Deleting, Eliminating and Decomposing to Hereditary Classes Are All FPT-Equivalent

Akanksha Agrawal* Lawqueen Kanesh† Daniel Lokshtanov‡ Fahad Panolan§
M. S. Ramanujan¶ Saket Saurabh‖ Meirav Zehavi**

Abstract

Vertex-deletion problems have been at the heart of parameterized complexity throughout its history. Here, the aim is to determine the minimum size (denoted by $\mathbf{mod}_{\mathcal{H}}$) of a modulator to a graph class $\mathcal{H}$, i.e., a set of vertices whose deletion results in a graph in $\mathcal{H}$. Recent years have seen the development of a research programme where the complexity of modulators is measured in ways other than size. For instance, for a graph class $\mathcal{H}$, the graph parameters elimination distance to $\mathcal{H}$ (denoted by $\mathbf{ed}_{\mathcal{H}}$) [Bulian and Dawar, Algorithmica, 2016], and $\mathcal{H}$-treewidth (denoted by $\mathbf{tw}_{\mathcal{H}}$) [Eiben et al. JCSS, 2021] aim to minimize the treedepth and treewidth, respectively, of the "torso" of the graph induced on a modulator to the graph class $\mathcal{H}$. Here, the torso of a vertex set S in a graph G is the graph with vertex set S and an edge between two vertices $u, v \in S$ if there is a path between u and v in G whose internal vertices all lie outside S.

In this paper, we show that from the perspective of (non-uniform) fixed-parameter tractability (FPT), the three parameters described above give equally powerful parameterizations for every hereditary graph class $\mathcal{H}$ that satisfies mild additional conditions. In fact, we show that for every hereditary graph class $\mathcal{H}$ satisfying mild additional conditions, with the exception of $\mathbf{tw}_{\mathcal{H}}$ parameterized by $\mathbf{ed}_{\mathcal{H}}$, for every pair of these parameters, computing one parameterized by itself or any of the others is FPT-equivalent to the standard vertex-deletion (to $\mathcal{H}$) problem. As an example, we prove that an FPT algorithm for the vertex-deletion problem implies a non-uniform FPT algorithm for computing $\mathbf{ed}_{\mathcal{H}}$ and $\mathbf{tw}_{\mathcal{H}}$.

The conclusions of non-uniform FPT algorithms being somewhat unsatisfactory, we essentially prove that if $\mathcal{H}$ is hereditary, union-closed, CMSO-definable, and (a) the canonical equivalence relation (or any refinement thereof) for membership in the class can be efficiently computed, or (b) the class admits a "strong irrelevant vertex rule", then there exists a uniform FPT algorithm for $\mathbf{ed}_{\mathcal{H}}$. Using these sufficient conditions, we obtain *uniform* FPT algorithms for computing $\mathbf{ed}_{\mathcal{H}}$, when $\mathcal{H}$ is defined by excluding a finite number of connected (a) minors, or (b) topological minors, or (c) induced subgraphs, or when $\mathcal{H}$ is any of bipartite, chordal or interval graphs. For most of these problems, the existence of a uniform FPT algorithm has remained open in the literature. In fact, for some of them, even a non-uniform FPT algorithm was not known. For example, Jansen et al. [STOC 2021] ask for such an algorithm when $\mathcal{H}$ is defined by excluding a finite number of connected topological minors. We resolve their question in the affirmative.

*Indian Institute of Technology Madras, Chennai, India. agrawal@post.bgu.ac.il

†National University of Singapore, Singapore, Singapore, lawqueen@comp.nus.edu.sg

‡University of California Santa Barbara, USA. daniello@ucsb.edu

§Indian Institute of Technology, Hyderabad, India, panolan@iith.ac.in

¶University of Warwick, Coventry, UK, R.Maadapuzhi-Sridharan@warwick.ac.uk

‖The Institute of Mathematical Sciences, HBNI, Chennai, India, and University of Bergen, Bergen, Norway. saket@imsc.res.in

**Ben-Gurion University of the Negev, Beersheba, Israel. meiravze@bgu.ac.il

Contents

1 Introduction

Delineating borders of fixed-parameter tractability (FPT) is arguably the biggest quest of parameterized complexity. Central to this goal is the class of vertex-deletion problems (to a graph class $\mathcal{H}$). In the Vertex Deletion to $\mathcal{H}$ problem, the goal is to compute a minimum set of vertices to delete from the input graph, in order to obtain a graph contained in $\mathcal{H}$. For a graph G, $\mathbf{mod}_{\mathcal{H}}(G)$ denotes the size of a smallest vertex set S such that $G-S \in \mathcal{H}$. If $G-S \in \mathcal{H}$, then S is called a *modulator to* $\mathcal{H}$. The parameterized complexity of vertex-deletion problems has been extensively studied for numerous choices of $\mathcal{H}$, e.g., when $\mathcal{H}$ is planar, bipartite, chordal, interval, acyclic or edgeless, respectively, we get the classical Planar Vertex Deletion, Odd Cycle Transversal, Chordal Vertex Deletion, Interval Vertex Deletion, Feedback Vertex Set, and Vertex Cover problems (see, for example, [23]).

In the study of the parameterized complexity of vertex-deletion problems, solution size and various graph-width measures have been the most frequently considered parameterizations, and over the past three decades their power has been well understood. In particular, numerous vertex-deletion problems are known to be fixed-parameter tractable (i.e., can be solved in time $f(k)n^{\mathcal{O}(1)}$, where k is the parameter and n is the input size) under these parameterizations. In light of this state of the art, recent efforts have shifted to the goal of identifying and exploring "hybrid" parameters such that:

(a) they are upper bounded by the solution size as well as certain graph-width measures,

(b) they can be arbitrarily (and simultaneously) smaller than both the solution size and graph-width measures, and

(c) a significant number of the problems that are in the class FPT when parameterized by solution size or graph-width measures, can be shown to be in the class FPT also when parameterized by these new parameters.

Two recently introduced parameters in this line of research are: (a) $\mathcal{H}$-elimination distance and (b) $\mathcal{H}$-treewidth of G. The $\mathcal{H}$-elimination distance of a graph G to $\mathcal{H}$ (denoted $\mathbf{ed}_{\mathcal{H}}(G)$) was introduced by Bulian and Dawar [11] and roughly speaking, it expresses the number of rounds needed to obtain a graph in $\mathcal{H}$ by removing one vertex from every connected component in each round. We refer the reader to Section 2 for a more formal definition. The reader familiar with the notion of treedepth [64] will be able to see that this closely follows the recursive definition of treedepth. That is, if $\mathcal{H}$ is the class of empty graphs, then the $\mathcal{H}$-elimination distance of G is nothing but the treedepth of G. In fact, if $\mathcal{H}$ is union-closed (*as will be the case for all graph classes we consider in this paper*), then one gets the following equivalent perspective on this notion. The $\mathcal{H}$-elimination distance of G is defined as the minimum possible *treedepth* of the torso of a modulator of G to $\mathcal{H}$. Here, the torso of a vertex set S in a graph G is the graph with vertex set S and an edge between two vertices $u, v \in S$ if there is a path between u and v in G whose internal vertices all lie outside S. Consequently, it is easy to see that $\mathcal{H}$-elimination distance of G is always upper bounded by both the size of the smallest modulator of G to $\mathcal{H}$ (i.e., $\mathbf{mod}_{\mathcal{H}}(G)$) as well as the treedepth of G[1].

The second parameter of interest for us is $\mathcal{H}$-treewidth which, roughly speaking, aims to "generalize" treewidth and solution size, the same way that elimination distance aims to generalize treedepth and solution size. This notion was recently introduced by Eiben et al. [27] and builds on a similar hybrid parameterization which was first developed in the context of solving CSPs [38] and found applications also in algorithms for SAT [37] and Mixed ILPs [36]. Specifically, a tree $\mathcal{H}$-decomposition of a graph G of width ℓ is a tree decomposition of G along with a set $L \subseteq V(G)$ (called base vertices), such that (i) each base vertex appears in exactly

[1]For this, we always assume that $\mathcal{H}$ contains the empty graph and so $V(G)$ is a trivial modulator to $\mathcal{H}$.

one bag, (ii) the base vertices in a bag induce a subgraph belonging to $\mathcal{H}$, and (iii) the number of non-base vertices in any bag is at most $\ell + 1$. The value of $\mathcal{H}$-treewidth of G (denoted $\mathbf{tw}_{\mathcal{H}}(G)$) is the minimum width taken over all tree $\mathcal{H}$-decompositions of G. The utility of this definition arises from the fact that $\mathcal{H}$-treewidth of G is always upper bounded by the treewidth of G (indeed, one could simply take a tree-decomposition of G attaining the treewidth of G and set $L = \emptyset$) and moreover, one can design fixed-parameter algorithms for several problems parameterized by the $\mathbf{tw}_{\mathcal{H}}$ of the graph [27, 47]. On the other hand, for union-closed graph classes $\mathcal{H}$, the $\mathcal{H}$-treewidth of G is nothing but the minimum possible *treewidth* of the torso of a modulator of G to $\mathcal{H}$. This immediately implies that the $\mathcal{H}$-treewidth of G is always upper bounded by both the treewidth of G as well as the size of the smallest modulator of G to $\mathcal{H}$ (i.e., $\mathbf{mod}_{\mathcal{H}}(G)$).

It is fairly easy to see that $\mathbf{ed}_{\mathcal{H}}(G)$ (respectively, $\mathbf{tw}_{\mathcal{H}}(G)$) can be arbitrarily smaller than both $\mathbf{mod}_{\mathcal{H}}(G)$ and the treedepth of G (respectively, the treewidth of G). Similarly, $\mathbf{tw}_{\mathcal{H}}(G)$ itself can be arbitrarily smaller than $\mathbf{ed}_{\mathcal{H}}(G)$. We refer the reader to [39, 48] for some illustrative examples of this fact. As both these parameters satisfy Properties **(a)** and **(b)** listed above, there has been a sustained effort in the last few years to investigate the extent to which Property **(c)** is satisfied by these two parameters. In this effort, one naturally encounters the following two fundamental algorithmic questions:

Question 1: For which families $\mathcal{H}$ of graphs is Elimination Distance to $\mathcal{H}$ (Treewidth Decomposition to $\mathcal{H}$) FPT when parameterized by $\mathbf{ed}_{\mathcal{H}}$ (respectively, $\mathbf{tw}_{\mathcal{H}}$)?

Question 2: For which families $\mathcal{H}$ of graphs is Vertex Deletion to $\mathcal{H}$ parameterized by $\mathbf{ed}_{\mathcal{H}}(G)$ ($\mathbf{tw}_{\mathcal{H}}(G)$) FPT?

In Elimination Distance to $\mathcal{H}$ (Treewidth Decomposition to $\mathcal{H}$), the input is a graph G and integer k and the goal is to decide whether $\mathbf{ed}_{\mathcal{H}}(G)$ (respectively, $\mathbf{tw}_{\mathcal{H}}(G)$) is at most k. The parameter in both problems is k. Both the questions listed above are extremely wide-ranging and challenging. Indeed, for Question 1, notice that not even an XP algorithm (running in time $n^{\mathcal{O}(g(k))}$, where k is $\mathbf{ed}_{\mathcal{H}}$ or $\mathbf{tw}_{\mathcal{H}}$) is obvious even for well-understood graph classes $\mathcal{H}$, such as bipartite graphs. On the other hand, Vertex Deletion to $\mathcal{H}$ in this case has a trivial $n^{\mathcal{O}(k)}$-time algorithm where one simply guesses the minimum modulator to $\mathcal{H}$ and checks whether the graph induced by the rest of the vertices is bipartite, in linear time. In the absence of a resolution to Question 1, Question 2 then brings with it the challenge of solving Vertex Deletion to $\mathcal{H}$ (or indeed, any problem) without necessarily being able to efficiently compute $\mathbf{ed}_{\mathcal{H}}$ or $\mathbf{tw}_{\mathcal{H}}$.

State of the art for Question 1. In their work, Bulian and Dawar [12] showed that the Elimination Distance to $\mathcal{H}$ problem is FPT, when $\mathcal{H}$ is a minor-closed class of graphs and asked whether it is FPT, when $\mathcal{H}$ is the family of graphs of degree at most d. In a partial resolution to this question, Lindermayr et al. [56] showed that Elimination Distance to $\mathcal{H}$ is FPT when we restrict the input graph to be planar. Finally, Agrawal et al. [2], resolved this question completely by showing that the problem is (non-uniformly) FPT (we refer the reader to Section 1.4 for a definition of non-uniform FPT). In fact, they obtained their result for all $\mathcal{H}$ that are characterized by a finite family of induced subgraphs. Recently, Jansen and de Kroon [46] extended the aforementioned result of Agrawal et al. [2] further, and showed that Treewidth Decomposition to $\mathcal{H}$ is also (non-uniformly) FPT for $\mathcal{H}$ that are characterized by a finite family of induced subgraphs. In the same paper they also showed that Treewidth Decomposition to $\mathcal{H}$ (Elimination Distance to $\mathcal{H}$) is non-uniformly FPT for $\mathcal{H}$ being

the family of bipartite graphs. Even more recently, Fomin et al. [30] showed that for every graph family $\mathcal{H}$ expressible by a first order-logic formula Elimination Distance to $\mathcal{H}$ is (non-uniformly) FPT. Since a family of graphs characterized by a finite set of forbidden induced subgraphs is expressible in this fragment of logic, this result also generalizes the result of Agrawal et al. [2]. Until this result of Fomin et al., the research on Question 1 has essentially proceeded on a case-by-case basis, where each paper considers a specific choice of $\mathcal{H}$.

State of the art for Question 2. In a recent paper, Jansen et al. [48] provide a general framework to design *FPT-approximation* algorithms for $\mathbf{ed}_{\mathcal{H}}$ and $\mathbf{tw}_{\mathcal{H}}$ for various choices of $\mathcal{H}$. For instance, when $\mathcal{H}$ is bipartite or characterized by a finite set of forbidden (topological) minors, they give FPT algorithms (parameterized by $\mathbf{tw}_{\mathcal{H}}$) that compute a tree $\mathcal{H}$-decomposition of G whose width is *not necessarily optimal*, but polynomially bounded in the $\mathcal{H}$-treewidth of the input, i.e., an approximation. These approximation algorithms enable them to address Question 2 for various classes $\mathcal{H}$ without having to exactly compute $\mathbf{ed}_{\mathcal{H}}(G)$ or $\mathbf{tw}_{\mathcal{H}}(G)$ (i.e., without resolving Question 1 for these classes). Towards answering Question 2, they give the following FPT algorithms for Vertex Deletion to $\mathcal{H}$ parameterized by $\mathbf{tw}_{\mathcal{H}}$. Let $\mathcal{H}$ be a hereditary class of graphs that is defined by a finite number of forbidden connected (a) minors, or (b) induced subgraphs, or (c) $\mathcal{H} \in \{\mathsf{bipartite\ graphs}, \mathsf{chordal\ graphs}\}$. There is an algorithm that, given an n-vertex graph G, computes a minimum vertex set X such that $G - X \in \mathcal{H}$ in time $f(\mathbf{tw}_{\mathcal{H}}(G)) \cdot n^{\mathcal{O}(1)}$. Note that all of these FPT algorithms are uniform.

1.1 Our Motivation

The starting point of our work lies in the aforementioned recent advances made by Agrawal et al. [2], Fomin et al. [30] and Jansen et al. [48]. A closer look at these algorithms shows an interesting property of these algorithms: known algorithms for Elimination Distance to $\mathcal{H}$ and Treewidth Decomposition to $\mathcal{H}$ utilize the corresponding (known) algorithms for Vertex Deletion to $\mathcal{H}$ in a non-trivial manner. This fact, plus the recent successes in designing (typically, non-uniform) FPT algorithms for Elimination Distance to $\mathcal{H}$ and Treewidth Decomposition to $\mathcal{H}$ naturally raises the following questions.

Question 3: When (if at all) is the parameterized complexity of Elimination Distance to $\mathcal{H}$ or Treewidth Decomposition to $\mathcal{H}$ different from that of Vertex Deletion to $\mathcal{H}$?

Question 4: When (if at all) is the parameterized complexity of Vertex Deletion to $\mathcal{H}$ is different from that of Vertex Deletion to $\mathcal{H}$ parameterized by $\mathbf{ed}_{\mathcal{H}}(G)$, or that of Vertex Deletion to $\mathcal{H}$ parameterized by $\mathbf{tw}_{\mathcal{H}}(G)$?

Question 5: Could one obtain uniform FPT algorithms for Elimination Distance to $\mathcal{H}$ and Treewidth Decomposition to $\mathcal{H}$ for natural families of graphs?

The main objective of the paper is to provide satisfactory answers to Questions 3, 4 and 5.

1.2 Answering Questions 3 and 4: FPT-equivalence of deletion, elimination and decomposition

Roughly speaking, we show:

For Question 3: If $\mathcal{H}$ has certain properties which are also possessed by numerous well-studied graph classes, then there is *no difference* in the parameterized complexity of Elimination Distance to $\mathcal{H}$, Treewidth Decomposition to $\mathcal{H}$, and Vertex Deletion to $\mathcal{H}$. This explains, unifies and extends almost all known results aimed at Question 1 in the literature.

For Question 4: If $\mathcal{H}$ has the properties referred to above, then there is *no difference* in the parameterized complexity of Vertex Deletion to $\mathcal{H}$, Vertex Deletion to $\mathcal{H}$ parameterized by $\mathbf{ed}_{\mathcal{H}}(G)$, and Vertex Deletion to $\mathcal{H}$ parameterized by $\mathbf{tw}_{\mathcal{H}}(G)$. This explains, unifies and extends almost all known results aimed at Question 2 in the literature.

For Question 5: There are several fundamental graph classes $\mathcal{H}$ for which we are able to give sufficiency conditions that imply the *first uniform* FPT algorithms for Elimination Distance to $\mathcal{H}$. In some of these cases, not even a non-uniform FPT algorithm was previously known.

In what follows, we formally state our results and go into more detail about their implications and significance. Towards that we first define a notion of FPT-equivalence. We say that two parameterized problems are (non-uniformly) *FPT-equivalent* if, given an FPT algorithm for any one of the two problems we can obtain a (non-uniform) FPT algorithm for the other problem. Let $\mathcal{H}$ be a family of graphs. Then, by parameterizing Vertex Deletion to $\mathcal{H}$, Elimination Distance to $\mathcal{H}$, and Treewidth Decomposition to $\mathcal{H}$, by any of $\mathbf{mod}_{\mathcal{H}}(G)$, $\mathbf{ed}_{\mathcal{H}}(G)$, and $\mathbf{tw}_{\mathcal{H}}(G)$ we get nine different problems. Our first result shows FPT-equivalences among eight of these.

Theorem 1.1. *$\mathcal{H}$ be a hereditary family of graphs that is* CMSO[2] *definable and closed under disjoint union. Then the following problems are (non-uniformly) FPT-equivalent.*

1. Vertex Deletion to $\mathcal{H}$ *parameterized by* $\mathbf{mod}_{\mathcal{H}}(G)$
2. Vertex Deletion to $\mathcal{H}$ *parameterized by* $\mathbf{ed}_{\mathcal{H}}(G)$
3. Vertex Deletion to $\mathcal{H}$ *parameterized by* $\mathbf{tw}_{\mathcal{H}}(G)$
4. Elimination Distance to $\mathcal{H}$ *parameterized by* $\mathbf{mod}_{\mathcal{H}}(G)$
5. Elimination Distance to $\mathcal{H}$ *parameterized by* $\mathbf{ed}_{\mathcal{H}}(G)$
6. Treewidth Decomposition to $\mathcal{H}$ *parameterized by* $\mathbf{mod}_{\mathcal{H}}(G)$
7. Treewidth Decomposition to $\mathcal{H}$ *parameterized by* $\mathbf{ed}_{\mathcal{H}}(G)$
8. Treewidth Decomposition to $\mathcal{H}$ *parameterized by* $\mathbf{tw}_{\mathcal{H}}(G)$

Notice that because $\mathbf{tw}_{\mathcal{H}}(G) \leq \mathbf{ed}_{\mathcal{H}}(G) \leq \mathbf{mod}_{\mathcal{H}}(G)$, an FPT algorithm for one problem parameterized by a smaller parameter also implies an FPT algorithm for the problem parameterized by the larger parameter. However, the implications in the other direction are surprising and insightful.

1.2.1 Implications of Theorem 1.1

We now describe the various applications and consequences of our first main theorem (Theorem 1.1).

[2]In this paper, when we say CMSO, we refer to the fragment that is sometimes referred to as CMSO_2 in the literature. We refer the reader to Section 2.3 for the formal description of this fragment.

As a classification tool that unifies and extends many known results. Theorem 1.1 is a powerful classification tool which states that as far as the (non-uniform) fixed-parameter tractability of computing any of the parameters $\mathbf{mod}_{\mathcal{H}}(G)$, $\mathbf{ed}_{\mathcal{H}}(G)$ and $\mathbf{tw}_{\mathcal{H}}(G)$ is concerned, they are essentially the "same parameter" for many frequently considered graph classes $\mathcal{H}$. In other words, to obtain an FPT algorithm for any of the problems mentioned in Theorem 1.1, it is sufficient to design an FPT algorithm for the standard vertex-deletion problem, namely, Vertex Deletion to $\mathcal{H}$. This implication unifies several known results in the literature.

For example, let $\mathcal{H}$ be the family of graphs of degree at most d and recall that it was only recently that Agrawal et al. [5] and Jansen et al. [46] showed that Elimination Distance to $\mathcal{H}$ and Treewidth Decomposition to $\mathcal{H}$ are (non-uniformly) FPT, respectively. However, using Theorom 1.1, the fixed-parameter tractability of these two problems and in fact, even the fixed-parameter tractability of Vertex Deletion to $\mathcal{H}$ parameterized by $\mathbf{ed}_{\mathcal{H}}(G)$ (or $\mathbf{tw}_{\mathcal{H}}(G)$), is implied by the straightforward $d^k n^{\mathcal{O}(1)}$-time branching algorithm for Vertex Deletion to $\mathcal{H}$ (i.e., the problem of deleting at most k vertices to get a graph of degree at most d).

Moreover, for various well-studied families of $\mathcal{H}$, we immediately derive FPT algorithms for all combinations of Vertex Deletion to $\mathcal{H}$, Elimination Distance to $\mathcal{H}$, Treewidth Decomposition to $\mathcal{H}$ parameterized by any of $\mathbf{mod}_{\mathcal{H}}(G)$ $\mathbf{ed}_{\mathcal{H}}(G)$ and $\mathbf{tw}_{\mathcal{H}}(G)$, which are covered in Theorem 1.1. For instance, we can invoke this theorem using well-known FPT algorithms for Vertex Deletion to $\mathcal{H}$ for several families of graphs that are CMSO definable and closed under disjoint union, such as families defined by a finite number of forbidden connected (a) minors, or (b) topological minors, or (c) induced subgraphs, or (d) $\mathcal{H}$ being bipartite, chordal, proper-interval, interval, and distance-hereditary; to name a few [73, 14, 15, 13, 28, 33, 31, 49, 58, 63, 70, 71, 72, 54]. Thus, Theorem 1.1 provides a unified understanding of many recent results and resolves the parameterized complexity of several questions left open in the literature.

Of particular significance among the new results is the case where, invoking Theorem 1.1 and taking $\mathcal{H}$ to be a class defined by a finite number of forbidden connected topological minors gives the *first* FPT algorithms for computing $\mathbf{ed}_{\mathcal{H}}$ and $\mathbf{tw}_{\mathcal{H}}$, resolving an open problem posed by Jansen et al. [48].

Deletion to Families of Bounded Rankwidth. We observe that Theorem 1.1 can be invoked by taking $\mathcal{H}$ to be the class of graphs of bounded *rankwidth*, extending a result of Eiben et al. [27].

Rankwidth is a graph parameter introduced by Oum and Seymour [69] to approximate yet another graph parameter called Cliquewidth. The notion of cliquewidth was defined by Courcelle and Olariu [21] as a measure of how "clique-like" the input graph is. One of the main motivations was that several NP-complete problems become tractable on the family of cliques (complete graphs), the assumption was that these algorithmic properties extend to "clique-like" graphs [20]. However, computing cliquewidth and the corresponding cliquewidth decomposition seems to be computationally intractable. This then motivated the notion of rankwidth, which is a graph parameter that approximates cliquewidth well while also being algorithmically tractable [69, 66]. For more information on cliquewidth and rankwidth, we refer to the surveys by Hlinený et al. [43] and Oum [68].

For a graph G, we will use $\mathbf{rwd}(G)$ to denote the rankwidth of G. Let $\eta \geq 1$ be a fixed integer and let $\mathcal{H}_\eta$ denote the class of graphs of rankwidth at most η. It is known that Vertex Deletion to $\mathcal{H}_\eta$ is FPT [22]. The algorithm is based on the fact that for every integer η, there is a finite set $\mathcal{C}_\eta$ of graphs such that for every graph G, $\mathbf{rwd}(G) \leq \eta$ if and only if no vertex-minors of G are isomorphic to a graph in $\mathcal{C}_\eta$ [65, 67]. Further, it is known that vertex-minors can be expressed in CMSO, this together with the fact that we can test whether a graph H is a vertex-minor of G or not in $f(|H|)n^{\mathcal{O}(1)}$ time on graphs of bounded rankwidth leads

to the desired algorithm [22, Theorem 6.11]. It is also important to mention that for Vertex Deletion to $\mathcal{H}_1$, also known as the Distance-Hereditary Vertex-Deletion problem, there is a dedicated algorithm running in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ [28]. For us, two properties of $\mathcal{H}_\eta$ are important: (a) expressibility in CMSO and (b) being closed under disjoint union. These two properties, together with the result in [22] imply that Theorem 1.1 is also applicable to $\mathcal{H}_\eta$. Thus, we are able to generalize and extend the result of Eiben et al. [27], who showed that for every η, computing $\mathbf{tw}_{\mathcal{H}_\eta}$ is FPT. Since we do not need the notion of rankwidth and vertex-minors in this paper beyond this application, we refer the reader for further details to [43, 68].

Beyond graphs: Cut problems. Notice that in the same spirit as we have seen so far, one could also consider the parameterized complexity of other classical problems such as cut problems (e.g., Multiway Cut) as long as the parameter is smaller than the standard parameter studied so far. With this view, we obtain the first such results for several cut problems such as Multiway Cut, Subset FVS and Subset OCT. That is, we obtain FPT algorithms for these problems that is parameterized by a parameter whose value is upper bounded by the standard parameter (i.e., solution size) and which can be arbitrarily smaller. For instance, consider the Multiway Cut problem, where one is given a graph G and a set of vertices S (called terminals) and an integer ℓ and the goal is to decide whether there is a set of at most ℓ vertices whose deletion separates every pair of these terminals. The standard parameterization for this problem is the solution size ℓ. Jansen et al. [48] propose to consider annotated graphs (i.e., undirected graphs with a distinguished set of terminal vertices) and study the parameterized complexity of Multiway Cut parameterized by the elimination distance to a graph where each component has at most one terminal. Notice that this new parameter is always upper bounded by the size of a minimum solution.

Thus, an FPT algorithm for Multiway Cut with such a new parameter would naturally extend the boundaries of tractability for the problem. We are able to obtain such an algorithm by using Theorem 1.1. We then proceed to obtain similar FPT algorithms for the other cut problems mentioned in this paragraph. Recall that in the Subset FVS problem, one is given a graph G, a set S of terminals and an integer ℓ and the goal is to decide whether there is a set of at most ℓ vertices that hits every cycle in G that contains a terminal. Similarly, in the Subset OÇT problem, one is given a graph G, a set S of terminals and an integer ℓ and the goal is to decide whether there is a set of at most ℓ vertices that hits every *odd* cycle in G that contains a terminal.

Building on our new result for Multiway Cut, we obtain an FPT algorithm for Subset FVS parameterized by the elimination distance to a graph where no terminal is part of a cycle, and an FPT algorithm for Subset OCT parameterized by the elimination distance to a graph where no terminal is part of an odd cycle. We summarize all three results as follows:

Theorem 1.2 (Informal version of Theorem 5.1)**.** *The following problems are FPT:*

1. Multiway Cut *parameterized by elimination distance to an annotated graph where each component has at most one terminal.*
2. Subset Feedback Vertex Set *parameterized by elimination distance to an annotated graph where no terminal occurs in a cycle.*
3. Subset Odd Cycle Transversal *parameterized by elimination distance to an annotated graph where no terminal occurs in an odd cycle.*

In fact, we also strengthen the parameterization to an analogue of $\mathcal{H}$-treewidth in the natural way and obtain corresponding results. The details can be found in Section 5. To achieve these results, we use Theorem 1.1. However, note that that Theorem 1.1 is defined only when $\mathcal{H}$ is

a family of graphs. In order to capture problems such as Multiway Cut, Subset FVS and Subset OCT, we express our problems in terms of appropriate notions of structures and then give a reduction to a pure graph problem on which Theorem 1.1 can be invoked.

These results make concrete advances in the direction proposed by Jansen et al. [48] to develop FPT algorithms for Multiway cut parameterized by the elimination distance to a graph where each component has at most one terminal.

1.2.2 Modulators to Scattered Families

Recent years have seen another new direction of research on Vertex Deletion to $\mathcal{H}$ – instead of studying the computation of a modulator to a single family of graphs $\mathcal{H}$, one can aim to compute small vertex sets whose deletion leaves a graph where each connected component comes from a particular pre-specified graph class [40, 45, 44]. As an example of problems in this line of research, consider the following. Given a graph G, and a number k, find a modulator S of size at most k (or decide whether one exists) such that in $G - S$, each connected component is either chordal, or bipartite or planar. Let us call such an S, *a scattered modulator*. Such scattered modulators (if small) can be used to design new FPT algorithms for certain problems by taking separate FPT algorithms for the problems on each of the pre-specified graph classes and then combining them in a non-trivial way "through" the scattered modulator. However, the quality of the modulators considered in this line of research so far has been measured in the traditional way, i.e., in terms of the size. In this paper, by using Theorem 1.1, we initiate a new line of research where, again, it is not the modulator size that is the measure of structure, but in some sense, the treedepth or treewidth of the torso of the graph induced by the modulator. That is, we introduce the first extensions of "scattered modulators" to "scattered elimination distance" and "scattered $\mathcal{H}$-tree decompositions" and obtain results regarding the computation of the corresponding modulators as well as their role in the design of FPT algorithms for other problems (i.e., cross parameterization).

The first study of scattered modulators was undertaken by Ganian et al. [40], who introduced this notion in their work on constraint satisfaction problems. Recently, Jacob et al. [44, 45] initiated the study of scattered modulators explicitly for "scattered" families of graphs. In particular, let $\mathcal{H}_1, \ldots, \mathcal{H}_d$ be families of graphs. Then, the scattered family of graphs $\otimes(\mathcal{H}_1, \ldots, \mathcal{H}_d)$ is defined as the set of all graphs G such that every connected component of G belongs to $\bigcup_{i=1}^{d} \mathcal{H}_i$. That is, each connected component of G belongs to some $\mathcal{H}_i$. As their main result, Jacob et al. [44] showed that Vertex Deletion to $\mathcal{H}$ is FPT whenever Vertex Deletion to $\mathcal{H}_i$, $i \in \{1, \ldots, d\}$, is FPT, and each of $\mathcal{H}_i$ is CMSO expressible. Here, $\mathcal{H}$ is the scattered family $\otimes(\mathcal{H}_1, \ldots, \mathcal{H}_d)$. Notice that if each of $\mathcal{H}_i$ is CMSO expressible then so is $\mathcal{H}$. Further, it is easy to observe that if each of $\mathcal{H}_i$ is closed under disjoint union then so is $\mathcal{H}$. The last two properties together with the result of Jacob et al. [44] enable us to invoke Theorem 1.1 even when $\mathcal{H}$ is a scattered graph family. The effect can be formalized as follows.

Theorem 1.3. *Let $\mathcal{H}_1, \ldots, \mathcal{H}_d$ be hereditary, union-closed,* CMSO *expressible families of graphs such that* Vertex Deletion to $\mathcal{H}_i$ *is* FPT *for every $i \in [d]$. Let $\mathcal{H} = \otimes(\mathcal{H}_1, \ldots, \mathcal{H}_d)$ and $\mathcal{H}' = \bigcup_{i=1}^{d} \mathcal{H}_i$. Then,* Elimination Distance to $\mathcal{H}$ *and* Treewidth Decomposition to $\mathcal{H}$ *are also* FPT.

Notice in the above statement that if we take $\mathcal{H}' = \bigcup_{i=1}^{d} \mathcal{H}_i$, then the size of a modulator to $\mathcal{H}$ is different from that of a smallest modulator to $\mathcal{H}'$, whereas the elimination distance to $\mathcal{H}$ and $\mathcal{H}'$ are the same.

1.3 New Results on Cross Parameterizations

Another popular direction of research in Parameterized Complexity is cross parameterizations: that is parameterization of one problem with respect to alternate parameters. For an illustration, consider Odd Cycle Transversal (OCT) on chordal graphs. Let $\mathcal{H}$ denote the family of chordal graphs. It is well known that OCT is polynomial-time solvable on chordal graphs. Further, given a graph G and a modulator to chordal graphs of size $\mathbf{mod}_{\mathcal{H}}(G)$, OCT admits an algorithm with running time $2^{\mathcal{O}(\mathbf{mod}_{\mathcal{H}}(G))}n^{\mathcal{O}(1)}$. It is therefore natural to ask whether OCT admits an algorithm with running time $f(\mathbf{ed}_{\mathcal{H}}(G))n^{\mathcal{O}(1)}$ or $f(\mathbf{tw}_{\mathcal{H}}(G))n^{\mathcal{O}(1)}$, given an $\mathcal{H}$-elimination forest of G of depth $\mathbf{ed}_{\mathcal{H}}(G)$ and an $\mathcal{H}$-decomposition of G of width $\mathbf{tw}_{\mathcal{H}}(G)$, respectively. The question is also relevant, in fact more challenging, when an $\mathcal{H}$-elimination forest of G of depth $\mathbf{ed}_{\mathcal{H}}(G)$ or an $\mathcal{H}$-decomposition of G of width $\mathbf{tw}_{\mathcal{H}}(G)$ is *not* given. Jansen et al. [48] specifically asked to consider this research direction in their paper. Quoting them:

> "...the elimination distance can also be used as a parameterization away from triviality for solving other parameterized problems Π, when using classes $\mathcal{H}$ in which Π is polynomial-time solvable. This can lead to interesting challenges of exploiting graph structure. For problems which are FPT parameterized by deletion distance to $\mathcal{H}$, does the tractability extend to elimination distance to $\mathcal{H}$? For example, is Undirected Feedback Vertex Set FPT when parameterized by the elimination distance to a subcubic graph or to a chordal graph? The problem is known to be FPT parameterized by the deletion distance to a chordal graph [51] or the edge-deletion distance to a subcubic graph [61]."

A step in this direction can be seen in the work of Eiben et al. [27, Thm. 4]. They present a meta-theorem that yields non-uniform FPT algorithms when Π satisfies several conditions, which require a technical generalization of an FPT algorithm for Π parameterized by deletion distance to $\mathcal{H}$. Here, we avoid resorting to such requirements and instead, provide sufficient conditions *on the problem itself*, which are usually very easily checked and which enables us to obtain fixed-parameter algorithms for vertex-deletion problems (or edge-deletion problems) parameterized by $\mathbf{ed}_{\mathcal{H}}(G)$ ($\mathbf{tw}_{\mathcal{H}}(G)$) when given an $\mathcal{H}$-elimination forest of G of depth $\mathbf{ed}_{\mathcal{H}}(G)$ (respectively, an $\mathcal{H}$-decomposition of G of width $\mathbf{tw}_{\mathcal{H}}(G)$). As a consequence, we resolve the aforementioned open problem of Jansen et al. [48] on the parameterized complexity of Undirected Feedback Vertex Set parameterized by the elimination distance to a chordal graph.

Let us now state an informal version of our result so that we can convey the main message and highlight some consequences without delving into excessive detail at this point. We refer the reader to Section 6 for the formal statement and full proof.

Theorem 1.4 (Informal version of Theorem 6.1)**.** *Let $\mathcal{H}$ be a hereditary family of graphs and Π be a parameterized graph problem satisfying the following properties.*

1. *Π has the property of finite integer index (FII).*
2. *Π is FPT parameterized by $\mathbf{mod}_{\mathcal{H}}(G)$.*
3. *Either, a $\mathcal{H}$-decomposition of G of width $\mathbf{tw}_{\mathcal{H}}(G)$ (or a $\mathcal{H}$-elimination forest of G of depth $\mathbf{ed}_{\mathcal{H}}(G)$) is given; or $\mathcal{H}$ is CMSO definable, closed under disjoint union and Vertex Deletion to $\mathcal{H}$ is FPT.*

Then, Π is FPT parameterized by $\mathbf{ed}_{\mathcal{H}}(G)$ or $\mathbf{tw}_{\mathcal{H}}(G)$.

FII is a technical property satisfied by numerous graph problems and often easily verified. The term FII first appeared in the works of [9, 25] and is similar to the notion of finite state [1, 10, 18].

An intuitive way (though, formally, not correct) to understand FII is as follows. Let Π be a graph problem. Further, for a graph G, let $\mathsf{opt}_\Pi(G)$ denote the optimum (minimum or maximum) value of solution to Π. For example, if Π is Dominating Set then $\mathsf{opt}_\Pi(G)$ denotes the size of a minimum dominating set; and if Π is Cycle Packing then $\mathsf{opt}_\Pi(G)$ denotes the cardinality of a set containing maximum number of pairwise vertex disjoint cycles. In a simplistic way we can say that a graph problem has FII, if for every graph G, and a separation (L, R) $(L \cup R = V(G))$ we have that:

$$\mathsf{opt}_\Pi(G) = \mathsf{opt}_\Pi(G[L \setminus R]) + \mathsf{opt}_\Pi(G[R \setminus L]) \pm h(|L \cap R|).$$

Here, h is a function of the order of separation $(|L \cap R|)$ only. This immediately implies that problems such as Dominating Set and Cycle Packing have FII. In the context of this work, it is sufficient for the reader to know that Vertex Deletion to $\mathcal{H}$ has FII, whenever $\mathcal{H}$ is hereditary, CMSO definable, and closed under disjoint union. We refer the reader to [9, 25, 8] for more details on FII.

Now as a corollary to Theorem 1.4 we get that Undirected Feedback Vertex Set is FPT parameterized by $\mathbf{ed}_\mathcal{H}(G)$ or $\mathbf{tw}_\mathcal{H}(G)$, where $\mathcal{H}$ is a family of chordal graphs, answering the problem posed by Jansen et al. [48]. Similarly, we can show that Dominating Set is FPT parameterized by $\mathbf{ed}_\mathcal{H}(G)$ or $\mathbf{tw}_\mathcal{H}(G)$, where $\mathcal{H}$ is a family of interval graphs.

1.4 Answering Question 5: Towards Uniform FPT Algorithms

The FPT algorithms obtained via Theorems 1.1 and 1.4 and the extension to families of structures are non-uniform. In fact, to the best of our knowledge, all of the current known FPT algorithms for Elimination Distance to $\mathcal{H}$ or Treewidth Decomposition to $\mathcal{H}$, are non-uniform; except for Elimination Distance to $\mathcal{H}$, when $\mathcal{H}$ is the family of empty graphs (which, as discussed earlier, is simply the problem of computing treedepth). However, we note that the FPT-approximation algorithms in Jansen et al. [48] (in fact, all the algorithms obtained in [48]) are uniform.

For the sake of clarity, we formally define the notion of uniform and non-uniform FPT algorithms [26, Definition 2.2.1].

Definition 1.1 (Uniform and non-uniform FPT)**.** *Let Π be a parameterized problem.*

(i) We say that Π is uniformly *FPT if there is an algorithm $\mathcal{A}$, a constant c, and an arbitrary function $f : \mathbb{N} \to \mathbb{N}$ such that: the running time of $\mathcal{A}(\langle x, k\rangle)$ is at most $f(k)|x|^c$ and $\langle x, k\rangle \in \Pi$ if and only if $\mathcal{A}(\langle x, k\rangle) = 1$.*

(ii) We say that Π is non-uniformly *FPT if there is collection of algorithms $\{\mathcal{A}_k : k \in \mathbb{N}\}$, a constant c, and an arbitrary function $f : \mathbb{N} \to \mathbb{N}$, such that : for each $k \in \mathbb{N}$, the running time of $\mathcal{A}_k(\langle x, k\rangle)$ is $f(k)|x|^c$ and $\langle x, k\rangle \in \Pi$ if and only if $\mathcal{A}_k(\langle x, k\rangle) = 1$.*

Towards unification, we first present a general set of demands that, if satisfied, shows that Elimination Distance to $\mathcal{H}$ parameterized by $\mathbf{ed}_\mathcal{H}(G)$ is uniformly FPT. Like before, we consider a hereditary family $\mathcal{H}$ of graphs such that $\mathcal{H}$ is CMSO definable, closed under disjoint union and Vertex Deletion to $\mathcal{H}$ is FPT. However, we strengthen the last two demands. To explain the new requirements, we briefly (and informally) discuss a few notions concerning boundaried graphs and equivalence classes. Essentially, a *boundaried graph* (*a t-boundaried graph*) is a graph with an injective labelling of some of its vertices by positive integers (upper bounded by t). When *gluing* two boundaried graphs G and H, denoted by $G \oplus H$, we just take their disjoint union, and unify vertices having the same label. Concerning some $\mathcal{H}$, two boundaried graphs G_1 and G_2 are equivalent (under the *canonical equivalence relation*) if, for any boundaried graph H, $G_1 \oplus H \in \mathcal{H}$ if and only if $G_2 \oplus H \in \mathcal{H}$. A *refinement of the canonical equivalence relation* (or just a refinement, for short) is an equivalence relation where any two

boundaried graphs considered equivalent, are equivalent according to the canonical equivalence relation (but not vice versa).

Now, the new requirements are to define a refinement R whose number of equivalence classes for t-boundaried graphs is a function of t only, here called a *finite (per t) refinement*, such that:

- R **is closed under disjoint union:** That is, if we have a boundaried graph that belongs to some equivalence class $R \in \mathsf{R}$, then the disjoint union of that boundaried graph and a non-boundaried graph from $\mathcal{H}$ also belongs to R. [Strengthens the closeness under disjoint union property of $\mathcal{H}$.]
- Vertex Deletion to R **is FPT:** We have a uniform FPT-algorithm (with parameters t and k) that, given a t-boundaried graph G, an equivalence class $R \in \mathsf{R}$, and $k \in \mathbb{N}$, decides whether there exists $S \subseteq V(G)$ of size at most k such that $G - S$ belongs to an equivalence class "at least as good as" R. [Strengthens that Vertex Deletion to $\mathcal{H}$ is FPT.]

We prove the following.

Theorem 1.5 (Informal)**.** *Let $\mathcal{H}$ be a hereditary family of graphs and a finite (per t) refinement R satisfying the following properties.*

1. *$\mathcal{H}$ is CMSO definable.*
2. *R is closed under disjoint union.*
3. *Vertex Deletion to R is FPT (parameterized by boundary and solution sizes).*

Then, Elimination Distance to $\mathcal{H}$ *is uniformly* FPT *parameterized by* $\mathbf{ed}_{\mathcal{H}}(G)$.

We also give two conditions that seem easier to implement. Together with $\mathcal{H}$ being CMSO definable and closed under disjoint union, the satisfaction of the two new condition yields the conditions of Theorem 1.5. In particular, they allow the user to not deal with equivalence classes at all, and to deal with boundaried graphs only with respect to a condition posed on obstructions. Roughly speaking, the simpler conditions are as follows.

- **Finite Boundaried Partial-Obstruction Witness Set:** $\mathcal{H}$ admits a characterization by a (possibly infinite) set $\mathbb{O}$ of obstructions as induced subgraphs.[3] Let $\mathcal{O}_t$ be the set of t-boundaried graphs that are subgraphs of obstructions from $\mathbb{O}$. Then, there exists $\mathcal{O}'_t \subseteq \mathcal{O}_t$ of finite size (depending only on t) such that: for any boundaried graph G, if there exists $O \in \mathcal{O}_t$ such that $G \oplus O \notin \mathcal{H}$, then there exists $O' \in \mathcal{O}'_t$ such that $G \oplus O' \notin \mathcal{H}$.
- **A "Strong" Irrelevant Vertex Rule:** There exist families of graphs $\mathcal{X}$ and $\mathcal{Y}$ (possibly $\mathcal{X} = \mathcal{Y}$), such that: *(i)* Each "large" graph in $\mathcal{X}$ contains a "large" graph from $\mathcal{Y}$ as an induced (or not) subgraph. *(ii)* Given a (not boundaried) graph of "large" treewidth, it contains a "large" graph in $\mathcal{X}$ as an induced subgraph. *(iii)* Given $k \in \mathbb{N}$ and a (not boundaried) graph that contains a "large" graph in $\mathcal{X}$ as an induced subgraph, which, in turn, (by condition (i)), contains a "large" graph Y in $\mathcal{Y}$ as an induced subgraph, "almost all" (depending on k) of the vertices in Y are *k-irrelevant.*[4]

The reason why we claim that these conditions are "simple" is that known algorithms (for the problems considered here) already implicitly yield them as part of their analysis. So, the satisfaction of these conditions do not seem (in various cases) to require much "extra" work compared to the design of an FPT algorithm (or a kernel) to the problem at hand. Using these

[3] Even if the more natural characterization is by forbidden minors/topological minors/subgraphs, we can translate this characterization to one by induced subgraphs (which can make a finite obstruction set become infinite).

[4] A vertex v in G is k-irrelevant if the answers to (G, k) and $(G - v, k)$ are the same (as instances of Vertex Deletion to $\mathcal{H}$).

sufficient conditions, we obtain *uniform* FPT algorithms for computing $\mathbf{ed}_{\mathcal{H}}$, when $\mathcal{H}$ is defined by excluding a finite number of connected (a) minors, or (b) topological minors, or (c) induced subgraphs, or when $\mathcal{H}$ is any of bipartite, chordal or interval graphs. For most of these problems, the existence of a uniform FPT algorithm has remained open in the literature. In fact, for some of them, even a non-uniform FPT algorithm was not known.

2 Preliminaries

2.1 Generic Notations

We begin by giving all the basic notations we use in this paper. For a graph G, we use $V(G)$ and $E(G)$ to denote its vertex set and edge set, respectively. For a graph G, whenever the context is clear, we use n and m to denote $|V(G)|$ and $|E(G)|$, respectively. Consider a graph G. For $X \subseteq V(G)$, $G[X]$ denotes the graph with vertex set X and the edge set $\{\{x,y\} \in E(G) \mid x,y \in X\}$. By $G - X$, we denote the graph $G[V(G) \setminus X]$. For a vertex $v \in V(G)$, $N_G(v)$ and $N_G[v]$ denote the set of open neighbors and closed neighbors of v in G. That is, $N_G(v) = \{u \in V(G) \mid \{u,v\} \in E(G)\}$ and $N_G[v] = N_G(v) \cup \{v\}$. For a set $U \subseteq V(G)$, $N_G(U) = \bigcup_{u \in U} N_G(u) \setminus U$ and $N_G[U] = N_G(U) \cup U$. For a vertex set $C \subseteq V(G)$, by slightly abusing the notation, we use $N_G(C)$, and $N_G[C]$ to denote $N(V(C))$, and $N[V(C)]$, respectively. In all the notations above, we drop the subscript G whenever the context is clear.

A *path* $P = (v_1, v_2, \cdots, v_\ell)$ in G is a subgraph of G where $V(P) = \{v_1, v_2, \cdots, v_\ell\} \subseteq V(G)$ is a set of distinct vertices and $E(P) = \{\{v_i, v_{i+1}\} \mid i \in [\ell - 1]\} \subseteq E(G)$, where $|V(P)| = \ell$ for some $\ell \in [|V(G)|]$. The above defined path P is called as $v_1 - v_\ell$ path. We say that the graph G is *connected* if for every $u, v \in V(G)$, there exists a $u - v$ path in G. A *connected component* of G is an inclusion-wise maximal connected induced subgraph of G.

A *rooted tree* is a tree with a special vertex designated to be the *root*. Let T be a rooted tree with root $r \in V(T)$. We say that a vertex $v \in V(T) \setminus \{r\}$ is a *leaf* of T if v has exactly one neighbor in T. Moreover, if $V(T) = \{r\}$, then r is the leaf (as well as the root) of T.[5] A vertex which is not a leaf, is called a *non-leaf* vertex. Let $t, t' \in V(T)$ such that $\{t, t'\} \in E(T)$ and t' is not contained in the $t - r$ path in T, then we say that t is the *parent* of t' and t' is a *child* of t. A vertex $t' \in V(T)$ is a *descendant* of t (t' can possibly be the same as t), if there is a $t - t'$ path in $T - \{\mathsf{par}_T(t)\}$, where $\mathsf{par}_T(t)$ is the parent of t in T. Note that when $t = r$, then $T - \{\mathsf{par}_T(t)\} = T$, as the parent of r does not exist. That is, every vertex in T is a descendant of r.

A *rooted forest* is a forest where each of its connected component is a rooted tree. For a rooted forest F, a vertex $v \in V(F)$ that is not a root of any of its rooted trees is a *leaf* if it is of degree exactly one in F. The *depth* of a rooted tree T is the maximum number of edges in a root to leaf path in T. The *depth* of a rooted forest is the maximum depth among its rooted trees.

2.2 Graph Classes and Decompositions

We always assume that $\mathcal{H}$ is a hereditary class of graphs, that is, closed under taking induced subgraphs. A set $X \subseteq V(G)$ is called an $\mathcal{H}$-deletion set if $G - X \in \mathcal{H}$. The task of finding a smallest $\mathcal{H}$-deletion set is called the $\mathcal{H}$-DELETION problem (also referred to as $\mathcal{H}$-VERTEX DELETION but we abbreviate it since we do not consider edge deletion problems). Next, we give the notion of *elimination-distance* introduced by Bulian and Dawar [11]. We rephrase their definition, and our definition is (almost) in-line with the equivalent definition given by Jansen et al. [48].

[5] A root is not a leaf in a tree, if the tree has at least two vertices.

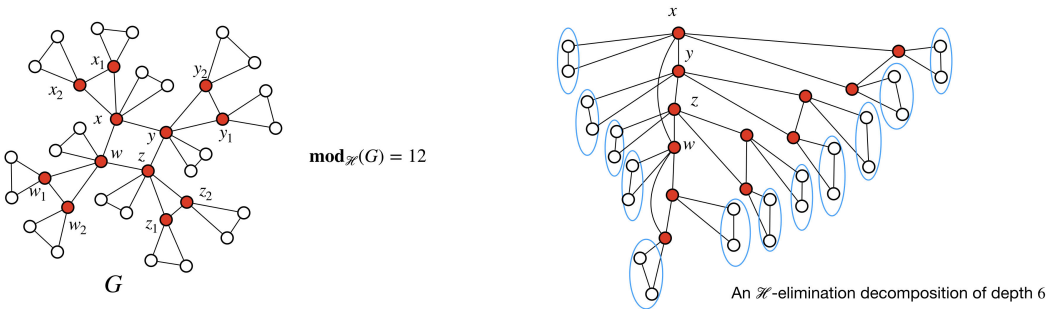

Figure 1: Let $\mathcal{H}$ be the family of triangle-free graphs. The figure shows a graph G with a modulator to $\mathcal{H}$ of size 12 (in red), and an $\mathcal{H}$-elimination decomposition of depth 6.

Definition 2.1. For a graph class $\mathcal{H}$, an *$\mathcal{H}$-elimination decomposition* of graph G is pair (T, χ, L) where T is a rooted forest and $\chi\colon V(T) \to 2^{V(G)}$ and $L \subseteq V(G)$, such that:

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

The *depth* of T is the maximum number of **edges** on a root-to-leaf path (see Figure 1). We refer to the union of base components as the set of base vertices. The $\mathcal{H}$-elimination distance of G, denoted $\mathbf{ed}_{\mathcal{H}}(G)$, is the minimum depth of an $\mathcal{H}$-elimination forest for G. A pair (T, χ) is a (standard) elimination forest if $\mathcal{H}$ is the class of empty graphs, i.e., the base components are empty. The treedepth of G, denoted $\mathbf{td}(G)$, is the minimum depth of a standard elimination forest.

It is straight-forward to verify that for any G and $\mathcal{H}$, the minimum depth of an $\mathcal{H}$-elimination forest of G is equal to the $\mathcal{H}$-elimination distance as defined recursively in the introduction. (This is the reason we have defined the depth of an $\mathcal{H}$-elimination forest in terms of the number of edges, while the traditional definition of treedepth counts vertices on root-to-leaf paths.) The following definition captures the relaxed notion of tree decomposition.

Definition 2.2 ([48])**.** For a graph class $\mathcal{H}$, an *$\mathcal{H}$-tree decomposition* of graph G is a triple (T, χ, L) where $L \subseteq V(G)$, T is a rooted tree, and $\chi\colon V(T) \to 2^{V(G)}$, such that:

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

The *width* of a tree $\mathcal{H}$-decomposition is defined as $\max(0, \max_{t \in V(T)} |\chi(t) \setminus L| - 1)$. The $\mathcal{H}$-treewidth of a graph G, denoted $\mathbf{tw}_{\mathcal{H}}(G)$, is the minimum width of a tree $\mathcal{H}$-decomposition of G. The connected components of $G[L]$ are called base components and the vertices in L are called base vertices.

A pair (T, χ) is a (standard) *tree decomposition* if $(T, \chi, \emptyset)$ satisfies all conditions of an $\mathcal{H}$-decomposition; the choice of $\mathcal{H}$ is irrelevant.

In the definition of width, we subtract one from the size of a largest bag to mimic treewidth. The maximum with zero is taken to prevent graphs $G \in \mathcal{H}$ from having $\mathbf{tw}_{\mathcal{H}}(G) = -1$.

Topological minors: Roughly speaking, a graph H is a *topological minor* of G if G contains a subgraph G' on $|V(H)|$ vertices, where the edges in H correspond to (vertex disjoint) paths in G'. We now formally define the above. Let $\mathsf{Paths}(G)$ be the set of all paths in G. We say that a graph H is a *topological minor* of G if there are injective functions $\phi : V(H) \to V(G)$ and $\varphi : E(H) \to \mathsf{Paths}(G)$ such that (i) for all $e = \{h, h'\} \in E(H)$, $\phi(h)$ and $\phi(h')$ are the endpoints of $\varphi(e)$, (ii) for distinct $e, e' \in E(H)$, the paths $\varphi(e)$ and $\varphi(e')$ are internally vertex-disjoint, and (iii) there does not exist a vertex v such that v is in the image of ϕ and there is an edge $e \in E(H)$ where v is an internal vertex in the path $\varphi(e)$.

Unbreakable Graphs. To formally introduce the notion of unbreakability, we rely on the definition of a separation:

Definition 2.3. [**Separation**] *A pair (X, Y) where $X \cup Y = V(G)$ is a* separation *if $E(X \setminus Y, Y \setminus X) = \emptyset$. The order of (X, Y) is $|X \cap Y|$.*

Roughly speaking, a graph is breakable if it is possible to "break" it into two large parts by removing only a small number of vertices. Formally,

Definition 2.4. [**(s, c)-Unbreakable graph**] *Let G be a graph. If there exists a separation (X, Y) of order at most c such that $|X \setminus Y| \geq s$ and $|Y \setminus X| \geq s$, called an* (s, c)-witnessing separation*, then G is* (s, c)-breakable*. Otherwise, G is* (s, c)-unbreakable*.*

We next state an observation which immediately follows from the above definition.

Observation 2.5. *Consider a graph G, an integer k and a set $S \subseteq V(G)$ of size at most k, where G is $(\alpha(k), k)$-unbreakable graph with $|V(G)| > 2\alpha(k) + k$. Then, there is exactly one connected component C^* in $G - S$ that has at least $\alpha(k)$ vertices and $|V(G) \setminus V(C^*)| < \alpha(k) + k$.*

2.3 Counting Monadic Second Order Logic

The syntax of Monadic Second Order Logic (MSO) of graphs includes the logical connectives $\vee$, $\wedge$, $\neg$, $\Leftrightarrow$, $\Rightarrow$, variables for vertices, edges, sets of vertices and sets of edges, the quantifiers $\forall$ and $\exists$, which can be applied to these variables, and five binary relations:

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

Counting Monadic Second Order Logic (CMSO) extends MSO by including atomic sentences testing whether the cardinality of a set is equal to q modulo r, where q and r are integers such that $0 \leq q < r$ and $r \geq 2$. That is, CMSO is MSO with the following atomic sentence: $\mathbf{card}_{q,r}(S) = \mathbf{true}$ if and only if $|S| \equiv q \pmod{r}$, where S is a set. We refer to [6, 18, 19] for a detailed introduction to CMSO.

We will crucially use the following result of Lokshtanov et al. [60] that allows one to obtain a (non-uniform) FPT algorithm for CMSO-expressible graph problems by designing an FPT algorithm for the problem on unbreakable graphs.

Proposition 2.6 (Theorem 1, [60]). *Let ψ be a CMSO sentence and let $d > 4$ be a positive integer. There exists a function $\alpha : \mathbb{N} \to \mathbb{N}$, such that for every $c \in \mathbb{N}$ there is an $\alpha(c) \in \mathbb{N}$, if there exists an algorithm that solves* CMSO$[\psi]$ *on $(\alpha(c), c)$-unbreakable graphs in time $\mathcal{O}(n^d)$, then there exists an algorithm that solves* CMSO$[\psi]$ *on general graphs in time $\mathcal{O}(n^d)$.*

2.4 Parameterized graph problems

A parameterized graph problem Π is usually defined as a subset of $\Sigma^* \times \mathbb{Z}^+$ where, in each instance (x, k) of Π, x encodes a graph and k is the parameter (we denote by $\mathbb{Z}^+$ the set of all non-negative integers). In this paper we use an extension of this definition (also used by Bodlaender et al. [8] and Fomin et al. [34]) that permits the parameter k to be negative with the additional constraint that either all pairs with non-positive values of the parameter are in Π or that no such pair is in Π. Formally, a parametrized problem Π is a subset of $\Sigma^* \times \mathbb{Z}$ where for all $(x_1, k_1), (x_2, k_2) \in \Sigma^* \times \mathbb{Z}$ with $k_1, k_2 < 0$ it holds that $(x_1, k_1) \in \Pi$ if and only if $(x_2, k_2) \in \Pi$. This extended definition encompasses the traditional one and is needed for technical reasons (see Subsection 2.6). In an instance of a parameterized problem (x, k), the integer k is called the parameter.

2.5 Boundaried Graphs

Here we define the notion of *boundaried graphs* and various operations on them.

Definition 2.7. [**Boundaried Graphs**] *A boundaried graph is a graph G with a set $B \subseteq V(G)$ of distinguished vertices and an injective labelling λ_G from B to the set $\mathbb{Z}^+$. The set B is called the* boundary *of G and the vertices in B are called* boundary vertices *or* terminals. *Given a boundaried graph G, we denote its boundary by $\delta(G)$, we denote its labelling by λ_G, and we define its* label set *by $\Lambda(G) = \{\lambda_G(v) \mid v \in \delta(G)\}$. Given a finite set $I \subseteq \mathbb{Z}^+$, we define $\mathcal{F}_I$ to denote the class of all boundaried graphs whose label set is I. We also denote by $\mathcal{F}$ the class of all boundaried graphs. Finally we say that a boundaried graph is a* t-boundaried *graph if $\Lambda(G) \subseteq \{1, \ldots, t\}$.*

Definition 2.8. [**Gluing by $\oplus$**] *Let G_1 and G_2 be two boundaried graphs. We denote by $G_1 \oplus G_2$ the graph (not boundaried) obtained by taking the disjoint union of G_1 and G_2 and identifying equally-labeled vertices of the boundaries of G_1 and G_2. In $G_1 \oplus G_2$ there is an edge between two vertices if there is an edge between them either in G_1 or in G_2, or both.*

We remark that if G_1 has a label which is not present in G_2, or vice-versa, then in $G_1 \oplus G_2$ we just forget that label.

Definition 2.9. [**Gluing by $\oplus_\delta$**] *The* boundaried gluing operation $\oplus_\delta$ *is similar to the normal gluing operation, but results in a boundaried graph rather than a graph. Specifically $G_1 \oplus_\delta G_2$ results in a boundaried graph where the graph is $G = G_1 \oplus G_2$ and a vertex is in the boundary of G if it was in the boundary of G_1 or of G_2. Vertices in the boundary of G keep their label from G_1 or G_2.*

Let $\mathcal{G}$ be a class of (not boundaried) graphs. By slightly abusing notation we say that a boundaried graph *belongs to a graph class* $\mathcal{G}$ if the underlying graph belongs to $\mathcal{G}$.

Definition 2.10. [**Replacement**] *Let G be a t-boundaried graph containing a set $X \subseteq V(G)$ such that $\partial_G(X) = \delta(G)$. Let G_1 be a t-boundaried graph. The result of* replacing X with G_1 *is the graph $G^\star \oplus G_1$, where $G^\star = G \setminus (X \setminus \partial(X))$ is treated as a t-boundaried graph with $\delta(G^\star) = \delta(G)$.*

2.6 Finite Integer Index

Definition 2.11. [**Canonical equivalence on boundaried graphs.**] *Let* Π *be a parameterized graph problem whose instances are pairs of the form* (G, k)*. Given two boundaried graphs* $G_1, G_2 \in \mathcal{F}$*, we say that* $G_1 \equiv_\Pi G_2$ *if* $\Lambda(G_1) = \Lambda(G_2)$ *and there exists a* transposition constant $c \in \mathbb{Z}$ *such that*

$$\forall (F, k) \in \mathcal{F} \times \mathbb{Z} \qquad (G_1 \oplus F, k) \in \Pi \Leftrightarrow (G_2 \oplus F, k + c) \in \Pi.$$

Here, c *is a function of the two graphs* G_1 *and* G_2*.*

Note that the relation $\equiv_\Pi$ is an equivalence relation. Observe that c could be negative in the above definition. This is the reason we allow the parameter in parameterized problem instances to take negative values.

Next we define a notion of "transposition-minimality" for the members of each equivalence class of $\equiv_\Pi$.

Definition 2.12. [**Progressive representatives [8]**] *Let* Π *be a parameterized graph problem whose instances are pairs of the form* (G, k) *and let* $\mathcal{C}$ *be some equivalence class of* $\equiv_\Pi$*. We say that* $J \in \mathcal{C}$ *is a* progressive representative *of* $\mathcal{C}$ *if for every* $H \in \mathcal{C}$ *there exists* $c \in \mathbb{Z}^-$*, such that*

$$\forall (F, k) \in \mathcal{F} \times \mathbb{Z} \quad (H \oplus F, k) \in \Pi \Leftrightarrow (J \oplus F, k + c) \in \Pi. \tag{1}$$

The following lemma guarantees the existence of a progressive representative for each equivalence class of $\equiv_\Pi$.

Lemma 2.13 ([8])**.** *Let* Π *be a parameterized graph problem whose instances are pairs of the form* (G, k)*. Then each equivalence class of* $\equiv_\Pi$ *has a progressive representative.*

Notice that two boundaried graphs with different label sets belong to different equivalence classes of $\equiv_\Pi$. Hence for every equivalence class $\mathcal{C}$ of $\equiv_\Pi$ there exists some finite set $I \subseteq \mathbb{Z}^+$ such that $\mathcal{C} \subseteq \mathcal{F}_I$. We are now in position to give the following definition.

Definition 2.14. [**Finite Integer Index**] *A parameterized graph problem* Π *whose instances are pairs of the form* (G, k) *has* Finite Integer Index *(or is* FII*), if and only if for every finite* $I \subseteq \mathbb{Z}^+$*, the number of equivalence classes of* $\equiv_\Pi$ *that are subsets of* $\mathcal{F}_I$ *is finite. For each* $I \subseteq \mathbb{Z}^+$*, we define* $\mathcal{S}_I$ *to be a set containing exactly one progressive representative of each equivalence class of* $\equiv_\Pi$ *that is a subset of* $\mathcal{F}_I$*. We also define* $\mathcal{S}_{\subseteq I} = \bigcup_{I' \subseteq I} \mathcal{S}_{I'}$*.*

The proof of next lemma is identical to the one given for [8, Lemma 8.4]

Lemma 2.15 ([8])**.** *Let* $\mathcal{H}$ *family of graphs that is CMSO definable and union closed. Then,* Vertex Deletion to $\mathcal{H}$ *has FII.*

Lemma 2.16. ([46, Lemma 2])*. Let* $\mathcal{H}$ *family of graphs that is CMSO definable and union closed. Then,* Elimination Distance to $\mathcal{H}$ *and* Treewidth Decomposition to $\mathcal{H}$ *is CMSO definable.*

2.7 Replacement lemma

This subsection is verbatim taken from Fomin et al. [34, Section 3.3] and is provided here only for completion. We only need to make few simple modifications to suit our need.

Definition 2.17. *Let* $\mathcal{G}$ *denote the set of all graphs. A graph* parameter *is a function* $\Psi \colon \mathcal{G} \to \mathbb{Z}^+$*. That is,* Ψ *associates a non-negative integer to a graph* $G \in \mathcal{G}$*. The parameter* ψ *is called* monotone*, if for every* $G \in \mathcal{G}$*, and for every* $V_1 \subseteq V_2$*,* $\Psi(G[V_2]) \geq \Psi(G[V_1])$*.*

We can use Ψ to define several graph parameters such as *treewidth*, or given a family $\mathcal{F}$ of graphs, a minimum sized vertex subset S of G, called modulator, such that $G - S \in \mathcal{F}$. Next we define a notion of monotonicity for parameterized problems.

Definition 2.18. ([34, Definition 3.9]). *We say that a parameterized graph problem* Π *is* positive monotone *if for every graph* G *there exists a unique* $\ell \in \mathbb{N}$ *such that for all* $\ell' \in \mathbb{N}$ *and* $\ell' \geq \ell$, $(G, \ell') \in \Pi$ *and for all* $\ell' \in \mathbb{N}$ *and* $\ell' < \ell$, $(G, \ell') \notin \Pi$. *A parameterized graph problem* Π *is* negative monotone *if for every graph* G *there exists a unique* $\ell \in \mathbb{N}$ *such that for all* $\ell' \in \mathbb{N}$ *and* $\ell' \geq \ell$, $(G, \ell') \notin \Pi$ *and for all* $\ell' \in \mathbb{N}$ *and* $\ell' < \ell$, $(G, \ell') \in \Pi$. Π *is monotone if it is either positive monotone or negative monotone. We denote the integer* ℓ *by* $\textsc{Threshold}(G, \Pi)$ *(in short* $\textsc{Thr}(G, \Pi)$*).*

We first give an intuition for the next definition. We are considering monotone functions and thus for every graph G there is an integer k where the answer flips. However, for our purpose we need a corresponding notion for boundaried graphs. If we think of the representatives as some "small perturbation", then it is the max threshold over all small perturbations ("adding a representative = small perturbation"). This leads to the following definition.

Definition 2.19. ([34, Definition 3.10]). *Let* Π *be a monotone parameterized graph problem that has FII and* Ψ *be a graph parameter. Let* $\mathcal{S}_t$ *be a set containing exactly one progressive representative of each equivalence class of* $\equiv_\Pi$ *that is a subset of* $\mathcal{F}_I$*, where* $I = \{1, \ldots, t\}$*. For a* t*-boundaried graph* G*, we define*

$$\begin{aligned} \iota(G) &= \max_{G' \in \mathcal{S}_t} \textsc{Thr}(G \oplus G', \Pi), \\ \mu(G) &= \max_{G' \in \mathcal{S}_t} \Psi(G \oplus G') \end{aligned}$$

The next lemma says the following. Suppose we are dealing with some FII problem and we are given a boundaried graph with boundary size t. We know it has a representative of size $h(t)$ and we want to find this representative. In general finding a representative for a boundaried graph is more difficult than solving the corresponding problem. The next lemma says basically that if we can find "OPT" of a boundaried graph efficiently then we can efficiently find its representative. Here by "OPT" we mean $\iota(G)$, which is a robust version of the threshold function (under adding a representative). And by efficiently we mean as efficiently as solving the problem on normal (unboundaried) graphs.

Lemma 2.20. ([34, Lemma 3.11]). *Let* Π *be a monotone parameterized graph problem that has FII and* Ψ *be a graph parameter. Furthermore, let* $\mathcal{A}$ *be an algorithm for* Π *that, given a pair* (G, k)*, decides whether it is in* Π *in time* $f(|V(G)|, \Psi(G))$*. Then for every* $t \in \mathbb{N}$*, there exists a* $\xi_t \in \mathbb{Z}^+$ *(depending on* Π *and* t*), and an algorithm that, given a* t*-boundaried graph* G *with* $|V(G)| > \xi_t$*, outputs, in* $\mathcal{O}(\iota(G)(f(|V(G)| + \xi_t, \mu(G)))$ *steps, a* t*-boundaried graph* $G^\star$ *such that* $G \equiv_\Pi G^\star$ *and* $|V(G^\star)| < \xi_t$*. Moreover we can compute the translation constant* c *from* G *to* $G^\star$ *in the same time.*

Proof. We give prove the claim for positive monotone problems Π; the proof for negative monotone problems is identical. Let $\mathcal{S}_t$ be a set containing exactly one progressive representative of each equivalence class of $\equiv_\Pi$ that is a subset of $\mathcal{F}_I$, where $I = \{1, \ldots, t\}$, and let $\xi_t = \max_{Y \in \mathcal{S}_t} |V(Y)|$. The set $\mathcal{S}_t$ is hardwired in the description of the algorithm. Let $Y_1, \ldots, Y_\rho$ be the set of progressive representatives in $\mathcal{S}_t$. Let $\mathcal{F}_t = \mathcal{F}_I$. Our objective is to find a representative Y_ℓ for G such that

$$\forall (F, k) \in \mathcal{F}_t \times \mathbb{Z} \qquad (G \oplus F, k) \in \Pi \Leftrightarrow (Y_\ell \oplus F, k - \vartheta(X, Y_\ell)) \in \Pi. \tag{2}$$

Here, $\vartheta(X, Y_\ell)$ is a constant that depends on G and Y_ℓ. Towards this we define the following matrix for the set of representatives. Let

$$A[Y_i, Y_j] = \textsc{Thr}(Y_i \oplus Y_j, \Pi)$$

The size of the matrix A only depends on Π and t and is also hardwired in the description of the algorithm. Now given G we find its representative as follows.

- Compute the following row vector $\mathcal{X} = [\textsc{Thr}(G \oplus Y_1, \Pi), \ldots, \textsc{Thr}(G \oplus Y_\rho, \Pi))]$. For each Y_i we decide whether $(G \oplus Y_i, k) \in \Pi$ using the assumed algorithm for deciding Π, letting k increase from 1 until the first time $(G \oplus Y_i, k) \in \Pi$. Since Π is positive monotone this will happen for some $k \leq \iota(G)$. Thus the total time to compute the vector $\mathcal{X}$ is $\mathcal{O}(\iota(G)(f(|V(G)| + \xi_t, \mu(G)))$.

- Find a translate row in the matrix $A(\Pi)$. That is, find an integer n_o and a representative Y_ℓ such that

$$\begin{aligned}&[\textsc{Thr}(G \oplus Y_1, \Pi), \textsc{Thr}(G \oplus Y_2, \Pi), \ldots, \textsc{Thr}(G \oplus Y_\rho, \Pi)]\\ = &[\textsc{Thr}(Y_\ell \oplus Y_1, \Pi) + n_0, \textsc{Thr}(Y_\ell \oplus Y_2, \Pi) + n_0, \ldots, \textsc{Thr}(Y_\ell \oplus Y_\rho, \Pi) + n_0]\end{aligned}$$

  Such a row must exist since $\mathcal{S}_t$ is a set of representatives for Π; the representative Y_ℓ for the equivalence class to which G belongs, satisfies the condition.

- Set Y_ℓ to be $G^\star$ and the translation constant to be $-n_0$.

From here it easily follows that $G \equiv_\Pi G^\star$. This completes the proof. □

We remark that the algorithm whose existence is guaranteed by the Lemma 2.20 assumes that the set $\mathcal{S}_t$ of representatives are hardwired in the algorithm. In its full generality we currently donot known of a procedure that for problems having FII outputs such a representative set. Thus, the algorithms using Lemma 2.20 are not uniform.

Next we illustrate a situation in which one can can apply Lemma 2.20 to reduce a portion of a graph. Let $\mathcal{F}$ be a family of interval graphs. Further, let Π be the Dominating Set problem and Ψ denote the modulator to $\mathcal{F}$. That is, given a graph G,

$$\Psi(G) = \min_{S \subseteq V(G), G-S \in \mathcal{F}} |S|.$$

It is possible to show that Dominating Set parameterized by $\Psi(G)$ is FPT. That is, we can design an algorithm that can decide whether an instance (G, k) of Dominating Set is an Yes-instance in time $f(\Psi(G)) \cdot n^{\mathcal{O}(1)}$. In fact, in time $2^{\mathcal{O}(\Psi(G))}n^{\mathcal{O}(1)}$. This implies that if we have a t-boundaried graph G, then we can find a representative of it with respect to Dominating Set in time $2^{\mathcal{O}(\mu(G))}n^{\mathcal{O}(1)}$. We will see its uses in this way in Section 6.

3 Structural Results

3.1 Bounded Modulators on Unbreakable Graphs

In this section we show that for any $(\alpha(k), k)$-unbreakable graph G that has more than $3(\alpha(k)+k)$ vertices and its $\mathcal{H}$-elimination decomposition (resp. $\mathcal{H}$-tree decomposition), (T, χ, L) of depth at most k, we have $|V(G) \setminus L| \leq \alpha(k) + k$ and there is a large connected component in $G[L]$, by proving the following two lemmas.

Lemma 3.1. *Consider a graph G, an integer k, and any $\mathcal{H}$-elimination decomposition (resp. $\mathcal{H}$-tree decomposition) (T, χ, L) of depth (resp. width) at most k (resp. $k-1$), where G is $(\alpha(k), k)$-unbreakable graph. Then, one of the following holds:*

1. $|V(G)| \leq 3(\alpha(k)+k)$, *or*
2. *there is exactly one connected component* C^* *in* $G[L]$ *that has at least* $\alpha(k)$ *vertices, and* $|V(G) \setminus V(C^*)| \leq \alpha(k)+k$.

Towards proving the lemma, we begin by stating a folklore result regarding weighted trees (an explicit proof can be found, for instance, in the full version of [5]).

Proposition 3.2. *Consider a tree* T *and a weight function* $w : V(T) \to \mathbb{N}$, $\tau = \sum_{t \in V(T)} w(t)$ *such that* $1 \leq w(t) \leq 2\tau/5$, *for each* $t \in V(T)$. *Then, there exists a non-leaf vertex* $\widehat{t}$ *in* T *such that the connected components of* $T - \{\widehat{t}\}$ *can be partitioned into two sets* $\mathcal{C}_1$ *and* $\mathcal{C}_2$, *with* $\sum_{C \in \mathcal{C}_1} w(C) \leq 2\tau/3$ *and* $\sum_{C \in \mathcal{C}_2} w(C) \leq 2\tau/3$, *where* $w(C) = \sum_{t \in V(C)} w(t)$.

Next we prove useful lemma(s) about unbreakable graphs.

Lemma 3.3. *Consider a family of hereditary graphs* $\mathcal{H}$. *Furthermore, consider an integer* k, *an* $(\alpha(k), k)$*-unbreakable graph* G *with more than* $3(\alpha(k)+k)$ *vertices, and an* $\mathcal{H}$*-elimination decomposition* (T, χ, L) *of depth at most* k *for* G. *Then,* $G[L]$ *has a connected component with at least* $\alpha(k)$ *vertices.*

Proof. As $\mathcal{H}$ is hereditary, note that G must also admit an $\mathcal{H}$-elimination decomposition (T', χ', L), such that: i) for each $t \in V(T')$ we have $|\chi(t)| \geq 1$, ii) for each $t \in V(T')$, if $\chi(t) \cap L \neq \emptyset$, then $G[\chi(t) \cap L]$ is connected, and iii) the depth of T' is at most the depth of T. Hereafter, we will consider an $\mathcal{H}$-elimination decomposition (T', χ', L) for G that satisfies the above conditions. Towards a contradiction we suppose that the size of each connected component in $G[L]$ is strictly less than $\alpha(k)$. With the above assumption, we will exhibit an $(\alpha(k), k)$-witnessing separation, which will contradict that G is $(\alpha(k), k)$-unbreakable.

Let T^* be the tree obtained from T', by arbitrarily connecting one of the roots in T' to all the other roots. Formally, let $\{r_1, r_2, \cdots, r_p\}$ be the set of roots in T' (note that p must be the number of connected components in T'). We let $V(T^*) = V(T')$ and $E(T^*) = E(T') \cup \{\{r_1, r_i\} \mid i \in [p] \setminus \{1\}\}$. Also, let $\mathsf{wht} : V(T^*) \to \mathbb{N}$, such that for $t \in V(T^*) = V(T')$, we have $\mathsf{wht}(t) = \chi(t)$.

By the construction of T^* and wht, we can obtain that for each $t \in V(T^*)$, $1 \leq \mathsf{wht}(t) < \alpha(k)$ and $\tau = \sum_{t \in V(T^*)} \mathsf{wht}(t) = n$, where $n = |V(G)|$. As $n \geq 3(\alpha(k)+k)$, we have $2\tau/5 > (\alpha(k)+k)$, and hence for each $t \in V(T^*)$, $1 \leq \mathsf{wht}(t) \leq 2\tau/5$. The tree T^* and the weight function wht, satisfies the premises of Proposition 3.2. Thus, using the proposition, there is a non-leaf node t^* in T^* such that connected components in $T^* - \{t^*\}$ can be partitioned into two sets $\mathcal{C}_1$ and $\mathcal{C}_2$, such that for each $j \in [2]$, $\mathsf{wht}(\mathcal{C}_j) = \sum_{C \in \mathcal{C}_j} \mathsf{wht}(C) \leq 2\tau/3 \leq 2n/3 \leq 2(\alpha(k)+k)$, where $\mathsf{wht}(C) = \sum_{t \in V(C)} \mathsf{wht}(t)$.

For $i \in [2]$, let $Z_i = \bigcup_{C \in \mathcal{C}_i} V(C)$ and $\widehat{V}_i = \{v \in V(G) \setminus L \mid \text{ for some } t \in Z_i, \chi(t) = \{v\}\}$. For $i \in [2]$, let $\widehat{U}_i = \cup_{t \in Z_i} (\chi(t) \cap L)$. Let $V_i = \widehat{V}_i \cup \widehat{U}_i$, for each $i \in [2]$. From construction of T^* and wht, we obtain that $|V_1| \leq 2(\alpha(k)+k)$. As $n > 3(\alpha(k)+k)$, we have $|V_2| > \alpha(k)+k$. By similar arguments, we obtain $|V_1| > \alpha(k)+k$. Observe that $V_1 \cup V_2 = V(G) \setminus \chi(t^*)$. Let S be the set containing each vertex $v \in V(G) \setminus L$, such that $t_v \in V(T^*)$, where $\chi(t_v) = \{v\}$, t_v is an ancestor of t^* (possibly $t_v = t^*$). Note that $|S| \leq k$.

Let $Y_1 = V_1 \cup S$ and $Y_2 = V_2 \cup S$. Note that $S = Y_1 \cap Y_2$, $|Y_1 \setminus Y_2| = |Y_1 \setminus S| \geq \alpha(k)$, $|Y_2 \setminus Y_1| = |Y_2 \setminus S| \geq \alpha(k)$, $|S| \leq k$, and $Y_1 \cup Y_2 = V(G)$. Moreover, by the construction of Y_1 and Y_2, we have that there is no edge $\{u, v\} \in E(G)$, such that $u \in Y_1 \setminus S$ and $v \in Y_2 \setminus S$. This implies that (Y_1, Y_2) is a separation of order k in G such that $|Y_1 \setminus Y_2| \geq \alpha(k)$, $|Y_2 \setminus Y_2| \geq \alpha(k)$ This contradicts the assumption that G is $(\alpha(k), k)$-unbreakable. □

Analogous to the above, we can prove a result regarding $\mathcal{H}$-tree decompositions.

Lemma 3.4. *Consider a family of hereditary graphs $\mathcal{H}$. Furthermore, consider an integer $k \geq 2$, an $(\alpha(k), k)$-unbreakable graph G with more than $3(\alpha(k)+k)$ vertices, and an $\mathcal{H}$-tree decomposition (T, χ, L) of width at most $k-1$ for G. Then, $G[L]$ has a connected component with at least $\alpha(k)$ vertices.*

Proof. If $L = V(G)$, then note G must have a connected component that have exactly one connected component with at least $\alpha(k)$ vertices, as G is $(\alpha(k), k)$-unbreakable. Hereafter we assume that $V(G) \setminus L \neq \emptyset$. As $\mathcal{H}$ is hereditary, G must also admit an $\mathcal{H}$-tree decomposition (T', χ', L), such that:

1. for every distinct vertices $t, t' \in V(T')$ with $\chi(t) \cap L = \emptyset$ and $\chi(t') \cap L = \emptyset$, we have $\chi(t) \not\subseteq \chi(t')$,
2. for each $t \in V(T')$ with $\chi(t) \cap L \neq \emptyset$, $\chi(t) \setminus L \subseteq \chi(t_{\mathsf{par}})$, where t_{par} is the parent of t in T'.
3. for each $t \in V(T')$, if $\chi(t) \cap L \neq \emptyset$, then $G[\chi(t) \cap L]$ is connected,
4. the width of T' is at most the width of T, and
5. T' is connected.

Hereafter, we will consider an $\mathcal{H}$-tree decomposition (T', χ', L) for G that satisfies the above conditions. Towards a contradiction we suppose that the size of each connected component in $G[L]$ is strictly less than $\alpha(k)$ vertices. We define a (partition) function $\rho : V(T') \to 2^{V(G)}$ using χ and the properties of (T', χ', L) as follows. For each $t \in V(T')$, where: i) $\chi'(t) \cap L \neq \emptyset$, we set $\rho(t) = \chi'(t) \cap L$, and ii) otherwise, we set $\rho(t) = \chi'(t) \setminus \chi'(t_{\mathsf{par}})$, where t_{par} is the parent (if it exists) of t in T'.[6] Notice that for each $t \in V(T')$, $\rho(t) \neq \emptyset$ (recall $V(G) \setminus L \neq \emptyset$) and $\{\rho(t) \mid t \in V(T')\}$ is a partition of $V(G)$. We define the weight function $\mathsf{wht} : V(T') \to \mathbb{N}$, by setting, for each $t \in V(T')$, $\mathsf{wht}(t) = |\rho(t)|$. By the construction of wht we can obtain that, for each $t \in V(T)$, $1 \leq \mathsf{wht}(t) \leq \max\{\alpha(k) - 1 + k\} \leq \alpha(k) + k$ (recall our assumption that each connected component in $G[L]$ has less than $\alpha(k)$ vertices). Furthermore, we have $\tau = \sum_{t \in V(T')} \mathsf{wht}(t) = n = |V(G)|$.

As $n \geq 3(\alpha(k)+k)$, we have $2\tau/5 > (\alpha(k)+k)$, and hence for each $t \in V(T')$, $1 \leq \mathsf{wht}(t) \leq 2\tau/5$. The tree T' and the weight function wht, satisfies the premises of Proposition 3.2. Thus, there is a non-leaf node t^* in T' such that connected components in $T' - \{t^*\}$ can be partitioned into two sets $\mathcal{C}_1$ and $\mathcal{C}_2$, such that for each $j \in [2]$, $\mathsf{wht}(\mathcal{C}_j) = \sum_{C \in \mathcal{C}_j} \mathsf{wht}(C) \leq 2\tau/3 = 2n/3$, where $\mathsf{wht}(C) = \sum_{t \in V(C)} \mathsf{wht}(t)$. Moreover, $\mathsf{wht}(\mathcal{C}_1), \mathsf{wht}(\mathcal{C}_2) \geq n/3 \geq \alpha(k)+k$. Let $S = \chi'(t^*)$, and note that as t^* is a non-leaf node, we have $|S| \leq k$. For $i \in [2]$, $V_i = \cup_{t \in \mathcal{C}_i} \rho(t)$. Notice that $(S \cup V_1, S \cup V_2)$ is a separation of order k, where $|V_1 \setminus S|, |V_2 \setminus S| \geq \alpha(k)$, which contradicts that G is $(\alpha(k), k)$-unbreakable. □

Using the above two lemmas we obtain the desired result (Lemma 3.1).

Proof of Lemma 3.1. Consider an integer k, an $(\alpha(k), k)$-unbreakable graph G, and an $\mathcal{H}$-elimination decomposition (resp. $\mathcal{H}$-tree decomposition) (T, χ, L) for G, of depth (resp. width) k (resp. $k-1$). If $|V(G)| \leq 3(\alpha(k)+k)$, then the condition required by the lemma is trivially satisfied. Now consider the case when $|V(G)| > 3(\alpha(k)+k)$. Note that G must also admit an $\mathcal{H}$-elimination decomposition (resp. $\mathcal{H}$-tree decomposition), say, (T', χ', L) such that for each $t \in V(T')$, $G[\chi'(t) \cap L]$ is connected. From Lemma 3.3, $G[L]$ has a connected component of size at least $\alpha(k)$, and let D be such a connected component, and $t^* \in V(T')$ be a vertex such that $V(D) \subseteq \chi'(t^*)$. Let $S = \{v \in V(G) \setminus L \mid$ for some $t \in V(T') \setminus \{t^*\}, \chi'(t) = \{v\}$, where t is an ancestor of $t^*\}$ (resp., let $S = \chi'(t^*) \setminus L$). Note that $|S| \leq k$, and $N_G(D) \subseteq$

[6] If t is a root, then $\rho(t) = \chi'(t)$.

S. The above, together with the assumption that G is $(\alpha(k), k)$-unbreakable implies that $V(G) \setminus (S \cup V(D)) < \alpha(k)$. From the above we can obtain that $|V(G) \setminus L| \leq \alpha(k) + k$ and there is exactly one connected component in $G[L]$ that has more than $\alpha(k)$ vertices. This concludes the proof. □

3.2 Computing $\mathcal{H}$-Elimination Decomposition Using its Decision Oracle

The objective of this section is to prove the following lemma.

Lemma 3.5. *Consider an algorithm $\mathcal{A}$, for* Elimination Distance to $\mathcal{H}$, *that runs in time $f(\ell') \cdot g(|V(G')|)$, for an instance (G', ℓ') of the problem.*[7] *Then, for any given graph G on n vertices, we can compute an $\mathcal{H}$-elimination decomposition for G of depth $\ell = \mathbf{ed}_{\mathcal{H}}(G)$, in time bounded by $f(\ell) \cdot g(n) \cdot n^{\mathcal{O}(1)}$.*

Consider a family of graphs $\mathcal{H}$, for which Treewidth Decomposition to $\mathcal{H}$ admits an algorithm, say, $\mathbb{M}_{\mathsf{elm}}$, which given a graph G on n vertices and an integer k, runs in time $f(k) \cdot g(n)$, and output 1 if $\mathbf{ed}_{\mathcal{H}}(G) \leq k$ and 0, otherwise. We design a recurive algorithm that given a graph G and an integer k, and returns an $\mathcal{H}$-elimination decomposition of depth at most k, or returns that no such decomposition exists.

Consider a given graph G and an integer k. We assume that the graph G is connected, as otherwise, we can apply our algorithm for each of its connected components. We will explicitly ensure that, while making recursive calls, we maintain the connectivity requirement. We now state the base cases of our recursive algorithm.

Base Case 1. If $G \in \mathcal{H}$ and $k \geq 0$, then the algorithm returns $(T = (\{r\}, \emptyset), \chi : \{r\} \to 2^{V(G)}, V(G))$, where $\chi(r) = V(G)$, as the $\mathcal{H}$-elimination decomposition of G.

Base Case 2. If Base Case 1 is not applicable and $k \leq 0$, then return that $\mathbf{ed}_{\mathcal{H}}(G) > k$.

For each $v \in V(G)$, let $\mathcal{C}_v$ be the set of connected components in $G - \{v\}$

Base Case 3. If Base Case 1 and 2 are not applicable, and there is no $v \in V(G)$, such that for every $C \in \mathcal{C}_v$, $(C, k-1)$ is a yes-instance of Elimination Distance to $\mathcal{H}$, then return that $\mathbf{ed}_{\mathcal{H}}(G) > k$.

The correctnesses of Base Case 1 and 2 are immediate from their descriptions. If the first two base cases are not applicable, then $k \geq 1$ and $\mathbf{ed}_{\mathcal{H}}(G) \geq 1$ must hold. Thus, for any $\mathcal{H}$-elimination decomposition, say, (T, χ, L) for G, T must have at least one vertex which is not a leaf. The third base case precisely returns that $\mathbf{ed}_{\mathcal{H}}(G) > k$, when the above condition cannot be satisfied, thus its correctness follows. Using $\mathbb{M}_{\mathsf{elm}}$, we can test if $(G, 0)$ is a yes-instance of Elimination Distance to $\mathcal{H}$ in time bounded by $f(k) \cdot g(n)$. Thus, we can test/apply Base Case 1, 2 and 3 in time bounded by $f(k) \cdot g(n) \cdot n^{\mathcal{O}(1)}$. Hereafter we assume that the base cases are not applicable.

Recursive Step. Find a vertex $v^* \in V(G)$, such that for every $C \in \mathcal{C}_{v^*}$, $(C, k-1)$ is a yes-instance of Elimination Distance to $\mathcal{H}$, using $\mathbb{M}_{\mathsf{elm}}$. Such a v^* exists as Base Case 3 is not applicable.

Recursively obtain an $\mathcal{H}$-elimination decomposition (T_C, χ_C, L_C) for the instance $(C, k-1)$, for each $C \in \mathcal{C}_{v^*}$. Let $L^* = \bigcup_{C \in \mathcal{C}_{v^*}} L_C$, and let T^* be the forest defined as follows. We have $V(T^*) = \{r^*\} \cup (\bigcup_{C \in \mathcal{C}_{v^*}} V(T_C))$, where r^* is a new vertex, and $E(T^*)$ contains all edges in $E(T_C)$, for each $C \in \mathcal{C}_{v^*}$, and for each root r in some forest in T_C, for some $\mathcal{C}_{v^*}$, the edge $\{r^*, r\}$ belongs to $E(T^*)$. Finally, let $\chi^* : V(T^*) \to 2^{V(G)}$ be the function such that $\chi^*(r^*) = \{v^*\}$, and for each $C \in \mathcal{C}_{v^*}$ and $a \in V(T_C)$, we have $\chi^*(a) = \chi_C(a)$. Return (T^*, χ^*, L^*) as the $\mathcal{H}$-elimination decomposition for G.

[7] We will use the standard assumption from Parameterized Complexity that the functions f and g are non-decreasing. For more details on this, please see Chapter 1 of the book [23].

The correctness of the above recursive step follows from its description. Moreover, it can be execute in time bounded by $f(k) \cdot g(n) \cdot n^{\mathcal{O}(1)}$, using $\mathcal{E}_{\mathsf{mod}}$.

The overall correctness of the algorithm follows from the correctness of each of its base cases and recursive step. Moreover, as we can always assume that $k \leq n$ and the depth of the recursion tree can be bounded by $k+1$, we can obtain that our algorithm runs in time $f(k) \cdot g(n) \cdot n^{\mathcal{O}(1)}$, using $\mathcal{E}_{\mathsf{mod}}$. By exhibiting the above algorithm, we have obtained a proof of Lemma 3.5.

3.3 Computing $\mathcal{H}$-Tree Decomposition Using its Decision Oracle

The objective of this section is to prove the following lemma.

Lemma 3.6. *Consider an algorithm $\mathcal{A}$, for* Treewidth Decomposition to $\mathcal{H}$, *that runs in time $f(\ell') \cdot g(|V(G')|)$, for an instance (G', ℓ') of the problem.*[8] *Then, for any given graph G on n vertices, we can compute an $\mathcal{H}$-tree decomposition for G of width $\ell = \mathbf{tw}_{\mathcal{H}}$, in time bounded by $\left(f(\ell) \cdot g(n) + \ell^{\mathcal{O}(\ell^2)}\right) \cdot n^{\mathcal{O}(1)} + h(\mathcal{F})$, where $h(\mathcal{F})$ depends only on the family $\mathcal{H}$.*

Consider a family of graphs $\mathcal{H}$, for which Treewidth Decomposition to $\mathcal{H}$ admits an algorithm, say, $\mathcal{T}_{\mathsf{tw}}$, which given a graph G on n vertices and an integer k, runs in time $f(k) \cdot g(n)$, and output 1 if $\mathbf{tw}_{\mathcal{H}}(G) \leq k$ and 0, otherwise. We will assume that $\mathcal{H}$ is not the family of all graphs, otherwise, the problem is trivial, i.e., we can return $(T = (\{t\}, \emptyset), \chi, V(G))$, where $\chi(t) = V(G)$, as the $\mathcal{H}$-tree decomposition of width 0.

We will design an algorithm $\mathcal{D}_{\mathsf{tw}}$ which, for given a graph G on n vertices, will construct an $\mathcal{H}$-tree decomposition for G of $\mathcal{H}$-treewidth $\ell = \mathbf{tw}_{\mathcal{H}}$, in time bounded by $\ell^{\mathcal{O}(\ell^3)} \cdot f(\ell) \cdot |V(G)|^{\mathcal{O}(1)} + h(\mathcal{F})$, where $h(\mathcal{F})$ is a number depending on the family $\mathcal{F}$. Intuitively speaking, we will attach a flower of obstructions on each vertex and check if the resulting graph has its $\mathcal{H}$-treewidth exactly the same as $\mathbf{tw}_{\mathcal{H}}(G)$. If the $\mathcal{H}$-treewidth does not increase, then we will be able to obtain that this vertex can be part of the modulator. We repeat this procedure to identify the vertices $S \subseteq V(G)$ that go the the modulator. After this, we take the torso of $G[S]$ in G, to obtain the graph (to be denoted by) $\widetilde{G}_S$. Then using the known algorithm of Bodlaender [7], we compute a tree decomposition for $\widetilde{G}_S$, using which we construct an $\mathcal{H}$-tree decomposition for G.

We next state an observation that will be useful in constructing an obstruction, i.e., a graph outside $\mathcal{H}$.

Observation 3.7. *There exists a number $h = h(\mathcal{H})$,*[9] *such that we can find a graph $H \notin \mathcal{H}$ in h many steps, where each step can be execute in constant time.*

Proof. We initialise $i = 1$ and do following steps:

1. Construct the set, $\mathcal{G}_i$, that contains all graph on exactly i vertices.
2. For each $H \in \mathcal{G}_i$, check if $(H', 0)$ is a no-instance of Treewidth Decomposition to $\mathcal{H}$, using the algorithm $\mathcal{T}_{\mathsf{tw}}$, and if it is a no-instance, then return the graph H (and exit). Otherwise, increment i by 1 and go to Step 1.

Let $q \geq 1$, such that $\mathcal{H}$ does not contain some graph on (exactly) q vertices. Note that q is well-defined, as $\mathcal{H}$ is not the family of all graphs by our assumption. Notice that at the iteration where $i = q$, we will be able to output a graph that is not in $\mathcal{H}$. Also the number of steps executed by the procedure we described depends only on q (which in turn depends only on $\mathcal{H}$). This concludes the proof. □

[8] Again, we assume that the functions f and g are non-decreasing.

[9] That is, h depends on the family $\mathcal{H}$.

We next state an easy observation using which we can compute $\ell = \mathbf{tw}_{\mathcal{H}}(G)$ with the help of the algorithm $\mathcal{T}_{\mathsf{tw}}$.

Observation 3.8. *For a given graph G on n vertices, we can compute (using $\mathcal{T}_{\mathsf{tw}}$) $\ell = \mathbf{tw}_{\mathcal{H}}(G)$ in time bounded by $\mathcal{O}(n) \cdot f(\mathbf{tw}_{\mathcal{H}}(G)) \cdot g(n)$.*

Proof. We iterate over $i \in \mathbb{N}$ (starting from 0) and check whether (G, i) is a yes-instance of Treewidth Decomposition to $\mathcal{H}$ using $\mathcal{T}_{\mathsf{tw}}$, and stop at the iteration where the instance is a yes-instance. Note that, the iteration i at which we stop, it must hold that $i = \mathbf{tw}_{\mathcal{H}}(G)$. As $\mathbf{tw}_{\mathcal{H}}(G) \leq n$ and f is a non-decreasing function by our assumption, our procedure achieves the claimed running time bound. □

We now move to formal description of our algorithm. We fix an arbitrary ordering of vertices in G, and let $V(G) = \{v_1, v_2, \cdots, v_n\}$. We compute $\ell = \mathbf{tw}_{\mathcal{H}}(G)$, using Observation 3.8. Let H^* be the graph returned by Observation 3.7, and let $V(H^*) = \{u_1^*, u_2^*, \cdots, u_q^*\}$. We will construct a graph G'_i and a set $S_i \subseteq V(G)$, for each $i \in [n]$, where we add a flower of obstruction at v_i (and add v_i to S_i) if and only if after adding such obstructions, the $\mathcal{H}$-treewidth of the resulting graph doesn't change. Formally, we do the following.

1. Set $G'_0 = G$ and $S_0 = \emptyset$.
2. For each $i \in [n]$ (in increasing order), we do the following:
   (a) Initialize $G'_1 = G'_{i-1}$ and $S_i = S_{i-1}$.
   (b) We obtain the graph $\widehat{G}_i$ obtained from G'_{i-1} by adding $k+2$ copies of H at v as follows. For $j \in [k+2]$, let H_j^* be the graph such that $E(H_j^*) = \{u_{1,j}^*, u_{2,j}^*, \cdots, u_{q,j}^*\}$ and $E(H_j^*) = \{\{u_{p,j}^*, u_{r,j}^*\} \mid p, r \in [q], \{u_p^*, u_r^*\} \in E(H)\}$. Furthermore, let $H'_j = H_j^* - \{u_{1j}^*\}$. We let $\widehat{G}_i$ be the graph with $V(\widehat{G}_i) = V(G'_{i-1}) \bigcup \left(\cup_{j \in [k+2]} V(H'_j)\right)$ and $E(\widehat{G}_i) = E(G'_{i-1}) \bigcup \left(\cup_{j \in [k+2]} E(H'_j)\right) \bigcup \left\{\{v_i, u_{p,j}^*\} \mid j \in [k+2], p \in [q] \text{ and } \{u_1^*, u_p^*\} \in E(H)\right\}$.
   (c) Check if $(\widehat{G}_i, \ell)$ is a yes-instance of Treewidth Decomposition to $\mathcal{H}$ using $\mathcal{T}_{\mathsf{tw}}$. If the above is true, then set $G'_i = \widehat{G}_i$ and $S_i = S_{i-1} \cup \{v_i\}$, and otherwise, set $G'_i = G'_{i-1}$ and $S_i = S_{i-1}$.

Next we show that, there is an $\mathcal{H}$-tree decomposition of optimal width for G which puts exactly the vertices in S_n in the modulator.

Lemma 3.9. *There is an $\mathcal{H}$-tree decomposition, $(T, \chi, V(G) \setminus S_n)$, for G of width $\ell = \mathbf{tw}_{\mathcal{H}}(G)$.*

Proof. For each $i \in [n]_0$, the construction of G'_i implies that G'_i must admit an $\mathcal{H}$-tree decomposition (T'_i, χ'_i, L'_i) of width $\mathbf{tw}_{\mathcal{H}}(G)$, such that $S_i = (V(G'_i) \setminus L'_i) \cap \{v_1, v_2, \cdots, v_i\}$.[10] As $\mathcal{H}$ is a hereditary family of graphs, we can obtain that for each $i \in [n]$ and $\chi_i : V(T'_i) \to 2^{V(G)}$, where for $t \in V(T'_i)$, $\chi_i(t) = \chi'_i(t) \cap V(G)$, $(T'_i, \chi_i, L'_i \cap V(G))$ is an $\mathcal{H}$-tree decomposition for G of width $\mathbf{tw}_{\mathcal{H}}(G)$, such that $S_i = V(G) \setminus L'_i$. The above in particular implies that, $(T'_n, \chi_n, V(G) \setminus S_n)$ is an $\mathcal{H}$-tree decomposition for G of width $\mathbf{tw}_{\mathcal{H}}(G)$. This concludes the proof. □

Let $S = S_n$ and $L = V(G) \setminus S$. Let $\widetilde{G}$ be the graph obtained from G with vertex set $V(G)$, by taking a torso with respect to the connected components in $G[L]$. That is, $V(\widetilde{G}) = V(G)$ and for $u, v \in V(\widetilde{G})$, $\{u, v\}$ is an edge in $\widetilde{G}$ if and only if one of the following holds: i) $\{u, v\} \in E(G)$,

[10] Here we use the fact that H^* is a graph that has smallest number of vertices, which does not belong to $\mathcal{H}$. Thus, deletion of v_i from G'_i would imply that each of the newly attached obstructions at v_i (if any) are intersected.

or ii) there is a connected component C in $G - S$ ($= G[L]$), such that $u, v \in N_G(C)$ and $u \neq v$. We let $\widetilde{G}_S = \widetilde{G}[S]$

We have the following observation regarding $\widetilde{G}$, which follows from its construction and Lemma 3.9.

Observation 3.10. *The following properties hold:*

1. *A tuple $(T, \chi, V(G) \setminus S)$ is either an $\mathcal{H}$-tree decomposition for both G and $\widetilde{G}$, or none. Moreover, there is at least one such $\mathcal{H}$-tree decomposition of width $\mathbf{tw}_{\mathcal{H}}(G)$ for $\widetilde{G}$.*
2. *Treewidth of $\widetilde{G}_S$ is $\mathbf{tw}_{\mathcal{H}}(G)$.*

Due to the above observation, it is now enough to compute a tree decomposition of $\widetilde{G}$ of width $\mathbf{tw}_{\mathcal{H}}(G)$, to obtain an $\mathcal{H}$-tree decomposition for G of width $\mathbf{tw}_{\mathcal{H}}(G)$. We next use the following result, which immediately follows as a corollary from the result of Bodlaender et al. [7].

Proposition 3.11 (see, [7] or Theorem 7.17 [23])**.** *There is an algorithm, which given a graph G on n vertices, in time bounded by $\mathsf{tw}(G)^{\mathcal{O}(\mathsf{tw}(G)^2)} \cdot n$, computes a tree decomposition of G of width $\mathsf{tw}(G)$.*

We are now ready to prove Lemma 3.6.

Proof of Lemma 3.6. Consider a graph G on n vertices. We construct the graph $\widetilde{G}$ (and $\widetilde{G}_S$) as described previously. From Observation 3.7 and the constructions of G'_i and S_i, for $i \in [n]$, implies that $\widetilde{G}$ (and $\widetilde{G}_S$) can be constructed in time bounded by $f(\ell) \cdot g(n) \cdot n^{\mathcal{O}(1)} + h(\mathcal{F})$, where $\ell = \mathbf{tw}_{\mathcal{H}}(G)$. Then using Proposition 3.11 we can compute a tree decomposition, $(\widetilde{T}_S, \widetilde{\chi}_S : V(\widetilde{T}_S) \to 2^{V(\widetilde{G}_S)})$ of width ℓ, for $\widetilde{G}_S$ in time bounded by $\ell^{\mathcal{O}(\ell^2)} \cdot n^{\mathcal{O}(1)}$ (see item 2 of Observation 3.10). Recall that $S = V(\widetilde{G}_S)$. For each connected component C in $G - S$, the construction of $\widetilde{G}_S$ implies that $N_G(C)$ ($\subseteq S$) induces a clique in $\widetilde{G}_S$. Thus there must exist $t \in V(\widetilde{T}_S)$ such that $N_G(C) \subseteq \widetilde{\chi}_S(t)$. We construct a tree from $\widetilde{T}_S$ and a function $\chi : V(T) \to 2^{V(G)}$ as follows. Initialize $T = \widetilde{T}_S$ and $\chi = \widetilde{\chi}_S$. For each connected component C in $G - S$, we add a new node t_C and add the edge $\{t_C, t_C^*\}$ to $E(T)$, where t_C^* is an arbitrary selected node (if it exists) in $\widetilde{T}_S$, such that $N_G(C) \subseteq \widetilde{\chi}_S(t_C^*)$.[11] Furthermore, we set $\chi(t_C) = N_G(C) \cup V(C)$. The above construction together with Observation 3.10 implies that $(T, \chi, V(G) \setminus S)$ is an $\mathcal{H}$-tree decomposition for G. Note that we can construct $(T, \chi, V(G) \setminus S)$ in time bounded by $\left(f(\ell) \cdot g(n) + \ell^{\mathcal{O}(\ell^2)}\right) \cdot n^{\mathcal{O}(1)} + h(\mathcal{F})$. This concludes the proof. □

4 Equivalences Among Deletion, Decomposition and Elimination

The overall schema of proof of the theorem is presented in Figure 2. Notice that ones the implications depicted in the figure are obtained, we can conclude the proof of Theorem 1.1. We will next discuss the results that are used to obtain the proof, and we begin with a simple observation which directly follows from the fact that $\mathbf{tw}_{\mathcal{H}}(G) \leq \mathbf{ed}_{\mathcal{H}}(G) \leq \mathbf{mod}_{\mathcal{H}}(G)$.

Observation 4.1. *The following implications hold:*

1. *Statement 3 ⇒ Statement 2 ⇒ Statement 1.*
2. *Statement 6 ⇒ Statement 5 ⇒ Statement 4.*
3. *Statement 9 ⇒ Statement 8 ⇒ Statement 7.*

[11] If t_C^* does not exist, in particular, when $S = \emptyset$, then we just add the node t_C.

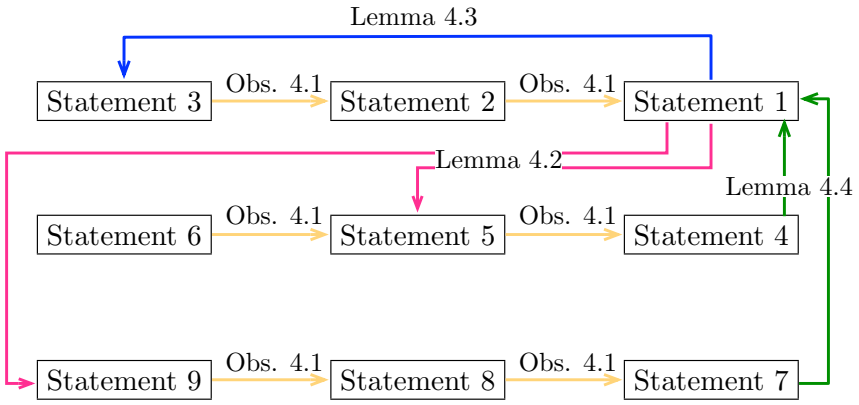

Figure 2: Implications used to obtain proof of Theorem 1.1.

We prove that Statement 1 implies Statement 5 and 9 in Section 4.1, by proving the following lemma.

Lemma 4.2. *Consider a family $\mathcal{H}$ of graphs that is CMSO definable and is closed under disjoint union and induced subgraphs. If* Vertex Deletion to $\mathcal{H}$ *parameterized by* $\mathbf{mod}_{\mathcal{H}}(G)$ *is FPT, then i)* Elimination Distance to $\mathcal{H}$ *parameterized by* $\mathbf{ed}_{\mathcal{H}}(G)$ *is FPT, and ii)* Treewidth Decomposition to $\mathcal{H}$ *parameterized by* $\mathbf{tw}_{\mathcal{H}}(G)$ *is FPT.*

Intuitively speaking, we obtain the proof of the above lemma as follows. Suppose that Vertex Deletion to $\mathcal{H}$, parameterized by $\mathbf{mod}_{\mathcal{H}}(G)$, admits an FPT algorithm, say, $\mathcal{M}_{\mathsf{mod}}$.[12] We will intuitively explain how we obtain an FPT algorithm for Elimination Distance to $\mathcal{H}$ parameterized by $\mathbf{ed}_{\mathcal{H}}(G)$, using $\mathcal{M}_{\mathsf{mod}}$. Consider an $\mathcal{H}$-elimination decomposition (T, χ, L) of depth at most k for G (if it exists). From Lemma 3.1, either the number of vertices in G is bounded by $3(\alpha(k)+k)$, in which we can resolve the instance by a brute-force procedure, or $G[L]$ has exactly one large connected component, denoted by D^*, and $V(G) \setminus V(D^*)$ has size bounded by $\alpha(k)+k$. Let t be the parent (if it exists) of t^*, where t^* is the leaf containing $V(D^*)$. Roughly speaking, we will try to determine the large component D^* completely, and then resolve the remaining instance. To this end, we will maintain a subset, $A^* \subseteq V(G)$, which will also be the subset of vertices from G that are associated with the root-to-t path in T, and thus, we will always have $|S| \leq k$. We will look at the unique large connected component C^* in $G - A^*$ (see Observation 2.5), and try to fix it as much as possible in the following sense. We will (roughly speaking) show that either an arbitrary solution for (C^*, k) as an instance for Vertex Deletion to $\mathcal{H}$, obtained using the assumed algorithm $\mathcal{M}_{\mathsf{mod}}$, is enough for us to completely determine D^*, or we will be able to find a small connected set contained in C^*, with small neighborhood containing an obstruction to $\mathcal{H}$. In the latter case, we will further be able to show that any such maximal connected set (with bounded neighborhood) must have a non-empty intersection with the set of vertices in G associated with the root-to-t path in T. Thus, we will either be able to "grow" our set A^* (upto size at most k), or resolve the instance by brute-force. The above will give us an algorithm as required by the lemma. We note that the algorithm for the case of Treewidth Decomposition to $\mathcal{H}$ parameterized by $\mathbf{tw}_{\mathcal{H}}(G)$ can be obtained in a very similar fashion, but for this case, we will maintain that the set S is

[12] For ease in readability, as much as possible, we will use the letters $\mathcal{M}, \mathcal{E}$ and $\mathcal{T}$ for algorithms for the problems Vertex Deletion to $\mathcal{H}$, Elimination Distance to $\mathcal{H}$ and Treewidth Decomposition to $\mathcal{H}$, respectively. Moreover, the subscripts mod, elm and tw will denote the parameterizations $\mathbf{mod}_{\mathcal{H}}(G)$, $\mathbf{ed}_{\mathcal{H}}(G)$ and $\mathbf{tw}_{\mathcal{H}}(G)$, respectively. We note that we aren't fixing such algorithms, but whenever a need to assume/obtain such algorithms arises, we will be using the above letters/subscripts.

the subset of vertices present in the bag of $\mathcal{H}$-tree decomposition, that contains all the vertices in the large component in $G[L]$.

In Section 4.2 we show that Statement 1 implies Statement 3, assuming that Lemma 4.2 holds, by proving the following result.

Lemma 4.3. *Consider a family $\mathcal{H}$ of graphs that is CMSO definable and is closed under disjoint union and induced subgraphs. If* Vertex Deletion to $\mathcal{H}$ *parameterized by* $\mathbf{mod}_{\mathcal{H}}(G)$ *is FPT, then the problem is also FPT when parameterized by* $\mathbf{tw}_{\mathcal{H}}(G)$.

Intuitively speaking, we obtain a proof of the above lemma as follows. Suppose that Vertex Deletion to $\mathcal{H}$ parameterized by $\mathbf{mod}_{\mathcal{H}}(G)$ admits an FPT algorithm, say, $\mathbb{M}_{\mathsf{mod}}$. Consider an instance (G, k) of the problem Vertex Deletion to $\mathcal{H}$. From Lemma 4.2, we can obtain that Treewidth Decomposition to $\mathcal{H}$ parameterized by $\mathbf{tw}_{\mathcal{H}}(G)$ has an FPT algorithm, say, $\mathbb{T}_{\mathsf{tw}}$. Using the algorithm $\mathbb{T}_{\mathsf{tw}}$ and Lemma 3.6, we compute an $\mathcal{H}$-tree decomposition for G. For each leaf t in T, where the graph $G[\chi(t) \cap L]$ has large number of vertices, we replace $G[\chi(t) \cap L]$ by another graph, using Lemma 2.20, still maintaining equivalence without increasing the parameter. After this, we are obtain to bound the (standard) treewidth of the graph, and resolve the instance using Courcelle's Theorem [18]. The above gives us an FPT algorithm for Vertex Deletion to $\mathcal{H}$, when parameterized by $\mathbf{tw}_{\mathcal{H}}$.

In Section 4.3 we prove that Statement 4 (resp. Statement 7) implies Statement 1, by proving the following lemma.

Lemma 4.4. *Consider a family $\mathcal{H}$ of graphs that is CMSO definable and is closed under disjoint union and induced subgraphs. If* Elimination Distance to $\mathcal{H}$ *(resp.* Treewidth Decomposition to $\mathcal{H}$*) parameterized by* $\mathbf{mod}_{\mathcal{H}}(G)$ *is FPT, then* Vertex Deletion to $\mathcal{H}$ *parameterized by* $\mathbf{mod}_{\mathcal{H}}(G)$ *is also FPT.*

Roughly speaking, the above lemma is proved as follows. Consider an instance (G, k) of Vertex Deletion to $\mathcal{H}$. From Proposition 2.6, it is enough for us to focus in $(\alpha(k), k)$-unbreakable graphs, and thus we assume that G is such a graph. If G has at most $2\alpha(k) + k$ vertices, we resolve the instance by trying all possible subsets. Otherwise, using an assumed FPT algorithm $\mathbb{X}_{\mathsf{mod}}$ for Elimination Distance to $\mathcal{H}$ (resp. Treewidth Decomposition to $\mathcal{H}$) parameterized by $\mathbf{mod}_{\mathcal{H}}(G)$, we compute an $\mathcal{H}$-elimination decomposition (resp. $\mathcal{H}$-tree decomposition), say, (T, χ, L). Now using Observation 2.5 we will to able to conclude that: i) there is exactly one connected component C^* in $G[L]$ which has more than $\alpha(k)$ vertices, ii) $S^* = N_G(C^*)$ has size at most k, and iii) $Z^* = V(G) \setminus V(C^*)$ has size at most $\alpha(k) + k$. Using the above, we are either able branch on vertices of Z^*, or conclude that S^* is already a solution for the given Vertex Deletion to $\mathcal{H}$ instance. The above gives us an algorithm for Vertex Deletion to $\mathcal{H}$, when parameterized by $\mathbf{mod}_{\mathcal{H}}(G)$, using the assumed FPT algorithm for Elimination Distance to $\mathcal{H}$ (resp. Treewidth Decomposition to $\mathcal{H}$) parameterized by $\mathbf{mod}_{\mathcal{H}}(G)$.

4.1 Proof of Lemma 4.2

The objective of this section is to prove Lemma 4.2. We will present the result for Elimination Distance to $\mathcal{H}$, and later comment how we can adapt exactly the same idea for Treewidth Decomposition to $\mathcal{H}$. Fix any family $\mathcal{H}$ of graphs that is CMSO definable and is hereditary, such that Vertex Deletion to $\mathcal{H}$ admits an FPT algorithm, say, $\mathbb{M}_{\mathsf{mod}}$, which given an instance (G', ℓ), where G' is an n vertex graph, correctly resolves the instance in time bounded by $f_{\mathsf{mod}}(\ell) \cdot n^{\mathcal{O}(1)}$.

Let (G, k) is an instance of the problem Elimination Distance to $\mathcal{H}$. From Proposition 2.6, it is enough for us to design an algorithm for $(\alpha(k), k)$-unbreakable graphs, and thus, we

assume that G is $(\alpha(k),k)$-unbreakable. If G has at most $3(ak+k)$ vertices, then we can resolve the instance in FPT time, by brute force. Thus, the interesting case is when G has more than $3(ak+k)$ vertices and it is $(\alpha(k),k)$-unbreakable. We will begin by defining an extension version of the problem for (large) unbreakable graphs, called α-Extension $\mathcal{H}$-Elimination Distance (α-Ext $\mathcal{H}$-ED, for short), which will lie at the heart of our FPT algorithm for Elimination Distance to $\mathcal{H}$ (using $\mathbb{M}_{\mathsf{mod}}$ as a subroutine). Roughly speaking, the problem α-Ext $\mathcal{H}$-ED will (recursively) try to compute (some of) the vertices that will be mapped to the root-to-leaf path leading the the large connected component (see Lemma 3.1), which will be enough for us to identify the large connected component in a decomposition. Once we have the above set, we will be able to determine the unique large connected component in the final decomposition, and then solve the remainder of the problem using brute force as the number of vertices outside the large connected component can be bounded by a function of k. We would like to remark that, an intuitive level the above illustrates how the deletion and the elimination problems coincide.

(Problem Definition) α-Extension $\mathcal{H}$-Elimination Distance (α-Ext $\mathcal{H}$-ED)
Input: A graph G, an integer k, a set $A^* \subseteq V(G)$ of size at most k, such that G is an $(\alpha(k),k)$-unbreakable graph and $|V(G)| > 3(\alpha(k)+k)$.
Question: Test if there is an $\mathcal{H}$-elimination decomposition (T,χ,L), where D^* is the unique connected component in $G[L]$ of size at least $\alpha(k)$ (see Lemma 3.1) and $t^* \in V(T)$ is the vertex with $V(D^*) \subseteq \chi(t^*)$, such that the following holds:

1. $L \cap A^* = \emptyset$.
2. For each $v \in N_G(D^*) \cup A^*$, there is (unique non-leaf vertex) $t \in V(T) \setminus \{t^*\}$, such that $\chi(t) = \{v\}$ and t is an ancestor of t^* in T.

In the above, we say that (T,χ,L) is a *solution* to the α-Ext $\mathcal{H}$-ED instance (G,k,A^*).
The objective of the remainder of this section is to prove the following lemma.

Lemma 4.5. *Equipped with the algorithm $\mathbb{M}_{\mathsf{mod}}$, we can design an algorithm for α-Ext $\mathcal{H}$-ED, that given an instance (G,k,A^*), where G is a graph on n vertices, correctly decides whether or not it is a yes-instance of the problem in time bounded by $(f_{\mathsf{mod}}(k)+(\alpha(k)+k)^{\mathcal{O}(\alpha(k)+k)})\cdot n^{\mathcal{O}(1)}$.*

We prove the above lemma by exhibiting such an algorithm for α-Ext $\mathcal{H}$-ED. Let (G,k,A^*) be an instance of α-Ext $\mathcal{H}$-ED. As $|V(G)| \geq 3(\alpha(k)+k)$, from Observation 2.5, $G - A^*$ has a unique connected component of size at least $\alpha(k)$, we denote that connected component by C^*. Note that from the observation we also have $|V(G) \setminus V(C^*)| < \alpha(k)+k$.

We design a branching algorithm for the problem, and we will use $k - |A^*|$ as the measure to analyse the running time of our algorithm. We start with some simple cases, when we can directly resolve the given instance. (The time required for the execution of our base cases and branching rules will be provided in the runtime analysis of the algorithm.)

Notice that Elimination Distance to $\mathcal{H}$ is a special case of α-Ext-$\mathcal{H}$-Elimination Distance, namely, when $A^* = \emptyset$. Thus to prove Lemma 4.2, it is sufficient to prove the more general Lemma 4.5. Next, we give description of our branching algorithm for the α-Ext-$\mathcal{H}$-Elimination Distance problem.

Base Case 1: If $k - |A| = 0$ and $C^* \notin \mathcal{H}$, then return that the instance has no solution.

Notice that if the instance admits a solution, say, (T,χ,L), where D^* is the unique large connected component in $G[L]$, then $V(D^*) \subseteq V(C^*)$ (see Observation 2.5). All the vertices in $N_G(D^*) \cup A^*$ must be mapped to non-leaf vertices in a single root-to-leaf path. In Base Case 1, as $C^* \notin \mathcal{H}$, some neighbor of D^* in G outside A^* must also be mapped to (a non-leaf in) the same root-to-leaf path. Thus the correctness of Base Case 1 follows.

Intuitively speaking, our next base case deals with the case when an arbitrary deletion set for C^* of size at most k can be added to A^* to make the resulting C^* belong to $\mathcal{H}$. We will later

prove that if our branching rules (to be described later) and Base Case 1 is not applicable, then indeed we can obtain a solution to our instance using this base case. Before formally describing our next base case, we state a simple observation and introduce some notations.

The following observation uses the well-known self-reducibility property of NP-complete problem, using which we can find a solution with the help of a decision oracle.

Observation 4.6. *Given an instance (G^*, ℓ) of* Vertex Deletion to $\mathcal{H}$*, using the FPT algorithm $\mathcal{M}_{\mathsf{mod}}$ of* Vertex Deletion to $\mathcal{H}$*, in time $f_{\mathsf{mod}}(\ell) \cdot n^{\mathcal{O}(1)}$ we can obtain a minimum sized set $\widehat{S} \subseteq V(G^*)$ of size at most ℓ such that $G^* - \widehat{S} \in \mathcal{H}$ or return that no such set exists.*

We define a boolean variable bool as follows. If $(C^*, k - |A|)$ is a no-instance of Vertex Deletion to $\mathcal{H}$, then we set $\mathsf{bool} = 0$. Otherwise, let S^* be a minimum sized set computed using Observation 4.6. If for any subset $Z \subseteq V(G) \setminus (V(C^*) \cup A^*)$, there is a solution $(T, \chi, V(G) \setminus (A^* \cup S^* \cup Z))$, for the instance $(G, k, A^* \cup S^*)$, then set $\mathsf{bool} = 1$, and otherwise, set $\mathsf{bool} = 0$. By Cayley's Theorem [16], the number of distinct labelled forest on q vertices is bounded by $q^{\mathcal{O}(q)}$. Using the above together with Observation 4.6 and the fact that $|V(G) \setminus (V(C^*) \cup A^*)| < \alpha(k) + k$ (see Observation 2.5), allows us to construct bool in time bounded by $(f_{\mathsf{mod}}(k) + (\alpha(k) + k)^{\mathcal{O}(\alpha(k)+k)}) \cdot n^{\mathcal{O}(1)}$. We are now ready to state our next base case.

Base Case 2: If $\mathsf{bool} = 1$, then return that (G, k, A^*) is a yes-instance of the problem.

The correctness of the above base case follows from the fact that whenever bool is set to 1, then we do have a solution for the instance (G, k, A^*).

Roughly speaking, we will be able to say that, if we have a yes-instance, then for any solution (T, χ, L) for the instance, when our base cases are not applicable, there is a small connected subset $X^* \subseteq V(C^*)$, with bounded neighborhood containing an obstruction, that must be separated from the large connected component in $G[L]$ (see Lemma 3.1). Moreover, as X^* is contained in the large connected component C^*, we will be able to guarantee that at least one vertex in the closed neighborhood of X^* must be mapped to some node in the root-to-leaf path of the node to which vertices in D^* are mapped. Before discussing further, we introduce some notations.

For a graph $\widehat{G}$ and integers $p, q \in \mathbb{N}$, a set $B \subseteq V(\widehat{G})$ is a (p, q)-*connected set* in $\widehat{G}$, if $\widehat{G}[B]$ is connected, $|B| \leq p$ and $|N_{\widehat{G}}(B)| \leq q$. We say that a (p, q)-connected set B in $\widehat{G}$ is *maximal* if there does not exist another (p, q)-connected set B^* in $\widehat{G}$, such that $B \subset B^*$. Below we state a result regarding computation of maximal (p, q)-connected sets in a graph, which directly follows from Lemma 3.1 of Fomin and Villanger [35].

Proposition 4.7. *For a graph $\widehat{G}$ on n vertices and integers $p, q \in [n]$, the number of (p, q)-connected sets in $\widehat{G}$ is bounded by $2^{p+q} \cdot n$. Moreover, the set of all maximal (p, q)-connected sets in $\widehat{G}$ can be computed in time bounded by $2^{p+q} \cdot n^{\mathcal{O}(1)}$.*

We enumerate the set of all maximal $(\alpha(k), k)$-connected sets in C^*, and let $\mathcal{C}_{\mathsf{conn}}$ be the set containing all maximal $(\alpha(k) + k, k)$-connected sets $Q \subseteq V(C^*)$, such that $G[N_{C^*}[Q]]$ does not belong to $\mathcal{H}$. Note that using $\mathcal{M}_{\mathsf{mod}}$ and Proposition 4.7, we can construct $\mathcal{C}_{\mathsf{conn}}$ in time bounded by $f_{\mathsf{mod}}(0) \cdot 2^{\mathcal{O}(\alpha(k)+k)} \cdot n^{\mathcal{O}(1)}$. We are now ready to state our branching rule.

Branching Rule 1: If there is some $C \in \mathcal{C}_{\mathsf{conn}}$, then for each $v \in N_G[C]$ $(= N_G[C] \setminus A^*)$, solve (recursively) the instance $(G, k, A^* \cup \{v\})$. Moreover, return yes, if and only if one such instance is a yes-instance of the problem.

The next lemma lies at the crux of our algorithm, which will help us establish that one of the base cases or branching rule must be applicable. For the following lemma, we suppose that (G, A, k) has a solution and Base Case 1 and 2 are not applicable, and we consider a solution (T, χ, L) for the instance, which maximizes $|V(G) \setminus L|$. As G is $(\alpha(k), k)$-unbreakable, from Lemma 3.1 we can obtain that there is exactly one connected component, say, D^*, in $G[L]$,

that has at least $\alpha(k)$ vertices, and we let $t^* \in V(T)$ such that $\chi(t^*) \subseteq V(D^*)$. We define let the set $S \subseteq V(G) \setminus L$ contain all the vertices mapped to the root-to-t^* path in T, that is $S = \{v \in V(G) \mid \chi(t) = \{v\}, t \neq t^*, t \text{ is an ancestor of } t^* \text{ in } T\}$.

Lemma 4.8. *We have $\mathcal{C}_{\mathsf{conn}} \neq \emptyset$, and for each $C \in \mathcal{C}_{\mathsf{conn}}$, we have $S \cap N_{C^*}[C] \neq \emptyset$.*

Proof. As C^* is the (unique) connected component in $G - A^*$ that has at least $\alpha(k)$ vertices, thus, we can obtain that $V(D^*) \subseteq V(C^*)$. Let Y be the set of vertices in C^* that do not belong to the set $S \cup V(D^*)$, i.e., $Y = V(C^*) \setminus (V(D^*) \cup S)$. Let $\mathcal{Y}_{\mathsf{obs}}$ be the set of connected components in $G[Y]$ that do not belong to $\mathcal{H}$. We let $S_{\mathsf{obs}} \subseteq S \cap V(C^*)$ be the set of vertices in $S \cap V(C^*)$ that are neighbors of some connected components in $\mathcal{Y}_{\mathsf{obs}}$, i.e., $S_{\mathsf{obs}} = \{v \in S \cap V(C^*) \mid v \in N_G(C) \text{ for some } C \in \mathcal{Y}_{\mathsf{obs}}\}$. Furthermore, let $S_{\mathsf{rem}} = (S \cap V(C^*)) \setminus S_{\mathsf{obs}}$. Notice that $V(D^*), S_{\mathsf{rem}}, S_{\mathsf{obs}}, Y$ is a partition of $V(C^*)$ (possibly with empty-sets).

Firstly consider the case when $S_{\mathsf{obs}} = \emptyset$, and we will argue that Base Case 2 must be applicable, which will lead us to a contradiction. Note that for the above case, $S_{\mathsf{rem}} = S \cap V(C^*)$, where $|S| \leq k$, and each connected component in $C^* - S_{\mathsf{rem}} = C^* - S$ must belong to $\mathcal{H}$. As (T, χ, L) is a solution for the instance, we must have $A^* \subseteq S$. From the above two statements we can obtain that $(C^*, k - |A^*|)$ is a yes-instance of Vertex Deletion to $\mathcal{H}$. Let $S^* \subseteq V(C^*)$ be the (same) minimum sized set computed using Observation 4.6, while computation of bool, such that each connected component in $C^* - S^*$ is in $\mathcal{H}$. Note that $|S^*| \leq |S_{\mathsf{rem}}| = |S \cap V(C^*)|$ (recall that $S_{\mathsf{obs}} = \emptyset$). Let $Z = V(G) \setminus (L \cup A^*) \subseteq V(G) \setminus (V(C^*) \cup A^*)$. We will argue that (G, k, A^*) admits a solution $(T, \chi', V(G) \setminus (A^* \cup S^* \cup Z))$, for an appropriately constructed χ', in which case $\mathsf{bool} = 1$, contradicting that Base Case 2 is not applicable. Set $\chi(t^*) = \chi(t^*) \cup (V(C^*) \setminus S^*)$ and for each $t \in V(T) \setminus \{t^*\}$, such that $t \cap L \neq \emptyset$, we set $\chi'(t) = \chi(t) \setminus V(C^*)$. Let $S_{\mathsf{rem}} = \{u_1, u_2, \cdots, u_p\}$ and $S^* = \{w_1, w_2, \cdots, w_q\}$, where note that $q \leq p$. For each $i \in [q]$, we set $\chi'(t_i) = \{w_i\}$, where t_i is the vertex in T, such that $\chi(t_i) = \{u_i\}$. Notice that, as $C^* - S$ is in $\mathcal{H}$ and $\mathcal{H}$ is closed under disjoint union, we can obtain that for each $t \in V(T)$, $G[\chi'(t)]$ is in $\mathcal{H}$. By the construction of χ', note that $(T, \chi', V(G) \setminus (A^* \cup S^* \cup Z))$ satisfies item 1 to 3 of Definition 2.1. Thus we can conclude that $(T, \chi', V(G) \setminus (A^* \cup S^* \cup Z))$ is a solution for (G, k, A^*).

We will next consider the case when $S_{\mathsf{obs}} \neq \emptyset$. As $|V(G) \setminus V(D^*)| < \alpha(k) + k$ (see Observation 3.1), we can obtain that $|Y|$, and thus, the number of vertices in each connected component in $\mathcal{Y}_{\mathsf{obs}}$ is less than $\alpha(k) + k$. Notice that for each $C \in \mathcal{Y}_{\mathsf{obs}}$, $|N_G(C)| \leq |S| \leq k$, and thus, C is an $(\alpha(k) + k, k)$-connected set in G. Thus, for each $C \in \mathcal{Y}_{\mathsf{obs}}$, there must exists some $C' \in \mathcal{C}_{\mathsf{conn}}$, such that $V(C) \subseteq V(C')$. The above together with the assumption $S_{\mathsf{obs}} \neq \emptyset$ (and thus, $\mathcal{Y}_{\mathsf{obs}} \neq \emptyset$) we can obtain that $\mathcal{C}_{\mathsf{conn}} \neq \emptyset$. By construction, for each $C \in \mathcal{C}_{\mathsf{conn}}$, $V(C) \subseteq V(C^*)$ and $N_{C^*}[C]$ is not in $\mathcal{H}$. Moreover, as C^* is connected and C is a maximal $(\alpha(k) + k, k)$-connected set in C^*, we can obtain that $N_{C^*}[C] \cap S \neq \emptyset$. This concludes the proof. □

We are now ready to prove Lemma 4.5.

Proof of Lemma 4.5. From Proposition 2.6, it is enough for us to design an algorithm for $(\alpha(k), k)$-unbreakable graphs, therefore we design algorithm for case when the input graph is $(\alpha(k), k)$-unbreakable. Consider an instance (G, A^*, k) of α-Ext $\mathcal{H}$-ED, where G is an $(\alpha(k), k)$-unbreakable graph. If the instance can be resolved using Base Case 1 or 2, then the algorithm resolve it. Otherwise, from Lemma 4.8 we know that the branching rule must be applicable. By our previous discussion, the we can test/apply our base cases in time bounded by $(f_{\mathsf{mod}}(k) + (\alpha(k) + k)^{\mathcal{O}(\alpha(k)+k)}) \cdot n^{\mathcal{O}(1)}$. Recall that for each set in $\mathcal{C}_{\mathsf{conn}}$, we have $|N_{C^*}[C]| \leq \alpha(k) + 2k$, and $\mathcal{C}_{\mathsf{conn}}$ can be constructed in time bounded by $f_{\mathsf{mod}}(0) \cdot 2^{\mathcal{O}(\alpha(k)+k)} \cdot n^{\mathcal{O}(1)}$. Also, the depth of the recursion tree can be bounded by $k+1$ (see Base Case 1). Thus, the number of nodes in the recursion tree can be bounded by $(\alpha(k) + 2k)^{\mathcal{O}(\alpha(k)+2k)}$). Thus we can bound the

running time of our algorithm for α-Ext $\mathcal{H}$-ED by $(f_{\mathsf{mod}}(k)+f_{\mathsf{mod}}(0))\cdot(\alpha(k)+k)^{\mathcal{O}(\alpha(k)+k)}\cdot n^{\mathcal{O}(1)}$. This concludes the proof. □

We will end this section with a remark regarding how we can obtain an FPT algorithm for Treewidth Decomposition to $\mathcal{H}$ parameterized by $\mathbf{tw}_{\mathcal{H}}(G)$, using $\mathcal{M}_{\mathsf{mod}}$. Consider an instance (G,k) of Treewidth Decomposition to $\mathcal{H}$, and an $\mathcal{H}$-tree decomposition, (T,χ,L) for it, if it exists. For the above case, we will define our extension version with respect to $A^*\subseteq V(G)$ of size at most k, where we would want the unique large connected component in $G[L]$ to have all its neighbors in A^*. Notice that all our argument will work exactly with the above minor modification. Thus, using exactly the same ideas as we presented for the case of Elimination Distance to $\mathcal{H}$, we will be able to obtain an FPT algorithm for Treewidth Decomposition to $\mathcal{H}$ parameterized by $\mathbf{tw}_{\mathcal{H}}(G)$, as required by the lemma.

4.2 Prove of Lemma 4.3

Fix any family $\mathcal{H}$ of graphs that is CMSO definable and is closed under disjoint union and taking induced subgraphs, such that Vertex Deletion to $\mathcal{H}$ admits an FPT algorithm, say, $\mathcal{M}_{\mathsf{mod}}$, when parameterized by $\mathbf{mod}_{\mathcal{H}}(G)$, running in time $f_{\mathsf{mod}}(k)\cdot n^{\mathcal{O}(1)}$. From Lemma 4.2 (see Section 4.1), we can obtain that Treewidth Decomposition to $\mathcal{H}$ admits an FPT algorithm, say, $\mathcal{T}_{\mathsf{tw}}$, when parameterized by $\mathbf{tw}_{\mathcal{H}}(G)$, running in time $f_{\mathsf{tw}}(k)\cdot n^{\mathcal{O}(1)}$. We will design an FPT algorithm, $\mathcal{M}_{\mathsf{tw}}$, for Vertex Deletion to $\mathcal{H}$, parameterized by $\mathbf{tw}_{\mathcal{H}}(G)$, using $\mathcal{M}_{\mathsf{mod}}$ and $\mathcal{T}_{\mathsf{tw}}$ as subroutines.

Let $\ell=\mathbf{tw}_{\mathcal{H}}(G)$. By instantiating Lemma 3.6 with the algorithm $\mathcal{T}_{\mathsf{tw}}$, we can compute an $\mathcal{H}$-tree decomposition, (T,χ,L) of $\mathcal{H}$-treewidth exactly ℓ, for G in time bounded by $\big(f_{\mathsf{tw}}(\ell)+\ell^{\mathcal{O}(\ell^2)}\big)n^{\mathcal{O}(1)}+h(\mathcal{F})$. We let $S=V(G)\setminus L$. We will assume that the constants ξ_i, for each $i\in[\ell+1]$ is hardcoded in the algorithm, and due to this and Lemma 2.20 our algorithm will be non-uniform. If we are able to bound the size of each connected component in $G[L]$ by $\sum_{i\in[\ell+1]}\xi_i$, where ξ_i is the number from Lemma 2.20, then notice that the treewidth of G can be bounded by $\ell+\sum_{i\in[\ell+1]}\xi_i$. As Vertex Deletion to $\mathcal{H}$ is CMSO expressible, for the case when G has bounded treewidth, we can check whether (G,k) is a yes-instance of Vertex Deletion to $\mathcal{H}$ using Courcelle's Theorem [18] in time bounded by $f(\ell)\cdot n^{\mathcal{O}(1)}$, where f is some computable function. Thus (roughly speaking) our next objective will be to bound the size of $\chi^{-1}(t)$, for each $t\in V(T)$ by replacements using Lemma 2.20. Let $\widetilde{A}$ be the set of nodes in T whose bag contain at least $\sum_{i\in[\ell+1]}\xi_i$ vertices from L, i.e., $\widetilde{A}=\{t\in V(T)\mid|\chi(t)\cap L|\geq\sum_{i\in[\ell+1]}\xi_i\}$. Furthermore, let $\widetilde{\mathcal{G}}$ be the set of graphs induced by vertices in L, in each of the bags of nodes in $\widetilde{A}$, i.e., $\widetilde{\mathcal{G}}=\{G[\chi(t)\cap L]\mid t\in\widetilde{A}\}$. We let $\widetilde{\mathcal{G}}=\{\widetilde{G}_1,\widetilde{G}_2,\cdots,\widetilde{G}_q\}$.

We create a sequence of $G_0,G_1,\cdots,G_q$ graphs and a sequence of constant $c_0,c_1,\cdots,c_q$ as follows. Intuitively speaking, we will obtain the above sequence of graph by replacing $\widetilde{G}_i$, for $i\in[q]$, by some graph obtained using Lemma 2.20. Set $G_0=G$ and $c_0=0$. We iteratively compute G_i, for each $i\in[q]$ (in increasing order) as follows. For the graph $\widetilde{G}_i$, let $\widetilde{t}_i$ be the unique leaf in T, such that $V(\widetilde{G}_i)\subseteq\chi(\widetilde{t}_i)$, and let $\widetilde{B}_i=\chi(\widetilde{t}_i)\setminus V(\widetilde{G}_i)$ and $\widetilde{b}_i=|\widetilde{B}_i|$. Note that $|\widetilde{B}_i|\leq\ell+1$, as (T,χ,L) is an $\mathcal{H}$-tree decomposition of G. Fix an arbitrary injective function $\lambda_{\widetilde{G}_i}:\widetilde{B}\to\{1,2,\cdots,\widetilde{b}_i\}$, and then $\widetilde{G}_i$ is the boundaried graph with boundary $\widetilde{B}_i$.[13] Note that $V(\widetilde{G}_i)\subseteq V(G_{i-1})$. Also, let G_i' be the boundaried graph $G_{i-1}-V(\widetilde{G}_i)$, with boundary $\widetilde{B}_i$. Using Lemma 2.20 and the algorithm $\mathcal{M}_{\mathsf{mod}}$, we find the graph $\widetilde{G}_i^*$ and the translation constant c_i, such that $\widetilde{G}_i\equiv_\Pi\widetilde{G}_i^*$ and $|V(\widetilde{G}_i^*)|\leq\xi_{\widetilde{b}_i}$ in time bounded by $\mathcal{O}(f_{\mathsf{mod}}(\ell)\cdot n^{\mathcal{O}(1)})$.[14]

[13] We have slightly abused the notation, and used $\widetilde{G}_i$ to denote both a graph and a boundaried graph.

[14] Note that for any graph $\widehat{G}$, $\iota(\widehat{G})\leq|V(\widehat{G})|$ (see Definition 2.19).

We let G_i be the graph $\widetilde{G}_i^* \oplus G_i'$, which can be computed in time bounded by $\mathcal{O}(f_{\mathsf{mod}}(\ell) \cdot n^{\mathcal{O}(1)})$. Let $c^* = \sum_{i \in [q]} c_i$. With the constructions described above, we are now in a position to prove Lemma 4.3.

Proof of Lemma 4.3. As $\mathcal{H}$ is CMSO definable to prove the lemma, it is enough to establish the following statements (together with Courcelle's Theorem [18]).

1. The instance $(G_q, k + c^*)$ can be constructed in time bounded by $\mathcal{O}(f(\ell) \cdot n^{\mathcal{O}(1)})$, for some function f,
2. $(G_q, k + c^*)$ and (G, k) are equivalent instances of Vertex Deletion to $\mathcal{H}$, and
3. The treewidth of G_q is at most $\ell + \max_{i \in [q]} c_i + \sum_{i \in [\ell]} \chi_i$ and the $\mathcal{H}$-treewidth of G_q is at most $\ell + \max_{i \in [q]} c_i$.[15]

As stated perviously, we will assume that the constants ξ_i, for $i \in [\ell + 1]$ are hardcoded in the algorithm. Thus, we can construct the set $\widetilde{\mathcal{G}}$ in polynomial time. Also, note that for any $i \in [q]$, $\iota(\widetilde{G}_i^* \oplus G_i') \leq \iota(G) \leq |V(G)|$ (see Definition 2.19). Thus, for some function f, we can construct the instance $(G_q, k + c^*)$ in time bounded by $f(\ell) \cdot n^{\mathcal{O}(1)}$.

We will inductively argue that for each $i \in [q]_0$, $(G_i, k + \sum_{j \in [i]_0} c_j)$ and (G, k) are equivalent instances of Vertex Deletion to $\mathcal{H}$. As $G_0 = G$ and $k + c_0 = k$, the claim trivially follows for the case when $i = 0$. Next we assume that for some $q' \in [q-1]_0$, for each $i' \in [q']_0$, $(G_{i'}, k + \sum_{j \in [i']_0} c_j)$ and (G, k) are equivalent instances of the problem. We will next prove the statement for $i = i' + 1$. It is enough to argue that $(G_{i-1}, k + \sum_{j \in [i-1]_0} c_j)$ and $(G_i, k + \sum_{j \in [i]_0} c_j)$ are equivalent instances. Recall that, by construction, $G[V(\widetilde{G}_i)] = G_{i-1}[V(\widetilde{G}_i)] = \widetilde{G}_i$ and $G_i' = G_{i-1} - V(\widetilde{G}_i)$ are boundaried graphs with boundary $\widetilde{B}_i$, and $G_i = \widetilde{G}_i \oplus G_i'$. From Lemma 2.20, $\widetilde{G}_i^* \equiv_\Pi \widetilde{G}_i$. Thus by definition, we have that $(G_{i-1}, k + \sum_{j \in [i-1]_0} c_j)$ and $(G_i, k + \sum_{j \in [i]_0} c_j)$ are equivalent instances of Vertex Deletion to $\mathcal{H}$.

To prove the third statement, note that it is enough to construct an $\mathcal{H}$-tree decomposition, (T_q, χ_q, L_q), of G_q, where for each $t \in V(T_q)$, we have $|\chi_q^{-1}(t)| \leq \ell + \max_{i \in [q]} c_i + \sum_{i \in [\ell]} \chi_i$ and $|\chi_q^{-1}(t) \setminus L_q| \leq \ell + \max_{i \in [q]} c_i$. Let $X = \cup_{i \in [q]} V(\widetilde{C}_i)$, and $L_q = (L \setminus X) \cup (V(G_q) \setminus V(G))$ and $T_q = T$. For each $t \in V(T) \setminus \widetilde{A}$, we set $\chi_q(t) = \chi(t)$, and for each $i \in [q]$, we set $\chi_q(\widetilde{t}_i) = (\chi(\widetilde{t}_i) \setminus V(\widetilde{G}_i)) \cup V(\widetilde{G}_i^*)$. For each $i \in [q]$, note that $|V(\widetilde{G}_i^*)| \leq \xi_{\widetilde{b}_i} \leq \sum_{j \in [\ell]} \xi_j$. Thus we can obtain that (T_q, χ_q, L_q) is an $\mathcal{H}$-tree decomposition of G_q that satisfies all the required properties. This concludes the proof. □

4.3 Proof of Lemma 4.4

Fix any family $\mathcal{H}$ of graphs that is CMSO definable and is closed under disjoint union and taking induced subgraphs, such that Elimination Distance to $\mathcal{H}$ (resp. Treewidth Decomposition to $\mathcal{H}$) admits an FPT algorithm, say, $\mathfrak{X}_{\mathsf{mod}}$ running in time $f(\ell) \cdot n^{\mathcal{O}(1)}$, where n is the number of vertices in the given graph G and $\ell = \mathbf{mod}_{\mathcal{H}}(G)$.

Let (G, k) be an instance of the problem Vertex Deletion to $\mathcal{H}$. From Proposition 2.6 it is enough for us to design an algorithm for $(\alpha(k), k)$-unbreakable graphs, and thus, we assume that G is $(\alpha(k), k)$-unbreakable. We begin with the following simple sanity checks.

Base Case 1. If $G \in \mathcal{H}$ and $k \geq 0$, then return that (G, k) is a yes-instance of the problem. Moreover, if $k < 0$, then return that the instance is a no-instance.

[15] We remark that although $k + c^*$ can possibly be much larger than k, both the treewidth and the $\mathcal{H}$-treewidth of G_q are at most some additive constants (depending on $\mathcal{H}$) away from k.

Base Case 2. If $|V(G)| \leq 2\alpha(k)+k$, then for each $S \subseteq V(G)$, check if $G-S \in \mathcal{H}$, by calling $\mathcal{X}_{\mathsf{mod}}$ for the instance $(G-S,0)$. If for any such S we obtain that $G-S \in \mathcal{H}$, return that (G,k) is a yes-instance, and otherwise return that it is a no instance.

Base Case 3. If G does not admit an $\mathcal{H}$-elimination decomposition (resp. $\mathcal{H}$-tree decomposition) of depth (resp. width) at most k, then return that (G,k) is a no-instance of the problem.

The correctness of the Base Case 1 and 2 is immediate from their descriptions. Note that $\mathbf{tw}_{\mathcal{H}}(G) \leq \mathbf{ed}_{\mathcal{H}}(G) \leq \mathbf{mod}_{\mathcal{H}}(G)$. Thus, if (G,k) is a yes-instance of Vertex Deletion to $\mathcal{H}$, then it must admit an $\mathcal{H}$-elimination decomposition (resp. $\mathcal{H}$-tree decomposition) of depth (resp. width) at most k. The above implies the correctness of Base Case 3. Note that using $\mathcal{X}_{\mathsf{mod}}$, we can test/apply all the base cases in time bounded by $\max\{2^{\alpha(k)+k}, f(k)\} \cdot n^{\mathcal{O}(1)}$.

Hereafter we assume that the base cases are not applicable. We compute an $\mathcal{H}$-elimination decomposition (resp. $\mathcal{H}$-tree decomposition), say, (T,χ,L), by instantiating Lemma 3.5 (resp. Lemma 3.6) with the algorithm $\mathcal{X}_{\mathsf{mod}}$, of depth (resp. width) at most k. Note that the above decomposition can be computed as Base Case 3 is not applicable.

Let C^* be a connected component in $G[L]$ with maximum number of vertices, and let $S^* = N_G(C^*)$ and $Z^* = V(G) \setminus N_G[C^*]$. Note that as (T,χ,L) is an $\mathcal{H}$-elimination decomposition (resp. $\mathcal{H}$-tree decomposition) of depth (resp. width) at most k, we can obtain that $|S^*| \leq k$. As G is $(\alpha(k),k)$-unbreakable, the above together with Observation 2.5 implies that $|Z^* \cup S^*| \leq \alpha(k)+k$. We have the following observation which immediately follows from the fact that (T,χ,L) is an $\mathcal{H}$-elimination decomposition (resp. $\mathcal{H}$-tree decomposition) for G of depth (resp. width) at most k.

Observation 4.9. *For any $S \subseteq V(G)$, where $|S| \leq k$, either $Z^* \cap S \neq \emptyset$, or $G - S^* \in \mathcal{H}$.*

The above observation leads us to the following base case and our branching rule.

Base Case 4. $G - S^* \in \mathcal{H}$, then return that (G,k) is a yes-instance.

As $|S^*| \leq k$, the correctness of the above base case immediately follows. Moreover, we can apply Base Case 3 in time bounded by $f(k) \cdot n^{\mathcal{O}(1)}$ using $\mathcal{X}_{\mathsf{mod}}$.

Branching Rule. For each $z \in Z^*$, (recursively) solve the instance $(G-\{z\}, k-1)$. Return that (G,k) is a yes-instance if and only if for some $z \in Z$, $(G-\{z\},k-1)$ is a yes-instance.

The correctness of the branching rule follows from Observation 4.9 and non-applicability of Base Case 4. Moreover, we can create instances in the branching rule in polynomial time, given the decomposition (T,χ,L).

Note that the depth of the recursion tree is bounded by $k+1$. Also, each fo the steps can be applied in time bounded by $\max\{2^{\alpha(k)+k}, f(k)\} \cdot n^{\mathcal{O}(1)}$. Thus we can bound the running time of our algorithm by $k^{\mathcal{O}(k)} \cdot \max\{2^{\alpha(k)+k}, f(k)\} \cdot n^{\mathcal{O}(1)}$. The correctness of the algorithm is immediate from the description and Observation 4.9. The above implies proof of Lemma 4.4.

5 Applications of Theorem 1.1 – Beyond Graphs

In this section we see applications of our main theorem (Theorem 1.1) for problems that are not precisely captured by the families of graphs mentioned in the premise of Theorem 1.1. This allows us to obtain the first analogous results for problems such as Multiway Cut, Subset Feedback Vertex Set (Subset FVS, for short) and Subset Odd Cycle Transversal (Subset OCT, for short). That is, we obtain FPT algorithms for these problems that is parameterized by a parameter whose value is upper bounded by the standard parameter (i.e., solution size) and which can be arbitrarily smaller. For instance, consider the Multiway Cut problem, where one is given a graph G and a set of vertices S (called terminals) and an integer ℓ and the goal is to decide whether there is a set of at most ℓ vertices whose deletion separates every pair of these terminals. The standard parameterization for this problem is the

solution size. Jansen et al. [48] propose to consider undirected graphs with a distinguished set of terminal vertices and study the parameterized complexity of Multiway Cut parameterized by the elimination distance to a graph where each component has at most one terminal. Notice that this new parameter is always upper bounded by the size of a minimum solution. Thus, an FPT algorithm for Multiway Cut with this new parameter would naturally extend the boundaries of tractability for the problem and we obtain such an algorithm by using Theorem 1.1 in an appropriate way. We then proceed to obtain similar FPT algorithms for the other cut problems mentioned in this paragraph.

That is, we obtain an FPT algorithm for Subset FVS parameterized by the elimination distance to a graph where no terminal is part of a cycle, and an FPT algorithm for Subset OCT parameterized by the elimination distance to a graph where no terminal is part of an odd cycle.

To this end, we begin by formally defining structures in the context of this paper and an extension of $\mathcal{H}$-elimination decomposition and $\mathcal{H}$-tree decomposition to families of structures.

Definition 5.1. [**Structure**] A *structure* α is a tuple whose first element is a graph, say, G, and each of the remaining elements is a subset of $V(G)$, a subset of $E(G)$, a vertex in $V(G)$ or an edge in $E(G)$. The number of elements in the tuple is the *arity* of the structure.

We will be concerned with structures of arity 2 where we have the second element as a vertex subset, and we denote the family of such structures by $\mathcal{S}$. For a family $\mathcal{S}' \subseteq \mathcal{S}$, we will next define the notion of $\mathcal{S}_{\mathsf{str}}$-elimination decompositions and $\mathcal{S}_{\mathsf{str}}$-tree decompositions that is tailored to our purpose.

Definition 5.2. For a family of structures, $\mathcal{S}' \subseteq \mathcal{S}$, an *$\mathcal{S}'$-elimination decomposition* of a structure (G, S), where $S \subseteq V(G)$, is a triplet (T, χ, L), where $L \subseteq V(G)$, T is a rooted forest, and $\chi \colon V(T) \to 2^{V(G)}$, such that:

1. For each internal node t of T we have $|\chi(t)| \leq 1$ and $\chi(t) \subseteq V(G) \setminus L$.
2. The sets $(\chi(t))_{t \in V(T)}$ form a partition of $V(G)$.
3. For each edge $\{u, v\} \in E(G)$, if $u \in \chi(t_1)$ and $v \in \chi(t_2)$, then t_1, t_2 are in ancestor-descendant relation in T.
4. For each leaf t of T, we have $\chi(t) \subseteq L$ and the structure $(G[\chi(t)], S \cap \chi(t))$ belongs to $\mathcal{S}'$.

The *depth* of (T, χ, L) is same as the depth of T. The *$\mathcal{S}'$-elimination distance* of the structure (G, S), denoted $\mathbf{ed}_{\mathcal{S}'}(G, S)$, is the minimum depth of an $\mathcal{S}'$-elimination decomposition of (G, S).

Definition 5.3. For a family of structures, $\mathcal{S}' \subseteq \mathcal{S}$, an *$\mathcal{S}'$-elimination decomposition* of a structure (G, S), where $S \subseteq V(G)$, is a triplet (T, χ, L), where $L \subseteq V(G)$, T is a rooted tree, and $\chi \colon V(T) \to 2^{V(G)}$, such that:

1. For each $v \in V(G)$ the nodes $\{t \mid v \in \chi(t)\}$ form a non-empty connected subtree of T.
2. For each edge $\{u, v\} \in E(G)$ there is a node $t \in V(G)$ with $\{u, v\} \subseteq \chi(t)$.
3. For each vertex $v \in L$, there is a unique $t \in V(T)$ for which $v \in \chi(t)$. Moreover, t must a leaf of T.
4. For each node $t \in V(T)$, the structure $(G[\chi(t) \cap L], \chi(t) \cap L \cap S)$ belongs to $\mathcal{S}'$.

The *width* of an $\mathcal{S}'$-tree decomposition is $\max(0, \max_{t \in V(T)} |\chi(t) \setminus L| - 1)$. The $\mathcal{S}'$-treewidth of a structure (G, S), denoted $\mathbf{tw}_{\mathcal{S}'}(G, S)$, is the minimum width of an $\mathcal{S}'$-tree decomposition of (G, S).

We remark that though the above two definitions are extendable to more general notions of structures, we choose to give it for this restricted version as it is enough for our purpose and it is more insightful for the reader. We now will be able to capture problems such as Multiway Cut, Subset FVS and Subset OCT. To this end, we begin by defining the following families of structures.

$\mathcal{S}_{\mathsf{mway}} = \{(G, S) \,|\, \text{every connected component of } G \text{ has at most one vertex from } S \subseteq V(G)\}$
$\mathcal{S}_{\mathsf{fvs}} = \{(G, S) \mid G \text{ has no cycle containing a vertex from } S \subseteq V(G)\}$
$\mathcal{S}_{\mathsf{oct}} = \{(G, S) \mid G \text{ has no odd length cycle containing a vertex from } S \subseteq V(G)\}$

We are now ready to state the main result of this section.

Theorem 5.1. *Each of the following parameterized problems admits an FPT algorithm:*

1. Multiway Cut *parameterized by* $\mathbf{tw}_{\mathcal{S}_{\mathsf{mway}}}(G, S)$ *(and thus,* $\mathbf{ed}_{\mathcal{S}_{\mathsf{mway}}}(G, S)$*).*
2. Subset FVS *parameterized by* $\mathbf{tw}_{\mathcal{S}_{\mathsf{fvs}}}(G, S)$ *(and thus,* $\mathbf{ed}_{\mathcal{S}_{\mathsf{fvs}}}(G, S)$*), and*
3. Subset OCT *parameterized by* $\mathbf{tw}_{\mathcal{S}_{\mathsf{oct}}}(G, S)$ *(and thus,* $\mathbf{ed}_{\mathcal{S}_{\mathsf{oct}}}(G, S)$*).*

We will begin by explaining the intuitive idea behind the proof of the above theorem. For simplicity, let us fix the problem Multiway Cut. Our first goal is to construct a CMSO definable family of graph $\mathcal{H}_{\mathsf{mway}}$ that is closed under disjoint union. Then, for a given instance (G, S, k) of Multiway Cut, by appropriate "gadgeteering", we will construct an (equivalent) instance (G', k) of Vertex Deletion to $\mathcal{H}_{\mathsf{mway}}$, ensuring that $\mathbf{tw}_{\mathcal{H}_{\mathsf{mway}}}(G')$ is at most a polynomial factor away from $\mathbf{tw}_{\mathcal{S}_{\mathcal{H}_{\mathsf{mway}}}}(G)$. Using a known FPT algorithm for Multiway Cut parameterized by the solution, we will be able to obtain an FPT algorithm for Vertex Deletion to $\mathcal{H}_{\mathsf{mway}}$, parameterized by $\mathbf{mod}_{\mathcal{H}_{\mathsf{mway}}}(G')$. The above statement together with Theorem 1.1 will imply that Vertex Deletion to $\mathcal{H}_{\mathsf{mway}}$ admits an FPT algorithm, when parameterized by $\mathbf{tw}_{\mathcal{H}_{\mathsf{mway}}}(G')$. The above, together with the equivalence of the instance (G, S, k) of Multiway Cut and the instance (G', k) of Vertex Deletion to $\mathcal{H}_{\mathsf{mway}}$, and the property that $\mathbf{tw}_{\mathcal{H}_{\mathsf{mway}}}(G')$ is at most a polynomial factor away from $\mathbf{tw}_{\mathcal{S}_{\mathcal{H}_{\mathsf{mway}}}}(G)$, will imply an FPT algorithm for Vertex Deletion to $\mathcal{H}_{\mathsf{mway}}$ parameterized by $\mathbf{tw}_{\mathcal{S}_{\mathcal{H}_{\mathsf{mway}}}}(G)$.

Before moving to the formal description, we briefly discuss the construction of $\mathcal{H}_{\mathsf{mway}}$. For an instance (G, S, k) of Multiway Cut, we will subdivide the edges of G and then attach a K_3 at each vertex in S to obtain the graph G' in the Vertex Deletion to $\mathcal{H}_{\mathsf{mway}}$ instance (G', k). Roughly speaking, the family $\mathcal{H}_{\mathsf{mway}}$ will trivially contain all "ill-formed" graphs, i.e., the graph that cannot be obtained via a reduction that we discussed above. Apart from the above graphs, the family $\mathcal{H}_{\mathsf{mway}}$ will contain all "solved graphs" (very roughly speaking). The above will be categorized based on cut vertices whose removal results in connected components that are K_3s. We will now move towards the formal discussion of our proof of Theorem 5.1.

Some useful graphs and graph classes. For a graph G, the *subdivision* of G, denoted by G_{sd} is the graph obtained by sub-dividing the edges of G exactly once, i.e., we have $V(G_{\mathsf{sd}}) = V(G) \cup \{w_e \mid e \in E(G)\}$ and $E(G_{\mathsf{sd}}) = \{\{u, w_e\}, \{w_e, v\} \mid e = \{u, v\} \in E(G)\}$. A *double subdivision* of a graph G is the graph G_{dsd}, obtained by sub-dividing each of the edges in G with two vertices, i.e., we have $V(G_{\mathsf{dsd}}) = V(G) \cup \{w_e, w'_e \mid e \in E(G)\}$ and $E(G_{\mathsf{dsd}}) = \{\{u, w_e\}, \{w_e, w'_e\}, \{w'_e, v\} \mid e = \{u, v\} \in E(G)\}$.

For a graph G and a set $S \subseteq V(G)$, by $G^{S,\Delta}$, we denote the graph obtained from G by attaching a K_3 on vertices in S, i.e., $V(G^{S,\Delta}) = V(G) \cup \{a_s, a'_s \mid s \in S\}$ and $E(G^{S,\Delta}) = E(G) \cup \{\{a_s, a'_s\}, \{a_s, s\}, \{a'_s, s\} \mid s \in S\}$.

Consider a graph G. A *cut* vertex in G is a vertex $v \in V(G)$, such that the number of connected components in $G - \{v\}$ is strictly more than the number of connected components in G. We say that a cut vertex $v \in V(G)$ is *relevant* if $G - \{v\}$ has a connected component C

with **exactly** two vertices and for each $u \in V(C)$, we have $\{u, v\} \in V(G)$. We now define the following hereditary families of graphs.

$\mathcal{H}_{\mathsf{mway}} = \{G \in \mathcal{G} \mid$ each connected component of G has at most one relevant cut vertex$\}$
$\mathcal{H}_{\mathsf{fvs}} = \{G \in \mathcal{G} \mid G$ has no cycle of length at least 4 containing a relevant cut vertex$\}$
$\mathcal{H}_{\mathsf{oct}} = \{G \in \mathcal{G} \mid G$ has no odd cycle of length at least 5 containing a relevant cut vertex$\}$

We have the following observation regarding the above defined families of graphs.

Observation 5.4. *Each $\mathcal{H} \in \{\mathcal{H}_{\mathsf{mway}}, \mathcal{H}_{\mathsf{fvs}}, \mathcal{H}_{\mathsf{oct}}\}$ is* CMSO *definable and closed under disjoint union.*

Proof. From the description of the families, it clearly follows that they are closed under disjoint union. We will next show that they are also CMSO definable. We note that checking whether a vertex subset $X \subseteq V(G)$ induces a connected subgraph of G, is CMSO definable, and we let this predicate be $\mathsf{conn}(X)$ (see, for example, Section 7.4.1 in the book of Cygan et al. [23]). Next we give a predicate which can check if a vertex is a relevant cut vertex in the given graph.

$\mathsf{rel\text{-}cut}(v) = \exists u, u' \in V(G) \setminus \{v\} \big[u \neq u' \wedge \mathsf{adj}(u, u') = 1 \wedge \mathsf{adj}(u, v) = 1 \wedge \mathsf{adj}(u', v) = 1 \wedge (\forall w \in V(G) \setminus \{u, u', v\}\ \forall x \in \{u, u'\}\ \mathsf{adj}(w, u) = 0)\big]$

Now we define a predicate which can test if a connected component of a graph contains more that contains two relevant cut vertices.

$\mathsf{in\text{-}}\mathcal{H}_{\mathsf{mway}} = \neg\exists u, v \in V(G) \big[u \neq v \wedge \mathsf{rel\text{-}cut}(u) = 1 \wedge \mathsf{rel\text{-}cut}(v) = 1 \wedge \big(\exists X \subseteq V(G) (\mathsf{conn}(X) = 1 \wedge u, v \in X)\big]$

Notice that using $\mathsf{in\text{-}}\mathcal{H}_{\mathsf{mway}}$ (and $\mathsf{rel\text{-}cut}(v)$) we can obtain that $\mathcal{H}_{\mathsf{mway}}$ is CMSO definable.

It is well known that checking if a graph contains a cycle of length at least 5 containing a particular vertex is CMSO definable. We denote such a predicate by $\mathsf{cycle}_{\geq 5}(v)$. The above together with the definition of $\mathsf{rel\text{-}cut}(v)$ implies that $\mathcal{H}_{\mathsf{fvs}}$ is CMSO definable. Since we are allowed to have condition on cardinality of a set size modulo a number q, using $\mathsf{cycle}_{\geq 5}(v)$ and $\mathsf{rel\text{-}cut}(v)$ we can obtain that $\mathcal{H}_{\mathsf{oct}}$ is CMSO definable. □

Reduction. In the following lemma we show how we can obtain our reduction.

Lemma 5.5. *Each of the following holds:*

1. *An instance (G, S, k) of* Multiway Cut *is a yes-instance if and only if $(G^{S,\Delta}_{\mathsf{sd}}, k)$ is a yes-instance of* Vertex Deletion to $\mathcal{H}_{\mathsf{mway}}$. *Moreover,* $\mathbf{tw}_{\mathcal{H}_{\mathsf{mway}}}(G^{S,\Delta}_{\mathsf{sd}}) < 3 \cdot (\ell + 1) + \binom{\ell+1}{2}$, *where* $\ell = \mathbf{tw}_{\mathcal{S}_{\mathsf{mway}}}(G, S)$.
2. *An instance (G, S, k) of* Subset FVS *is a yes-instance if and only if $(G^{S,\Delta}_{\mathsf{sd}}, k)$ is a yes-instance of* Vertex Deletion to $\mathcal{H}_{\mathsf{fvs}}$. *Moreover,* $\mathbf{tw}_{\mathcal{H}_{\mathsf{fvs}}}(G^{S,\Delta}_{\mathsf{sd}}) < 3 \cdot (\ell + 1) + \binom{\ell+1}{2}$, *where* $\ell = \mathbf{tw}_{\mathcal{S}_{\mathsf{fvs}}}(G, S)$.
3. *An instance (G, S, k) of* Subset OCT *is a yes-instance if and only if $(G^{S,\Delta}_{\mathsf{dsd}}, k)$ is a yes-instance of* Vertex Deletion to $\mathcal{H}_{\mathsf{oct}}$. *Moreover,* $\mathbf{tw}_{\mathcal{H}_{\mathsf{oct}}}(G^{S,\Delta}_{\mathsf{dsd}}) < 3 \cdot (\ell + 1) + 2 \cdot \binom{\ell+1}{2}$, *where* $\ell = \mathbf{tw}_{\mathcal{S}_{\mathsf{oct}}}(G, S)$.

Proof. We will prove the third statement as it is the most involved. The proof of other two statements can be obtained by following similar arguments. Let $G' = G^{S,\Delta}_{\mathsf{dsd}}$. Note that a cycle C in G has even (resp. odd) number of vertices, if and only if the cycle C' with vertex set $V(C') = \{u, w_e, w'_e, v \mid e = \{u, v\} \in E(C)\}$ and edge set $E(C') = \{\{u, w_e\}, \{w_e, w'_e\}, \{w'_e, v\} \mid e = \{u, v\} \in E(C)\}$ in G', has even (resp. odd) number of vertices. Moreover, no (simple) cycle in G' of length at least 4 can contain a vertex from $\{a_s, a'_s \mid s \in S\}$. Also, for any cycle C' in G' of odd (resp. even) length at least 4, we can obtain a cycle C in G of odd (resp. even) length at least 3, by contracting the edges incident to vertices in $W = \{w_e, w'_e \mid e \in E(G)\}$. Notice that any minimal set $S \subseteq V(G')$, such that $G' - S$ is in $\mathcal{H}_{\mathsf{odd}}$ does not contain a vertex

from $S' = \{a_s, a'_s \mid s \in S\}$, as there is no odd cycle in G' of length at least 5 that contains a vertex from S'. Also, each vertex in W has degree exactly 2 in G, and thus any cycle in G' containing them, must also contain their neighbors. From the above we can obtain that if (G', k) is a yes-instance of Vertex Deletion to $\mathcal{H}_{\mathsf{oct}}$, then there is $B \subseteq V(G) \subseteq V(G')$ of size at most k, such that $G' - B \in \mathcal{H}_{\mathsf{oct}}$. From the above discussions we can obtain that (G, S, k) is a yes-instance of Subset FVS if and only if (G', k) is a yes-instance of Vertex Deletion to $\mathcal{H}_{\mathsf{oct}}$, and in particular, we can obtain that for $B \subseteq V(G)$, $G - B$ has no odd cycle containing a vertex from S if and only if $G' - B$ has no odd length cycle with at least 5 vertices, containing a vertex from S. This concludes the first part of the proof.

We will now argue that $\mathbf{tw}_{\mathcal{H}_{\mathsf{oct}}}(G') \leq \mathbf{tw}_{\mathcal{S}_{\mathsf{oct}}}(G, S)$. To this end, consider an $\mathcal{S}_{\mathsf{oct}}$-tree decomposition, (T, χ, L) of width ℓ, for (G, S). Let T' be the tree obtained from T on $2|V(T)|$ vertices as follows. Initialize $T' = T$. For each $t \in V(T)$, we add a vertex t_{cpy} to $V(T')$ and the edge $\{t, t_{\mathsf{cpy}}\}$ to $E(T')$. We define the sets X' and L' as follows. Initialize $X' = X$. For each edge $e = \{u, v\} \in E(G)$, such that $u, v \in X$, we add w_e, w'_e to X'. Furthermore, for each $s \in S \cap X$, we add a_s, a'_s to X'. This completes the construction of X'. We set $L' = V(G') \setminus X'$. We next define a function $\chi' : V(T') \to 2^{V(G')}$, where we initialize $\chi'(t') = \emptyset$, for each $t' \in V(T')$. Roughly speaking, for each non-leaf vertex t in T', $\chi'(t)$ will contain the vertices corresponding to the subdivision of the edges that are contained in $\chi(t) \setminus L$. Furthermore, $\chi'(t)$ will also contain the K_3, if any, that are attached to the vertices in $\chi(t) \setminus L$. For $t \in V(T) \subseteq V(T')$, set $\chi'(t) = (\chi(t) \setminus L) \cup \{w_e, w'_e \mid e = \{u, v\} \in E(G)$ and $u, v \in \chi(t) \setminus L\} \cup \{a_s, a'_s \mid s \in (S \cap \chi(t)) \setminus L\}$. Note that for each $t \in V(T)$, we have $\chi'(t) \subseteq X'$ and $|\chi'(t)| \leq 3 \cdot |\chi(t) \cap L| + 2 \cdot \binom{|\chi(t) \cap L|}{2} \leq 3(\ell + 1) + 2 \cdot \binom{\ell+1}{2}$. Next (roughly speaking) for leaf-nodes t_{cpy}, we will assign the remaining vertices in accordance with $\chi(t)$ as follows. For each $t \in V(T)$, where $\chi(t) \cap L \neq \emptyset$, we let $Z^t_{\mathsf{cpy}} = \{w_e, w'_e \mid e = \{u, v\} \in E(G), u \in \chi(t) \cap L$ and $v \in \chi(t)\} \cup \{a_s, a'_s \mid s \in S \cap L \cap \chi(t)\}$. Furthermore, we set $\chi(t_{\mathsf{cpy}}) = \chi(t) \cup Z^t_{\mathsf{cpy}}$ (note that by the construction of L', we have $L \cup Z^t_{\mathsf{cpy}} \subseteq L'$.)

Recall that for any $B \subseteq V(G)$, $G - B$ has no odd cycle containing a vertex from S if and only if $G' - B$ has no odd length cycle with at least 5 vertices, containing a vertex from S. The above together with the construction of G' and (T', χ', L'), and the assumption that (T, χ, L) is an $\mathcal{S}_{\mathsf{oct}}$-tree decomposition of (G, S), implies that (T', χ', L') is an $\mathcal{H}_{\mathsf{oct}}$-tree decomposition of G' of width less than $3 \cdot (\ell + 1) + 2 \cdot \binom{\ell+1}{2}$. This concludes the proof. □

We will now show that for each $\mathcal{H} \in \{\mathcal{H}_{\mathsf{mway}}, \mathcal{H}_{\mathsf{fvs}}, \mathcal{H}_{\mathsf{oct}}\}$, Vertex Deletion to $\mathcal{H}$ admits an FPT algorithm, using the known FPT algorithms for Multiway Cut, Subset FVS and Subset OCT *parameterized by the solution size.*

Lemma 5.6. *For each $\mathcal{H} \in \{\mathcal{H}_{\mathsf{mway}}, \mathcal{H}_{\mathsf{fvs}}, \mathcal{H}_{\mathsf{oct}}\}$, Vertex Deletion to $\mathcal{H}$ admits an FPT algorithm, when parameterized by $\mathbf{mod}_{\mathcal{H}}(G)$.*

Proof. Consider $\mathcal{H} \in \{\mathcal{H}_{\mathsf{mway}}, \mathcal{H}_{\mathsf{fvs}}, \mathcal{H}_{\mathsf{oct}}\}$, and an instance (G, k) of Vertex Deletion to $\mathcal{H}$, where $k \geq 0$ (as otherwise, the problem is trivial). Let Q be the set of relevant cut vertices in G. Note that Q can be computed in polynomial time. If $Q = \emptyset$, then clearly, $G \in \mathcal{H}$, and thus we can return that (G, k) is a yes-instance of the problem. We will next consider the case when $Q \neq \emptyset$. For each $v \in Q$, let $\mathcal{D}_v$ be the set of connected components D in $G - \{v\}$ which is an edge whose both endpoints are adjacent to v, i.e., D that has exactly two vertices and for each $u \in V(D)$, we have $\{u, v\} \in E(G)$. Note that for any $v \in Q$, we have $\mathcal{D}_v \neq \emptyset$, and for each $D \in \mathcal{D}_v$, $V(D) \cap Q = \emptyset$. Let $Z = \cup_{v \in Q, D \in \mathcal{D}_v} V(D)$, and $G' = G - Z$. Notice that all of the following hold:

1. (G, k) is yes-instance of Vertex Deletion to $\mathcal{H}_{\mathsf{mway}}$ if and only if (G', Q, k) is a yes-instance of Multiway Cut.
2. (G, k) is yes-instance of Vertex Deletion to $\mathcal{H}_{\mathsf{fvs}}$ if and only if (G', Q, k) is a yes-instance of Subset FVS.

3. (G, k) is yes-instance of Vertex Deletion to $\mathcal{H}_{\mathsf{oct}}$ if and only if (G', Q, k) is a yes-instance of Subset OCT.

Thus, to resolve the instance (G, k) of Vertex Deletion to $\mathcal{H}$, when $Q \neq \emptyset$, we can now invoke the known FPT algorithms (see [62, 24, 57, 52]) for each problem, as discussed above. This concludes the proof. □

We are now ready to prove Theorem 5.1 by arguing the applicability of Theorem 1.1.

Proof of Theorem 5.1. For each $\mathcal{H} \in \{\mathcal{H}_{\mathsf{mway}}, \mathcal{H}_{\mathsf{fvs}}, \mathcal{H}_{\mathsf{oct}}\}$, we have the following properties: i) $\mathcal{H}$ is CMSO definable and closed under disjoint union (see Observation 5.4) and ii) Vertex Deletion to $\mathcal{H}$ admits an FPT algorithm, when parameterized by $\mathbf{mod}_{\mathcal{H}}(G)$. Thus, from Theorem 1.1, for each $\mathcal{H} \in \{\mathcal{H}_{\mathsf{mway}}, \mathcal{H}_{\mathsf{fvs}}, \mathcal{H}_{\mathsf{oct}}\}$, there is an FPT algorithm for Vertex Deletion to $\mathcal{H}$, parameterized by $\mathbf{tw}_{\mathcal{H}}(G)$. The above combined with Lemma 5.6 implies the proof of the theorem. □

6 Cross Parameterizations

In this section we extend our results from previous section to problems where the parameterizations is with respect to other problems. For an illustration consider Odd Cycle Transversal on chordal graphs. Let $\mathcal{H}$ denotes the family of chordal graphs. It is well known that Odd Cycle Transversal is polynomial time solvable on chordal graphs. Further, given a graph G and a modulator to chordal graphs of size $\mathbf{mod}_{\mathcal{H}}(G)$, Odd Cycle Transversal admits an algorithm with running time $2^{\mathcal{O}(\mathbf{mod}_{\mathcal{H}}(G)))}n^{\mathcal{O}(1)}$. It is natural to ask whether Odd Cycle Transversal admits an algorithm with running time $f(\mathbf{ed}_{\mathcal{H}}(G))n^{\mathcal{O}(1)}$ or $f(\mathbf{tw}_{\mathcal{H}}(G))n^{\mathcal{O}(1)}$, given a $\mathcal{H}$-elimination forest of G of depth $\mathbf{ed}_{\mathcal{H}}(G)$ and $\mathcal{H}$-decomposition of G of width $\mathbf{tw}_{\mathcal{H}}(G)$, respectively. The question is also relevant, in fact more challenging, when $\mathcal{H}$-elimination forest of G of depth $\mathbf{ed}_{\mathcal{H}}(G)$ or $\mathcal{H}$-decomposition of G of width $\mathbf{tw}_{\mathcal{H}}(G)$ are not given. We provide sufficient conditions which allows us to have an algorithm for vertex deletion problems (or edge deletion problems) when given a $\mathcal{H}$-elimination forest of G of depth $\mathbf{ed}_{\mathcal{H}}(G)$ or $\mathcal{H}$-decomposition of G of width $\mathbf{tw}_{\mathcal{H}}(G)$. Here, $\mathcal{H}$ is a family of graphs.

Theorem 6.1. *Let $\mathcal{H}$ be a family of graphs and Π be a monotone parameterized graph problem and (G, k) be an instance of Π. Further assume that we have following.*

1. *Π has FII.*
2. *Π is FPT parameterized by $\mathbf{mod}_{\mathcal{H}}(G)$.*
3. *An $\mathcal{H}$-tree decomposition (resp. $\mathcal{H}$-elimination decomposition) of G of width $\mathbf{tw}_{\mathcal{H}}(G)$ (resp. depth $\mathbf{ed}_{\mathcal{H}}(G)$) is given.*

Then, there is an algorithm that, given an n-vertex graph G and an integer k, decides whether $(G, k) \in \Pi$ in time $f(\mathbf{tw}_{\mathcal{H}}(G)) \cdot n^{\mathcal{O}(1)}$ (resp. $f(\mathbf{ed}_{\mathcal{H}}(G)) \cdot n^{\mathcal{O}(1)}$). That is, Π is FPT parameterized by $\mathbf{tw}_{\mathcal{H}}(G)$ (resp. $\mathbf{ed}_{\mathcal{H}}(G)$).

Proof. The proof of this theorem is similar to the proof of Lemma 4.3, for the sake of completeness we give a complete proof here. Consider an instance (G, k) of Π, and let (T, χ, L) be the given $\mathcal{H}$-tree decomposition (resp. $\mathcal{H}$-elimination decomposition) of width (resp. depth) at most $\mathbf{tw}_{\mathcal{H}}(G)$ (resp. $\mathbf{ed}_{\mathcal{H}}(G)$) for G, and let $\ell = \mathbf{tw}_{\mathcal{H}}(G)$ (resp. $\ell = \mathbf{ed}_{\mathcal{H}}(G)$). Let $\mathcal{P}_{\mathsf{mod}}$ be an FPT algorithm for Π, running in time $f(\mathbf{mod}_{\mathcal{H}}(G)) \cdot n^{\mathcal{O}(1)}$. Using Lemma 2.20, we will next bound the size of $\chi(t)$, for each leaf t in T.

We will assume that the constants ξ_i from Lemma 2.20, for each $i \in [\ell + 1]$ are hardcoded in the algorithm. We will bound the number of vertices in $G[\chi(t) \cap L]$ by $\sum_{i \in [\ell+1]} \xi_i$, for

each $t \in V(T)$, then the (standard) treewidth of G can be bounded by $\ell + \sum_{i\in[\ell+1]} \xi_i$ (resp. $\ell + 1 + \sum_{i\in[\ell+1]} \xi_i$). Using the property that Π has FII, (roughly speaking) our next objective will be to bound the size of $\chi(t)$, for each $t \in V(T)$ by replacements using Lemma 2.20. Let $\widetilde{A}$ be the set of nodes in T whose bag contains more than $\sum_{i\in[\ell+1]} \xi_i$ vertices from L, i.e., $\widetilde{A} = \{t \in V(T) \mid |\chi(t) \cap L| > \sum_{i\in[\ell+1]} \xi_i\}$. Furthermore, let $\widetilde{\mathcal{G}}$ be the set of graphs induced by vertices in L, in each of the bags of nodes in $\widetilde{A}$, i.e., $\widetilde{\mathcal{G}} = \{G[\chi(t) \cap L] \mid t \in \widetilde{A}\}$. We let $\widetilde{\mathcal{G}} = \{\widetilde{G}_1, \widetilde{G}_2, \cdots, \widetilde{G}_q\}$.

We create a sequence of $G_0, G_1, \cdots, G_q$ graphs and a sequence of constant $c_0, c_1, \cdots, c_q$ as follows. Intuitively speaking, we will obtain the above sequence of graph by replacing $\widetilde{G}_i$, for $i \in [q]$, by some graph obtained using Lemma 2.20. Set $G_0 = G$ and $c_0 = 0$. We iteratively compute G_i, for each $i \in [q]$ (in increasing order) as follows. For the graph $\widetilde{G}_i$, let $\widetilde{t}_i$ be the unique leaf in T, such that $V(\widetilde{G}_i) \subseteq \chi(\widetilde{t}_i)$, and let $\widetilde{B}_i = \chi(\widetilde{t}_i) \setminus V(\widetilde{G}_i)$ and $\widetilde{b}_i = |\widetilde{B}_i|$. Note that $|\widetilde{B}_i| \leq \ell$ (resp. $|\widetilde{B}_i| \leq \ell+1$), as (T, χ, L) is an $\mathcal{H}$-elimination decomposition (resp. $\mathcal{H}$-tree decomposition) of G of depth (resp. width) ℓ. Fix an arbitrary injective function $\lambda_{\widetilde{G}_i} : \widetilde{B} \to \{1, 2, \cdots, \widetilde{b}_i\}$, and then $\widetilde{G}_i$ is the boundaried graph with boundary $\widetilde{B}_i$.[16] Note that $V(\widetilde{G}_i]) \subseteq V(G_{i-1})$. Also, let G'_i be the boundaried graph $G_{i-1} - V(\widetilde{G}_i)$, with boundary $\widetilde{B}_i$. Using Lemma 2.20 and the algorithm $\mathcal{P}_{\mathsf{mod}}$, we find the graph $\widetilde{G}^*_i$ and the translation constant c_i, such that $\widetilde{G}_i \equiv_\Pi \widetilde{G}^*_i$ and $|V(\widetilde{G}^*_i)| \leq \xi_{\widetilde{b}_i}$ in time bounded by $\mathcal{O}(f(\ell) \cdot n^{\mathcal{O}(1)})$.[17]

We let G_i be the graph $\widetilde{G}^*_i \oplus G'_i$, which can be computed in time bounded by $\mathcal{O}(f(\ell) \cdot n^{\mathcal{O}(1)})$. Let $c^* = \sum_{i\in[q]} c_i$. With the constructions described above, we are now in a position to prove Lemma 4.3. Note that to obtain the desired result, it is enough to prove the following statements.

1. The instance $(G_q, k + c^*)$ can be constructed in time bounded by $\mathcal{O}(g(\ell) \cdot n^{\mathcal{O}(1)})$, for some function g,
2. $(G_q, k + c^*)$ and (G, k) are equivalent instances of Π, and
3. The treewidth of G_q is at most $\ell + \max_{i\in[q]} c_i + \sum_{i\in[\ell]} \zeta_i$ and the $\mathcal{H}$-treewidth (resp. $\mathcal{H}$-elimination distance) of G_q is at most $\ell + \max_{i\in[q]} c_i$.[18]

As stated perviously, we will assume that the constants ξ_i, for $i \in [\ell + 1]$ are hardcoded in the algorithm. Thus, we can construct the set $\widetilde{\mathcal{G}}$ in polynomial time. Also, note that for any $i \in [q]$, $\iota(\widetilde{G}^*_i \oplus G'_i) \leq \iota(G) \leq |V(G)|$ (see Definition 2.19). Thus, for some function f, we can construct the instance $(G_q, k + c^*)$ in time bounded by $g(\ell) \cdot n^{\mathcal{O}(1)}$.

We will inductively argue that for each $i \in [q]_0$, $(G_i, k + \sum_{j\in[i]_0} c_j)$ and (G, k) are equivalent instances of Π. As $G_0 = G$ and $k + c_0 = k$, the claim trivially follows for the case when $i = 0$. Next we assume that for some $q' \in [q-1]_0$, for each $i' \in [q']_0$, $(G_{i'}, k + \sum_{j\in[i']_0} c_j)$ and (G, k) are equivalent instances of the problem. We will next prove the statement for $i = i' + 1$. It is enough to argue that $(G_{i-1}, k + \sum_{j\in[i-1]_0} c_j)$ and $(G_i, k + \sum_{j\in[i]_0} c_j)$ are equivalent instances. Recall that, by construction, $G[V(\widetilde{G}_i)] = G_{i-1}[V(\widetilde{G}_i)] = \widetilde{G}_i$ and $G'_i = G_{i-1} - V(\widetilde{G}_i)$ are boundaried graphs with boundary $\widetilde{B}_i$, and $G_i = \widetilde{G}_i \oplus G'_i$. From Lemma 2.20, $\widetilde{G}^*_i \equiv_\Pi \widetilde{G}_i$. Thus by definition, we have that $(G_{i-1}, k + \sum_{j\in[i-1]_0} c_j)$ and $(G_i, k + \sum_{j\in[i]_0} c_j)$ are equivalent instances of Π.

To prove the third statement, note that it is enough to construct an $\mathcal{H}$-tree decomposition, (T_q, χ_q, L_q), of G_q, where for each $t \in V(T_q)$, we have $|\chi_q(t)| \leq \ell + \max_{i\in[q]} c_i + \sum_{i\in[\ell]} \chi_i$ and $|\chi_q^{-1}(t) \setminus L_q| \leq \ell + \max_{i\in[q]} c_i$. (Using similar arguments we can also obtain the statement regarding $\mathcal{H}$-elimination decomposition.) Let $X = \cup_{i\in[q]} V(\widetilde{C}_i)$, and $L_q = (L \setminus X) \cup (V(G_q) \setminus$

[16] We have slightly abused the notation, and used $\widetilde{G}_i$ to denote both a graph and a boundaried graph.

[17] Note that for any graph $\widehat{G}$, $\iota(\widehat{G}) \leq |V(\widehat{G})|$ (see Definition 2.19).

[18] We remark that although $k + c^*$ can possibly be much larger than k, both the treewidth and the $\mathcal{H}$-treewidth of G_q are at most some additive constants (depending on $\mathcal{H}$) away from k.

$V(G)$) and $T_q = T$. For each $t \in V(T) \setminus \widetilde{A}$, we set $\chi_q(t) = \chi(t)$, and for each $i \in [q]$, we set $\chi_q(\widetilde{t}_i) = (\chi(\widetilde{t}_i) \setminus V(\widetilde{G}_i)) \cup V(\widetilde{G}_i^*)$. For each $i \in [q]$, note that $|V(\widetilde{G}_i^*)| \leq \xi_{\widetilde{b}_i} \leq \sum_{j \in [\ell]} \xi_j$. Thus we can obtain that (T_q, χ_q, L_q) is an $\mathcal{H}$-tree decomposition of G_q that satisfies all the required properties. This concludes the proof. □

7 Uniform FPT Algorithm

In this section we design a uniform FPT algorithm for Elimination Distance to $\mathcal{H}$ parameterized by $\mathbf{ed}_{\mathcal{H}}(G)$ assuming we have a *specific type* of algorithm for solving Vertex Deletion to $\mathcal{H}$ parameterized by $\mathbf{mod}_{\mathcal{H}}(G)$. Our algorithm is generic and in the next section we explain the corollaries of this result for various families $\mathcal{H}$. Throughout this section we assume that $\mathcal{H}$ is a hereditary family of graphs and it is closed under disjoint union. Throughout the section, k is the parameter in the input instance of Elimination Distance to $\mathcal{H}$. Unless specified all the graphs mentioned in this section are $2k$-boundaried graphs and we assume that $k \geq 2$. Our algorithm uses the recursive understanding technique [17, 42, 53].

Here, we identify a (q, k)-unbreakable induced subgraph from an input graph and we replace it with a smaller representative graph, where q is bounded by function of k. Eventually, when the size of the graph become bounded by a function of k, we use a brute-force algorithm to solve the problem.

Definition 7.1 (Canonical equivalence relation)**.** *The canonical equivalence relation $\equiv_{\mathcal{H}}$ for $\mathcal{H}$ over the set of boundaried graphs is defined as follows. Two graphs G_1 and G_2 are equivalent if for any graph H, $G_1 \oplus H \in \mathcal{H} \Leftrightarrow G_2 \oplus H \in \mathcal{H}$.*

We assume that we are given a refinement of the equivalence relation $\equiv_{\mathcal{H}}$ which we call a *user defined equivalence relation*. We use $\equiv_u$ to denote this refinement of the equivalence relation $\equiv_{\mathcal{H}}$. We use $\mathcal{Q}(\equiv_u)$ to denote the set of equivalence classes of $\equiv_u$.

Definition 7.2 (Comparison of equivalence classes)**.** *Let Q_1 and Q_2 be two equivalence classes in $\equiv_u$. Let $G_1 \in Q_1$ and $G_2 \in Q_2$. We say that Q_1 is at least as good as Q_2 if the following holds. For any graph H, if $G_2 \oplus H \in \mathcal{H}$, then $G_1 \oplus H \in \mathcal{H}$.*

Throughout the section we also assume that along with $\equiv_u$, we are given a user defined function $\mathsf{ug}\colon \mathcal{Q}(\equiv_u) \times \mathcal{Q}(\equiv_u) \mapsto \{0, 1\}$ such that $\mathsf{ug}(Q, Q) = 1$ for all $Q \in \mathcal{Q}(\equiv_u)$ and it has the following property. If $\mathsf{ug}(Q_1, Q_2) = 1$, then Q_1 is at least as good as Q_2. We would like to mention that the reverse may not be true. That is, even if Q_1' is at least as good as Q_2', the user may define that $\mathsf{ug}(Q_1', Q_2') = 0$. We also assume that equivalence classes in $\equiv_u$ satisfies the following property.

Definition 7.3 (Component deletion property)**.** *Let G be a graph and Q is an equivalence class in $\equiv_u$ such that $G \in Q$. Let C be a connected component of G such that $V(C) \cap \delta(G) = \emptyset$. Then, $G - C$ belongs to Q. Moreover, if F is a graph in $\mathcal{H}$ and $\delta(F) = \emptyset$, then $G \uplus F \in Q$.*

Moreover, we assume that we have an access to an algorithm, called *user's algorithm*, denoted by $\mathcal{A}_u$ that, given a graph G, a non-negative integer k' and equivalence class $Q \in \mathcal{Q}(\equiv_u)$, outputs a vertex subset S of size at most k' such that $G - S$ belongs to an equivalence class which is at least as good as Q (if it exists). If no such set exists, then the algorithm outputs No. We use $f_u(k', |\mathcal{Q}(\equiv_u)|, n)$ to denote the running time of $\mathcal{A}_u$ where $n = |V(G)|$.

Next, we define the notion of state-tuple. Here two "graphs having same set of state-tuples" can be treated as identical.

Definition 7.4 (State-tuple)**.** *A state-tuple is a tuple $(D, T, \chi, L, \mathcal{P}, \mathsf{eq})$, where $L \subseteq D \subseteq [2k]$, $\mathcal{P}$ is a partition of L, $\mathsf{eq}\colon \mathcal{P} \to \mathcal{Q}(\equiv_u)$, T is a rooted forest of depth at most k and $|V(T)| \leq (2k)^2$, and $\chi\colon V(T) \to 2^D \cup \{\spadesuit, \heartsuit\}$ such that the following holds.*

1. *For each internal node t of T we have $\chi(t) \in \{\spadesuit, \heartsuit, \emptyset\} \cup \{\{x\} \colon x \in (D \setminus L)\}$.*
2. *For each leaf t of T, $\chi(t) \in \mathcal{P} \cup \{\{x\} \colon x \in (D \setminus L)\}$.*
3. *The sets $(\chi(t))_{t \in N}$ form a partition of D, where $N = \{t \in V(T) \colon \chi(t) \notin \{\spadesuit, \heartsuit, \emptyset\}\}$*

Next we define a notation which we use throughout the section. Let G be a graph and T be a rooted forest. Let χ be a function from $V(T)$ to a super set of $2^{V(G)}$. Let Y be the nodes of T that are mapped to subsets containing at least one vertex from $\delta(G)$ by the function χ. Then, we use $I_T(G, \chi)$ to denote the set of nodes of T that belong to the unique paths in T from the vertices in Y to the corresponding roots.

Definition 7.5. *Let G and be a graph and $s = (D, T, \chi, L, \mathcal{P}, \mathsf{eq})$ be a state-tuple. We say that G satisfies s if $\Lambda(G) = D$ and there is an $\mathcal{H}$-elimination decomposition $(\widehat{T}, \widehat{\chi}, \widehat{L})$ of G, of depth at most k with the following properties.*

(a) T is isomorphic to $\widehat{T}[Z]$, where $Z = I_{\widehat{T}}(G, \widehat{\chi})$. For any node $t \in Z$, we use the same notation to represent the node in T that maps to t by the isomorphism function. That is, $V(T) = Z$.

(b) For any node t of T with $\widehat{\chi}(t) \cap \delta(G) \neq \emptyset$, $\chi(t) = \lambda_G(\widehat{\chi}(t) \cap \delta(G))$

(c) For any node t of T with $\widehat{\chi}(t) \neq \emptyset$ and $\widehat{\chi}(t) \subseteq V(G) \setminus \delta(G)$, $\chi(t) = \spadesuit$.

(d) Let $t_1, \ldots, t_\ell$ be the leaf nodes of $\widehat{T}$ that belong to Z. Then, $\mathcal{P} = (\chi(t_i))_{i \in [\ell]}$.

(e) For each node t of T with $\chi(t) \in \mathcal{P}$, $G[\widehat{\chi}(t)]$ belongs to an equivalence class $Q \in \mathcal{Q}(\equiv_u)$ which is at least as good as $\mathsf{eq}(\chi(t))$. That is, $\mathsf{ug}(Q, \mathsf{eq}(\chi(t)) = 1$.

Also, we say that G satisfies s through $(\widehat{T}, \widehat{\chi}, \widehat{L})$.

Definition 7.6. *We say that a graph G exactly satisfies a state-tuple $(D, T, \chi, L, \mathcal{P}, \mathsf{eq})$ if G satisfies $(D, T, \chi, L, \mathcal{P}, \mathsf{eq})$ such that a stricter condition than condition (e) in Definition 7.5 holds. That is, for each node t of T with $\chi(t) \in \mathcal{P}$, $G[\widehat{\chi}(t)]$ belongs to the equivalence class $\mathsf{eq}(\chi(t))$.*

Definition 7.7. *Signature of a graph G, denoted by $\mathsf{real\text{-}sig}(G)$, is the set of all state-tuples s such that G satisfies s.*

Observation 7.8. *Signature of a graph G is unique and its cardinality is bounded by a function k and $|\mathcal{Q}(\equiv_u)|$.*

For two graphs G and H, and an $\mathcal{H}$-elimination decomposition $(\widehat{T}, \widehat{\chi}, \widehat{L})$ of $G \oplus H$, we use $\mathsf{Restricted}_G(\widehat{T}, \widehat{\chi}, \widehat{L})$ to denote the $\mathcal{H}$-elimination decomposition $(\widehat{T}, \psi, F)$ of G obtained by restricting $(\widehat{T}, \widehat{\chi}, \widehat{L})$ to G. Formally $F = \widehat{L} \cap V(G)$ and for any node $t \in V(\widehat{T})$, $\psi(t) = \widehat{\chi}(t) \cap V(G)$.

Definition 7.9. *Let G and H be two graphs and $s = (D, T, \chi, L, \mathcal{P}, \mathsf{eq})$ be a state-tuple. We say that (G, H) realizes s, if $\Lambda(G) = D$ and there is an $\mathcal{H}$-elimination decomposition $(\widehat{T}, \widehat{\chi}, \widehat{L})$ of $G^\star = G \oplus H$, of depth at most k with the following conditions.*

(i) G satisfies s through $\mathsf{Restricted}_G(\widehat{T}, \widehat{\chi}, \widehat{L})$. That is, conditions (a)-(e) in Definition 7.5 hold.

(ii) For each node t of T with $\widehat{\chi}(t) \neq \emptyset$ and $\widehat{\chi}(t) \subseteq V(H) \setminus \delta(G)$, $\chi(t) = \heartsuit$. (Recall that T is isomorphic to $\widehat{T}[Z]$, where $Z = I_{\widehat{T}}(G, \widehat{\chi})$).

Then, we say that (G, H) realizes s through $(\widehat{T}, \widehat{\chi}, \widehat{L})$. We say that (G, H) exactly realizes s, if instead of conditions (i), we have that G exactly satisfies s through $\mathsf{Restricted}_G(\widehat{T}, \widehat{\chi}, \widehat{L})$. In that case, we say that (G, H) exactly realizes s through $(\widehat{T}, \widehat{\chi}, \widehat{L})$.

Definition 7.10. *A marked signature is a subset of $\{s\colon s \text{ is a state-tuple}\} \times \{0, 1\}$.*

Observation 7.11. *The number of marked signatures are bounded by a function of k and the number of equivalence classes in $\equiv_u$.*

In a marked signature every state-tuple is marked as 0 or 1, where we interpret the marking with 0 as marking "do not care". Later we will define *marked signature of a graph G* where for any tuple s that is marked with 0 (do not care), there is a tuple in the signature which is "strictly better than" s. Towards that we define the following.

Definition 7.12. *Let $s_1 = (D_1, T_1, \chi_1, L_1, \mathcal{P}_1, \mathsf{eq}_1)$ and $s_2 = (D_2, T_2, \chi_2, L_2, \mathcal{P}_2, \mathsf{eq}_2)$ be two state tuples. We say that s_2 is strictly better than s_1 if the following holds.*

- *The set $D_2 \setminus L_2$ is a strict super set of $D_1 \setminus L$.*
- *For any two graphs G and H, if (G, H) exactly realizes s_1, then (G, H) realizes s_2.*

Definition 7.13 (Validity)**.** *Let sig be a marked signature. We say that sig is valid if the following holds. For any pair $(s_1 = (D_1, T_1, \chi_1, L_1, \mathcal{P}_1, \mathsf{eq}_1), 0)$ in sig (i.e., a tuple that is marked with 0), there is a pair $(s_2 = (D_2, T_2, \chi_2, L_2, \mathcal{P}_2, \mathsf{eq}_2), b)$ in sig, such that s_2 is strictly better than s_1, where $b \in \{0, 1\}$.*

Definition 7.14 (compatibility)**.** *Let G be a graph and sig be a marked signature. We say that sig is compatible with G if sig is valid and the following holds.*

- *For any $s \in \mathsf{real\text{-}sig}(G)$, $\{(s, 0), (s, 1)\} \cap sig \neq \emptyset$.*
- *For any $s \notin \mathsf{real\text{-}sig}(G)$, if $(s, b) \in sig$, then $b = 0$.*

We would like to mention that for a graph G, there could be many marked signatures that are compatible with G.

Definition 7.15 (Similarity)**.** *Let sig_1 and sig_2 be two marked signatures. We say that sig_1 and sig_2 are similar if the following holds.*

- *$\{s\colon (s, 1) \in sig_1\} \subseteq \{s'\colon (s', 0) \in sig_2$ or $(s', 1) \in sig_2\}$, and*
- *$\{s\colon (s, 1) \in sig_2\} \subseteq \{s'\colon (s', 0) \in sig_1$ or $(s', 1) \in sig_1\}$.*

Next we prove that if the "real signatures" of two graphs G_1 and G_2 are same, then for any two marked signatures sig_1 and sig_2 that are compatible with G_1 and G_2, respectively, the marked signatures sig_1 and sig_2 are similar.

Lemma 7.16. *Let G_1 and G_2 be two graphs such that $\mathsf{real\text{-}sig}(G_1) = \mathsf{real\text{-}sig}(G_2)$. Let sig_1 and sig_2 be two marked signatures such that sig_1 is compatible with G_1 and sig_2 is compatible with G_2. Then sig_1 and sig_2 are similar.*

Proof. Suppose sig_1 and sig_2 are not similar. Then, at least one of the following statements is true.

(i) There exists a state-tuple s such that $(s, 1) \in sig_1$ and $(s, 0), (s, 1) \notin sig_2$.

(ii) There exists a state-tuple s such that $(s, 1) \in sig_2$ and $(s, 0), (s, 1) \notin sig_1$.

Since the above statements are symmetric we assume that statement (i) is true and derive a contradiction. Since $(s,1) \in sig_1$, we have that $s \in \mathsf{real\text{-}sig}(G_1) = \mathsf{real\text{-}sig}(G_2)$. Since $s \in \mathsf{real\text{-}sig}(G_2)$ we have that either $(s,1) \in sig_2$ or $(s,0) \in sig_2$. This is a contradiction to the assumption that $(s,0),(s,1) \notin sig_2$. □

Definition 7.17. *We say that a graph R* represents *a graph G, if the following holds. For any graph H, $(G \oplus H, k)$ is yes-instance of* Elimination Distance to $\mathcal{H}$ *if and only if $(R \oplus H, k)$ is yes-instance of* Elimination Distance to $\mathcal{H}$.

Observation 7.11 implies the following lemma.

Lemma 7.18. *There is a function r such that for any graph G, there is a graph R of size $r(k, |\mathcal{Q}(\equiv_u)|)$, that represents G.*

Next we state two important lemmas and then using them prove our results. Later we prove both the lemmas.

Lemma 7.19. *Let G_1 and G_2 be two graphs such that $\Lambda(G_1) = \Lambda(G_2)$. Let sig_1 and sig_2 be two marked signatures such that sig_1 is compatible with G_1 and sig_2 is compatible with G_2. Moreover, sig_1 and sig_2 are similar. Then, for any graph H, $(G_1 \oplus H, k)$ is yes-instance of* Elimination Distance to $\mathcal{H}$ *if and only if $(G_2 \oplus H, k)$ is yes-instance of* Elimination Distance to $\mathcal{H}$.

We prove Lemma 7.19 in Section 7.1. Next we prove that given a graph (q,k)-unbreakable graph G, a marked signature that is compatible with G can be computed efficiently. Recall that we are given an algorithm $\mathcal{A}_u$ to find a deletion set of an input graph to a given equivalence class in $\mathcal{Q}(\equiv_u)$ and its running time is denoted by the function f_u.

Lemma 7.20. *There is a function g and an algorithm that given a (q,k)-unbreakable graph G (recall that G is also a $2k$-boundaried graph) as input, runs in time $g(k,q,|\mathcal{Q}(\equiv_u)|) \cdot f_u(k,|\mathcal{Q}(\equiv_u)|,n) \cdot n^{\mathcal{O}(1)}$, and outputs a marked signature that is compatible with G, where $n = |V(G)|$.*

We prove Lemma 7.20 in Section 7.2. Now, using Lemmas 7.19 and 7.20, we define a uniform FPT algorithm for Elimination Distance to $\mathcal{H}$. We know that by Lemma 7.18, for any graph G, there is a representative of G and its size bounded a function r of k and $|\mathcal{Q}(\equiv_u)|$. We remark that this is an existential result and we do not the function r. So first we design an algorithm for Elimination Distance to $\mathcal{H}$ assuming we know the value $r(k,|\mathcal{Q}(\equiv_u)|)$. Later we explain how to get rid of this assumption. In the rest of the section we use $q = 2^{k+1} \cdot r(k,|\mathcal{Q}(\equiv_u)|)$.

Lemma 7.21. *There is a function g' and an algorithm $\mathcal{A}_r$ that given a (q,k)-unbreakable graph G, runs in time $g'(k,q,|\mathcal{Q}(\equiv_u)|) \cdot f_u(k,|\mathcal{Q}(\equiv_u)|,n) \cdot n^{\mathcal{O}(1)}$ and outputs a graph R of size at most $r(k,|\mathcal{Q}(\equiv_u)|)$ such that R represents G, where $n = |V(G)|$. Here f_u is the running time of the user's algorithm $\mathcal{A}_u$.*

Proof. Our algorithm uses Lemma 7.20. The pseudocode of our algorithm is given in Algo-

rithm 7.1.

Result: A graph R that represents the input (q,k)-unbreakable graph G
Using Lemma 7.20 compute a marked signature sig_G that is compatible with G;
for $i = 1$ *to* $r(k, |\mathcal{Q}(\equiv_u)|)$ **do**
  for *every graph* R *on* i *vertices* **do**
    Using Lemma 7.20 compute a marked signature sig_R that is compatible with R;
    if sig_G *and* sig_R *are similar* **then**
      Output R;
    end
  end
end

Algorithm 7.1: Algorithm $\mathcal{A}_r$

Next we prove the correctness of the algorithm. By Lemma 7.18, we know that there is a representative of G, of size at most $r(k, |\mathcal{Q}(\equiv_u)|)$. Thus, there is a graph R of size at most $r(k, |\mathcal{Q}(\equiv_u)|)$ such that $\mathsf{real\text{-}sig}(G) = \mathsf{real\text{-}sig}(R)$. Hence, by Lemma 7.16, for any marked signatures sig_R and sig_G that are compatible with R and G, respectively, we have that sig_G and sig_R are similar. Moreover, by Lemma 7.19 if there is a graph R and a marked signature sig_R compatible with R such that sig_G and sig_R are similar, then R represents G. Therefore, the algorithm is correct.

Notice that the algorithm runs the algorithm of Lemma 7.20 at most $r(k, |\mathcal{Q}(\equiv_u)|)+1$ times. This implies that the total running time of the algorithm is upper bounded by $g'(k, q, |\mathcal{Q}(\equiv_u)|) \cdot f_u(k, |\mathcal{Q}(\equiv_u)|, n) \cdot n^{\mathcal{O}(1)}$ for some function g'. This completes the proof of the lemma. □

There is an algorithm to determine (approximately) whether a graph is unbreakable. We use the statement from [60], although lemmas similar to it can be found in [17].

Proposition 7.22. [60] *There is an algorithm* Break-ALG*, that given two positive integer* $s, c \in \mathbb{N}$ *and a graph* G*, runs in time* $2^{\mathcal{O}(c \log(s+c))} \cdot n^3 \log n)$ *and either returns an* $(\frac{s}{2^c}, c)$*-witnessing separation or correctly concludes that* G *is* (s,c)*-unbreakable.*

Next we prove the following theorem. The proof of the theorem has the same *general template* employed by algorithms based on the recursive understanding technique. So, we give a proof sketch of Theorem 7.1.

Theorem 7.1. *There is a function* f *and an algorithm for* Elimination Distance to $\mathcal{H}$ *running in time* $f(k, q, |\mathcal{Q}(\equiv_u)|) \cdot f_u(k, |\mathcal{Q}(\equiv_u)|, n) \cdot n^{\mathcal{O}(1)}$*. Here, we assume that we know the value* $r(k, |\mathcal{Q}(\equiv_u)|)$*.*

Proof sketch. We design a recursive algorithm, denoted by $\mathcal{A}$, where the input is a $2k$-boundaried graph G' and the output is a graph of size at most $r(k, |\mathcal{Q}(\equiv_u)|)$ that represents the input graph. The following are the steps of the recursive algorithm $\mathcal{A}$.

(1) Apply Proposition 7.22 on (G', q, k) and it outputs either an $(\frac{q}{2^k}, k)$-witnessing separation (X, Y) or conclude that G' is (q,k)-unbreakable.

(2) If G' is (q,k)-unbreakable, then compute a representative R of G' using Lemma 7.21.

(3) Suppose we get an $(\frac{q}{2^k}, k)$-witnessing separation (X, Y) in Step (1). Let G_1 be the boundaried graph $G[X]$ with $\delta(G_1) = (\delta(G) \cup S) \cap V(G_1)$ and G_2 be the boundaried graph $G[Y]$ with $\delta(G_2) = (\delta(G) \cup S) \cap V(G_2)$. Since $|\delta(G)| \leq 2k$, either G_1 or G_2 is a $2k$-boundaried graph. Without loss of generality let G_1 is be a $2k$-boundaried graph. Since (X, Y) is a $(\frac{q}{2^k}, k)$-witnessing separation, we have that $|V(G_1)| \geq \frac{q}{2^k} \geq 2r(k, |\mathcal{Q}(\equiv_u)|)$.

(4) Recursively call $\mathcal{A}$ and compute a representative R of G_1 of size at most $r(k, |\mathcal{Q}(\equiv_u)|)$.

(5) Recursively call $\mathcal{A}$ on $R \oplus_\delta G_2$ and output the result.

Let (G, k) be the given input instance of Elimination Distance to $\mathcal{H}$. We assume that G is a $2k$-boundaried graph with $\delta(G) = \emptyset$. Then run $\mathcal{A}$ on G and let R be the output. Then we do a brute force computation on R and output accordingly. The correctness of our algorithm follows from the correctness of Lemma 7.21, Proposition 7.22, and Definition 7.17.

Now we analyse the running time. Let $T(n, k, |\mathcal{Q}(\equiv_u)|)$ be the running time of the algorithm $\mathcal{A}$. In the algorithm we make two recursive calls where the size of the input graphs are at most $|V(G_1)|$ and $|V(G_2)| + r(k, |\mathcal{Q}(\equiv_u)|)$. Here $|V(G_1)| + |V(G_2)| \leq n + 2k$. This implies that the running time has the following recurrence relation.

$$T(n, k, |\mathcal{Q}(\equiv_u)|) = T(n_1, k, |\mathcal{Q}(\equiv_u)|) + T(n_2 + r(k, |\mathcal{Q}(\equiv_u)|), k, |\mathcal{Q}(\equiv_u)|) + d(k, \mathcal{Q}(\equiv_u), n)$$

where $n_1 + n_2 \leq n + 2k$, and $d(k, \mathcal{Q}(\equiv_u), n)$ is the time taken for Steps (1) and (2). The base case is $T(2q - 1, k, |\mathcal{Q}(\equiv_u)|) \leq d(k, \mathcal{Q}(\equiv_u), 2q - 1)$. By solving the above recursive formula, we get that the total running time is at most $f_1(k, q, |\mathcal{Q}(\equiv_u)|) \cdot f_u(k, |\mathcal{Q}(\equiv_u)|, n) \cdot n^{\mathcal{O}(1)}$ for some function f_1.

Now to solve the problem, first we run $\mathcal{A}$ on the input graph and get a representative $G^\star$ of size at most $r(k, |\mathcal{Q}(\equiv_u)|)$ for the input graph. Then, we do a brute force computation on $G^\star$ and output accordingly. The running time of the algorithm follows from the running time of $\mathcal{A}$ and the fact that the size of $G^\star$ is at most $r(k, |\mathcal{Q}(\equiv_u)|)$. This completes the proof of the theorem. □

Next, we explain how to get rid of the assumption that we know the value $r(k, |\mathcal{Q}(\equiv_u)|)$ in Theorem 7.1. First, we notice that if we choose a value r' instead of $r(k, |\mathcal{Q}(\equiv_u)|)$ and if the algorithm in Lemma 7.21 succeeds, then the output is a representative of the input graph. If the algorithm terminates without producing an output, then our choice r' for $r(k, |\mathcal{Q}(\equiv_u)|)$ is wrong. In that case we say that the algorithm fails. Now, we run the algorithm in Theorem 7.1 by substituting values $1, 2, 3, \ldots$ instead of $r(k, |\mathcal{Q}(\equiv_u)|)$ in the order. If for a value r' the algorithm in Lemma 7.21 fails restart the whole algorithm with the next value $r' + 1$. Clearly we will succeed on or before we reach the value $r(k, |\mathcal{Q}(\equiv_u)|)$. Thus, we have the following theorem.

Theorem 7.2. *There is a function f and an algorithm for* Elimination Distance to $\mathcal{H}$ *running in time $f(k, q, |\mathcal{Q}(\equiv_u)|) \cdot f_u(k, |\mathcal{Q}(\equiv_u)|, n) \cdot n^{\mathcal{O}(1)}$. Here f_u is the running time of the user's algorithm $\mathcal{A}_u$.*

7.1 Proof of Lemma 7.19

To prove the lemma it is enough to prove that for any graph H such that $(G_1 \oplus H, k)$ is a yes-instance of Elimination Distance to $\mathcal{H}$, $(G_2 \oplus H, k)$ is also a yes-instance of Elimination Distance to $\mathcal{H}$. Towards that let H be an arbitrary graph such that $(G_1 \oplus H, k)$ is a yes-instance of Elimination Distance to $\mathcal{H}$. That is, $\mathbf{ed}_{\mathcal{H}}(G_1 \oplus H) \leq k$. First, let us fix some notations. Let $G_1^\star = G_1 \oplus H$, $G_2^\star = G_2 \oplus H$, $B_1 = \delta(G_1)$, $B_2 = \delta(G_2)$, and $D = \Lambda(G_1) = \Lambda(G_2)$. Among all the $\mathcal{H}$-elimination decompositions of $G_1^\star$, of depth at most k (recall that $\mathbf{ed}_{\mathcal{H}}(G_1^\star) \leq k$), let $(\widehat{T}_1, \widehat{\chi}_1, \widehat{L}_1)$ be an $\mathcal{H}$-elimination decomposition of $G_1^\star$ with maximum number of vertices from B_1 are deleted (i.e., the number of internal nodes in $\widehat{T}_1$ that are mapped to subsets of B_1 by $\widehat{\chi}$ is maximized).

Now we will derive a state-tuple $s = (D, T, \chi, L, \mathcal{P}, \mathsf{eq})$ in $\mathsf{real\text{-}sig}(G_1)$ from $(\widehat{T}_1, \widehat{\chi}_1, \widehat{L}_1)$ such that (G_1, H) exactly realizes s through $(\widehat{T}_1, \widehat{\chi}_1, \widehat{L}_1)$. Recall that $D = \Lambda(G_1) = \Lambda(G_2)$.

(*i*) The rooted forest T is isomorphic to $\widehat{T}_1[Z_1]$, where $Z_1 = I_{\widehat{T}_1}(G_1, \widehat{\chi}_1)$. For each node $t \in Z_1$, we use the same notation to represent the node in T that is mapped to t by the isomorphism function. That is, $V(T) = Z_1$. Since $|B_1| \leq 2k$ and the depth of the rooted forest $\widehat{T}_1$ is at most k, we have that $|V(T)| \leq (2k)^2$.

(*ii*) Next we define χ. For any node t in T, if $\widehat{\chi}_1(t) \cap B_1 \neq \emptyset$, then we set $\chi(t) = \lambda_{G_1}(\widehat{\chi}_1(t) \cap B_1)$. For any node t in T, if $\widehat{\chi}_1(t) \subseteq V(G_1) \setminus B_1$, then we set $\chi(t) = \spadesuit$. For any node t in T, if $\widehat{\chi}_1(t) \subseteq V(G_1^\star) \setminus V(G_1)$, then we set $\chi(t) = \heartsuit$.

(*iii*) We define $L = \lambda_{G_1}(B_1 \cap \widehat{L}_1)$. From the construction of T and χ, notice that for any leaf node t of $\widehat{T}_1$, if $t \in V(T)$, then $\chi(t) \subseteq L$. Let $t_1, \ldots, t_\ell$ be the leaves of $\widehat{T}_1$ that belong to $V(T)$. Then, $(\chi(t_i))_{i \in [\ell]}$ forms a partition of L. We set $\mathcal{P} = (\chi(t_i))_{i \in [\ell]}$.

(*iv*) Now, for each $i \in [\ell]$, we set $\mathsf{eq}(\chi(t_i))$ to be the the equivalence class of $\equiv_u$ that contains $G_1[\widehat{\chi}_1(t_i)]$.

This completes the construction of the state-tuple $s = (D, T, \chi, L, \mathcal{P}, \mathsf{eq})$. From the construction of s, we have that $s \in \mathsf{real\text{-}sig}(G_1)$ and (G_1, H) exactly realizes s through $(\widehat{T}_1, \widehat{\chi}_1, \widehat{L}_1)$ (see Definitions 7.6 and 7.9).

Claim 7.23. $(s, 1)$ *belongs to* sig_1.

Proof. Since $s \in \mathsf{real\text{-}sig}(G_1)$ and sig_1 is compatible with G_1, we have that at least one among $(s, 0)$ and $(s, 1)$ belongs to sig_1. To prove that $(s, 1) \in sig_1$, we prove that $(s, 0)$ does not belong to sig_1. Since sig_1 is a valid marked signature, if $(s, 0) \in sig_1$, then there is a pair $(s_1 = (D_1, T_1, \chi_1, L_1, \mathcal{P}_1, \mathsf{eq}_1), b) \in sig_1$ such that $D_1 \setminus L_1$ is a strict super set of $D \setminus L$ and s_1 is at least as good as s. This implies that since (G_1, H) exactly realizes s, (G_1, H) realizes s_1 and more labels of the boundary vertices are there in $D_1 \setminus L_1$ than in $D \setminus L$. That is, there is an $\mathcal{H}$-elimination decomposition $(T^\star, \chi^\star, L^\star)$ of $G_1^\star$ such that the number of internal nodes in $T^\star$ that are mapped to subsets of B_1 by $\chi^\star$ is strictly more than the number of internal nodes in $\widehat{T}$ that are mapped to subsets of B_1 by $\widehat{\chi}$. This contradicts our choice of the $\mathcal{H}$-elimination decomposition $(\widehat{T}_1, \widehat{\chi}_1, \widehat{L}_1)$ of $G_1^\star$. □

Since $(s, 1) \in sig_1$, and sig_1 and sig_2 are similar, we have that either $(s, 1) \in sig_2$ or $(s, 0) \in sig_2$. Next we prove that indeed $(s, 1) \in sig_2$.

Claim 7.24. $(s, 1)$ *belongs to* sig_2.

Proof. We know that either $(s, 1) \in sig_2$ or $(s, 0) \in sig_2$. Suppose $(s, 1) \notin sig_2$. Then $(s, 0) \in sig_2$. Since sig_2 is a valid marked signature and $(s, 0) \in sig_2$, there is a pair $(s_2 = (D_2, T_2, \chi_2, L_2, \mathcal{P}_2, \mathsf{eq}_2), b) \in sig_2$, for some $b \in \{0, 1\}$, such that $D_2 \setminus L_2$ is a strict super set of $D \setminus L$ and s_2 is at least as good as s. This implies that since (G_1, H) exactly realizes s, (G_1, H) realizes s_2 and more labels of the boundary vertices are there in $D_2 \setminus L_2$ than in $D \setminus L$. As like in the proof of Claim 7.23, this contradicts our choice of the $\mathcal{H}$-elimination decomposition $(\widehat{T}_1, \widehat{\chi}_1, \widehat{L}_1)$ of $G_1^\star$. □

Because of Claim 7.24, $s \in \mathsf{real\text{-}sig}(G_2)$. From this we will prove that $G_2^\star$ is a yes-instance of Elimination Distance to $\mathcal{H}$. Since $s \in \mathsf{real\text{-}sig}(G_2)$, G_2 satisfies s. This implies that there is an $\mathcal{H}$-elimination decomposition (T', χ', L') of G_2 with the following properties.

(a) T is isomorphic to $T'[Z_2]$, where $Z_2 = I_{T'}(G_2, \chi')$. For any node $t \in Z_2$, we use the same notation to represent the node in T that maps to t by the isomorphism function. That is, $V(T) = Z_2$.

(b) For any node t of T with $\chi'(t) \cap B_2 \neq \emptyset$, $\chi(t) = \lambda_G(\chi'(t) \cap B_2)$

(c) For any node t of T with $\chi'(t) \neq \emptyset$ and $\chi'(t) \subseteq V(G_2) \setminus B_2$, $\chi(t) = \spadesuit$.

(d) Let $d_1, \ldots, d_{\ell'}$ be the leaf nodes of T' that belong to Z_2. Then, $\mathcal{P} = (\chi(d_i))_{i \in [\ell']}$.

(e) For each node t of T with $\chi(t) \in \mathcal{P}$, the equivalence class $Q \in \mathcal{Q}(\equiv_u)$ that contains $G_2[\chi'(t)]$ is at least as good as $\mathsf{eq}(\chi(t))$. That is, $\mathsf{ug}(Q, \mathsf{eq}(\chi(t)) = 1$.

Next we construct an $\mathcal{H}$-elimination decomposition $(\widehat{T}_2, \widehat{\chi}_2, \widehat{L}_2)$ of $G_2^\star$ from s, $(\widehat{T}_1, \widehat{\chi}_1, \widehat{L}_1)$ and (T', χ', L'), and prove that that indeed $(\widehat{T}_2, \widehat{\chi}_2, \widehat{L}_2)$ is an $\mathcal{H}$-elimination decomposition of $G_2^\star$, of depth at most k.

- First we will construct the rooted forest $\widehat{T}_2$. Recall that $Z_1 = I_{\widehat{T}_1}(G_1, \widehat{\chi}_1)$, $Z_2 = I_{T'}(G_2, \chi')$, and $V(T) = Z_1 = Z_2$. Moreover, $V(T) \subseteq V(\widehat{T}_1)$ and $V(T) \subseteq V(T')$. The vertex set of $\widehat{T}_2$ is $V(T') \uplus (V(\widehat{T}_1) \setminus V(T))$. The edge set of $\widehat{T}_2$ is the union of the edge sets of T' and $\widehat{T}_1$. All the roots of T' and $\widehat{T}_1$ are the roots of $\widehat{T}_2$. From the construction of $\widehat{T}_2$, it is easy to prove that $\widehat{T}_2$ is a forest. For any leaf node t of $\widehat{T}_2$, the unique path in $\widehat{T}_2$ from t to the root, is a leaf to root path either in T' or in $\widehat{T}_1$. Therefore, the depth of the forest $\widehat{T}_2$ is at most k because the depths of T' and $\widehat{T}_1$ are at most k.

- Next we define $\widehat{\chi}_2$.
  - For any internal node t of $\widehat{T}_2$, if $t \in V(T')$ and $\chi'(t) \neq \emptyset$, then we set $\widehat{\chi}_2(t) = \chi'(t)$.
  - For any internal node t of $\widehat{T}_2$, if $t \in V(\widehat{T}_1)$ and $\widehat{\chi}_1(t) \subseteq V(G_1^\star) \setminus V(G_1)$, then we set $\widehat{\chi}_2(t) = \widehat{\chi}_1(t)$. Notice that because of conditions (ii), (b), and (c), for such nodes t, we have that $\chi'(t) = \emptyset$.
  - Now let t be a leaf node of $\widehat{T}_2$. If t is a leaf node in T', then let $U' = \chi'(t)$. If t is a leaf node in $\widehat{T}_1$, then let $U_1 = \widehat{\chi}_1(t) \setminus V(G_1)$. Then, we set $\widehat{\chi}_2(t) = U' \cup U_1$.
  - For all other nodes t in $\widehat{T}_2$ (which are not considered above), we set $\widehat{\chi}_2(t) = \emptyset$.

- Next, we define $\widehat{L}_2 = (V(H) \cap \widehat{L}_1) \cup (V(G_2) \cap L')$.

Next, we prove that $(\widehat{T}_2, \widehat{\chi}_2, \widehat{L}_2)$ is indeed an $\mathcal{H}$-elimination decomposition of $G_2^\star$ of depth at most k. Since we have already proved that the depth of $\widehat{T}_2$ is at most k, the depth of the decomposition $(\widehat{T}_2, \widehat{\chi}_2, \widehat{L}_2)$ is at most k. Now we prove that $(\widehat{T}_2, \widehat{\chi}_2, \widehat{L}_2)$ satisfies all the conditions of Definition 2.1.

Let t be an internal node in $\widehat{T}_2$ and suppose that $\widehat{\chi}_2(t) \neq \emptyset$. Then, exactly one of the following is true.

- t is an internal node in T' and $\widehat{\chi}_2(t) = \chi'(t)$.
- t is an internal node in $\widehat{T}_1$ and $\widehat{\chi}_2(t) = \widehat{\chi}_1(t)$.

Thus, since $(\widehat{T}_1, \widehat{\chi}_1, \widehat{L}_1)$ and (T', χ', L') are $\mathcal{H}$-elimination decompositions of $G_1^\star$ and G_2, we have that $|\widehat{\chi}_2(t)| \leq 1$ and $\widehat{\chi}_2(t) \subseteq V(G) \setminus \widehat{L}_2$. Therefore, condition (1) of Definition 2.1 is satisfied.

From the construction of $\widehat{\chi}_2$, notice that for any $v \in V(G_2^\star)$, there is a node $t \in V(\widehat{T}_2)$ such that $v \in \widehat{\chi}_2(t)$. Moreover, since $(\chi'(t))_{t \in V(T')}$ is a partition of $V(G_2)$ and $(\widehat{\chi}_1(t) \cap (V(G_1^\star) \setminus V(G_1)))_{t \in V(\widehat{T}_1)}$ forms a partition of $V(G_1^\star) \setminus V(G_1)$, we have that $(\widehat{\chi}_2(t))_{t \in V(\widehat{T}_2)}$ forms a partition of $V(G_2^\star)$. Thus, condition (2) of Definition 2.1 is satisfied.

Now we prove that $(\widehat{T}_2, \widehat{\chi}_2, \widehat{L}_2)$ satisfies condition (3) of Definition 2.1. Let $uv \in E(G^\star)$ be an arbitrary edge. Let $u \in \widehat{\chi}_2(t)$ and $u \in \widehat{\chi}_2(t')$. Suppose $uv \in E(G_2)$. Then, since (T', χ', L') is an $\mathcal{H}$-elimination decomposition of G_2 and T' is a subgraph of $\widehat{T}_2$, we have that t and t' are in ancestor-descendant relation of $\widehat{T}_2$. Suppose $uv \in E(H)$. Then, t and t' are also belong to

$\widehat{T}_1$. Since $\widehat{T}_1$ is a subgraph of $\widehat{T}_2$ and $(\widehat{T}_1, \widehat{\chi}_1, \widehat{L}_)$ is an $\mathcal{H}$-elimination decomposition of $G_1^\star$, we have that t and t' are in ancestor-descendant relation of $\widehat{T}_2$.

Next we prove that $(\widehat{T}_2, \widehat{\chi}_2, \widehat{L}_2)$ satisfies condition (4) of Definition 2.1. That is, for each leaf node t of $\widehat{T}_2$, $G_2^\star[\widehat{\chi}_2(t)]$ belongs to $\mathcal{H}$. If $\widehat{\chi}_2(t) \subseteq V(G_2)$, then $G_2^\star[\widehat{\chi}_2(t)] = G_2[\widehat{\chi}_2(t)]$ is a base component in (T', χ', L') and hence it belongs to $\mathcal{H}$. If $\widehat{\chi}_2(t) \subseteq V(H)$, then $G_2^\star[\widehat{\chi}_2(t)] = H[\widehat{\chi}_2(t)]$ is an induced subgraph of a base component in $(\widehat{T}_1, \widehat{\chi}_1, \widehat{L})$ and hence it belongs to $\mathcal{H}$ because $\mathcal{H}$ is a hereditary family.

Now, we are in the case when $\widehat{\chi}_2(t) \cap (V(G_2) \setminus B_2) \neq \emptyset$ and $\widehat{\chi}_2(t) \cap (V(G_2^\star) \setminus V(G_2)) \neq \emptyset$. Let $H' = H[\widehat{\chi}_2(t) \cap V(H)]$ and $G_2' = G_2[\widehat{\chi}_2(t) \cap V(G_2)]$. Then $G_2' \oplus H'$ is the graph $G_2^\star[\widehat{\chi}_2(t)]$. If $B_2 \cap \widehat{\chi}_2(t) = \emptyset$, then by the construction of $\widehat{T}_2$, either $V(G_2') = \emptyset$ or $V(H') = \emptyset$ which is a contradiction to the assumption that $\widehat{\chi}_2(t) \cap (V(G_2) \setminus B_2) \neq \emptyset$ and $\widehat{\chi}_2(t) \cap (V(G_2^\star) \setminus V(G_2)) \neq \emptyset$. So we assume that $B_2 \cap \widehat{\chi}_2(t) \neq \emptyset$. This implies that t is a leaf node in T' as well as a leaf node in $\widehat{T}_1$. Let $G_1' = G_1[\widehat{\chi}_1(t) \cap V(G_1)]$. Notice that $G_1' \oplus H'$ is a base component in $(\widehat{T}_1, \widehat{\chi}_1, \widehat{L}_1)$, and hence $G_1' \oplus H'$ belongs to $\mathcal{H}$. Since (G_1, H) exactly realizes s through $(\widehat{T}_1, \widehat{\chi}_1, \widehat{L}_1)$, we have that G_1' belongs to the equivalence class $\mathsf{eq}_1(\widehat{\chi}_1(t))$. Since G_2 satisfies s, we have that G_2' belongs to an equivalence class which is at least as good as $\mathsf{eq}_1(\chi_1(t))$. This implies that $G_2' \oplus H'$ belongs to $\mathcal{H}$ because G_1' belongs to the equivalence class $\mathsf{eq}_1(\widehat{\chi}_1(t))$ and $G_1' \oplus H'$ belongs to $\mathcal{H}$.

This completes the proof of the lemma.

7.2 Proof of Lemma 7.20

The proof of Lemma 7.20 is identical to the proof of Lemma 4.5, where we use the algorithm $\mathcal{A}_u$ instead of $\mathcal{M}_{\mathsf{mod}}$. That is, we have a branching algorithm where the main computation boils down to executions of $\mathcal{A}_u$ for each state tuple s. Here, the algorithm is notationally cumbersome compared to the proof of Lemma 4.5. Throughout the section G is a (q,k)-unbreakable graph.

If $|V(G)| \leq 3q + k$, then we do a brute force computation. Otherwise, to prove the lemma we do the following. For each state-tuple $s = (D, T, \chi, L, \mathcal{P}, \mathsf{eq})$, either we identify that $s \in \mathsf{real\text{-}sig}(G)$ or we identify a state-tuple $\tilde{s} = (\tilde{D}, \tilde{T}, \tilde{\chi}, \tilde{L}, \tilde{\mathcal{P}}, \tilde{\mathsf{eq}}) \in \mathsf{real\text{-}sig}(G)$ such that $\tilde{s}$ is strictly better than s. or we may "fail". If we identify that $s \in \mathsf{real\text{-}sig}(G)$, then we include $(s, 1)$ in the output signature. If we identify that $\tilde{s} \in \mathsf{real\text{-}sig}(G)$, then we include $(s, 0)$ and $(\tilde{s}, 1)$ in the output signature. If we fail to identify s or $\tilde{s}$, then we prove that indeed $s \notin \mathsf{real\text{-}sig}(G)$. This is formalized in the following lemma.

Lemma 7.25. *Suppose $|V(G)| > 3q + k$. There is a function g_1 and an algorithm* Sig-Test *that given a graph G and a state-tuple s, runs in time $g_1(q,k) \cdot f_u(k, |\mathcal{Q}(\equiv_u)|, n) \cdot n^{\mathcal{O}(1)}$, and outputs exactly one of the following: (i) $(s, 1)$, (ii) $(s, 0)$ and $(\tilde{s} = (D, \tilde{T}, \tilde{\chi}, \tilde{L}, \tilde{\mathcal{P}}, \tilde{\mathsf{eq}}), 1)$, or (iii)* Fail. *The algorithm has the following properties.*

- *(a) If the output is $(s, 1)$, then $s \in \mathsf{real\text{-}sig}(G)$.*
- *(b) If the output is $(s, 0)$ and $(\tilde{s}, 1)$, then $\tilde{s} \in \mathsf{real\text{-}sig}(G)$, and $\tilde{s}$ is strictly better than s.*
- *(c) If $s \in \mathsf{real\text{-}sig}(G)$, then the output is either (i) or (ii).*

First assuming Lemma 7.25, we prove Lemma 7.20.

Proof of Lemma 7.20. Initially we set $sig = \emptyset$. For each state-tuple s, we run Sig-Test. If the output is $(s, 1)$, then we add $(s, 1)$ to sig. If the output is $(s, 0)$ and $(\tilde{s}, 1)$, then we add $(s, 0)$ and $(\tilde{s}, 1)$ to sig. Finally we output sig. The correctness of the algorithm follows from the correctness of Sig-Test. Since the number of state-tuples is bounded by k and $|\mathcal{Q}(\equiv_u)|$, by Lemma 7.25, the running time of the algorithm follows. □

7.2.1 Proof of Lemma 7.25

First we give an overview of our algorithm Sig-Test. Let $s \in \mathsf{real\text{-}sig}(G)$. To identify that s is indeed belongs to $\mathsf{real\text{-}sig}(G)$ we have to test the existence of an $\mathcal{H}$-elimination decomposition $(\widehat{T}, \widehat{\chi}, \widehat{L})$ of G, of depth at most k, such that G satisfies s through $(\widehat{T}, \widehat{\chi}, \widehat{L})$. Since G is (q,k)-unbreakable and $|V(G)| > 3q + k$, by Lemma 3.1 we know that there is exactly one connected component $C^\star$ in $G[\widehat{L}]$ that has at least q vertices and $|V(G) \setminus V(C^\star)| \leq q + k$. This implies that $|V(\widehat{T})| \leq q + k$. Thus, we can guess the rooted forest $\widehat{T}$ and the leaf node $t^\star$ in $\widehat{T}$ such that $V(C^\star) \subseteq \widehat{\chi}(t^\star)$. Let $EQ^\star$ be the equivalence class mentioned in the state-tuple corresponding to the leaf node $t^\star$ (assuming $C^\star$ contains some boundary vertices). Then $\widehat{\chi}(t^\star)$ belongs to an equivalence class which is at least as good as $EQ^\star$. Using branching rules we *almost* identify the values $\widehat{\chi}(t)$ for all the nodes t except the nodes in the unique path $P^\star$ from $t^\star$ to the root. Finally we will have a connected component $D^\star$ such that $N(D^\star) \subseteq \bigcup_{t \in V(P^\star)} \widehat{\chi}(t)$. In this step we run the user's algorithm $\mathcal{A}_u$ to identify the vertices S of $D^\star$ to be placed in $\widehat{\chi}(t)$ for the internal nodes of T in the path from $t^\star$ to the root such that $\chi(t) = \spadesuit$ and $D^\star - S$ belongs to an equivalence class which is at least as good EQ. If S does not contain any vertex from $\delta(G)$, then we identify a witness (i.e., the $\mathcal{H}$-elimination decomposition $(\widehat{T}, \widehat{\chi}, \widehat{L})$) for the fact that $s \in \mathsf{real\text{-}sig}(G)$. In that case the algorithm Sig-Test outputs $(s, 1)$. If $S \cap \delta(G) \neq \emptyset$, then we get a witness for another state-tuple $\tilde{s}$ and we prove that $\tilde{s}$ is at least as good as s. In that case the algorithm Sig-Test outputs $(s, 0)$ and $(\tilde{s}, 1)$. If $\mathcal{A}_u$ outputs No, then Sig-Test outputs Fail.

Now we formally prove Lemma 7.25. We start by defining the notion of a partial solution.

Definition 7.26 (Partial solution)**.** *Given a graph G and a state-tuple $s = (D, T, \chi, L, \mathcal{P}, \mathsf{eq})$, a partial solution of (G, s) is a tuple $e = (D, T^\star, \chi^\star, L^\star)$ where $T^\star$ is a rooted forest of depth at most k and $\chi^\star \colon V(T^\star) \to 2^{V(G)}$ and $L^\star \subseteq V(G)$, such that the following properties hold.*

(a) $\Lambda(G) = D$ *and* $\lambda_G(L^\star \cap \delta(G)) = L$.

(b) For each internal node t of $T^\star$ we have $|\chi^\star(t)| \leq 1$ and $\chi^\star(t) \subseteq V(G) \setminus L$.

(c) The sets $(\chi^\star(t))_{t \in V(T^\star)}$ form a partition of a subset of $V(G)$.

(d) T is isomorphic to $T^\star[Z]$, where $Z = I_{T^\star}(G, \chi^\star)$. For any node $t \in Z$, we use the same notation to represent the node in T that maps to t by the isomorphism function. That is, $V(T) = Z$.

(e) For any node t of T with $\chi^\star(t) \cap \delta(G) \neq \emptyset$, $\chi(t) = \lambda_G(\chi^\star(t) \cap \delta(G))$

(f) For each node t of T with $\chi^\star(t) \neq \emptyset$ and $\chi^\star(t) \subseteq V(G) \setminus \delta(G)$, $\chi(t) = \spadesuit$.

(g) Let $t_1, \ldots, t_\ell$ be the leaf nodes of $T^\star$ that belong to Z. Then, $\mathcal{P} = (\chi(t_i))_{i \in [\ell]}$.

(h) For any two nodes t and t' in $T^\star$ such that they are not in an ancestor-descendant relationship, there is no edge in G between a vertex in $\chi^\star(t)$ and $\chi^\star(t')$.

For a partial solution $e = (D, T^\star, \chi^\star, L^\star)$ of (G, s), we use $\mathsf{Used}_{\chi^\star}(G)$ to denote the set $\bigcup_{t \in V(T^\star)} \chi^\star(t)$. That is, in the partial solution e we have identified the places of $\mathsf{Used}_{\chi^\star}(G)$. The objective is to keep including vertices from $V(G) \setminus \mathsf{Used}_{\chi^\star}(G)$ to the sets $\{\chi^\star(t)\}_{t \in V(T^\star)}$ through branching rules.

Recall that $(G, s = (D, T, \chi, L, \mathcal{P}, \mathsf{eq}))$ is the input of the algorithm Sig-Test. To understand the steps of the algorithm let us assume that $s \in \mathsf{real\text{-}sig}(G)$ and $(\widehat{T}, \widehat{\chi}, \widehat{L})$ be an $\mathcal{H}$-elimination decomposition of G, of depth at most k, such that G satisfies s through $(\widehat{T}, \widehat{\chi}, \widehat{L})$. As mentioned earlier, since G is (q,k)-unbreakable and $|V(G)| > 3q + k$, by Lemma 3.1, there is exactly one connected component $C^\star$ in $G[\widehat{L}]$ that has at least q vertices and $|V(G) \setminus V(C^\star)| \leq q + k$ and

hence $|V(\widehat{T})| \leq q+k$. Therefore, as a first step we guess the tree $\widehat{T}$, all the nodes in t such that $\widehat{\chi}(t)$ contains at least one vertex from $\delta(G)$, $\widehat{\chi}(t) \cap \delta(G)$, and the node $t^\star$ such that $\widehat{\chi}(t^\star) = C^\star$. Since $|\delta(G)| \leq 2k$ and $|V(\widehat{T})| \leq q+k$ the number of choices for this guess is bounded by $(q+k)^{\mathcal{O}(q+k)}$. Thus, we assume that initially we have a partial solution $e = (D, T^\star, \chi^\star, L^\star)$, where $T^\star = \widehat{T}$ with the following properties. (Recall that T is an induced subgraph of $T^\star$).

- For any node $t \in V(T)$, $\chi^\star(t) \subseteq \delta(G)$ and $\chi(t) = \lambda_G(\chi^\star(t) \cap \delta(G))$.
- $\delta(G) = \bigcup_{t \in V(T^\star)} \chi^\star(t)$.

We remind that we also know the node $t^\star \in V(T^\star) = V(T)$ such that the vertices of the "unknown" large component $C^\star$ belong to $\widehat{\chi}(t^\star)$. Now on, we assume that our instances contain $(G, s, e, t^\star)$ where e is a partial solution and $t^\star$ is the special leaf node in the rooted forest $T^\star$ of e. The objective is to construct an $\mathcal{H}$-elimination decomposition $(\widehat{T}, \widehat{\chi}, \widehat{L})$ of G (we call it as solution) *obeying* the partial solution e such that either G satisfies s through $(\widehat{T}, \widehat{\chi}, \widehat{L})$ or G satisfies $\tilde{s}$ (derived from the solution) through $(\widehat{T}, \widehat{\chi}, \widehat{L})$ with the property that $\tilde{s}$ is strictly better than s. We formally define when a solution obeys a partial solution.

Definition 7.27. *Let $s = (D, T, \chi, L, \mathcal{P}, \mathsf{eq})$ be a state-tuple and $e = (D, T^\star, \chi^\star, L^\star)$ be a partial solution. We say that a solution $(\widehat{T}, \widehat{\chi}, \widehat{L})$ obeys the partial solution e if $\widehat{T}$ is isomorphic to $T^\star$ (we use the same vertex to denote its image in the isomorphism function, i.e., $V(\widehat{T}) = V(T^\star)$) and for each $t \in V(\widehat{T}) = V(T^\star)$, $\chi^\star(t) \subseteq \widehat{\chi}(t)$.*

Next we explain about our branching rules. For each instances created in the branching rule, at least for one internal node $t \in V(T^\star)$ with $\chi^\star(t) = \emptyset$, we will have a assigned a vertex from $V(G)$ to it. That is, in the new instance $(G, s, e' = (D, T^\star, \chi', L'), t^\star)$ created there is at least one new internal node $t \in V(T^\star)$ with $\chi^\star(t) = \emptyset$ and $\chi'(t) \neq \emptyset$. Since $|V(T^\star)| \leq q+k$ and for each internal node t, $|\chi'(t)| \leq 1$, we use the number of internal nodes t with $\chi^\star(t) = \emptyset$ as a measure for our branching algorithm. In our branching rules, the measure decreases by at least one after the application of each branching rule. Next we explain our branching rules. A branching step creates more instances. If for any of the instance the output is $(s, 1)$, then the output of our algorithm is $(s, 1)$. If this is not the case and at least one of them outputs $(s, 0)$ and $(\tilde{s}, 1)$, then the output of our algorithm is $(s, 0)$ and $(\tilde{s}, 1)$. If for all the instances, the output is Fail, then we output Fail.

Our first branching rule attempt to fix "neighborhood" inconsistencies, below we begin by explaining what we mean by an inconsistency that our first branching rules attempts to fix. Suppose there are (not necessarily distinct) vertices u, u' in the same connected component of $G' = G - \mathsf{Used}_{\chi^\star}(T^\star)$ and two nodes t and t' in $T^\star$ such that t and t' are not in ancestor-descendant relationship, and u and u' have neighbors in $\chi^\star(t)$ and $\chi^\star(t')$, respectively. Then consider any $u - u'$ path P in G' (which exists, since u and u' are in the same connected component of G'). We remark that if $u = u'$, then P is a path on one vertex. For the above, we say that (u, u', P) is an *inconsistent triplet*.

We will observe that for any solution $(\widehat{T}, \widehat{\chi}, \widehat{L})$ obeying the partial solution e, $V(P) \cap (V(G) \setminus L) \neq \emptyset$. In fact, we can establish a stronger statement than the above, which guarantees that at least one among the first and the last q many vertices in P, belongs to $V(G) \setminus L$. The above property is obtained because at least one of u or u' cannot belongs to the unique connected component of size at least q in $G[L]$, for any solution $(\widehat{T}, \widehat{\chi}, \widehat{L})$ (see Lemma 3.1). The above leads us to the following branching rule.

Branching Rule 1: Let (u, u', P) is an inconsistent triplet in $(G, s, e = (D, T^\star, \chi^\star, L^\star), t^\star)$. Then for each vertex v in the first q vertices or in the last q vertices in the path P and for each internal node $t \in V(T^\star)$ such that $\chi^\star(t) = \emptyset$ and $\chi(t) = \spadesuit$, construct a new instance

$(G, s, e' = (D, T^\star, \chi', L^\star), t^\star)$ if e' is a partial solution for (G, s) and solve it. Here $\chi'(t) = \{v\}$ and $\chi'(t_1) = \chi^\star(t_1)$ for all $t_1 \neq t$.

The correctness of the branching rule follows from Lemma 3.1. Notice that if Branching Rule 1 is not applicable, then there is no "crossing edge" inconsistencies. That is, let $(G, s, e = (D, T^\star, \chi^\star, L^\star), t^\star)$ be an instance such that Branching Rule 1 is not applicable. Then for any connected component C of $G - \mathsf{Used}_{\chi^\star}(G)$, there is a leaf node t in $T^\star$ such that for any vertex $w \in N_G(V(C))$, there is an ancestor t' of t (t' maybe equal to t) with $w \in \chi^\star(t')$.

Next, we do a branching rule similar to the Branching Rule 1 in the proof of Lemma 4.2. Recall that for a graph G and integers $p, q' \in \mathbb{N}$, a set $B \subseteq V(G)$ is a (p, q')-*connected set* in G, if $\widehat{G}[B]$ is connected, $|B| \leq p$ and $|N_G(B)| \leq q'$. A (p, q')-connected set B in G is *maximal* if there does not exist another (p, q')-connected set B^* in G, such that $B \subset B^*$. Using Proposition 4.7 we can enumerate all maximal (p, q')-connected sets in time $2^{p+q'} \cdot n^{\mathcal{O}(1)}$.

Since the the family $\mathcal{H}$ is closed under disjoint union, if there is a graph $H \notin \mathcal{H}$, then there is a connected component H' of H such that $H' \notin \mathcal{H}$. Let $(G, s, e = (D, T^\star, \chi^\star, L^\star), t^\star)$ be an instance and let $(\widehat{T}, \widehat{\chi}, \widehat{L})$ be a hypothetical solution obeying the partial solution e. Let C be a maximal $(q+k, k)$-connected set in G such that $G[C'] \notin \mathcal{H}$ where $C' = V(C) \setminus \mathsf{Used}_{\chi^*}(G)]$. Then by the hereditary property of $\mathcal{H}$, there exists a connected component F in $G[C']$ such that $F \notin \mathcal{H}$. Then observe that for the solution $(\widehat{T}, \widehat{\chi}, \widehat{L})$ obeying the partial solution e, $V(F) \cap (V(G) \setminus \widehat{L}) \neq \emptyset$. This leads to the following branching rule. Let $\mathcal{C}$ be the set of all maximal $(q + k, k)$-connected set in G.

Branching Rule 2: Let $(G, s, e = (D, T^\star, \chi^\star, L^\star), t^\star)$ be an instance. Let $C \in \mathcal{C}$ such that $G[C'] \notin \mathcal{H}$ where $C' = Q \setminus \mathsf{Used}_{\chi^*}(G)$. Let F be a connected component in $G[C']$ such that $F \notin \mathcal{H}$. Then for each $v \in V(F)$ and for each internal node $t \in V(T^\star)$ such that $\chi^\star(t) = \emptyset$ and $\chi(t) = \spadesuit$, construct a new instance $(G, s, e' = (D, T^\star, \chi', L^\star), t^\star)$ if e' is a partial solution for (G, s) and solve. Here $\chi'(t) = \{v\}$ and $\chi'(t_1) = \chi^\star(t_1)$ for all $t_1 \neq t$.

Towards the correctness of Branching Rule 2, notice that $V(F)$ cannot be a subset of $\widehat{\chi}(t')$ for any leaf node t' because $F \notin \mathcal{H}$. Now suppose $V(F) \subseteq \bigcup_{t' \in R} \widehat{\chi}(t')$ where R is the set of leaf nodes of $\widehat{T}$. Then, since F is connected there exists a two distinct leaf nodes t_1 and t_2 such that there is an edge in G between a vertex in $\widehat{\chi}(t_1)$ and a vertex in $\widehat{\chi}(t_2)$. This contradicts the fact that $(\widehat{T}, \widehat{\chi}, \widehat{L})$ is an $\mathcal{H}$-elimination decomposition.

Now, we prove the following lemma.

Lemma 7.28. *Suppose Branching Rule 1 is not applicable for an instance* $(G, s, e = (D, T^\star, \chi^\star, L^\star), t^\star)$. *Let* $Y = \bigcup_{t' \in R} \chi^\star(t')$ *where* R *is the set of leaf nodes of* $T^\star$. *Then there is a unique connected component* J^* *in* $G - (\mathsf{Used}_{\chi^\star}(G) \setminus Y)$ *that has at least* q *vertices, and* $|V(G) \setminus V(J^*)| \leq q + k$.

Proof. Since Branching Rule 1 is not applicable for any connected component C in $G - (\mathsf{Used}_{\chi^\star}(G) \setminus Y)$, there is a leaf node t_C in $T^\star$ such that for any vertex $w \in N_G(C)$, there is a node t' in the unique path in $T^\star$ from t_C to the root such that $w \in \chi^\star(t_C)$.

Let $\mathcal{G}$ be the set of all graphs. Now we construct a $\mathcal{G}$-elimination decomposition $(T^\star, \chi_1, L_1)$ from e as follows. Initially set $\chi_1(t) = \chi^\star(t)$ for all $t \in V(T^\star)$. For each connected component C in $G - (\mathsf{Used}_{\chi^\star}(G) \setminus Y)$, add $V(C)$ to $\chi_1(t_C)$. Then $(T^\star, \chi_1, L_1)$ is a $\mathcal{G}$-elimination decomposition. Now the lemma follows by applying Lemma 3.1 on $(T^\star, \chi_1, L_1)$. □

Now let $(G, s, e = (D, T^\star, \chi^\star, L^\star), t^\star)$ be an instance such that Branching Rules 1 and 2 are not applicable. Let $(\widehat{T}, \widehat{\chi}, \widehat{L})$ be a hypothetical solution obeying the partial solution e. Let $Y = \bigcup_{t' \in R} \chi^\star(t')$ where R is the set of leaf nodes of $T^\star$. Then by Lemma 7.28, there is a unique connected component J^* in $G - (\mathsf{Used}_{\chi^\star}(G) \setminus Y)$ that has at least q vertices, and $|V(G) \setminus V(J^*)| \leq q + k$. Now since $|V(G) \setminus V(J^*)| \leq q + k$, for each vertex $v \in V(G) \setminus V(J^*)$, we can guess the node $t_v \in V(T^\star) = V(\widehat{T})$ such that $v \in \widehat{\chi}(t_v)$. Thus, according to our guess we update the function $\chi^\star$. That is, we update $\chi^\star(t_v) := \chi^\star(t_v) \cup \{v\}$. Also notice that if there is a

vertex $u \in V(J^\star)$, an edge $uw \in E(G)$ and a node $t \in V(T^\star)$ such that $w \in \chi^\star(t)$, and $t^\star, t$ are not in an ancestor-descendant relation, then clearly there is an internal node t_u in $T^\star$ such that $\widehat{\chi}(t_u) = \{u\}$. Thus, we also guess the node t_u for such vertices. This implies that after these steps for any vertex $x \in V(G) \setminus \mathsf{Used}_{\chi^\star}(G)$, and any edge $xy \in E(G)$ either $y \in V(G) \setminus \mathsf{Used}_{\chi^\star}(G)$ or there is a node t in the unique path $P^\star$ in $T^\star$ from $t^\star$ to the root such that $y \in \chi^\star(t)$.

Next, we prove that at this stage by running one execution of the user defined function $\mathcal{A}_u$ we get the output (i) $(s, 1)$ or (i) $(s, 0)$ and $(\tilde{s}, 1)$.

Lemma 7.29. *Let $I = (G, s, e, t^\star)$ be an instance such that Branching Rules 1 and 2 are not applicable, where $s = (D, T, \chi, L, \mathcal{P}, \mathsf{eq})$ and $e = (D, T^\star, \chi^\star, L^\star)$. Let $(\widehat{T}, \widehat{\chi}, \widehat{L})$ be a hypothetical solution obeying the partial solution e. Let $P^\star$ be the unique path in $T^\star$ from $t^\star$ to the root. Let k' be the number of nodes t in $V(P^\star)$ such that $\chi(t) = \spadesuit$ and $\chi^\star(t) = \emptyset$. Notice that for any leaf node t in $T^\star$ such that $\chi^\star(t) \cap \delta(G) \neq \emptyset$, we have $\lambda_G(\chi^\star(t) \cap \delta(G)) = \lambda_G(\widehat{\chi}(t) \cap \delta(G)) = \chi(t) \in \mathcal{P}$. Let $EQ^\star$ be the equivalence class in $\equiv_u$ such that $\mathsf{eq}(\chi(t^\star)) = EQ^\star$. Let $U = V(G) \setminus \mathsf{Used}_{\chi^\star}(G)$ and $G^\star = G[U \cup \chi^\star(t^\star)]$. Suppose we have the following two conditions.*

(a) $\delta(G) \subseteq \bigcup_{t \in V(T^\star)} \chi^\star(t)$.

(b) For any vertex $x \in U$ and any edge $xy \in E(G)$, either $y \in U$ or there is a node $t \in V(P^\star)$ such that $y \in \chi^\star(t)$.

Then, the below conditions are true.

(i) There is vertex subset S of size at most k' such that $G^\star - S$ belongs to an equivalence class which is at least as good as $EQ^\star$.

(ii) Given a vertex subset S of size at most k' such that $G^\star - S$ belongs to an equivalence class which is at least as good as $EQ^\star$, then we can construct an $\mathcal{H}$-elimination decomposition $(T^\star, \psi, Z^\star)$ of G of depth at most k, in polynomial time such that the following holds.

- *If $S \cap \delta(G) = \emptyset$, then G satisfies s through $(T^\star, \psi, Z^\star)$.*
- *If $S \cap \delta(G) \neq \emptyset$, then we can construct a state-tuple $\tilde{s} = (D, T, \chi', L', \mathcal{P}', \mathsf{eq}')$ in polynomial time such that G satisfies $\tilde{s}$ through $(T^\star, \psi, Z^\star)$, and $\tilde{s}$ is strictly better than s.*

Proof. First we prove property (i) of the lemma. For the sake of contradiction, suppose there is no vertex subset S of size at most k' in $G^\star$ such that $G^\star - S$ belongs to a class which is at least as good as $EQ^\star$. Recall that $(\widehat{T}, \widehat{\chi}, \widehat{L})$ is a solution obeying the partial solution e. Let $S^\star = V(G^\star) \setminus \widehat{L}$. Then $G^\star - S^\star$ belongs to a class which is at least as good as $EQ^\star$. This implies that $|S^\star| > k'$. Let $X = \bigcup_{t \in V(P^\star) \setminus \{t^\star\}} \widehat{\chi}(t) \cap S^\star$. Notice that $|X| \leq k'$. We prove that $G^\star - X$ belongs to a class which is at least as good as $EQ^\star$ and that will be a contradiction to our assumption. Let $F = S^\star \setminus X$. Since $\bigcup_{t \in V(P^\star)} \widehat{\chi}(t) \cap F = \emptyset$, F has neighbors only in $\bigcup_{t \in V(P^\star) \setminus \{t^\star\}} \widehat{\chi}(t)$ (because $F \subseteq V(G^\star)$ and $(\widehat{T}, \widehat{\chi}, \widehat{L})$ is a solution), and Branching Rule 2 is not applicable, we have that $G[F] \in \mathcal{H}$. Moreover, $F \cap \delta(G) = \emptyset$. Thus, by the component deletion property (see Definition 7.3), we have that $G^\star - X = (G^\star - S^\star) \uplus G[F]$ belongs to an equivalence which is at least as good as $EQ^\star$. This is a contradiction to our assumption that there is no vertex subset S of size at most k' in $G^\star$ such that $G^\star - S$ belongs to a class which is at least as good as $EQ^\star$. This completes the proof of property (i).

Now we prove property (ii). Suppose there is a vertex subset $S = \{v_1, \ldots, v_\ell\}$ such that $\ell \leq k'$ and $G^\star - S$ belongs to an equivalence class which is at least as good as $EQ^\star$. Now we construct an $\mathcal{H}$-elimination decomposition $(T^\star, \psi, Z^\star)$ as follows. Let $W = \{t_1, \ldots, t_\ell\}$ be a subset of nodes in $P^\star$ such that for all $i \in [\ell]$ $\chi(t_i) = \spadesuit$ and $\chi^\star(t_i) = \emptyset$. Now for each $i \in [\ell]$, we set $\psi(t_i) = \{v_i\}$. For each $t \in V(T^\star) \setminus (W \cup \{t^\star\})$, we set $\psi(t) = \chi^\star(t)$. Finally, we set $\psi(t^\star) = V(G^\star) \setminus S$. Let R be the set of leaf nodes in $T^\star$. We define $Z^\star = \bigcup_{t \in R} \psi(t)$.

Claim 7.30. *The graph $G[\psi(t^\star)]$ is in an equivalence class which is at least as good as $EQ^\star$. For each $t \in R \setminus \{t^\star\}$, $\psi(t) \subseteq \widehat{\chi}(t)$ and if $\psi(t) \cap \delta(G) \neq \emptyset$, then $G[\psi(t)]$ is in an equivalence class which is at least as good as $\mathsf{eq}(\chi(t))$.*

Proof. Since $\psi(t^\star) = V(G^\star) \setminus S$, by our assumption (i.e., premise of the condition (ii)), we have that $G[\psi(t^\star)]$ is in an equivalence class which is at least as good as $EQ^\star$.

From the construction of ψ we have that for each $t \in R \setminus \{t^\star\}$, $\psi(t) = \chi^\star(t) \subseteq \widehat{\chi}(t)$. Now fix a node $t \in R \setminus \{t^\star\}$ such that $\psi(t) \cap \delta(G) \neq \emptyset$. Then, we know that $G[\psi(t)]$ is an induced subgraph of $G[\widehat{\chi}(t)]$. In fact we prove that $G[\psi(t)]$ is a union of some connected components of $G[\widehat{\chi}(t)]$. Suppose not. Then there is an edge xy in the graph $G[\widehat{\chi}(t)]$ such that $x \in \psi(t)$ and $y \notin \psi(t)$. Since $y \in \widehat{\chi}(t)$, we have that $y \notin \chi^\star(t')$ for any $t' \neq t$ because $(\widehat{T}, \widehat{\chi}, \widehat{L})$ obeys the partial solution e. This implies that $y \in U$. This contradicts condition (b) in the lemma. So $G[\psi(t)]$ is obtained by deleting some connected components from $G[\widehat{\chi}(t)]$. Moreover, we know that $\psi(t) \cap \delta(G) = \widehat{\chi}(t) \cap \delta(G)$. Thus, by the component deletion property (see Definition 7.3), we have that $G[\psi(t)]$ and $G[\widehat{\chi}(t)]$ are in same equivalence class which is at least as good as $\mathsf{eq}(\chi(t))$. This completes the proof of the claim. □

Claim 7.31. *$(T^\star, \psi, Z^\star)$ is an $\mathcal{H}$-elimination decomposition of G of depth at most k.*

Proof. Since e is a partial solution, we have that the depth of $T^\star$ is at most k. Now we prove that $(T^\star, \psi, Z^\star)$ satisfies the conditions of Definition 2.1. From the construction of ψ, we have that $|\psi(t)| \leq 1$ and $\psi(t) \subseteq V(G) \setminus Z^\star$ for any $t \in W$. For any internal node t of $T^\star$ such that $t \notin W$, $\psi(t) = \widehat{\chi}(t) \subseteq V(G) \setminus Z^\star$ and $|\psi(t)| = |\widehat{\chi}(t)| \leq 1$ because $(\widehat{T}, \widehat{\chi}, \widehat{L})$ is $\mathcal{H}$-elimination decomposition. This implies that condition (1) of Definition 2.1 holds. From the construction of ψ, condition (2) of Definition 2.1 holds.

Now we prove condition (3) of Definition 2.1. Let $uv \in E(G)$ and $u \in \psi(t)$ and $v \in \psi(t')$ for some $t, t' \in V(T^\star)$. Suppose $V(P^\star) \cap \{t, t'\} = \emptyset$. Then, $\psi(t) = \chi^\star(t) \subseteq \widehat{\chi}(t)$ and $\psi(t') = \chi^\star(t') \subseteq \widehat{\chi}(t')$. Then, since $(\widehat{T}, \widehat{\chi}, \widehat{L})$ is an $\mathcal{H}$-elimination decomposition, we have that t and t' are in ancestor-descendant relationship in $T^\star$. Suppose $t, t' \in V(P^\star)$. Then t and t' are in ancestor-descendant relationship in $T^\star$. Now consider the case when $|V(P^\star) \cap \{t, t'\}| = 1$. Without loss of generality let $t \in V(P^\star)$ and $t' \notin V(P^\star)$. Thus, from the construction of ψ, we have that $v \in \psi(t') = \chi^\star(t') \subseteq \widehat{\chi}(t')$. If $u \in \chi^\star(t)$, then t and t' are in ancestor-descendant relation in $T^\star$. Otherwise we contradict the fact that e is a partial solution. If $u \notin \chi^\star(t)$, then $u \in U = V(G^\star) \setminus \chi^\star(t^\star)$. Then, it will contradict condition (b) of the lemma. Thus, we have proved condition (3) of Definition 2.1.

Next we prove condition (4) of Definition 2.1. Recall that for any leaf node $t \in R \setminus \{t^\star\}$, $\psi(t) = \chi^\star(t) \subseteq \widehat{\chi}(t)$. Also, since $(\widehat{T}, \widehat{\chi}, \widehat{L})$ is an $\mathcal{H}$-elimination decomposition, $G[\widehat{\chi}(t)] \in \mathcal{H}$. Thus, since $G[\psi(t)]$ is an induced subgraph of $G[\widehat{\chi}(t)]$, we get that $G[\psi(t)] \in \mathcal{H}$ because $\mathcal{H}$ is an hereditary family. Recall that $\psi(t^\star) = V(G^\star) \setminus S$. Since $G^\star - S$ is in an equivalence class which is at least as good as $EQ^\star$ (see Claim 7.30), we have that $G[\psi(t^\star)] \in \mathcal{H}$. This completes the proof of the claim. □

Claim 7.32. *If $S \cap \delta(G) = \emptyset$, then G satisfies s through $(T^\star, \psi, Z^\star)$.*

Proof. Since the hypothetical solution $(\widehat{T}, \widehat{\chi}, \widehat{L})$ obeys the partial solution $e = (D, T^\star, \chi^\star, L^\star)$, G satisfies s through $(\widehat{T}, \widehat{\chi}, \widehat{L})$, and for each $t \in V(\widehat{T}) = V(T^\star)$, $\chi^\star(t) \subseteq \widehat{\chi}(t)$. We need to prove that G satisfies s through $(T^\star, \psi, Z^\star)$. Since $T^\star$ is isomorphic to $\widehat{T}$, condition (a) of Definition 7.5 is true. Since for any node $t \in V(T^\star)$, $\psi(t) \cap \delta(G) = \widehat{\chi}(t) \cap \delta(G)$, condition (b) of Definition 7.5 is true. Notice that for any $v_i \in S$, we set $\psi(t_i) = \{v_i\}$, where $\chi(t_i) = \spadesuit$. This implies that condition (c) of Definition 7.5 is true. Notice that for any vertex $v \in \delta(G)$, there is a node t such that $v \in \psi(t)$ and $v \in \widehat{\chi}(t)$. This implies that condition (d) of Definition 7.5 is true. Condition (e) of Definition 7.5 follows from Claim 7.30. □

Claim 7.33. *If $S \cap \delta(G) \neq \emptyset$, then we can construct a state-tuple $\tilde{s} = (D, T, \chi', L', \mathcal{P}', \mathsf{eq}')$ in polynomial time such that G satisfies $\tilde{s}$ through $(T^\star, \psi, Z^\star)$, $\tilde{s}$ is strictly better than s.*

Proof. First we define the function χ'. Recall that $S = \{v_1, \ldots, v_\ell\}$, and $W = \{t_1, \ldots, t_\ell\}$ be a subset of nodes in $P^\star$ such that for all $i \in [\ell]$ $\chi(t_i) = \spadesuit$ and $\chi^\star(t_i) = \emptyset$. For any $t \in V(T^\star) \setminus \{t_1, \ldots, t_\ell, t^\star\}$, we set $\chi'(t) = \chi(t)$. For any $i \in [\ell]$, if $v_i \in \delta(G)$, then $\chi'(t_i) = \{\lambda_G(v_i)\}$. Otherwise $\chi'(t_i) = \spadesuit$. Finally, $\chi'(t^\star) = \lambda_G(\psi(t^\star) \cap \delta(G))$. The construction of χ' and the fact that $S \cap \delta(G) \neq \emptyset$ implies that $D \setminus L \subset D \setminus L'$. Let $R' \subseteq R$ be the set of leaf nodes in $T^\star$ such that for any $t \in R'$, $\psi(t) \cap \delta(G) \neq \emptyset$. Then $\mathcal{P}' = (\chi'(t))_{t \in R'}$. Notice that for each $t \in R' \setminus \{t^\star\}$, $\chi(t) = \chi'(t)$ and $\chi'(t^\star) \subset \chi(t)$. Now we define eq'. For any $t \in R'$, we define $\mathsf{eq}'(\chi'(t)) = \mathsf{eq}(\chi(t))$. Finally $L' = L \setminus \lambda_G(S \cap \delta(G))$. This completes the construction of $\tilde{s} = (D, T, \chi', L', \mathcal{P}', \mathsf{eq}')$. Using arguments similar to the one in Claim 7.32, one can prove that G satisfies $\tilde{s}$ through $(T^\star, \psi, Z^\star)$, and hence it is omitted here.

Next we prove that $\tilde{s}$ is strictly better than s. We have already proved that $D \setminus L \subset D \setminus L'$. Let H be a graph such that $G \oplus H$ exactly realizes s through $(\widehat{T}, \eta, L_1)$ and $(\widehat{T}, \widehat{\chi}, \widehat{L})$ is equal to $\mathsf{Restricted}_G(\widehat{T}, \eta, L_1)$. Thus we have the following.

(i) G exactly satisfies s through $(\widehat{T}, \widehat{\chi}, \widehat{L})$.

(ii) For each node t of T with $\eta(t) \neq \emptyset$ and $\eta(t) \subseteq V(H) \setminus \delta(G)$, $\chi(t) = \heartsuit$.

We will prove that $G \oplus H$ realizes $\tilde{s}$. Towards that we will construct an $\mathcal{H}$-elimination decomposition $(\widehat{T}, \phi, L_2)$ of $G \oplus H$, of depth at most k, such that $G \oplus H$ realizes $\tilde{s}$ through $(\widehat{T}, \phi, L_2)$. Statement (i) implies that for any $t \in R'$, $G[\widehat{\chi}(t)]$ belongs to $\mathsf{eq}(\chi(t))$. Recall that $T^\star$ is isomorphic to $\widehat{T}$ and $V(T^\star) = V(\widehat{T})$. From the construction of $\tilde{s}$, we have that for any node $t \in V(T)$, $\chi(t) = \heartsuit$ if and only if $\chi'(t) = \heartsuit$. Now we construct the function ϕ. Let $V = V(G)$ and $V' = V(G \oplus H) \setminus V(G)$. For any node $t \in V(\widehat{T})$, we define $\phi(t) = (\psi(t) \cap V) \cup (\eta(t) \cap V')$. Let $L_2 = \bigcup_{t \in R} \phi(t)$ where R is the set of leaf nodes in $\widehat{T}$.

Now we prove that $G \oplus H$ realizes $\tilde{s}$ through $(\widehat{T}, \phi, L_2)$. It is easy to verify that $(T^\star, \psi, Z^\star)$ is equal to $\mathsf{Restricted}_G(\widehat{T}, \phi, L_2)$. We have already mentioned that G satisfies $\tilde{s}$ through $(T^\star, \psi, Z^\star)$. Next we prove that indeed $(\widehat{T}, \phi, L_2)$ is an $\mathcal{H}$-elimination decomposition of $G \oplus H$. It is easy to verify that conditions (1) and (2) of Definition 2.1 holds. Now we will prove that condition (3) of Definition 2.1 holds. Let uv be an edge in $G \oplus H$ and $t, t' \in V(\widehat{T})$ such that $u \in \phi(t)$ and $v \in \phi(t')$. Suppose $u, v \in V(G)$. Then, since G satisfies $\tilde{s}$ through $(T^\star, \psi, Z^\star)$, which is equal to $\mathsf{Restricted}_G(\widehat{T}, \phi, L_2)$, we have that t and t' are in ancestor-descendant relation. Suppose $u, v \in V(H)$. Then, since $(\widehat{T}, \eta, L_1)$ is an $\mathcal{H}$-elimination decomposition of $G \oplus H$, from the construction of ϕ, we have that t and t' are in ancestor-descendant relation. Thus, we have proved that condition (3) of Definition 2.1 holds.

Next we prove that condition (4) of Definition 2.1 holds. Let $S_H = \{v \in \delta(H) \colon \lambda_G(v) \in \lambda_G(S \cap \delta(G))\}$. That is S_H is the set of boundary vertices in H that has the same label as the vertices in $S \cap \delta(G)$. Let t be a leaf node. Let $G_1 = G[\eta(t) \cap V(G)] = G[\widehat{\chi}(t)]$ and $H_1 = H[\eta(t) \cap V(H)]$. Let $G_2 = G[\phi(t) \cap V(G)] = G[\psi(t)]$ and $H_2 = H[(\eta(t) \cap V(H)) \setminus S_H]$. Notice that $G[\eta(t)] = G_1 \oplus H_1$ and $G[\phi(t)] = G_2 \oplus H_2$. Since $G \oplus H$ exactly realizes s through $(\widehat{T}, \eta, L_1)$, we have that G_1 belongs to the equivalence class $\mathsf{eq}(\chi(t))$ and $G_1 \oplus H_1 \in \mathcal{H}$. Since G satisfies $\tilde{s}$ through $(T^\star, \psi, Z^\star)$ G_2 belongs to an equivalence class which is at least as good as $\mathsf{eq}'(\chi'(t)) = \mathsf{eq}(\chi(t))$. This implies that $G_2 \oplus H_1 \in \mathcal{H}$. Since H_2 is an induced subgraph of H_1 and $G_2 \oplus H_1 \in \mathcal{H}$, by the hereditary property of $\mathcal{H}$, we have that $G_2 \oplus H_2 \in \mathcal{H}$. □

This completes the proof of the lemma. □

Because of Lemma 7.29, in the final step of our algorithm Sig-Test, we run the algorithm $\mathcal{A}_u$ on the input $(G^\star, EQ^\star, k')$ as defined in Lemma 7.29. If the algorithm $\mathcal{A}_u$ fails to output

a solution, then we output Fail. Otherwise, let S be the output of the algorithm $\mathcal{A}_u$. If $S \cap \delta(G) = \emptyset$, then we output $(s, 1)$. If $S \cap \delta(G) \neq \emptyset$, then we construct $\tilde{s}$ (as mentioned in Lemma 7.29) and output $(s, 0)$ and $(\tilde{s}, 1)$.

Running time analysis Each branching rules makes a function of $q+k$ many branches. Since the number of nodes in $T^\star$ is at most $q + k$ and in each branch of the branching algorithm the number of internal nodes t with $\chi^\star(t) = \emptyset$ decreases by 1, the depth of the branching algorithm is at most $q + k$. In the final step we run the algorithm $\mathcal{A}_u$. This implies that the running time of the algorithm is upper bounded $g_1(q, k) \cdot f_u(k, |\mathcal{Q}(\equiv_u)|, n) \cdot n^{\mathcal{O}(1)}$ for some function g_1.

8 Applications of the Uniform Algorithm

The objective of this section is to prove the following theorem.

Theorem 8.1. Elimination Distance to $\mathcal{H}$ *admits a uniform FPT algorithm, for the case when $\mathcal{H}$ is any one of the following:*

1. *the family of chordal graphs,*
2. *the family of interval graphs,*
3. *the family of bipartite graphs,*
4. *for a fixed finite family of graphs $\mathbb{O}$, a graph is in $\mathcal{H}$, if and only if it does not contain any graph from $\mathbb{O}$ as an induced subgraph,*
5. *for a fixed finite family of graphs $\mathbb{O}$, a graph is in $\mathcal{H}$, if and only if it does not contain any graph from $\mathbb{O}$ as a minor,*[19] *or*
6. *for a fixed finite family of graphs $\mathbb{O}$, a graph is in $\mathcal{H}$, if and only if it does not contain any graph from $\mathbb{O}$ as a topological minor.*

For any fixed finite family of graphs $\mathcal{F}$, we can find a finite family of graphs $\mathcal{F}'$, such that a graph does not contain any minor from $\mathcal{F}$ if and only if it does not contain any topological minor from $\mathcal{F}'$. Thus, item 5 in Theorem 8.1 is subsumed by item 6 (see for instance, [33] for a discussion on this). Thus, we will not focus on proving item 5, and our rest of the section will focus on proving the theorem for all other items its statement.

We denote the families of bipartite, chordal, interval graphs by $\mathcal{H}_{\mathsf{bip}}$, $\mathcal{H}_{\mathsf{cdl}}$, and $\mathcal{H}_{\mathsf{int}}$, respectively. For a fixed finite family of graphs $\mathcal{F}$, we denote the families of graphs that have no graph from $\mathcal{F}$ as an induced subgraph and a topological minor by $\mathcal{H}_{\mathcal{F}_{\mathsf{ind}}}$ and $\mathcal{H}_{\mathcal{F}_{\mathsf{top}}}$, respectively.

To invoke Theorem 7.2, we need to define a refinement of the canonical equivalence class (see Definition 7.1) for the graph families of our concern. To this end, we introduce the notion of obstructions.

Obstructions to a family of graphs. For a family of graphs $\mathcal{H}$, a family of (possibly infinite) set of graphs $\mathbb{O}$ is an *induced obstruction set* to $\mathcal{H}$, if a graph $G \in \mathcal{H}$ if and only if G does not contain any graph from $\mathbb{O}$ as an induced subgraph. Notice that we have the following: i) the family $\mathbb{O}_{\mathsf{bib}}$ of all cycles of odd length is an induced obstruction set for $\mathcal{H}_{\mathsf{bib}}$ and ii) the family $\mathbb{O}_{\mathsf{cdl}}$ of all cycles of length at least 4 is an induced obstruction set for $\mathcal{H}_{\mathsf{cdl}}$.

The set of obstructions to interval graphs have been completely characterized by Lekkerkerker and Boland, [55]. A graph is an interval graph if and only if it does not contain any of the following graphs as an induced subgraph (see Figure 3).[20]

[19] As minor closed families are characterized by a finite family of forbidden minors by Robertson-Seymour Theorem, so the condition on finiteness for this case is not necessary.

[20] The figure is borrowed from [4].

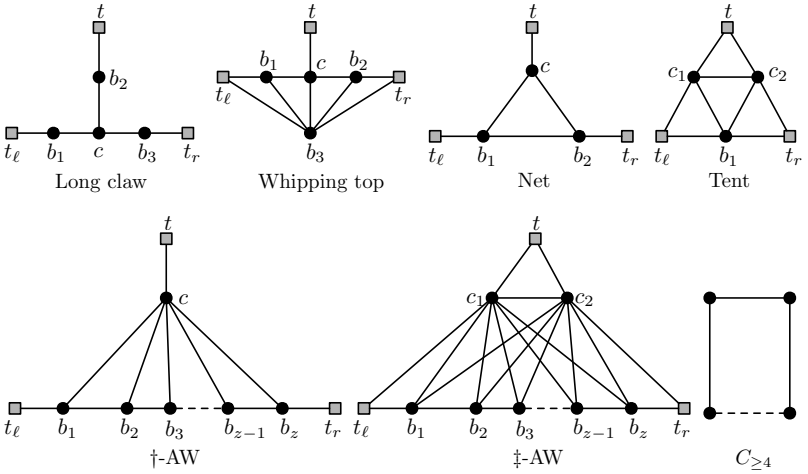

Figure 3: The set of obstructions for the family of interval graph.

- **Long Claw.** A graph $\mathbb{O}$ such that $V(\mathbb{O}) = \{t_\ell, t_r, t, c, b_1, b_2, b_3\}$ and $E(\mathbb{O}) = \{(t_\ell, b_1), (t_r, b_3), (t, b_2), (c, b_1), (c, b_2), (c, b_3)\}$.
- **Whipping Top.** A graph $\mathbb{O}$ such that $V(\mathbb{O}) = \{t_\ell, t_r, t, c, b_1, b_2, b_3\}$ and $E(\mathbb{O}) = \{(t_\ell, b_1), (t_r, b_2), (c, t), (c, b_1), (c, b_2), (b_3, t_\ell), (b_3, b_1), (b_3, c), (b_3, b_2), (b_3, t_r)\}$.
- **†-AW.** A graph $\mathbb{O}$ such that $V(\mathbb{O}) = \{t_\ell, t_r, t, c\} \cup \{b_1, b_2, \ldots, b_z\}$, where $t_\ell = b_0$ and $t_r = b_{z+1}$, $E(\mathbb{O}) = \{(t, c), (t_\ell, b_1), (t_r, b_z)\} \cup \{(c, b_i) \mid i \in [z]\} \cup \{(b_i, b_{i+1}) \mid i \in [z-1]\}$, and $z \geq 2$. A †-AW where $z = 2$ will be called a *net*.
- **‡-AW.** A graph $\mathbb{O}$ such that $V(\mathbb{O}) = \{t_\ell, t_r, t, c_1, c_2\} \cup \{b_1, b_2, \ldots, b_z\}$, where $t_\ell = b_0$ and $t_r = b_{z+1}$, $E(\mathbb{O}) = \{(t, c_1), (t, c_2), (c_1, c_2), (t_\ell, b_1), (t_r, b_z), (t_\ell, c_1), (t_r, c_2)\} \cup \{(c, b_i) \mid i \in [z]\} \cup \{(b_i, b_{i+1}) \mid i \in [z-1]\}$, and $z \geq 1$. A ‡-AW where $z = 1$ will be called a *tent*.
- **Hole.** A chordless cycle on at least four vertices.

We denote the above family of graph by $\mathbb{O}_{\mathsf{int}}$, which is an induced obstruction set for $\mathcal{H}_{\mathsf{int}}$.

Let $\mathcal{G}$ denote the family of all graphs. Consider a finite set of graphs $\mathcal{F}$. Notice that an induced obstruction set for $\mathcal{H}_{\mathcal{F}_{\mathsf{ind}}}$ is the same as $\mathbb{O}_{\mathcal{F}_{\mathsf{ind}}} = \mathcal{F}$. Let $\mathbb{O}_{\mathcal{F}_{\mathsf{top}}}$ be the family of all graphs that are not contained in $\mathcal{H}_{\mathcal{F}_{\mathsf{top}}}$, i.e., $\mathbb{O}_{\mathcal{F}_{\mathsf{top}}} = \mathcal{G} \setminus \mathcal{H}_{\mathcal{F}_{\mathsf{top}}}$. Notice that each graph in $\mathbb{O}_{\mathcal{F}_{\mathsf{top}}}$ must contain some $G \in \mathcal{F}$ as a topological minor. Thus, we can obtain that $\mathbb{O}_{\mathcal{F}_{\mathsf{top}}}$ is an induced obstruction set for $\mathcal{H}_{\mathcal{F}_{\mathsf{top}}}$.

Let $\mathbb{H} = \{\mathcal{H}_{\mathsf{cdl}}, \mathcal{H}_{\mathsf{int}}, \mathcal{H}_{\mathsf{bip}}\} \cup \{\mathcal{H}_{\mathcal{F}_{\mathsf{min}}}, \mathcal{H}_{\mathcal{F}_{\mathsf{top}}} \mid \mathcal{F}$ is a finite family of graphs$\}$. From our previous discussions we can obtain that each $\mathcal{H} \in \mathbb{H}$ has an induced obstruction set $\mathbb{O}$. In the rest of the section, for $\mathcal{H} \in \mathbb{H}$, we will work with the corresponding $\mathbb{O}$ that we stated previously. For the sake of simplicity, throughout this section we will assume that the label set of any boundaried graph G, we have $\Lambda(G) \subseteq \mathbb{N}$, where $0 \in \Lambda(G)$ and for no vertex $v \in \delta(G)$, we have $\lambda_G(v) = 0$.

Definition 8.1 (Boundaried Partial Obstruction)**.** Consider a family of graphs $\mathcal{H}$ and an induced obstruction set $\mathbb{O}$ for it. A *partial obstruction of* $\mathcal{H}$ from $\mathbb{O}$ is any graph that is an induced subgraph of some graph in $\mathbb{O}$. A *boundaried partial obstruction of* $\mathcal{H}$ is boundaried graph whose unboundaried counterpart (graph after forgetting the labels of the boundary) is a partial obstruction from $\mathbb{O}$. We denote the set of all boundaried partial obstruction from $\mathbb{O}$ of $\mathcal{H}$ by $\mathsf{BPO}_{\mathbb{O}}(\mathcal{H})$. (We skip the subscript in the above notation whenever the context is clear.)

Definition 8.2 (Relevant Boundaried Partial Obstruction)**.** Consider a family of graphs $\mathcal{H}$ and an induced obstruction set $\mathbb{O}$ for it. We say that $O \in \mathsf{BPO}(\mathcal{H})$ is *relevant* if there does not exist another boundaried partial obstruction $O' \in \mathsf{BPO}(\mathcal{H})$, such that:

1. $|V(O')| < |V(O)|$, and
2. for every boundaried graph G, $O \oplus G \notin \mathcal{H}$ if and only if $O' \oplus G \notin \mathcal{H}$.

In the next observation we show that for each $t \in \mathbb{N}$, we can compute the set of relevant boundaried partial obstructions in time bounded by t alone.

Observation 8.3. *For each $\mathcal{H} \in \mathbb{H}$, there exists a function $\zeta[\mathcal{H}] : \mathbb{N} \to \mathbb{N}$, such that for any $t \in \mathbb{N}$, we can find an inclusion-wise maximal set of relevant t-boundaried partial obstructions, denoted by, $\mathsf{RBPO}(\mathcal{H}, t)$ (with respect to $\mathbb{O}$) in time bounded by $\zeta[\mathcal{H}](t)$, such that $|\mathsf{RBPO}(\mathcal{H}, t)| \leq \zeta[\mathcal{H}](t)$.*

Proof. Note that for the family of chordal graphs, if we have a boundaried graph G with $\Lambda(G) \subseteq \{1, \ldots, t\}$, the following state information is enough: for every pair of labels in $\Lambda(G)$, whether there no path, a path with exactly one edge, or all paths have at least two edges, between the corresponding vertices associated with the labels. Notice that for each such state information, keeping one partial obstruction is enough for us, which can be found easily based on the state information. Similarly for all other graph classes, we can argue that only bounded number of state information are required, corresponding to each of which keeping one partial obstruction is enough for our purpose. □

For $\mathcal{H} \in \mathbb{H}$ and $t \in \mathbb{N}$ let $q[\mathcal{H}, t] = |\mathsf{RBPO}(\mathcal{H}, t)|$, we fix an arbitrary ordering among the t-boundaried graphs in $\mathsf{RBPO}(\mathcal{H}, t)$, and we let $\mathsf{RBPO}(\mathcal{H}, t) = \{P[\mathcal{H}, t, 1], P[\mathcal{H}, t, 2], \cdots, P[\mathcal{H}, t, q[\mathcal{H}, t]]\}$. We will next define the notion of "behaviour" of a boundaried graph, that will help us in defining our (refinement of) canonical equivalent class (see Definition 7.1).

Definition 8.4. Consider $\mathcal{H} \in \mathbb{H}$, $t \in \mathbb{N}$ and a t-boundaried graphs G. The $[\mathcal{H}, t]$-*behaviour* of G is the vector $\mathsf{bhv}[\mathcal{H}, t](G) = (z_1, z_2, \cdots, z_{q[\mathcal{H},t]})$, where for $i \in [q[\mathcal{H}, t]]$, $z_i = 1$ if and only if $G \oplus P[\mathcal{H}, i] \in \mathcal{H}$.

Definition 8.5. Consider $\mathcal{H} \in \mathbb{H}$ and $t \in \mathbb{N}$. Let G_1 and G_2 be two t-boundaried graphs, such that $\Lambda(G_1) = \Lambda(G_2)$ and t is the largest number in $\Lambda(G_1)$. We say that G_1 and G_2 are $\mathcal{H}$-*behaviour equivalent*, denoted by $G_1 \equiv_{u,\mathcal{H}} G_2$, if $\mathsf{bhv}[\mathcal{H}, t](G_1) = \mathsf{bhv}[\mathcal{H}, t](G_2)$.[21]

Notice that $\equiv_{u,\mathcal{H}}$ defines an equivalence class over the boundaried graphs. We use $\mathcal{Q}(\equiv_{u,\mathcal{H}})$ to denote the set of equivalence classes of $\equiv_{u,\mathcal{H}}$. The next observation follows from Definition 8.5.

Observation 8.6. *For each $\mathcal{H} \in \mathbb{H}$, and boundaried graphs G_1 and G_2, such that $G_1 \equiv_{u,\mathcal{H}} G_2$, we have $G_1 \equiv_{\mathcal{H}} G_2$. Also, $\equiv_{u,\mathcal{H}}$ satisfies the requirement of Definition 7.3.*

We will next give a definition that will be useful in obtaining the user defined function $\mathsf{ug}_{\mathcal{H}}$.

Definition 8.7. Consider $\mathcal{H} \in \mathbb{H}$, and two distinct equivalence classes $Q_1, Q_2 \in \mathcal{Q}(\equiv_{u,\mathcal{H}})$. Moreover, let t_1 and t_2 be the largest numbers in $\Lambda(G_1)$, for $G_1 \in Q_1$ and $\Lambda(G_2)$, for $G_2 \in Q_2$, respectively. We say that Q_1 is *at least as good as* Q_2 if each of the following holds:

1. $t_1 \leq t_2$, and
2. for each $H \in \mathsf{RBPO}(\mathcal{H}, t_1) \subseteq \mathsf{RBPO}(\mathcal{H}, t_2)$, whenever for any $G_2 \in Q_2$, $G_2 \oplus H \in \mathcal{H}$, then any $G_1 \in Q_1$ we have $G_1 \oplus H \in \mathcal{H}$.

Using the above definition we are now ready to define the function $\mathsf{ug}_{\mathcal{H}}$.

Definition 8.8. For $\mathcal{H} \in \mathbb{H}$, we define the function $\mathsf{ug}_{\mathcal{H}} : \mathcal{Q}(\equiv_{u,\mathcal{H}}) \times \mathcal{Q}(\equiv_{u,\mathcal{H}}) \to \{0, 1\}$ as follows: for $Q_1, Q_2 \in \mathcal{Q}(\equiv_{u,\mathcal{H}})$, if Q_1 is at least as good as Q_2 as per Definition 8.7, then set $\mathsf{ug}_{\mathcal{H}}(Q_1, Q_2) = 1$, otherwise set $\mathsf{ug}_{\mathcal{H}}(Q_1, Q_2) = 0$.

[21] The letter u in $\equiv_{u,\mathcal{H}}$ is to denote that it is a user defined equivalence class, as we are the user in this section.

We have now defined a refinement of $\equiv_{\mathcal{H}}$ and $\mathsf{ug}_{\mathcal{H}}$, for each $\mathcal{H} \in \mathbb{H}$. We will next focus on designing the "user's algorithm". To this end, we begin by defining the notion of an irrelevant vertex.

Definition 8.9 (Irrelevant Vertex)**.** Consider a family of graphs $\mathcal{H}$, a graph G, and an integer $k \in \mathbb{N}$. A vertex $v \in V(G)$ is $(\mathcal{H}, k)$-*irrelevant* if there exists $S \subseteq V(G)$ of size at most k such that $G - S \in \mathcal{H}$ if and only if there exists $S' \subseteq V(G - \{v\})$ of size at most k such that $(G - \{v\}) - S' \in \mathcal{H}$.

We now extend the above definition to boundaried graphs.

Definition 8.10 (Boundaried Irrelevant Vertex)**.** Consider a family of graphs $\mathcal{H}$, a boundaried graph G, and an integer $k \in \mathbb{N}$. A vertex $v \in V(G) \setminus \delta(G)$ is *boundaried* $(\mathcal{H}, k)$-*irrelevant* if for every boundaried graph H, v is $(\mathcal{H}, k)$-irrelevant in $G \oplus H$.

Definition 8.11 (Boundaried Obstruction Irrelevant Vertex)**.** Consider $\mathcal{H} \in \mathbb{H}$, a t-boundaried graph G, where t is the largest integer in $\Lambda(G)$, and an integer $k \in \mathbb{N}$. A vertex $v \in V(G) \setminus \delta(G)$ is *boundaried obstruction* $(\mathcal{H}, k)$-*irrelevant* if for every $H \in \mathsf{RBPO}(\mathcal{H}, t)$, v is $(\mathcal{H}, k)$-irrelevant in $G \oplus H$.

Lemma 8.12. *Consider $\mathcal{H} \in \mathbb{H}$, a t-boundaried graph G, where t is the largest integer in $\Lambda(G)$, and an integer $k \in \mathbb{N}$. A vertex $v \in V(G) \setminus \delta(G)$ is boundaried $(\mathcal{H}, k)$-irrelevant if and only if it is boundaried obstruction $(\mathcal{H}, k)$-irrelevant.*

Proof. Let $v \in V(G) \setminus \delta(G)$. One direction is trivial: If v is boundaried $(\mathcal{H}, k)$-irrelevant, then it is boundaried obstruction $(\mathcal{H}, k)$-irrelevant. Indeed, suppose that v is boundaried $(\mathcal{H}, k)$-irrelevant. Then, for every boundaried graph H, v is $(\mathcal{H}, k)$-irrelevant in $G \oplus H$. In particular, for every $O \in \mathsf{RBPO}(\mathcal{H}, t)$, v is $(\mathcal{H}, k)$-irrelevant in $G \oplus O$. So, v is boundaried obstruction $(\mathcal{H}, k)$-irrelevant.

Now, consider the reverse direction. Suppose v is boundaried obstruction $(\mathcal{H}, k)$-irrelevant. Then, for every $O \in \mathsf{RBPO}(\mathcal{H}, t)$, v is $(\mathcal{H}, k)$-irrelevant in $G \oplus O$. We need to prove that for any boundaried graph H, v is $(\mathcal{H}, k)$-irrelevant in $G \oplus H$. Towards that, let us fix a boundaried graph H. Without loss of generality, we can suppose that $\delta(H) \subseteq \{1, \ldots, t\}$ (else, we can "unlabel" the vertices in H having a larger label, and thereby we do not change $G \oplus H$). Targeting a contradiction, suppose that v is not $(\mathcal{H}, k)$-irrelevant in $G \oplus H$. So, because $\mathcal{H}$ is hereditary, this means that there exists $S' \subseteq V(G \oplus H - \{v\})$ of size at most k such that $(G \oplus H - \{v\}) - S' \in \mathcal{H}$, but there does not exist $S \subseteq V(G \oplus H)$ of size at most k such that $G \oplus H - S \in \mathcal{H}$. In particular, $G \oplus H - (S' \cup \{v\}) \notin \mathcal{H}$, so $G \oplus H$ must have an obstruction $O^\star$ in $\mathbb{O}$ (as an induced subgraph) that contains v and is disjoint from S'. Let O' be the subgraph of $O^\star[V(H)]$ whose boundary is the boundary vertices of H that occur in $O^\star$. So, O' is a t-boundaried partial obstruction of $\mathcal{H}$ such that $G \oplus O' - (S' \cap V(G)) \notin \mathcal{H}$ and $G \oplus O' - ((S' \cap V(G)) \cup \{v\}) \in \mathcal{H}$. By the definition of $\mathsf{RBPO}(\mathcal{H}, t)$, there exists $O \in \mathsf{RBPO}(\mathcal{H}, t)$ such that $G \oplus O - (S' \cap V(G)) \notin \mathcal{H}$ and $G \oplus O - ((S' \cap V(G)) \cup \{v\}) \in \mathcal{H}$. However, v is $(\mathcal{H}, k)$-irrelevant in $G \oplus O$, and thus we have reached a contradiction. □

For the families of graph that we are interested in, our objective will be to find an "irrelevant" vertex, towards obtaining a "user's algorithm". We first explain *intuitively* how we intend to obtain such a rule. For simplicity consider the family of chordal graphs. We know that any chordal graph with large treewidth must contain a large clique. More generally, any graph from which we can delete a small set of vertices so that the resulting graph is a chordal graph, either it contains a large clique, or its treewidth can be bounded by the size of the small deletion set. We remark that whenever a graph has a large clique, using the known (FPT/kernelization) algorithms for Chordal Vertex Deletion [63], we will be able to find an irrelevant vertex. We give generic definitions below, and then we will see how our families fit into these definitions.

Definition 8.13. Consider families of graphs $\widehat{\mathcal{G}}$ and $\widetilde{\mathcal{G}}$. We say that $\widehat{\mathcal{G}}$ has *treewidth property* with respect to $\widetilde{\mathcal{G}}$, if there is a function $\tau : \mathbb{N} \to \mathbb{N}$, such that any graph $G \in \widehat{\mathcal{G}}$ with $\mathbf{tw}(G) \geq \ell$, contains a graphs $G' \in \widetilde{\mathcal{G}}$ with $|V(G')| \geq \tau(\ell)$ as an induced subgraph. In that above, we say that τ is a *tw-certificate* for the treewidth-property of $\widehat{\mathcal{G}}$ with respect to $\widetilde{\mathcal{G}}$.

We will now extend the above definition to the classes where we can find large subgraphs of desired type in bounded amount of time.

Definition 8.14. Consider families of graphs $\widehat{\mathcal{G}}$ and $\widetilde{\mathcal{G}}$, such that $\widehat{\mathcal{G}}$ has treewidth-property with respect to $\widetilde{\mathcal{G}}$, with tw-certificate $\tau : \mathbb{N} \to \mathbb{N}$. We say that $\widehat{\mathcal{G}}$ has *efficient treewidth-property* with respect to $\widetilde{\mathcal{G}}$, if there a function $g : \mathbb{N} \to \mathbb{N}$ and an algorithm, which given an n-vertex graph $G \in \widehat{\mathcal{G}}$ and an integer $\ell \in \mathbb{N}$, in time bounded by $g(\ell) \cdot n^{\mathcal{O}(1)}$, either correctly concludes that $\mathbf{tw}(G) < \ell$, or outputs a graph $G' \in \widetilde{\mathcal{G}}$, such that: i) $|V(G')| \geq \tau(\ell)$ and ii) G' is an induced subgraph of G.

Definition 8.15. Consider families of graphs $\widehat{\mathcal{G}}$, $\widetilde{\mathcal{G}}$, and $\mathcal{H}$, where $\widetilde{\mathcal{G}} \subseteq \mathcal{H}$. We say that $\widehat{\mathcal{G}}$ is $(\widetilde{\mathcal{G}}, \mathcal{H})$-*deletable*, if there is a function $\mu : \mathbb{N} \to \mathbb{N}$, such that: for any graph $G \in \widehat{\mathcal{G}}$ which contains a graph $G' \in \widetilde{\mathcal{G}}$ with at least k vertices as an induced subgraph, there exists $Z \subseteq V(G') \subseteq V(G)$ of size at least $\mu(k)$, such that each vertex $u \in Z$ is $(\mathcal{H}, k)$-irrelevant in G. In the above, we say that μ is a *del-certificate* of $\widehat{\mathcal{G}}$ being $(\widetilde{\mathcal{G}}, \mathcal{H})$-deletable.

Definition 8.16. Consider families of graphs $\widehat{\mathcal{G}}$, $\widetilde{\mathcal{G}}$, and $\mathcal{H}$, where $\widetilde{\mathcal{G}} \subseteq \mathcal{H}$ and $\widehat{\mathcal{G}}$ is $(\widetilde{\mathcal{G}}, \mathcal{H})$-deletable, where μ is a del-certificate of $\widehat{\mathcal{G}}$ being $(\widetilde{\mathcal{G}}, \mathcal{H})$-deletable. A $(\widehat{\mathcal{G}}, \widetilde{\mathcal{G}}, \mathcal{H})$-*irrelevance detector* is an algorithm, which given an n-vertex graph $G \in \widehat{\mathcal{G}}$ and an induced subgraph $G' \in \widetilde{\mathcal{G}}$ of G, where $k = |V(G')|$, outputs a subset $Z \subseteq V(G') \subseteq V(G)$ of size at least $\mu(k)$, such that each vertex in Z is $(\mathcal{H}, k)$-irrelevant in G.

Now we will state results which follow from the known results, that will be important in obtaining our "user's algorithm". The next result says that, given a graph which admits a k-sized deletion set to chordal graphs, along with a large clique, then we will be able to obtain large set of vertices in this clique that are irrelevant for us. We remark that each of this vertices are "individually" irrelevant to us, and we won't be deleting all of them together. The above discussed result simply corresponds to the marking of vertices in a maximal clique in the clique-tree of the input graphs, obtained after removing a small deletion set. (A similar result can also be obtained for interval graphs.) Explicitly we can obtain such a result in various paper, like [63, 50, 3], where to the best of our knowledge, the first time such a result appeared in [63]. For an explicit reference, we direct the readers to Lemma 4.1 and Section 3.3 of [3]. We would like to remark that, whenever a reduction rule from the above result, deletes a vertex and decrements k by 1, we can instead add such a vertex to our "relevant set". Also, since we will assume that the input graph comes with the promise that it admits an at most k-sized deletion set to chordal graphs, the known algorithms cannot return no for such a graph with the integer k as input. Also, the kernelization algorithm for Interval Vertex Deletion given in, say, [3], never explicitly returns yes.

Observation 8.17. *There are computable (non-decreasing) functions $f_{\mathsf{cdl}}, g_{\mathsf{cdl}} : \mathbb{N} \to \mathbb{N}$, and an algorithm, which given an n-vertex graph G, an integer $k \geq 1$ and a maximal clique G' in G such that: i) $\mathbf{mod}_{\mathcal{H}_{\mathsf{cdl}}}(G) \leq k$, and ii) $|V(G')| \geq g_{\mathcal{H}_{\mathsf{cdl}}}(k)$; in time bounded by $f_{\mathcal{H}_{\mathsf{cdl}}}(k) \cdot n^{\mathcal{O}(1)}$, outputs a set $X \subseteq V(G')$, such that $|X| \leq g_{\mathcal{H}_{\mathsf{cdl}}}(k)$ and each vertex in $V(G') \setminus X$ is $(\mathcal{H}_{\mathsf{cdl}}, k)$-irrelevant in G.*

A result similar to the above can be obtained for the family of interval graphs, say, using Lemma 4.2 and Section 5 of [4].

Observation 8.18. *There are computable (non-decreasing) functions $f_{\mathsf{int}}, g_{\mathsf{int}} : \mathbb{N} \to \mathbb{N}$, and an algorithm, which given an n-vertex graph G, an integer $k \geq 1$ and a maximal clique G' in G such that: i) $\mathbf{mod}_{\mathcal{H}_{\mathsf{int}}}(G) \leq k$, and ii) $|V(G')| \geq g_{\mathcal{H}_{\mathsf{int}}}(k)$; in time bounded by $f_{\mathcal{H}_{\mathsf{int}}}(k) \cdot n^{\mathcal{O}(1)}$, outputs a set $X \subseteq V(G')$, such that $|X| \leq g_{\mathcal{H}_{\mathsf{int}}}(k)$ and each vertex in $V(G') \setminus X$ is $(\mathcal{H}_{\mathsf{int}}, k)$-irrelevant in G.*

Next we state a result which will help us find an irrelevant vertex for the case of $\mathcal{H}_{\mathsf{bip}}$, which can be obtained from Lemma 3.2 of [59].

Observation 8.19. *There are computable (non-decreasing) functions $f_{\mathsf{bip}}, g_{\mathsf{bip}} : \mathbb{N} \to \mathbb{N}$, and an algorithm, which given an n-vertex graph G, an integer $k \geq 1$, and a well-linked set Z in G such that: i) $\mathbf{mod}_{\mathcal{H}_{\mathsf{bip}}}(G) \leq k$, and ii) $|Z| \geq g_{\mathcal{H}_{\mathsf{bip}}}(k)$;[22] in time bounded by $f_{\mathcal{H}_{\mathsf{bip}}}(k) \cdot n^{\mathcal{O}(1)}$, outputs a set $X \subseteq Z$, such that $|X| \leq g_{\mathcal{H}_{\mathsf{bip}}}(k)$ and each vertex in $V(G') \setminus X$ is $(\mathcal{H}_{\mathsf{bip}}, k)$-irrelevant in G.*

In the next lemma we show how we can turn Observation 8.17 to 8.19 to find a boundaried $(\mathcal{H}, k)$-irrelevant vertex (together with the known algorithms for deletion to $\mathcal{H} \in \mathbb{H}$ [70, 63, 14, 33, 23, 59]).

Lemma 8.20. *Consider $\mathcal{H} \in \{\mathcal{H}_{\mathsf{chl}}, \mathcal{H}_{\mathsf{int}}, \mathcal{H}_{\mathsf{bip}}\}$. There are computable functions $f^*_{\mathcal{H}}, g^*_{\mathcal{H}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, and an algorithm, which given a t-boundaried graph G, where t is the largest number in $\Lambda(G)$, an integer $k \geq 0$, and an equivalence class $Q = (t^*, \Lambda \subseteq [t^*], (z_1, z_2, \cdots, z_{q[\mathcal{H}_{\mathsf{cdl}}, t^*]}))$, such that if $t^* \neq 0$, then $t^* \in \Lambda$; in time bounded by $f^*_{\mathcal{H}}(t,k) \cdot n^{\mathcal{O}(1)}$ it does one of the following:*

1. *correctly conclude that $\mathbf{tw}(G) \leq g^*_{\mathcal{H}}(t+k)$,[23],*
2. *find a boundaried $(\mathcal{H}, k)$-irrelevant vertex $v \in V(G) \setminus \delta(G)$, or*
3. *correctly conclude that there is no $S \subseteq V(G)$ of size at most k, such that $G - S$ belongs to an equivalence class Q', where $\mathsf{ug}_{\mathcal{H}_{\mathsf{cdl}}}(Q', Q) = 1$ (see Definition 8.8).*

Proof. We explain the proof for the case when $\mathcal{H} = \mathcal{H}_{\mathsf{cdl}}$, as the proof for all the other cases can be obtained in a similar fashion. Recall that we have an ordering on $\mathsf{RPBO}(\mathcal{H}_{\mathsf{cdl}}, t)$, which has allowed us to assume that $\mathsf{RBPO}(\mathcal{H}, t) = \{P[\mathcal{H}, t, 1], P[\mathcal{H}, t, 2], \cdots, P[\mathcal{H}, t, q[\mathcal{H}, t]]\}$. Let $q = |\mathsf{RPBO}(\mathcal{H}_{\mathsf{cdl}}, t)|$. The problem Chordal Vertex Deletion admits an FPT algorithm running in time $f(k) \cdot n^{\mathcal{O}(1)}$, where f is a computable function [63].

Let $g^*_{\mathcal{H}}(t', k') = |\mathsf{RPBO}(\mathcal{H}_{\mathsf{cdl}}, t')| \cdot g_{\mathcal{H}_{\mathsf{cdl}}}(k') + \sum_{i \in [q[\mathcal{H}_{\mathsf{cdl}}, t']]} |V(P[\mathcal{H}_{\mathsf{cdl}}, t'])| + k' + 1$, for $t', k' \in \mathbb{N}$, where $g_{\mathcal{H}_{\mathsf{cdl}}}$ is the function from Observation 8.17. For each $i \in [q]$, such that $z_i = 1$, we do the following:

1. If $(G \oplus P[\mathcal{H}, t, i], k)$ is a no-instance of Chordal Vertex Deletion, then report that there is no $S \subseteq V(G)$ of size at most k, such that $G - S$ belongs to an equivalence class Q', where $\mathsf{ug}_{\mathcal{H}_{\mathsf{cdl}}}(Q', Q) = 1$. The correctness of this step of the algorithm follows from its description. Moreover, it can be executed in time bounded by $f(k) \cdot n^{\mathcal{O}(1)}$.
2. We now assume that $(G \oplus P[\mathcal{H}, t, i], k)$ is a yes-instance of Chordal Vertex Deletion, and we compute a tree decomposition (T, χ) for $G \oplus P[\mathcal{H}, t, i]$, where each bag is a clique along with at most $k+1$ more vertices (see [41]). If each bag in the tree decomposition has at most $g^*_{\mathcal{H}}(t) + 1$ vertices, then the tree width of $G \oplus P[\mathcal{H}, t, i]$ (and thus, G) can be bounded by $g^*_{\mathcal{H}}(t, k)$. Note that the above step can be done in time bounded by $f(k) \cdot n^{\mathcal{O}(1)}$.

[22] For the definition of well-linked sets, please see [59]

[23] The treewidth of a boundaried graph is same as the treewidth of its unboundaried counterpart.

3. Otherwise, we can obtain that there is a clique G' in $G \oplus P[\mathcal{H}, t]$, such that $V(G') \subseteq V(G)$, and ii) $|V(G')| > |\mathsf{RPBO}(\mathcal{H}_{\mathsf{cdl}}, t)| \cdot g_{\mathcal{H}_{\mathsf{cdl}}}(k)$ vertices. Now using Observation 8.3, we obtain $X_i \subseteq V(G')$, such that: i) $|X_i| \leq g_{\mathcal{H}_{\mathsf{cdl}}}(k)$, and ii) each $v \in V(G') \setminus X_i$ is $(\mathcal{H}_{\mathsf{cdl}}, k)$-relevant in G.

Let $X = \cup_{i \in q} X_i$ and $Y = V(G' \setminus)$. We note that we use the same G' to compute X_is, for each $i \in [q]$. Note that $|X| \leq |\mathsf{RPBO}(\mathcal{H}_{\mathsf{cdl}}, t)| \cdot g_{\mathcal{H}_{\mathsf{cdl}}}(k)$ and $|V(G')| > |\mathsf{RPBO}(\mathcal{H}_{\mathsf{cdl}}, t)| \cdot g_{\mathcal{H}_{\mathsf{cdl}}}(k)$, and thus $Y \neq \emptyset$. Moreover, notice that each $v \in Y$ is a boundaried $(\mathcal{H}_{\mathsf{cdl}}, k)$-irrelevant vertex. This concludes the proof. □

Using Lemma 3.1, Theorem 6 and 7 from [32], we can obtain the following result for $\mathcal{H}_{\mathcal{F}_{\mathsf{top}}}$. Roughly speaking, the notion of "irrelevance" with respect to extended-folio presented in [32] is a stronger notion than the one we give in Definition 8.9 (which is equivalent to Definition 8.10, see Lemma 8.12). This allows us to directly borrow their algorithmic detection of irrelevant vertices for our case.

Observation 8.21. *Consider a fixed finite family of graphs $\mathcal{F}$. There are computable functions $f^*_{\mathcal{F}_{\mathsf{top}}}, g^*_{\mathcal{F}_{\mathsf{top}}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, and an algorithm, which given an n-vertex t-boundaried graph G and an integer $k \geq 1$, in time bounded $f^*_{\mathcal{F}_{\mathsf{top}}}(t,k) \cdot n^{\mathcal{O}(1)}$, either correctly concludes that $\mathbf{tw}(G) \leq g^*_{\mathcal{F}_{\mathsf{min}}}(t,k)$, or finds a boundaried $(\mathcal{H}_{\mathcal{F}_{\mathsf{top}}}, k)$-irrelevant vertex $v \in V(G) \setminus \delta(G)$.*

We will next prove a lemma that will allow us to find irrelevant vertices for $\mathcal{H}_{\mathcal{F}_{\mathsf{ind}}}$, where $\mathcal{F}$ is a finite family of graphs, using an application of the Sunflower Lemma [29, 23].

Lemma 8.22. *Consider a fixed finite family of graphs $\mathcal{F}$. There are computable functions $f_{\mathcal{F}_{\mathsf{ind}}}, g_{\mathcal{F}_{\mathsf{ind}}} : \mathbb{N} \to \mathbb{N}$, and an algorithm, which given an n-vertex t-boundaried graph G and an integer $k \geq 1$, where $n \geq g_{\mathcal{F}_{\mathsf{ind}}}(k)$, in time bounded $f_{\mathcal{F}_{\mathsf{ind}}}(k) \cdot n^{\mathcal{O}(1)}$, finds a boundaried $(\mathcal{H}_{\mathcal{F}_{\mathsf{ind}}}, k)$-irrelevant vertex $v \in V(G) \setminus \delta(G)$.*

Proof. Let $\mathcal{H} = \mathcal{H}_{\mathcal{F}_{\mathsf{ind}}}$. Let $c = c_{\mathcal{F}}$ be maximum among the size of the largest graph in $\mathcal{F}$, denoted by d, and the size of $\mathcal{F}$. So, c is a fixed constant. Let $\mathcal{O} = \{O : O$ is a t-boundaried graph that is an induced subgraph of at least one graph in $\mathcal{F}\}$. Notice that $|\mathcal{O}| \leq c \cdot 2^c \cdot \sum_{i=0}^{\min(t,c)} \binom{c}{i} \leq \mathcal{O}(1)$, and for each $O \in \mathcal{O}$, $|V(O)| \leq c$. We define $f_{\mathcal{F}_{\mathsf{ind}}}(x) = |\mathcal{O}|$, and $g_{\mathcal{F}_{\mathsf{ind}}}(x) = d^2 \cdot d! \cdot (x-1)^d \cdot |\mathcal{O}| + 1$ for all $x \in \mathbb{N}$.

Now, we describe the algorithm. Let (G, k) be its input. Then:

1. Initialize $M = \emptyset$.
2. For every $O \in \mathcal{O}$:
   (a) Let $G' = G \oplus O$.
   (b) Let $U = V(G')$ and $\mathcal{Q} = \{A \subseteq U : G'[A] \in \mathcal{F}\}$.
   (c) Call the algorithm in Theorem 2.26 [23] on $(U, \mathcal{Q}, k)$ to obtain $(U', \mathcal{H}', k)$.
   (d) Update $M \leftarrow M \cup (U' \cap V(G))$.
3. Return any vertex v from $V(G) \setminus M$.

We first argue that $V(G) \setminus M \neq \emptyset$, thus the algorithm returns a vertex. For this purpose, note that the number of iterations is $|\mathcal{O}|$. By Theorem 2.26 [23], in each one of them, at most $d^2 \cdot d! \cdot (k-1)^d$ new vertices are inserted into M. Thus, at the end, $|M| \leq d^2 \cdot d! \cdot (k-1)^d \cdot |\mathcal{O}| < g_{\mathcal{F}_{\mathsf{ind}}}(k)$. Second, we consider the time complexity of the algorithm. Notice that the algorithm in Theorem 2.26 [23] runs in polynomial time, hence each iteration is performed in polynomial time, which means that the overall runtime is $|\mathcal{O}| \cdot n^{\mathcal{O}(1)} \leq f_{\mathcal{F}_{\mathsf{ind}}}(k) \cdot n^{\mathcal{O}(1)} \leq n^{\mathcal{O}(1)}$.

We now consider the correctness of the algorithm. Trivially, $\mathsf{RBPO}(\mathcal{H}, t) \subseteq \mathcal{O}$. So, by Lemma 8.12, it suffice to show that v is k-irrelevant in $G \oplus O$ for every $O \in \mathcal{O}$. For this purpose, fix some $O \in \mathcal{O}$, and consider the iteration corresponding to this O. We need to show that (G', k) is a yes-instance if and only if $(G' - v, k)$ is a yes-instance. One direction is trivial: If (G', k) is a yes-instance, then $(G' - v, k)$ is a yes-instance. For the other direction, suppose that $(G' - v, k)$ is a yes-instance. Let $U^\star = V(G' - v)$ and $\mathcal{Q}^\star = \{A \subseteq U^\star : G'[A] \in \mathcal{F}\}$. Then, $(U^\star, \mathcal{Q}^\star, k)$ is a yes-instance of d-Hitting Set. As $v \notin M$, we have that $U' \subseteq U^\star$ and $\mathcal{Q}' \subseteq \mathcal{Q}^\star$. However, this means that $(U', \mathcal{Q}', k)$ is also a yes-instance of d-Hitting Set. In turn, because $(U, \mathcal{Q}, k)$ and $(U', \mathcal{Q}', k)$ are equivalent (see, for example, Theorem 2.26 [23]), we have that $(U, \mathcal{Q}, k)$ is also a yes-instance of d-Hitting Set. Hence, (G', k) is a yes-instance. This completes the proof. □

We next handle the bounded treewidth case.

Lemma 8.23. *There is a function $f_{\mathcal{H}}$, and an algorithm that given $\mathcal{H} \in \mathbb{H}$, a t-boundaried graph G, integers $k, \ell \in \mathbb{N}$, and an equivalence class $Q = (t^\star, \{t^\star\} \subseteq \Lambda \subseteq [t^\star], (z_1, \ldots, z_{q[\mathcal{H},t]}))$ such that the $\mathbf{tw}(G) \leq \ell$, runs in time $f_{\mathcal{H}}(k, \ell, t)n^{\mathcal{O}(1)}$ and outputs a vertex subset S of size at most k such that $G - S$ belongs to an equivalence class which is at least as good as Q.*

Proof sketch. Recall that $\mathsf{RBPO}(\mathcal{H}, t) = \{P[\mathcal{H}, t, 1], P[\mathcal{H}, t, 2], \cdots, P[\mathcal{H}, t, q[\mathcal{H}, t]]\}$. For simplicity, let us denote $H_i = P[\mathcal{H}, t, i]$ for all $i \in [q[\mathcal{H}, t]]$. Recall that a graph G' belongs to an equivalence class which is at least as good as Q if the following holds.

(i) $\max \Lambda(G') \leq t^\star$ and $\Lambda(G') \subseteq \Lambda$

(ii) For each $i \in q[\mathcal{H}, t]]$, if $z_i = 1$ then $G' \oplus H \in \mathcal{H}$.

One can write a CMSO formula ψ_1 such that a t-boundaried graph G' satisfies the formula ψ_1 if and only if the property (i) holds. Also, one can write a CMSO formula ψ_2 such that G' satisfies the formula ψ_2 if and only if $G' \in \mathcal{H}$.

Here our objective is to test whether there are k vertices $x_1, \ldots, x_k$ such that for each $i \in q[\mathcal{H}, t]]$, $(G \oplus H_i) - S$ satisfies ψ_1 and for each $i \in [q[\mathcal{H}, t]]]$ such that $z_i = 1$, $(G \oplus H_i) - S$ satisfies ψ_2, where $S = \{x_1, \ldots, x_k\}$. We can write one CMSO formula for it. Towards that we construct a graph

$$G^\star = ((G \oplus H_1) \oplus H_2) \ldots) \oplus H_{q[\mathcal{H},t]]}.$$

Then we will have free variables for each vertex subsets $V(G)$, $V(H_1), \ldots V(H_{q[\mathcal{H},t]]})$ and for each edge subsets $E(G)$, $E(H_1), \ldots E(H_{q[\mathcal{H},t]]})$. Then, each graph $G \oplus H_i$ is specified by four free variables $V(G), V(H_i), E(G)$, and $E(H_i)$. Thus, our objective is to test the existence of a vertex subset $S \subseteq V(G^\star)$ of size k such that for each i, the subgraph $G \oplus H_i$ satisfies ψ_1 and if $z_i = 1$, then it satisfies ψ_2. Thus, it can be expressed using a CMSO formula ψ. This formula size is upper bounded by a function of k, t and $\mathcal{H}$.

We know that $\mathbf{tw}(G) \leq \ell$ and $\{H_1, H_2, \ldots, H_{q[\mathcal{H}}\}$ is a fixed finite set of graphs independent of the input graph, we have that the $\mathbf{tw}(G^\star)$ is upper bounded by a function of k, ℓ, t and $\mathcal{H}$. Therefore the lemma follows from Courcelle's Theorem [18]. □

We are now ready to prove Theorem 8.1.

Proof of Theorem 8.1. Consider $\mathcal{H} \in \mathbb{H}$. Let $f^*, g^* : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be the appropriate functions based on $\mathcal{H}$ returned by Lemma 8.20, Observation 8.21 or Lemma 8.22. To prove the theorem, we will use Theorem 7.2, together with the refinement of the equivalence class given in Definition 8.5, the function $\mathsf{ug}_{\mathcal{H}}$, given in Definition 8.8, and the algorithm $\mathcal{A}$ we describe next. Our algorithm $\mathcal{A}$ will take as input a t-boundaried graph G, where $t \in \Gamma(G)$, an integer k, and an equivalence class $Q \in \mathcal{Q}(\equiv_{u,\mathcal{H}})$, in time bounded by $h(t, k) \cdot n^{\mathcal{O}(1)}$, it will check if there is $S \subseteq V(G)$ of size at most k, such that $G - S$ is in the equivalence class Q', where $\mathsf{ug}_{\mathcal{H}}(Q', Q) = 1$.

If Lemma 8.20, Observation 8.21 or Lemma 8.22 concludes that the treewidth of G is bounded by $g^*(t,k)$, then we resolve the instance using Lemma 8.23. Also, if using one of Lemma 8.20, Observation 8.21 or Lemma 8.22 we are able to obtain the a set S of desired type does not exists, the algorithm returns no. Otherwise, we have a boundaried $(\mathcal{H},k)$-irrelevant vertex $v \in V(G)$, and we do the following: i) If $v \notin \delta(G)$, we (recursively) solve the instance (G,k,Q); ii) Otherwise, let $Q' = t', \Lambda', \overline{z}'$ be the equivalence class obtained from $Q = (t^*, \Lambda, \overline{z})$, where $\Lambda' = \Lambda \setminus \lambda_G(v)$, t' is the largest number in Λ', and $\overline{z}'$ is the restriction of $\overline{z}$ to $\mathsf{RBPO}(\mathcal{H},t') \subseteq \mathsf{RBPO}(\mathcal{H},t^*)$. We (recursively) solve the instance $(G - \{v\}, k, Q')$. The correctness of the algorithm follows from its description. Also, from Lemma 8.20, Observation 8.21, Lemma 8.22 and Lemma 8.23, we can obtain that the algorithm is uniform and it runs in time bounded by $h(t,k) \cdot n^{\mathcal{O}(1)}$, for some function $h : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. □

9 Conclusion

In this paper, we have shown that as far as the task of identifying the boundaries of tractability in parameterized complexity is concerned, two recently popular hybrid sparameterizations that combine solution size and width measures ($\mathcal{H}$-elimination distance and $\mathcal{H}$-treewidth) are effectively only as powerful as the standard parameterization by the size of the modulator to $\mathcal{H}$ for a host of commonly studied graph classes $\mathcal{H}$. This is a surprising result, and one that unifies several recent results in the literature. Moreover, using our main result, we have resolved several open problems that were either stated explicitly or have naturally arisen in recent literature. We have also developed a framework for cross-parameterization and demonstrated how this can be applied to answer an open problem posed by Jansen et al. [48].

Hybrid parameterizations have been the subject of a flurry of interest in the last half-a-decade and it would be interesting to identify new, algorithmically useful, hybrid parameterizations that are provably stronger than existing ones. Further, as our characterization result gives non-uniform algorithms, we have developed a framework to design uniform FPT algorithms to compute elimination distance to $\mathcal{H}$ when $\mathcal{H}$ has certain properties. We have demonstrated that these properties are fairly mild by showing a number of well-studied graph classes that possess these properties.

We leave the following questions arising from our work as interesting future research directions.

- Is Elimination Distance to $\mathcal{H}$ parameterized by $\mathbf{tw}_{\mathcal{H}}(G)$ equivalent to the eight problems in Theorem 1.1 for hereditary, union-closed and CMSO definable $\mathcal{H}$? We conjecture that the answer is yes.
- Could one develop a framework to design uniform FPT algorithms for Treewidth Decomposition to $\mathcal{H}$ in the same spirit as our framework to design uniform FPT algorithms for Elimination Distance to $\mathcal{H}$?

Finally, although we provide powerful classification tools, the fact still remains that $\mathbf{tw}_{\mathcal{H}}(G) \leq \mathbf{ed}_{\mathcal{H}}(G) \leq \mathbf{mod}_{\mathcal{H}}(G)$ and that each parameter could be arbitrarily smaller than the parameters to its right. Thus, it is still an interesting direction of research to study Vertex Deletion to $\mathcal{H}$ for specific classes $\mathcal{H}$ parameterized by $\mathbf{ed}_{\mathcal{H}}(G)$ and $\mathbf{tw}_{\mathcal{H}}(G)$ and aim to optimize the running time, in the spirit of Jansen et al [48].

References

[1] Karl Abrahamson and Michael Fellows. Finite automata, bounded treewidth, and well-quasiordering. *Contemporary Mathematics*, 147:539–539, 1993. 8

[2] Akanksha Agrawal, Lawqueen Kanesh, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. An FPT Algorithm for Elimination Distance to Bounded Degree Graphs. In Markus Bläser and Benjamin Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science (STACS 2021)*, volume 187 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:11, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.STACS.2021.5. 2, 3

[3] Akanksha Agrawal, Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Feedback vertex set inspired kernel for chordal vertex deletion. *ACM Trans. Algorithms*, 15(1):11:1–11:28, 2019. URL: https://doi.org/10.1145/3284356, doi:10.1145/3284356. 57

[4] Akanksha Agrawal, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Interval vertex deletion admits a polynomial kernel. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1711–1730. SIAM, 2019. doi:10.1137/1.9781611975482.103. 53, 57

[5] Akanksha Agrawal and M. S. Ramanujan. On the parameterized complexity of clique elimination distance. In Yixin Cao and Marcin Pilipczuk, editors, *15th International Symposium on Parameterized and Exact Computation, IPEC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 180 of *LIPIcs*, pages 1:1–1:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. URL: https://doi.org/10.4230/LIPIcs.IPEC.2020.1, doi:10.4230/LIPIcs.IPEC.2020.1. 5, 18

[6] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12:308–340, 1991. 13

[7] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996. URL: https://doi.org/10.1137/S0097539793251219, doi:10.1137/S0097539793251219. 21, 23

[8] Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (meta) kernelization. *J. ACM*, 63(5):44:1–44:69, 2016. URL: https://doi.org/10.1145/2973749, doi:10.1145/2973749. 9, 14, 15

[9] Hans L. Bodlaender and Babette van Antwerpen-de Fluiter. Reduction algorithms for graphs of small treewidth. *Inf. Comput.*, 167(2):86–119, 2001. doi:10.1006/inco.2000.2958. 8, 9

[10] Richard B. Borie, R. Gary Parker, and Craig A. Tovey. Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica*, 7(5&6):555–581, 1992. URL: https://doi.org/10.1007/BF01758777, doi:10.1007/BF01758777. 8

[11] Jannis Bulian and Anuj Dawar. Graph isomorphism parameterized by elimination distance to bounded degree. *Algorithmica*, 75(2):363–382, 2016. doi:10.1007/s00453-015-0045-3. 1, 11

[12] Jannis Bulian and Anuj Dawar. Fixed-parameter tractable distances to sparse graph classes. *Algorithmica*, 79(1):139–158, 2017. URL: https://doi.org/10.1007/s00453-016-0235-7, doi:10.1007/s00453-016-0235-7. 2

[13] Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, 58(4):171–176, 1996. URL: https://doi.org/10.1016/0020-0190(96)00050-6, doi:10.1016/0020-0190(96)00050-6. 5

[14] Yixin Cao and Dániel Marx. Interval deletion is fixed-parameter tractable. *ACM Trans. Algorithms*, 11(3):21:1–21:35, 2015. doi:10.1145/2629595. 5, 58

[15] Yixin Cao and Dániel Marx. Chordal editing is fixed-parameter tractable. *Algorithmica*, 75(1):118–137, 2016. doi:10.1007/s00453-015-0014-x. 5

[16] Arthur Cayley. A theorem on trees. *Quarterly Journal of Pure and Applied Mathematics*, 23:376–378, 1889. 27

[17] Rajesh Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michal Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. *SIAM J. Comput.*, 45(4):1171–1229, 2016. URL: http://dx.doi.org/10.1137/15M1032077, doi:10.1137/15M1032077. 38, 42

[18] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990. URL: https://doi.org/10.1016/0890-5401(90)90043-H, doi:10.1016/0890-5401(90)90043-H. 8, 13, 25, 29, 30, 60

[19] Bruno Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. *Handbook of Graph Grammars*, pages 313–400, 1997. 13

[20] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000. URL: https://doi.org/10.1007/s002249910009, doi:10.1007/s002249910009. 5

[21] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discret. Appl. Math.*, 101(1-3):77–114, 2000. URL: https://doi.org/10.1016/S0166-218X(99)00184-5, doi:10.1016/S0166-218X(99)00184-5. 5

[22] Bruno Courcelle and Sang-il Oum. Vertex-minors, monadic second-order logic, and a conjecture by seese. *J. Comb. Theory, Ser. B*, 97(1):91–126, 2007. URL: https://doi.org/10.1016/j.jctb.2006.04.003, doi:10.1016/j.jctb.2006.04.003. 5, 6

[23] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3. 1, 20, 23, 34, 58, 59, 60

[24] Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. Subset feedback vertex set is fixed-parameter tractable. *SIAM J. Discret. Math.*, 27(1):290–309, 2013. 36

[25] Babette Lucie Elisabeth de Fluiter. *Algorithms for graphs of small treewidth*. PhD thesis, 1997. 8, 9

[26] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1. 9

[27] Eduard Eiben, Robert Ganian, Thekla Hamm, and O-joung Kwon. Measuring what matters: A hybrid approach to dynamic programming with treewidth. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019,*

Aachen, Germany, volume 138 of *LIPIcs*, pages 42:1–42:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL: https://doi.org/10.4230/LIPIcs.MFCS.2019.42, doi:10.4230/LIPIcs.MFCS.2019.42. 1, 2, 5, 6, 8

[28] Eduard Eiben, Robert Ganian, and O-joung Kwon. A single-exponential fixed-parameter algorithm for distance-hereditary vertex deletion. *J. Comput. Syst. Sci.*, 97:121–146, 2018. URL: https://doi.org/10.1016/j.jcss.2018.05.005, doi:10.1016/j.jcss.2018.05.005. 5, 6

[29] Paul Erdös and Richard Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, 1(1):85–90, 1960. 59

[30] Fedor V. Fomin, Petr A. Golovach, and Dimitrios M. Thilikos. Parameterized complexity of elimination distance to first-order logic properties. *CoRR (to appear in the proceedings fo LICS 2021)*, abs/2104.02998, 2021. URL: https://arxiv.org/abs/2104.02998, arXiv:2104.02998. 3

[31] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar $\mathcal{F}$-deletion: Approximation, kernelization and optimal FPT algorithms. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 470–479. IEEE Computer Society, 2012. URL: https://doi.org/10.1109/FOCS.2012.62, doi:10.1109/FOCS.2012.62. 5

[32] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Hitting topological minors is fpt. *CoRR*, abs/1904.02944, 2019. URL: http://arxiv.org/abs/1904.02944. 59

[33] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Hitting topological minors is FPT. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1317–1326. ACM, 2020. doi:10.1145/3357713.3384318. 5, 53, 58

[34] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Kernels for (connected) dominating set on graphs with excluded topological minors. *ACM Trans. Algorithms*, 14(1):6:1–6:31, 2018. URL: https://doi.org/10.1145/3155298, doi:10.1145/3155298. 14, 15, 16

[35] Fedor V. Fomin and Yngve Villanger. Treewidth computation and extremal combinatorics. *Comb.*, 32(3):289–308, 2012. 27

[36] Robert Ganian, Sebastian Ordyniak, and M. S. Ramanujan. Going beyond primal treewidth for (M)ILP. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 815–821. AAAI Press, 2017. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14272. 1

[37] Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Backdoor treewidth for SAT. In Serge Gaspers and Toby Walsh, editors, *Theory and Applications of Satisfiability Testing - SAT 2017 - 20th International Conference, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings*, volume 10491 of *Lecture Notes in Computer Science*, pages 20–37. Springer, 2017. doi:10.1007/978-3-319-66263-3_2. 1

[38] Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Combining treewidth and backdoors for CSP. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 36:1–36:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.STACS.2017.36. 1

[39] Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Combining treewidth and backdoors for CSP. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 36:1–36:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. URL: https://doi.org/10.4230/LIPIcs.STACS.2017.36, doi:10.4230/LIPIcs.STACS.2017.36. 2

[40] Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Discovering archipelagos of tractability for constraint satisfaction and counting. *ACM Trans. Algorithms*, 13(2):29:1–29:32, 2017. doi:10.1145/3014587. 7

[41] Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*. Elsevier, 2004. 58

[42] Martin Grohe, Ken-ichi Kawarabayashi, Dániel Marx, and Paul Wollan. Finding topological subgraphs is fixed-parameter tractable. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 479–488, 2011. URL: http://doi.acm.org/10.1145/1993636.1993700, doi:10.1145/1993636.1993700. 38

[43] Petr Hlinený, Sang-il Oum, Detlef Seese, and Georg Gottlob. Width parameters beyond tree-width and their applications. *Comput. J.*, 51(3):326–362, 2008. URL: https://doi.org/10.1093/comjnl/bxm052, doi:10.1093/comjnl/bxm052. 5, 6

[44] Ashwin Jacob, Jari J. H. de Kroon, Diptapriyo Majumdar, and Venkatesh Raman. Parameterized complexity of deletion to scattered graph classes. *CoRR*, abs/2105.04660, 2021. URL: https://arxiv.org/abs/2105.04660, arXiv:2105.04660. 7

[45] Ashwin Jacob, Diptapriyo Majumdar, and Venkatesh Raman. Parameterized complexity of deletion to scattered graph classes. In Yixin Cao and Marcin Pilipczuk, editors, *15th International Symposium on Parameterized and Exact Computation, IPEC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 180 of *LIPIcs*, pages 18:1–18:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. URL: https://doi.org/10.4230/LIPIcs.IPEC.2020.18, doi:10.4230/LIPIcs.IPEC.2020.18. 7

[46] Bart M. P. Jansen and Jari J. H. de Kroon. FPT algorithms to compute the elimination distance to bipartite graphs and more. *CoRR (to appear in the proceedings fo WG 2021)*, abs/2106.04191, 2021. URL: https://arxiv.org/abs/2106.04191, arXiv:2106.04191. 2, 5, 15

[47] Bart M. P. Jansen, Jari J. H. de Kroon, and Michal Wlodarczyk. Vertex deletion parameterized by elimination distance and even less. *CoRR*, abs/2103.09715, 2021. arXiv:2103.09715. 2

[48] Bart M. P. Jansen, Jari J. H. de Kroon, and Michal Wlodarczyk. Vertex deletion parameterized by elimination distance and even less. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1757–1769. ACM, 2021. URL: https://doi.org/10.1145/3406325.3451068, doi:10.1145/3406325.3451068. 2, 3, 5, 6, 7, 8, 9, 11, 12, 32, 61

[49] Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. A near-optimal planarization algorithm. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1802–1811. SIAM, SIAM, 2014. doi:10.1137/1.9781611973402.130. 5

[50] Bart M. P. Jansen and Marcin Pilipczuk. Approximation and kernelization for chordal vertex deletion. *SIAM J. Discret. Math.*, 32(3):2258–2301, 2018. doi:10.1137/17M112035X. 57

[51] Bart M. P. Jansen, Venkatesh Raman, and Martin Vatshelle. Parameter ecology for feedback vertex set. *Tsinghua Science and Technology*, 19(4):387–409, 2014. Special Issue dedicated to Jianer Chen. doi:10.1109/TST.2014.6867520. 8

[52] Naonori Kakimura, Ken-ichi Kawarabayashi, and Yusuke Kobayashi. Erdös-pósa property and its algorithmic applications: parity constraints, subset feedback set, and subset packing. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1726–1736. SIAM, 2012. 36

[53] Ken-ichi Kawarabayashi and Mikkel Thorup. The minimum k-way cut of bounded size is fixed-parameter tractable. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 160–169, 2011. URL: http://dx.doi.org/10.1109/FOCS.2011.53, doi:10.1109/FOCS.2011.53. 38

[54] 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 Trans. Algorithms*, 12(2):21:1–21:41, 2016. URL: https://doi.org/10.1145/2797140, doi:10.1145/2797140. 5

[55] C Lekkeikerker and J Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962. 53

[56] Alexander Lindermayr, Sebastian Siebertz, and Alexandre Vigny. Elimination distance to bounded degree on planar graphs. In Javier Esparza and Daniel Král', editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 65:1–65:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. URL: https://doi.org/10.4230/LIPIcs.MFCS.2020.65, doi:10.4230/LIPIcs.MFCS.2020.65. 2

[57] Daniel Lokshtanov, Pranabendu Misra, M. S. Ramanujan, and Saket Saurabh. Hitting selected (odd) cycles. *SIAM J. Discret. Math.*, 31(3):1581–1615, 2017. 36

[58] Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Trans. Algorithms*, 11(2):15:1–15:31, 2014. URL: https://doi.org/10.1145/2566616, doi:10.1145/2566616. 5

[59] Daniel Lokshtanov, M. S. Ramanujan, and Saket Saurabh. The half-integral erdös-pósa property for non-null cycles. *CoRR*, abs/1703.02866, 2017. URL: http://arxiv.org/abs/1703.02866, arXiv:1703.02866. 58

[60] Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Reducing CMSO model checking to highly connected graphs. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech*

Republic, volume 107 of *LIPIcs*, pages 135:1–135:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. URL: https://doi.org/10.4230/LIPIcs.ICALP.2018.135, doi:10.4230/LIPIcs.ICALP.2018.135. 13, 14, 42

[61] Diptapriyo Majumdar and Venkatesh Raman. Structural parameterizations of undirected feedback vertex set: FPT algorithms and kernelization. *Algorithmica*, 80(9):2683–2724, 2018. doi:10.1007/s00453-018-0419-4. 8

[62] Dániel Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006. doi:10.1016/j.tcs.2005.10.007. 36

[63] Dániel Marx. Chordal deletion is fixed-parameter tractable. *Algorithmica*, 57(4):747–768, 2010. doi:10.1007/s00453-008-9233-8. 5, 56, 57, 58

[64] Jaroslav Nesetril and Patrice Ossona de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *Eur. J. Comb.*, 27(6):1022–1041, 2006. doi:10.1016/j.ejc.2005.01.010. 1

[65] Sang-il Oum. Rank-width and vertex-minors. *J. Comb. Theory, Ser. B*, 95(1):79–100, 2005. URL: https://doi.org/10.1016/j.jctb.2005.03.003, doi:10.1016/j.jctb.2005.03.003. 5

[66] Sang-il Oum. Approximating rank-width and clique-width quickly. *ACM Trans. Algorithms*, 5(1):10:1–10:20, 2008. URL: https://doi.org/10.1145/1435375.1435385, doi:10.1145/1435375.1435385. 5

[67] Sang-il Oum. Rank-width and well-quasi-ordering. *SIAM J. Discret. Math.*, 22(2):666–682, 2008. URL: https://doi.org/10.1137/050629616, doi:10.1137/050629616. 5

[68] Sang-il Oum. Rank-width: Algorithmic and structural results. *Discret. Appl. Math.*, 231:15–24, 2017. URL: https://doi.org/10.1016/j.dam.2016.08.006, doi:10.1016/j.dam.2016.08.006. 5, 6

[69] Sang-il Oum and Paul D. Seymour. Approximating clique-width and branch-width. *J. Comb. Theory, Ser. B*, 96(4):514–528, 2006. URL: https://doi.org/10.1016/j.jctb.2005.10.006, doi:10.1016/j.jctb.2005.10.006. 5

[70] Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004. doi:10.1016/j.orl.2003.10.009. 5, 58

[71] Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006. 5

[72] Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. An fpt-algorithm for recognizing k-apices of minor-closed graph classes. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 95:1–95:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. URL: https://doi.org/10.4230/LIPIcs.ICALP.2020.95, doi:10.4230/LIPIcs.ICALP.2020.95. 5

[73] René van Bevern, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. Measuring indifference: Unit interval vertex deletion. In Dimitrios M. Thilikos, editor, *Graph Theoretic Concepts in Computer Science - 36th International Workshop, WG 2010, Zarós, Crete, Greece, June 28-30, 2010 Revised Papers*, volume 6410 of *Lecture Notes in Computer*

Science, pages 232–243, 2010. URL: https://doi.org/10.1007/978-3-642-16926-7_22, doi:10.1007/978-3-642-16926-7_22. 5

A Problem Definitions

Chordal Vertex Deletion
Input: An undirected graph G, and a positive integer k.
Output: Does there exist a vertex subset S of size at most k such that $G-S$ is a chordal graph?

Feedback Vertex Set (FVS)
Input: An undirected graph G, and a positive integer k.
Output: Does there exist a vertex subset S of size at most k that intersects all cycles in G?

Hitting Set
Input: A universe U, a family $\mathcal{A}$ of sets over U, and an integer k.
Output: Does there exist a set $X \subseteq U$ of size at most k that has a nonempty intersection with every element of $\mathcal{A}$?

Interval Vertex Deletion
Input: An undirected graph G, and a positive integer k.
Output: Does there exist a vertex subset S of size at most k such that $G-S$ is an interval graph?

Odd Cycle Transversal (OCT)
Input: An undirected graph G, and a positive integer k.
Output: Does there exist a vertex subset S of size at most k that intersects all odd cycles in G?

Mutiway Cut
Input: An undirected graph G, a vertex subset T, and a positive integer k.
Question: Does there exist a vertex subset S of size at most k such that each connected component of $G-S$ has at most one vertex of T?

Planar Vertex Deletion
Input: An undirected graph G, and a positive integer k.
Output: Does there exist a vertex subset S of size at most k such that $G-S$ is a planar graph?

Subset Feedback Vertex Set (Subset FVS)
Input: An undirected graph G, a vertex subset T, and a positive integer k.
Output: Does there exist a vertex subset S of size at most k that intersects all cycles containing a vertex of T?

Subset Odd Cycle Transversal (Subset OCT)
Input: An undirected graph G, a vertex subset T, and a positive integer k.
Output: Does there exist a vertex subset S of size at most k that intersects all odd cycles containing a vertex of T?

Topological Minor Deletion
Input: An undirected graph G, a family of undirected graphs $\mathcal{F}$, and a positive integer k.

Output: Does there exist a vertex subset S of size at most k such that $G - S$ contains no graph from $\mathcal{H}$ as a topological minor?

Vertex Cover
Input: An undirected graph G, and a positive integer k.
Output: Does there exist a vertex subset S of size at most k such that $G - S$ is edgeless?