S' \supseteq S_0$, there is a set $X_{S'} \supseteq S'$ with $\mathsf{bid}_{G \setminus (S_0 \cup A), W_4}(X_{S'}) \leq k(k+1)/2$ and $G \setminus X_{S'} \in \mathsf{exc}(\mathcal{F})$. We have that $|A \setminus X| \leq |A| \leq a$, so the algorithm can apply Find-Irrelevant-Vertex of Proposition 3.6 with input $(k(k+1)/2, a, G \setminus S_0, A, W_4, \mathfrak{R}_4)$ to obtain a vertex v such that for every k-elimination set $S' \supseteq S_0$, $G \setminus X_{S'} \in \mathsf{exc}(\mathcal{F})$ if and only if $G \setminus (X_{S'} \setminus v) \in \mathsf{exc}(\mathcal{F})$. It follows that (G, S_0, k) and $(G \setminus v, S_0, k)$ are equivalent instances of ANNOTATED $\mathcal{F}$-M-ELIMINATION DISTANCE.

Suppose now that (G, S_0, k) is a no-instance. In Step 1, the algorithm either reports a no-instance or finds a wall. In the latter case, the algorithm either goes to Step 2 or to Step 3. If it runs Step 2, the previous paragraph justifies that the algorithm finds a vertex v such that $(G \setminus v, S_0, k)$ is a no-instance. If the algorithm runs Step 3, then it either reports a no-instance or recursively runs on the instance $(G \setminus y, S_0 \cup \tilde{A}, k)$. If $(G \setminus y, S_0 \cup \tilde{A}, k)$ is yes-instance, then so is (G, k). Thus, $(G \setminus y, S_0 \cup \tilde{A}, k)$. is a no-instance. Hence, the algorithm always report a no-instance. Therefore, Theorem 9.1 follows.

Constructing the elimination ordering. Notice that the results of Theorem 8.1 and Theorem 9.1 solve the decision version of ELIMINATION DISTANCE TO $\mathcal{G}$. Using the dynamic programming algorithm of Section 7, we may find a k-elimination set X certifying that $\mathsf{ed}_{\mathcal{G}}(G) \leq k$. One may further determine, from X, the way the elimination ordering is applied on the vertices of X as follows. Let $\mathsf{torso}(G, X)$ be the graph obtained from $G[X]$ if, for every connected component C of $G - X$, we make adjacent all pairs of vertices in $N_G(V(C))$ in $G[X]$. Then we know that $\mathsf{td}(\mathsf{torso}(G, X)) \leq k$ and the required elimination ordering is the same as the one for $\mathsf{torso}(G, X)$, which can be computed by the algorithm of [61] in time $2^{O(k^2)} \cdot n$.

10. Bounding the obstructions of $\mathcal{E}_k(\mathsf{exc}(\mathcal{F}))$

In this section, we prove the following result that provides an upper bound on the size of the graphs in $\mathsf{obs}(\mathcal{E}_k(\mathsf{exc}(\mathcal{F})))$. The following theorem is a reformulation of Theorem 1.5.

THEOREM 10.1. *Let $\mathcal{F}$ be a non-empty finite collection of non-empty graphs and k be a positive integer. Every graph in $\mathsf{obs}(\mathcal{E}_k(\mathsf{exc}(\mathcal{F})))$ has $2^{2^{2^{2^{2^k}}^{O_{\ell_{\mathcal{F}}}(1)}}}$ vertices. In the particular case when $\mathcal{F}$ contains an apex-graph, every graph in $\mathsf{obs}(\mathcal{E}_k(\mathsf{exc}(\mathcal{F})))$ has $2^{2^{2^k}^{O_{\ell_{\mathcal{F}}}(1)}}$ vertices.*

Recall that when $\mathcal{F} = \{K_1\} = \mathsf{obs}(\mathcal{G}_\emptyset)$, it is known [22] that every graph in $\mathsf{obs}(\mathcal{E}_k(\mathcal{G}_\emptyset))$ has at most $2^{2^{k-1}}$ vertices.

Theorem 10.1 implies that one can construct an algorithm that receives as input $\mathsf{obs}(\mathsf{exc}(\mathcal{F}))$ and k, and outputs $\mathsf{obs}(\mathcal{E}_k(\mathsf{exc}(\mathcal{F})))$. This is done by enumerating all graphs on at most $f(k)$ vertices, where $f(k)$ is the bound on the number of vertices given by Theorem 10.1, and filtering out those that are members of $\mathcal{E}_k(\mathsf{exc}(\mathcal{F}))$ and taking those that are minor-minimal in what is left. The running time of the algorithm can be bounded by $O(2^{2^{2^{2^{2^k}}^{O_{\ell_{\mathcal{F}}}(1)}}} \cdot n^2)$ in the general case and by $O(2^{2^{2^k}^{O_{\ell_{\mathcal{F}}}(1)}} \cdot n^2)$ if $\mathcal{F}$ contains an apex-graph.

Note that this brute-force algorithm can be used to solve $\mathcal{F}$-M-ELIMINATION DISTANCE. Indeed, to solve $\mathcal{F}$-M-ELIMINATION DISTANCE, we can compute $\mathsf{obs}(\mathcal{E}_k(\mathsf{exc}(\mathcal{F})))$ and then check whether there is a graph in $\mathsf{obs}(\mathcal{E}_k(\mathsf{exc}(\mathcal{F})))$ that is a minor of the input graph. Of course, this algorithm is much less efficient than the ones presented in the previous sections.

The rest of the section is structured as follows: in Subsection 10.1 we bound the treewidth of a minor-minimal obstruction of $\mathcal{E}_k(\mathsf{exc}(\mathcal{F}))$, while in Subsection 10.2 we bound the size of a minor-minimal obstruction of $\mathcal{E}_k(\mathsf{exc}(\mathcal{F}))$ of small treewidth. This immediately implies Theorem 10.1.

10.1   Bounding the treewidth of an obstruction

In this subsection we aim to prove an upper bound on the treewidth of a minor-minimal obstruction of $\mathcal{E}_k(\text{exc}(\mathcal{F}))$.

LEMMA 10.2. *Let $\mathcal{F}$ be a finite collection of graphs. There exists a function $f_{15} : \mathbb{N}^3 \rightarrow \mathbb{N}$ such that if $G \in \text{obs}(\mathcal{E}_k(\text{exc}(\mathcal{F})))$, then $\text{tw}(G) \leq f_{15}(k, a_\mathcal{F}, s_\mathcal{F})$. Moreover, $f_{15}(k, a, s) = 2^{\log(k \cdot c) \cdot 2^{k^{a-1} \cdot 2^{O(s^2 \log s)}}}$, where $a = a_\mathcal{F}$, $s = s_\mathcal{F}$, and c is a constant depending on $\ell_\mathcal{F}$.*

Note that when $a_\mathcal{F} = 1$, $f_{15}(k, 1, s) = O_{\ell_\mathcal{F}}(k^{2^{2^{O(s^2 \log s)}}})$.

PROOF. For simplicity, we use s, a, and ℓ instead of $s_\mathcal{F}$, $a_\mathcal{F}$, and $\ell_\mathcal{F}$, respectively. We set

$$b = f_3(s) + k + 1, \qquad\qquad d = f_8(a - 1, \ell),$$
$$r_4 = f_7(a - 1, \ell, 3, k(k + 1)/2), \qquad r_3 = f_5(r_4, a - 1, b, d),$$
$$x = f_{10}(a, s, k(k + 1)/2), \qquad\qquad p = f_{11}(a, s, k(k + 1)/2),$$
$$l = (x - 1) \cdot b, \qquad\qquad r_2 = \text{odd}(\max\{f_{13}(l + 1, r_3, p), f_9(a, s, k(k + 1)/2)\}),$$
$$r_1 = f_2(s) \cdot r_2, \text{ and} \qquad\qquad w = f_{12}(s) \cdot r_1 + k + 1.$$

It is easy to verify that $w = 2^{\log(k \cdot c) \cdot 2^{k^{a-1} \cdot 2^{O(s^2 \log s)}}}$.

Suppose towards a contradiction that $\text{tw}(G) > w$. Since $G \in \text{obs}(\mathcal{E}_k(\text{exc}(\mathcal{F})))$, for each $v \in V(G)$, $G \setminus v \in \mathcal{E}_k(\text{exc}(\mathcal{F}))$. Therefore, there exists a $(k + 1)$-elimination set S of G for $\text{exc}(\mathcal{F})$. Thus, for each $C \in \text{cc}(G \setminus S)$, $C \in \text{exc}(\mathcal{F})$. According to Lemma 4.6, $\text{tw}(G \setminus S) > w - k - 1 = f_{12}(s) \cdot r_1$, so there is $C \in \text{cc}(G \setminus S)$, such that $\text{tw}(C) > f_{12}(s) \cdot r_1$. Moreover, K_s is not a minor of C. Therefore, according to Proposition 5.1, C contains an r_1-wall W_1.

Since $r_1 = f_2(s) \cdot r_2$, by Proposition 3.3, there is a set $A \subseteq V(C)$ of size at most $f_3(s)$ and a flatness pair $(W_2, \mathfrak{R}_2)$ of $C \setminus A$ of height r_2 such that W_2 is a tilt of a subwall of W_1. Due to Proposition 5.5, there is a regular flatness pair $(W_2', \mathfrak{R}_2')$ of $C \setminus A$ of height r_2.

Since S is a $(k + 1)$-elimination set of G for $\text{exc}(\mathcal{F})$ and $C \in \text{cc}(G \setminus S)$, there exists a set $P \subseteq S$ of size at most $k + 1$ such that $(L, R) := (V(G) \setminus V(C), V(C) \cup P)$ is a separation of G with $L \cap R = P$. Thus, if $\mathfrak{R}_2''$ is the 7-tuple obtained by adding the vertices of $G \setminus (C \cup P)$ to the set in the first coordinate of $\mathfrak{R}_2'$, $(W_2', \mathfrak{R}_2'')$ is a regular flatness pair of $G \setminus (A \cup P)$ of height r_2.

Let $\tilde{Q}$ be a W_2'-canonical partition of $G \setminus (A \cup P)$. Let B be the set of vertices of $A \cup P$ adjacent to vertices of at least x p-internal bags of $\tilde{Q}$. Let $\mathcal{W} = \{W^1, ..., W^{l+1}\}$ be a family of $l + 1$ r_3-subwalls of W_2' such that for every $i \in [l + 1]$, $\bigcup \text{influence}_{\mathfrak{R}_2''}(W^i)$ is a subgraph of $\bigcup\{Q \mid Q \text{ is a } p\text{-internal bag of } \tilde{Q}\}$ and for every $i, j \in [l + 1]$ with $i \neq j$, there is no internal bag $Q \in \tilde{Q}$ that contains vertices of both $V(\bigcup \text{influence}_{\mathfrak{R}_2''}(W^i))$ and $V(\bigcup \text{influence}_{\mathfrak{R}_2''}(W^j))$. The existence of $\mathcal{W}$ follows from the fact that $r_2 \geq f_{13}(l + 1, r_3, p)$ and Proposition 5.6. Notice that the set $N_G((A \cup P) \setminus B)$ intersects the vertex set of at most $(x - 1) \cdot |(A \cup P) \setminus B| \leq l$ p-internal

bags of $\tilde{Q}$. Thus, there is an $i \in [l+1]$ such that no vertex in $(A \cup P) \setminus B$ is adjacent to vertices of $\bigcup \text{influence}_{\mathfrak{R}_2''}(W^i)$.

Let $(W_3, \mathfrak{R}_3)$ be a W^i-tilt of $(W_2', \mathfrak{R}_2'')$. Since $|B| \leq |A \cup P| \leq f_3(s) + k + 1 = b$ and $r_3 = f_5(r_4, b, a-1, d)$, by Proposition 3.5, there is a flatness pair $(W_4, \mathfrak{R}_4)$ of $G \setminus B$ of height r_4 that is d-homogeneous with respect to $\binom{B}{<a}$ and is a tilt of a subwall of $(W_3, \mathfrak{R}_3)$. By Observation 5.2 and Observation 5.3, $(W_4, \mathfrak{R}_4)$ is regular.

Recall that $(W_3, \mathfrak{R}_3)$ is a W^i-tilt of $(W_2', \mathfrak{R}_2'')$, $(W_4, \mathfrak{R}_4)$ is a tilt of a subwall of $(W_3, \mathfrak{R}_3)$, and $(W_4, \mathfrak{R}_4)$ is a flatness pair of $G \setminus B$ with $B \subseteq A \cup P$. Thus, given a W_4-canonical partition $\tilde{Q}_1$ of $G \setminus B$, there is a W_2'-canonical partition $\tilde{Q}_2$ of $G \setminus (A \cup P)$ such that each internal bag of $\tilde{Q}_1$ is contained in an internal bag of $\tilde{Q}_2$. Therefore, for every set $X \subseteq V(G)$, $\text{bid}_{G \setminus B, W_4}(X) \leq \text{bid}_{G \setminus (A \cup P), W_2'}(X)$.

Moreover, since $r_2 \geq f_9(a, s, k(k+1)/2)$, according to Proposition 3.7, every subset of B of size a intersects every set $X \subseteq V(G)$ such that $G \setminus X \in \text{exc}(\mathcal{F})$ and $\text{bid}_{G \setminus (A \cup P), W_2'}(X) \leq k(k+1)/2$. Hence, for any such X, $|B \setminus X| < a$.

Thus, according to Proposition 3.6, since $r_4 = f_7(a-1, \ell, 3, k(k+1)/2)$, it holds that there is a vertex v such that, for every set $X \subseteq V(G)$ with $G \setminus X \in \text{exc}(\mathcal{F})$ and $\text{bid}_{G \setminus B, W_4}(X) \leq k(k+1)/2$, $G \setminus X \in \text{exc}(\mathcal{F})$ if and only if $G \setminus (X \setminus v) \in \text{exc}(\mathcal{F})$.

Lemma 5.9 implies that for any k-elimination set S' of $G \setminus v$ for $\text{exc}(\mathcal{F})$, there is a set $X \supseteq S'$ such that $G \setminus X \in \text{exc}(\mathcal{F})$ and $\text{bid}_{G \setminus (A \cup P), W_2'}(X) \leq k(k+1)/2$. Since $\text{bid}_{G \setminus B, W_4}(X) \leq \text{bid}_{G \setminus (A \cup P), W_2'}(X)$, we also have that $\text{bid}_{G \setminus B, W_4}(X) \leq k(k+1)/2$. Thus, $G \setminus X \in \text{exc}(\mathcal{F})$ if and only if $G \setminus (X \setminus v) \in \text{exc}(\mathcal{F})$ and $G \in \mathcal{E}_k(\text{exc}(\mathcal{F}))$ if and only if $G \setminus v \in \mathcal{E}_k(\text{exc}(\mathcal{F}))$. However, since $G \in \text{obs}(\mathcal{E}_k(\text{exc}(\mathcal{F})))$, it holds that $G \notin \mathcal{E}_k(\text{exc}(\mathcal{F}))$ and $G \setminus v \in \mathcal{E}_k(\text{exc}(\mathcal{F}))$, a contradiction. ∎

10.2 Bounding the size of an obstruction of small treewidth

In order to bound the size of an obstruction of small treewidth, we first present some additional notions on tree decompositions on boundaried graphs.

Treewidth of boundaried graphs. Let $\mathbf{G} = (G, B, \rho)$ be a boundaried graph. A *tree decomposition* of $\mathbf{G}$ is a rooted tree decomposition (T, β, r) of G such that $\beta(r) = B$. The *width* of (T, β, r) is the width of (T, β). The treewidth of a boundaried graph $\mathbf{G}$ is the minimum width over all its tree decompositions and is denoted by $\text{tw}(\mathbf{G})$. A *nice tree decomposition* of $\mathbf{G}$ is a tree decomposition (T, β, r) of $\mathbf{G}$ that is also a nice tree decomposition of G rooted at r.

Let $\mathbf{G} = (G, B, \rho)$ be a boundaried graph and $\mathcal{T} = (\mathsf{T}, \beta, r)$ be a tree decomposition of $\mathbf{G}$. Notice that if $a, b \in V(\mathsf{T})$ and $a \in \text{Anc}_{\mathsf{T}, r}(b)$, then G_b is a subgraph of G_a. We define the t_q-boundaried graph $\bar{\mathbf{G}}_q = (\bar{G}_q, \beta(q), \rho_q)$, where $\bar{G}_q = G \setminus (V(G_q) \setminus \beta(q))$. Notice that $\mathbf{G}_q$ and $\bar{\mathbf{G}}_q$ are compatible and $\mathbf{G}_q \oplus \bar{\mathbf{G}}_q = G$.

Linked tree decompositions. Our next step is to use a special type of tree decompositions, namely *linked tree decompositions*, defined by Robertson and Seymour in [62]. Thomas in [71]

proved that every graph G admits a linked tree decomposition of width tw(G) (see also [14, 24]). By combining the result of [71] and [18, Lemma 4], we can consider tree decompositions as asserted in the following result.

PROPOSITION 10.3 ([18]). *Let $t \in \mathbb{N}_{\geq 1}$. For every boundaried graph $\mathbf{G} = (G, B, \rho)$ of treewidth $t - 1$, there exists a tree decomposition $(\mathsf{T}, \beta, \mathsf{r})$ of $\mathbf{G}$ of width $t - 1$ such that*

1. *(T, r) is a binary tree,*
2. *for every $a, b \in V(\mathsf{T})$ where a is a child of b in (T, r), if $|\beta(a)| = |\beta(b)|$ then G_a is a proper subgraph of G_b, i.e., $|V(G_a)| < |V(G_b)|$,*
3. *for every $s \in \mathbb{N}$ and every pair $u_1, u_2 \in V(\mathsf{T})$, where $u_1 \in \mathsf{Anc}_{\mathsf{T},\mathsf{r}}(u_2)$ and $|\beta(u_1)| = |\beta(u_2)|$, either there is an internal vertex w of $u_1 \mathsf{T} u_2$ such that $|\beta(w)| < s$, or there exists a collection of s vertex-disjoint paths in G between $\beta(u_1)$ and $\beta(u_2)$, and*
4. *$|V(G)| \leq t \cdot |V(\mathsf{T})|$.*

In fact, linked tree decompositions are defined as the tree decompositions satisfying only property (3) [62, 71]. In our proofs, we will need the extra properties (1), (2), and (4) that are provided by [18, Lemma 4].

We bound the size of a minor-minimal obstruction of small treewidth in Lemma 10.5. To do so, we need the following result (for a proof see e.g. [31, Lemma 14]).

PROPOSITION 10.4 ([31]). *Let $r, m \in \mathbb{N}_{\geq 1}$ and w be a word of length m^r over the alphabet $[r]$. Then there is a number $k \in [r]$ and a subword u of w such that u contains only numbers not smaller than k and u contains the number k at least m times.*

We are now ready to prove Lemma 10.5. The idea is to apply the technique of Lagergren [52] combined with the bound on the number of characteristics provided in Subsection 7.6. The proof of Lemma 10.5 is very similar to the corresponding proof in [69] for $\mathrm{obs}(\mathcal{A}_k(\mathrm{exc}(\mathcal{F})))$.

LEMMA 10.5. *Let $\mathcal{F}$ be a finite non-trivial collection of graphs. There exists a function f_{16} : $\mathbb{N}^2 \to \mathbb{N}$ such that if k is an integer and G is a graph in $\mathrm{obs}(\mathcal{E}_k(\mathrm{exc}(\mathcal{F})))$ of treewidth tw, then $|V(G)| \leq f_{16}(\mathrm{tw}, k)$. Moreover, $f_{16}(t, k) = 2^{2^{O_{\ell_{\mathcal{F}}}(t^3 + k \cdot t^2)}}$.*

PROOF. Let $G \in \mathrm{obs}(\mathcal{E}_k(\mathrm{exc}(\mathcal{F})))$. We set $t := \mathrm{tw}(G) + 1$. For simplicity, we use ℓ instead of $\ell_{\mathcal{F}}$. We set

$$d = f_{14}(t - 1, k) + 1, \qquad\qquad m = 2^{\binom{t}{2}} \cdot (d - 1) + 1,$$
$$x = m^t, \text{ and} \qquad\qquad b = t \cdot 2^x.$$

It is easy to verify that $b = 2^{2^{O_{\ell_{\mathcal{F}}}(t^3 + k \cdot t^2)}}$.

Suppose towards a contradiction that $|V(G)| > b$. Let (T, β) be a tree decomposition of G of width tw(G) and let $\mathsf{r} \in V(\mathsf{T})$. We consider the rooted tree (T, r) and we set $B := \beta(\mathsf{r})$ and a

bijection $\rho : B \to [|B|]$. We set $\mathbf{G} = (G, B, \rho)$ and observe that (T, β, r) is a tree decomposition of $\mathbf{G}$ of width $\mathrm{tw}(G)$. Since $\mathrm{tw}(\mathbf{G}) = \mathrm{tw}(G) = t - 1$, by Proposition 10.3, we can assume that for the tree decomposition (T, β, r) of $\mathbf{G}$ of width $t - 1$, Properties (1) to (4) are satisfied.

Since $|V(G)| > b = t \cdot 2^x$, Property (4) implies that $|V(\mathsf{T})| > 2^x$. Also, by Property (1), (T, r) is a binary tree and therefore there exists a leaf u of T such that $|V(r\mathsf{T}u)| \geq x$. We set $l := |V(r\mathsf{T}u)|$.

We set $v_1 = r$ and for every $i \in [l-1]$, we set v_{i+1} to be the child of v_i in (T, r) that belongs to $V(rTu)$. Keep in mind that $v_l = u$. For every $i \in [l]$, we set $c_i := |\beta(v_i)|$ and observe that, since (T, β, r) has width $t - 1$, $c_i \in [t]$.

Let C be the word $c_1 \cdots c_x$. Since $x = m^t$ and every $c_i \in [t]$, then, due to Proposition 10.4, there is a $t' \in [t]$ and a subword C' of C such that, for every c in C', $c \geq t'$ and there are at least m numbers in C' that are equal to t'. Therefore, there exists a set $\{z_1, \ldots, z_m\} \subseteq V(\mathsf{T})$ such that for every $i \in [2, m]$, z_i is a descendant of z_{i-1} in (T, r), for every $z' \in V(z_1\mathsf{T}z_m)$ it holds that $|\beta(z')| \geq t'$, and, for every $i \in [m]$, $|\beta(z_i)| = t'$. Hence, Property (3) of the tree decomposition (T, β, r) of $\mathbf{G}$ implies that there exists a collection $\mathcal{P} = \{P_1, \ldots, P_{t'}\}$ of t' vertex-disjoint paths in G between $\beta(z_1)$ and $\beta(z_m)$.

For every $i \in [m]$, let ρ_i be the function mapping a vertex v in $\beta(z_i)$ to the index of the path of $\mathcal{P}$ it intersects, i.e., for every $j \in [t']$, if v is a vertex in $V(P_j) \cap \beta(z_i)$, where $P_j \in \mathcal{P}$, then $\rho_i(v) = j$. Also, for every $i \in [m]$, let $\mathbf{G}_{z_i}$ be the t'-boundaried graph $(G_{z_i}, \beta(z_i), \rho_i)$. Since, $m = 2^{\binom{t}{2}} \cdot (d-1) + 1$, there is a set $J \subseteq [m]$ of size d such that for every $i, j \in J$, the graph $G_{z_i}[\beta(z_i)]$ is isomorphic to the graph $G_{z_j}[\beta(z_j)]$. Therefore, for every $i, j \in J$, $\mathbf{G}_{z_i}$ and $\mathbf{G}_{z_j}$ are compatible. Furthermore, observe that for every $i, j \in J$ with $i \leq j$, $\mathbf{G}_{z_j} \preceq \mathbf{G}_{z_i}$. To see why this holds, for every $i, j \in J$ with $i < j$, let $\mathcal{P}_{i,j}$ be the collection of subpaths of $\mathcal{P}$ between the vertices of $\beta(z_i)$ and $\beta(z_j)$ and consider the graph $G_{z_i}[\beta(z_i)] \cup \bigcup \mathcal{P}_{i,j} \cup G_{z_j}$, which is a subgraph of G_{z_i}. By contracting the edges in $\mathcal{P}_{i,j}$, we obtain a boundaried graph isomorphic to $\mathbf{G}_{z_j}$.

Recall that $|J| = d = f_{14}(t-1, k) + 1$. Thus, by Lemma 7.6, there exist $i, j \in J$ such that j is the smallest element in J that is greater than i and $\mathrm{char}_k(\mathbf{G}_{z_i}) = \mathrm{char}_k(\mathbf{G}_{z_j})$. For simplicity, we set $a := z_i$ and $b := z_j$. Notice that, in G, by contracting the edges of the paths in $\mathcal{P}_{i,j}$ and removing the vertices of G_a that are not vertices of G_b, we obtain a graph isomorphic to $\overline{\mathbf{G}}_a \oplus \mathbf{G}_b$. Therefore, $\overline{\mathbf{G}}_a \oplus \mathbf{G}_b$ is a minor of G. Furthermore, $|V(\overline{\mathbf{G}}_a \oplus \mathbf{G}_b)| < |V(G)|$. To prove this, we argue that G_b is a proper subgraph of G_a. First recall that for every $y \in V(a\mathsf{T}b)$, $|\beta(y)| \geq t'$. If there is a $y \in V(a\mathsf{T}b)$ such that $|\beta(y)| > t'$, then there is a vertex $v \in \beta(y)$ that is a vertex of $V(G_a) \setminus V(G_b)$ and thus G_b is a proper subgraph of G_a, while in the case where for every $y \in V(a\mathsf{T}b)$, $|\beta(y)| = t'$, Property (2) of Proposition 10.3 implies that G_b is a proper subgraph of G_a.

Let $G' := \overline{\mathbf{G}}_a \oplus \mathbf{G}_b$. Since $|V(G')| < |V(G)|$, G' is a minor of G, and $G \in \mathrm{obs}(\mathcal{E}_k(\mathrm{exc}(\mathcal{F})))$, it holds that $G' \in \mathcal{E}_k(\mathrm{exc}(\mathcal{F}))$. By Lemma 7.16, since $\mathrm{char}_k(\mathbf{G}_a) = \mathrm{char}_k(\mathbf{G}_b)$, we have that $\mathrm{ed}_{\mathrm{exc}(\mathcal{F})}(\overline{\mathbf{G}}_a \oplus \mathbf{G}_a) = \mathrm{ed}_{\mathrm{exc}(\mathcal{F})}(\overline{\mathbf{G}}_a \oplus \mathbf{G}_b)$ and therefore $\mathrm{ed}_{\mathrm{exc}(\mathcal{F})}(G) = \mathrm{ed}_{\mathrm{exc}(\mathcal{F})}(G')$. This contradicts the fact that $\mathrm{ed}_{\mathrm{exc}(\mathcal{F})}(G) > k$ and $\mathrm{ed}_{\mathrm{exc}(\mathcal{F})}(G') \leq k$. ∎

11.　Concluding remarks

For a minor-closed graph class $\mathcal{G}$, we proved that VERTEX DELETION TO $\mathcal{G}$ can be solved in time $2^{\mathsf{poly}(k)} \cdot n^2$ and that ELIMINATION DISTANCE TO $\mathcal{G}$ can be solved in time $2^{2^{2^{\mathsf{poly}(k)}}} \cdot n^2$, and in time $2^{2^{c \cdot k^2 \log k}} \cdot n^2$ and $2^{\mathsf{poly}(k)} \cdot n^3$ in the case where the obstruction set of $\mathcal{G}$ contains an apex-graph. Here the degree of poly and c heavily depend on the size of the obstructions of $\mathcal{G}$. An open question is whether $\mathsf{poly}(k)$ could be replaced by $c \cdot k^d$ for some constant c depending on $\mathcal{G}$ and some *universal* constant d (independent of $\mathcal{G}$). We tend to believe that this dependence on $\mathcal{G}$ in the exponent of the polynomial is unavoidable, at least if we want to use the irrelevant vertex technique, and specially our definition of homogeneity.

　　On the other hand, we are not aware, for any of the two considered problems, of any lower bound, assuming the Exponential Time Hypothesis [36], stronger than $2^{o(k)} \cdot n^{O(1)}$, which follows quite easily from known results for VERTEX COVER. Proving stronger lower bounds seems to be quite challenging.

　　Another open problem is whether it is possible to drop the time complexity of ELIMINATION DISTANCE TO $\mathcal{G}$ to $2^{\mathsf{poly}(k)} \cdot n^2$ for *every* minor-closed graph class $\mathcal{G}$. We tend to believe that this should be possible. However, it seems to require to use branching ingeniously and, in particular, to find equivalent instances of ELIMINATION DISTANCE TO $\mathcal{G}$ with a decreasing value of k.

　　As for the polynomial running time of our FPT-algorithms, a priori, nothing prevents the existence of algorithms running in *linear time*, although we are quite far from achieving this. Kawarabayashi [41] presented such a linear FPT-algorithm for the PLANARIZATION problem, heavily relying on the embedding on the resulting planar graph. Extending this technique to general minor-closed classes would require a very compact encoding of the (entangled) structure of minor-free graphs [64] that would be possible to handle in linear time.

　　We also proposed an XP-algorithm for ELIMINATION DISTANCE TO $\mathcal{G}$ parameterized by the treewidth of the input graph (with running time $n^{O(\mathsf{tw}^2)}$). As mentioned in the introduction (see also [4]), the existence of an FPT-algorithm for ELIMINATION DISTANCE TO $\mathcal{G}$, parameterized by treewidth, remains wide open and this is the case even in the very special case where $\mathcal{G}$ contains only the empty graph, where ELIMINATION DISTANCE TO $\mathcal{G}$ is equivalent to the problem of computing treedepth.

　　A last direction is to improve the bounds on the size of the obstructions given in Theorem 1.5. We believe that any substantial improvement should demand novel methodologies that go beyond the irrelevant vertex technique.

References

[1] Isolde Adler, Frederic Dorn, Fedor V. Fomin, Ignasi Sau, and Dimitrios M. Thilikos. Faster parameterized algorithms for minor containment. *Theoretical Computer Science*, 412(50):7018–7028, 2011. DOI (19)

[2] Isolde Adler, Martin Grohe, and Stephan Kreutzer. Computing excluded minors. *Proc. of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 641–650, 2008. URL (4)

[3] Isolde Adler, Stavros G. Kolliopoulos, Philipp Klaus Krause, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Irrelevant vertices for the planar disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 122:815–843, 2017. DOI (14)

[4] 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. *Proc. of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1976–2004, 2022. DOI (5, 6, 72)

[5] 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 (5)

[6] Akanksha Agrawal and M. S. Ramanujan. On the parameterized complexity of clique elimination distance. *Proc. of the 15th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 180 of *LIPIcs*, 1:1–1:13, 2020. DOI (5)

[7] Ernst Althaus and Sarah Ziegler. Optimal tree decompositions revisited: a simpler linear-time fpt algorithm, *Graphs and Combinatorial Optimization: from Theory to Applications*. Volume 5, AIRO Springer Series, pages 67–78. Springer, 2021. DOI (20)

[8] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987. DOI (56)

[9] 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. *Proc. of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 951–970, 2020. DOI (8, 10, 12, 21, 24, 32, 39)

[10] 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)

[11] Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. Hitting minors on bounded treewidth graphs. II. Single-exponential algorithms. *Theoretical Computer Science*, 814:135–152, 2020. DOI (10)

[12] Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. Hitting minors on bounded treewidth graphs. III. Lower bounds. *Journal of Computer and System Sciences*, 109:56–77, 2020. DOI (10)

[13] Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. Hitting minors on bounded treewidth graphs. IV. an optimal algorithm. *SIAM J. Comput.* 52(4):865–912, 2023. DOI (14)

[14] Patrick Bellenbaum and Reinhard Diestel. Two short proofs concerning tree-decompositions. *Combinatorics, Probability and Computing*, 11(6):541–547, 2002. DOI (70)

[15] Hans L. Bodlaender, John R. Gilbert, Ton Kloks, and Hjálmtyr Hafsteinsson. Approximating treewidth, pathwidth, and minimum elimination tree height. *Proc. of the 17th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 570 of *LNCS*, pages 1–12, 1991. DOI (4, 5, 19)

[16] Jannis Bulian and Anuj Dawar. Fixed-parameter tractable distances to sparse graph classes. *Algorithmica*, 79(1):139–158, 2017. DOI (2, 4, 61)

[17] Jannis Bulian and Anuj Dawar. Graph isomorphism parameterized by elimination distance to bounded degree. *Algorithmica*, 75(2):363–382, 2016. DOI (2)

[18] Dimitris Chatzidimitriou, Dimitrios M. Thilikos, and Dimitris Zoros. Sparse obstructions for minor-covering parameters. *Discrete Applied Mathematics*, 278:28–50, 2020. DOI (70)

[19] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Parameterized Algorithms. Springer, 2015. DOI (3)

[20] Reinhard Diestel. Graph Theory, volume 173. Springer-Verlag, 5th edition, 2017. DOI (8, 35, 64)

[21] Rodney G. Downey and Michael R. Fellows. Fundamentals of Parameterized Complexity. Texts in Computer Science. Springer, 2013. DOI (3)

[22] Zdenek Dvorák, Archontia C. Giannopoulou, and Dimitrios M. Thilikos. Forbidden graphs for tree-depth. *European Journal of Combinatorics*, 33(5):969–979, 2012. DOI (7, 67)

[23] Eduard Eiben, Robert Ganian, Thekla Hamm, and O-joung Kwon. Measuring what matters: A hybrid approach to dynamic programming with treewidth. *Journal of Computer and System Sciences*, 121:57–75, 2021. DOI (5)

[24] Joshua Erde. A unified treatment of linked and lean tree-decompositions. *Journal of Combinatorial Theory, Series B*, 130:114–143, 2018. DOI (70)

[25] Jörg Flum and Martin Grohe. Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. DOI (3)

[26] Fedor V. Fomin, Petr A. Golovach, Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. Compound logics for modification problems. *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, 61:1–61:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. DOI (14)

[27] Fedor V. Fomin, Petr A. Golovach, and Dimitrios M. Thilikos. Parameterized complexity of elimination distance to first-order logic properties. *ACM Transactions on Computational Logic*, 23(3):17:1–17:35, 2022. DOI (5)

[28] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar $\mathcal{F}$-Deletion: Approximation, Kernelization and Optimal FPT Algorithms. *Proc. of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 470–479, 2012. DOI (4, 10)

[29] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Hitting topological minors is FPT. *Proc. of the 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1317–1326, 2020. DOI (4, 10)

[30] Archontia C. Giannopoulou, O-joung Kwon, Jean-Florent Raymond, and Dimitrios M. Thilikos. A menger-like property of tree-cut width. *J. Comb. Theory B*, 148:1–22, 2021. DOI (8)

[31] Archontia C. Giannopoulou, Michal Pilipczuk, Jean-Florent Raymond, Dimitrios M. Thilikos, and Marcin Wrochna. Cutwidth: obstructions and algorithmic aspects. *Algorithmica*, 81(2):557–588, 2019. DOI (8, 70)

[32] Archontia C. Giannopoulou, Michal Pilipczuk, Jean-Florent Raymond, Dimitrios M. Thilikos, and Marcin Wrochna. Linear kernels for edge deletion problems to immersion-closed graph classes. *Proc. of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 80 of *LIPIcs*, 57:1–57:15, 2017. DOI (8)

[33] Petr A. Golovach, Giannos Stamoulis, and Dimitrios M. Thilikos. Combing a linkage in an annulus. *SIAM J. Discret. Math.* 37(4):2332–2364, 2023. DOI (14)

[34] Petr A. Golovach, Giannos Stamoulis, and Dimitrios M. Thilikos. Model-checking for first-order logic with disjoint paths predicates in proper minor-closed graph classes. *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 3684–3699. SIAM, 2023. DOI (14)

[35] Jiong Guo, Falk Hüffner, and Rolf Niedermeier. A structural view on parameterizing problems: distance from triviality. *Proc. of the 1st International Workshop on Parameterized and Exact Computation (IWPEC)*, volume 3162 of *LNCS*, pages 162–173, 2004. DOI (2)

[36] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. DOI (72)

[37] Bart M. P. Jansen, Jari J. H. de Kroon, and Michał Włodarczyk. Vertex deletion parameterized by elimination distance and even less. *Proc. of the 53rd Annual ACM-SIGACT Symposium on Theory of Computing (STOC)*, pages 1757–1769, 2021. DOI (5)

[38] Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. A near-optimal planarization algorithm. *Proc. of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1802–1811, 2014. DOI (4)

[39] Mamadou Moustapha Kanté and O-joung Kwon. An upper bound on the size of obstructions for bounded linear rank-width, 2014. URL (8)

[40] Mamadou Moustapha Kanté and O-joung Kwon. Linear rank-width of distance-hereditary graphs II. vertex-minor obstructions. *European Journal of Combinatorics*, 74:110–139, 2018. DOI (8)

[41] Ken-ichi Kawarabayashi. Planarity allowing few error vertices in linear time. *Proc. of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 639–648, 2009. DOI (72)

[42] Ken-ichi Kawarabayashi and Yusuke Kobayashi. Linear min-max relation between the treewidth of an H-minor-free graph and its largest grid minor. *Journal of Combinatorial Theory, Series B*, 141:165–180, 2020. DOI (24)

[43] Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce A. Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, 2012. DOI (4, 22)

[44] Ken-ichi Kawarabayashi, Robin Thomas, and Paul Wollan. A new proof of the flat wall theorem. *Journal of Combinatorial Theory, Series B*, 129:204–238, 2018. DOI (7, 13, 22, 24, 25)

[45] Ken-ichi Kawarabayashi and Paul Wollan. A Shorter Proof of the Graph Minor Algorithm: The Unique Linkage Theorem. *Proc. of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 687–694, 2010. DOI (14, 15)

[46] Eun Jung Kim, Alexander Langer, Christophe Paul, Felix Reidl, Peter Rossmanith, Ignasi Sau, and Somnath Sikdar. Linear kernels and single-exponential algorithms via protrusion decompositions. *ACM Transactions on Algorithms*, 12(2):21:1–21:41, 2016. DOI (4, 10)

[47] Eun Jung Kim, Maria J. Serna, and Dimitrios M. Thilikos. Data-compression for parametrized counting problems on sparse graphs. *Proc. of the 29th International Symposium on Algorithms and Computation (ISAAC)*, volume 123 of *LIPIcs*, 20:1–20:13, 2018. DOI (10)

[48] Ton Kloks. Treewidth, Computations and Approximations, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994. DOI (3)

[49] Tomasz Kociumaka and Marcin Pilipczuk. Deleting Vertices to Graphs of Bounded Genus. *Algorithmica*, 81(9):3655–3691, 2019. DOI (4)

[50] Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. *Proc. of the 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 184–192, 2021. DOI (19)

[51] Jens Lagergren. An upper bound on the size of an obstruction, *Graph Structure Theory*. Volume 147, Contemporary Mathematics, pages 601–621. American Mathematical Society, 1991. DOI (8)

[52] Jens Lagergren. Upper bounds on the size of obstructions and intertwines. *Journal of Combinatorial Theory, Series B*, 73:7–40, 1998. DOI (8, 70)

[53] Jens Lagergren and Stefan Arnborg. Finding minimal forbidden minors using a finite congruence. *Proc. of the 18th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 510 of *LNCS*, pages 532–543, 1991. DOI (8)

[54] John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980. DOI (3)

[55] Dániel Marx and Ildikó Schlotter. Obtaining a planar graph by vertex deletion. *Algorithmica*, 62(3-4):807–822, 2012. DOI (4, 7, 15)

[56] Laure Morelle, Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. Faster Parameterized Algorithms for Modification Problems to Minor-Closed Classes. *Proc. of the 50th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 261 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 93:1–93:19, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. DOI (1)

[57] Jaroslav Nesetril and Patrice Ossona de Mendez. Sparsity - Graphs, Structures, and Algorithms, volume 28 of *Algorithms and combinatorics*. Springer, 2012. DOI (3)

[58] Rolf Niedermeier. Invitation to fixed parameter algorithms, volume 31. Oxford University Press, 2006. DOI (3)

[59] Ljubomir Perkovic and Bruce A. Reed. An improved algorithm for finding tree decompositions of small width. *International Journal of Foundations of Computer Science*, 11(3):365–371, 2000. DOI (56)

[60] Alex Pothen. The complexity of optimal elimination trees. *Technical Report. Pennsylvania State University. Dept. of Computer Science*, 1988. URL (3)

[61] Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. A faster parameterized algorithm for treedepth. *Proc. of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 8572 of *LNCS*, pages 931–942, 2014. DOI (4, 6, 8, 39, 40, 67)

[62] Neil Robertson and Paul D. Seymour. Graph Minors. V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41(2):92–114, 1986. DOI (69, 70)

[63] Neil Robertson and Paul D. Seymour. Graph Minors. XIII. The Disjoint Paths Problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995. DOI (4, 7, 24)

[64] Neil Robertson and Paul D. Seymour. Graph Minors. XVI. Excluding a non-planar graph. *Journal of Combinatorial Theory, Series B*, 89(1):43–76, 2003. DOI (72)

[65] Neil Robertson and Paul D. Seymour. Graph Minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004. DOI (4, 10)

[66] Neil Robertson and Paul D. Seymour. Graph Minors. XXI. Graphs with unique linkages. *Journal of Combinatorial Theory, Series B*, 99(3):583–616, 2009. DOI (14)

[67] Neil Robertson and Paul D. Seymour. Graph Minors. XXII. Irrelevant vertices in linkage problems. *Journal of Combinatorial Theory, Series B*, 102(2):530–563, 2012. DOI (14)

[68] Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. A more accurate view of the Flat Wall Theorem, 2021. URL (7, 11, 13, 14, 22, 24, 25, 27)

[69] Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. k-apices of minor-closed graph classes. i. bounding the obstructions. *J. Comb. Theory B*, 161:180–227, 2023. DOI (7, 8, 11, 14, 15, 22, 25, 29, 31, 70)

[70] 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 (4–7, 10–16, 22, 25, 31, 33, 56, 60)

[71] Robin Thomas. A Menger-like property of tree-width: The finite case. *Journal of Combinatorial Theory, Series B*, 48(1):67–76, 1990. DOI (69, 70)

[72] Andrew Thomason. The extremal function for complete minors. *Journal of Combinatorial Theory, Series B*, 81(2):318–338, 2001. DOI (10)