Uniform Kernelization Complexity of Hitting Forbidden Minors

Archontia C. Giannopoulou[*] Bart M. P. Jansen[†] Daniel Lokshtanov[‡]
Saket Saurabh[§]

Abstract

The $\mathcal{F}$-Minor-Free Deletion problem asks, for a fixed set $\mathcal{F}$ and an input consisting of a graph G and integer k, whether k vertices can be removed from G such that the resulting graph does not contain any member of $\mathcal{F}$ as a minor. It generalizes classic graph problems such as Vertex Cover and Feedback Vertex Set. This paper analyzes to what extent provably effective and efficient preprocessing is possible for $\mathcal{F}$-Minor-Free Deletion. Fomin et al. (FOCS 2012) showed that the special case Planar $\mathcal{F}$-Minor-Free Deletion (when $\mathcal{F}$ contains at least one planar graph) has a kernel of polynomial size: instances (G, k) can efficiently be reduced to equivalent instances (G', k) of size $f(\mathcal{F}) \cdot k^{g(\mathcal{F})}$ for some functions f and g. The degree g of the polynomial grows very quickly; it is not even known to be computable. Fomin et al. left open whether Planar $\mathcal{F}$-Minor-Free Deletion has kernels whose size is *uniformly polynomial*, i.e., of the form $f(\mathcal{F}) \cdot k^c$ for some universal constant c that does not depend on $\mathcal{F}$. Our results in this paper are twofold.

1. We prove that not all Planar $\mathcal{F}$-Minor-Free Deletion problems have uniformly polynomial kernels (unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$). Since a graph class has bounded treewidth if and only if it excludes a planar graph as a minor, a canonical Planar $\mathcal{F}$-Minor-Free Deletion problem is Treewidth-η Deletion: can k vertices be removed to obtain a graph of treewidth at most η? We prove that the Treewidth-η Deletion problem does not have a kernel with $\mathcal{O}(k^{\frac{\eta}{4}-\epsilon})$ vertices for any $\epsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. In fact, we prove the stronger result that even parameterized by the *vertex cover number* of the graph (a larger parameter), the Treewidth-η Deletion problem does not admit uniformly polynomial kernels unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. This resolves an open problem of Cygan et al. (IPEC 2011). It is a natural question whether further restrictions on $\mathcal{F}$ lead to uniformly polynomial kernels. However, we prove that even when $\mathcal{F}$ contains a *path*, the degree of the polynomial must, in general, depend on the set $\mathcal{F}$.
2. Since a graph class has bounded treedepth if and only if it excludes a path as a minor, a canonical $\mathcal{F}$-Minor-Free Deletion problem when $\mathcal{F}$ contains a path is Treedepth-η Deletion: can k vertices be removed to obtain a graph of treedepth at most η? We prove that Treedepth-η Deletion admits uniformly polynomial kernels: for every fixed η there is a polynomial kernel with $\mathcal{O}(k^6)$ vertices. In order to develop the kernelization we prove several new results about the structure of optimal treedepth-decompositions. These insights allow us to formulate a simple, fully explicit, algorithm to reduce the instance. As opposed to the kernels of Fomin et al. (FOCS 2012), our kernelization algorithm does not rely on "protrusion machinery", which is a source of algorithmic non-constructivity.

[*]School of Engineering and Computing Sciences, Durham University, United Kingdom. `Archontia.Giannopoulou@gmail.com`. Supported by ERC Grant Agreement nr. 267959.

[†]Department of Mathematics and Computer Science, Technical University Eindhoven, The Netherlands. `b.m.p.jansen@tue.nl`. Supported by NWO Veni grant "Frontiers in Parameterized Preprocessing" and NWO Gravity grant "Networks".

[‡]Department of Informatics, University of Bergen, Norway. `daniello@ii.uib.no`. Supported by Bergen Research Foundation grant BeHard.

[§]The Institute of Mathematical Sciences, Chennai, India. `saket@imsc.res.in`. Supported by Parameterized Approximation, ERC Starting Grant 306992.

1 Introduction

Kernelization is the subfield of parameterized and multivariate algorithmics that investigates the power of provably effective preprocessing procedures for hard combinatorial problems. While the origins of data reduction and preprocessing can be traced back far into the history of computing, the rigorous analysis of these topics developed over the last decade. In kernelization we study *parameterized problems*: decision problems where every instance x is associated with a parameter k that measures some aspect of its structure. A parameterized problem is said to admit a kernel of size $f\colon \mathbb{N} \to \mathbb{N}$ if every instance (x, k) can be reduced in polynomial time to an equivalent instance with both size and parameter value bounded by $f(k)$. For practical and theoretical reasons we are primarily interested in kernels whose size is polynomial, so-called *polynomial kernels*. The study of kernelization has recently been one of the main areas of research in parameterized complexity, yielding many important new contributions to the theory. These include general results showing that certain classes of parameterized problems have polynomial kernels, and results showing how to utilize algebra, matroid theory, and topology for data reduction [2, 3, 17, 24, 26, 28, 29, 33]. The development of a framework for ruling out polynomial kernels under certain complexity-theoretic assumptions [6, 8, 13, 21] has added a new dimension to the field and strengthened its connections to classical complexity.

One of the fundamental challenges in the area is the possibility of characterizing general classes of parameterized problems possessing a kernel of polynomial size. In other words, to obtain "kernelization meta-theorems". In general, algorithmic meta-theorems have the following form: problems definable in a certain logic admit a certain kind of algorithms on certain inputs. A typical example of a meta-theorem is Courcelle's celebrated theorem [10] which states that all graph properties definable in monadic second order logic can be decided in linear time on graphs of bounded treewidth. It seems very difficult to find a fragment of logic for which every problem expressible in this logic admits a polynomial kernel on all undirected graphs. The main obstacle in obtaining such results stems from the fact that even a simplest form of logic can formalize problems that are not even fixed parameter tractable (FPT). In graph theory, one can define a general family of problems as follows. Let $\mathcal{F}$ be a family of graphs. Given an undirected graph G and a positive integer k, is it possible to do at most k edits of G such that the resulting graph does not contain a graph from $\mathcal{F}$? Here one can define edits as either vertex/edge deletions, edge additions, or edge contraction. Similarly, one may consider containment as a subgraph, induced subgraph, or a minor. The topic of this paper is one such generic problem, namely, the $\mathcal{F}$-Minor-Free Deletion problem.

The $\mathcal{F}$-Minor-Free Deletion problem asks, for a fixed set of graphs $\mathcal{F}$ and an input consisting of a graph G and integer k, whether k vertices can be removed from G such that the resulting graph does not contain any member of $\mathcal{F}$ as a minor. It generalizes classic graph problems such as Vertex Cover, Feedback Vertex Set, and Vertex Planarization. The parameterized complexity of this general problem is well understood. By a celebrated result of Robertson and Seymour, every $\mathcal{F}$-Minor-Free Deletion problem is non-uniformly FPT. That is, for every k there is an algorithm solving the problem in time $f(k) \cdot n^3$ [37]. However, whenever $\mathcal{F}$ is given explicitly, the problem is uniformly FPT because the excluded minors for the class of graphs that are yes-instance of the $\mathcal{F}$-Minor-Free Deletion problem can by computed explicitly [1]. Thus, the $\mathcal{F}$-Minor-Free Deletion problem is an interesting subject from the kernelization perspective:

For which sets $\mathcal{F}$ does $\mathcal{F}$-Minor-Free Deletion admit a polynomial kernel?

Fomin et al. [20] studied the special case of $\mathcal{F}$-Minor-Free Deletion problem where $\mathcal{F}$ contains at least one planar graph, known as the Planar $\mathcal{F}$-Minor-Free Deletion problem. It is much more restricted than $\mathcal{F}$-Minor-Free Deletion, but still generalizes problems such as Vertex Cover and Feedback Vertex Set. These problems are essentially about deleting k vertices to get a graph of constant treewidth: graphs that exclude a planar graph H as a minor have treewidth at most $|V(H)|^{\mathcal{O}(1)}$ [9]. In fact, a graph class has bounded treewidth if and only if it excludes a planar graph as a minor. Fomin et al. [20] exploited the properties of graphs of bounded treewidth and obtained a constant factor approximation algorithm, a $2^{\mathcal{O}(k \log k)} \cdot n$ time parameterized algorithm, and—most importantly, from our perspective—a polynomial sized kernel for every Planar $\mathcal{F}$-Minor-Free Deletion problem. More precisely, they showed that Planar $\mathcal{F}$-Minor-Free Deletion admits a kernel of size $f(\mathcal{F}) \cdot k^{g(\mathcal{F})}$ for some functions f and g. The degree g of the polynomial in the kernel size grows very quickly; it is not even known to be computable. This result is the starting point of our research.

> Does Planar $\mathcal{F}$-Minor-Free Deletion have kernels whose size is *uniformly polynomial*, of the form $f(\mathcal{F}) \cdot k^c$ for a universal constant c that does not depend on $\mathcal{F}$?

We prove that some Planar $\mathcal{F}$-Minor-Free Deletion problems *do not* have uniformly polynomial kernels (unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$). Since a graph class has bounded treewidth if and only if it excludes a planar graph as a minor, a canonical Planar $\mathcal{F}$-Minor-Free Deletion problem is Treewidth-η Deletion: can k vertices be removed to obtain a graph of treewidth at most η? We denote by K_d and P_d a clique and path on d vertices, respectively. Our first theorem is the following lower bound result.

Theorem 1. *Let $d \geq 3$ be a fixed integer and $\epsilon > 0$. If the parameterization by solution size k of one of the problems*

1. *$\{K_{d+1}\}$-Minor-Free Deletion,*
2. *$\{K_{d+1}, P_{4d}\}$-Minor-Free Deletion, and*
3. *Treewidth-$(d-1)$ Deletion*

admits a compression of bitsize $\mathcal{O}(k^{\frac{d}{2}-\epsilon})$, or a kernel with $\mathcal{O}(k^{\frac{d}{4}-\epsilon})$ vertices, then $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. In fact, even if the parameterization by the size x of a vertex cover of the input graph admits a compression of bitsize $\mathcal{O}(x^{\frac{d}{2}-\epsilon})$ or a kernel with $\mathcal{O}(x^{\frac{d}{4}-\epsilon})$ vertices, then $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

Theorem 1 shows that the kernelization result of Fomin et al. [20] is tight in the following sense: the degree g of the polynomial in the kernel sizes for Planar $\mathcal{F}$-Minor-Free Deletion must depend on the family $\mathcal{F}$. In fact, the theorem gives the stronger result that even parameterized by the *vertex cover number* of the graph (a larger parameter), the Treewidth-η Deletion problem does not admit uniformly polynomial kernels unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. This resolves an open problem of Cygan et al. [11]. As observed earlier, a graph class has bounded treewidth if and only if it excludes a planar graph as a minor. Thus, by restricting the $\mathcal{F}$-Minor-Free Deletion problem to those $\mathcal{F}$ that contain a planar graph, one exploits the properties of graphs of bounded treewidth to design polynomial kernels for Planar $\mathcal{F}$-Minor-Free Deletion. It is a natural question whether further restrictions on $\mathcal{F}$ lead to uniformly polynomial kernels. However, the second item of Theorem 1 shows that even when $\mathcal{F}$ contains a *path*, the degree of the polynomial must, in general, depend on the set $\mathcal{F}$. This raises the question whether there are any general families of $\mathcal{F}$-Minor-Free Deletion problems that admit uniformly polynomial kernels.

Excluding planar minors results in graphs of bounded treewidth [36]; excluding forest minors results in graphs of bounded pathwidth [35]; and excluding path minors results in graphs of bounded treedepth [31]. Since a graph class has bounded treedepth if and only if it excludes a path as a minor, a canonical $\mathcal{F}$-Minor-Free Deletion problem when $\mathcal{F}$ contains a path is Treedepth-η Deletion.

TREEDEPTH-η DELETION	**Parameter:** k
Input: An undirected graph G and a positive integer k.	
Question: Does there exist a subset $Z \subseteq V(G)$ of size at most k such that $\mathbf{td}(G - Z) \leq \eta$?	

Here $\mathbf{td}(G)$ denotes the treedepth of a graph G. The set Z is called a *treedepth-η modulator* of G. Surprisingly, we show that TREEDEPTH-η DELETION admits uniformly polynomial kernels. More precisely, we obtain the following theorem.

Theorem 2. TREEDEPTH-η DELETION *admits a kernel with* $2^{\mathcal{O}(\eta^2)}k^6$ *vertices.*

We prove several new results about the structure of optimal treedepth decompositions and exploit this to obtain the desired kernel for TREEDEPTH-η DELETION. Unlike the kernelization algorithm of Fomin et al. [20], our kernel is completely explicit. It does not use the machinery of protrusion replacement, which was introduced to the context of kernelization by Bodlaender et al. [3] and has subsequently been applied in various scenarios [17, 19, 22, 25]. Using protrusion replacement one can prove that kernelization algorithms exist, but the technique generally does not explicitly give the algorithm nor a concrete size bound for the resulting kernel.

Techniques. The kernelization lower bound of Theorem 1 is obtained by reduction from EXACT d-UNIFORM SET COVER, parameterized by the number of sets in the solution. Existing lower bounds exist for these problems due to Dell and Marx [12] and Hermelin and Wu [23], showing that the degree of the kernel size must grow linearly with the cardinality d of the sets in the input. While the construction that proves Theorem 1 is relatively simple in hindsight, the fact that the construction applies to all three mentioned problems, and also applies to the parameterization by vertex cover number, makes it interesting.

Our main technical contribution lies in the kernelization algorithm for TREEDEPTH-η DELETION. Our algorithm starts by enriching the graph G by adding edges between vertices that are connected by many internally vertex-disjoint paths. Like in prior work on TREEWIDTH-η DELETION [11], adding such edges does not change the answer to the problem. We then apply an algorithm by Reidl et al. [34] to compute an approximate treedepth-η modulator S of the resulting graph. The remainder of the algorithm strongly exploits the structure of the bounded-treedepth graph $G - S$. By combining separators for vertices that are not linked through many disjoint paths, we compute a small set Y such that all the bounded-treedepth connected components of $G - (S \cup Y)$ have a special structure: their neighborhood in S forms a clique, while they have less than η neighbors in Y. For such components C we can prove that optimal treedepth-η modulators contain at most 2η vertices from C. This important fact allows us to infer that optimal solutions cannot disturb the structure of the graph $G[C]$ too much. Using ideas inspired by earlier work on PATHWIDTH [7], it is relatively easy to bound the number of connected components of $G - (S \cup Y)$. The main work consists of reducing the size of each such component.

We formulate three lemmata that analyze under which circumstances the structure of optimal treedepth-η modulators is preserved when adding edges, removing edges, and removing vertices of the graph. By exploiting the fact that the solution size within a particular part C of the graph is constant, these lemmata ensure that even after deleting an optimal modulator from C, the remainder of C forces a structure of treedepth decompositions of the remaining graph that is compatible with the graph modifications. Of particular interest is the lemma showing that if v dominates the neighborhood of component C, then edges of v into the component may be safely discarded if certain other technical conditions are met.

The three described lemmata are the main tool in the reduction algorithm. To shrink components of $G - (S \cup Y)$ we have to add some edges, while removing other edges, to create settings where vertices can be removed from the instance without changing its answer. The fact that we have to combine edge additions and removals makes our reduction algorithm quite delicate: we cannot simply formulate reduction rules for adding and removing edges and apply them exhaustively, as they would work against each other. We therefore present a recursive algorithm that processes a treedepth-η decomposition of $G - S$ from top to bottom, making suitable transformations that bound the degree of the modulator S into the remainder of the component C. Using a careful measure expressed in terms of this degree, we can then prove that our algorithm achieves the desired size reduction.

Related Results. PLANAR $\mathcal{F}$-MINOR-FREE DELETION has received considerable attention recently. To start with, Fomin et al. [20] gave a $2^{\mathcal{O}(k)} \cdot n$-time parameterized algorithm for a variant of PLANAR $\mathcal{F}$-MINOR-FREE DELETION where every graph in $\mathcal{F}$ is connected. Kim et al. [25] showed that PLANAR $\mathcal{F}$-MINOR-FREE DELETION has an FPT algorithm with running time $2^{\mathcal{O}(k)} \cdot n^2$ time. They also showed, among many other results, that PLANAR $\mathcal{F}$-MINOR-FREE DELETION has linear kernel on topological-minor-free graphs. Cygan et al. [11] studied the TREEWIDTH-η DELETION problem parameterized by the vertex cover number of a graph and obtained a kernel of size $k^{\mathcal{O}(\eta)}$. In a later paper, Fomin et al. [18] studied $\mathcal{F}$-MINOR-FREE DELETION parameterized by the vertex cover number of the graph. They obtained kernels of size $k^{\mathcal{O}(\Delta(\mathcal{F}))}$, where $\Delta(\mathcal{F})$ is an upper bound on the maximum degree of any graph in $\mathcal{F}$.

Recently, treedepth has been the focus of several works. Reidl et al. [34] gave an algorithm with running time $2^{\mathcal{O}(t^2)} \cdot n$ to test whether the treedepth of graph is at most t. Gajarský et al. [22] obtained meta-theorems for kernelization when parameterized by a treedepth-η modulator. They showed, for example, that problems satisfying certain technical conditions admit linear kernels on hereditary graphs of bounded expansion when parameterized by the size of a treedepth-η modulator.

2 Preliminaries

For a finite set X and non-negative integer n we use $\binom{X}{n}$ to denote the collection of size-n subsets of X. We abbreviate $\{1, \ldots, n\}$ by $[n]$.

2.1 Graphs

All graphs we consider are finite, undirected, and simple. For a graph G we use $V(G)$ to denote the vertex set and $E(G)$ to denote the edge set, which is a subset of $\binom{V(G)}{2}$. For graphs G and H we write $H \subseteq G$ if H is a subgraph of G, i.e., if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. For $S \subseteq V(G)$ we denote by $G - S$ the graph obtained from G after removing the vertices of S and their incident edges. In the case where $S = \{u\}$, we abuse notation and write $G - u$ instead of $G - \{u\}$. We denote by $G[S]$ the subgraph of G induced by the set S. For $S \subseteq V(G)$, the *open neighborhood* of S in G, denoted $N_G(S)$, is the set $\{u \in V(G) \setminus S \mid \exists v \in S \colon \{u, v\} \in E(G)\}$. Again, in the case where $S = \{v\}$ we abuse notation and write $N_G(v)$ instead of $N_G(\{v\})$. The *closed neighborhood* of S in G, denoted $N_G[S]$ is defined as $N_G(S) \cup S$. Similarly, $N_G[v] := N_G(v) \cup \{v\}$ for single vertices v. The degree of a vertex $v \in V(G)$, denoted by $\deg_G(v)$, is $\deg_G(v) = |N_G(v)|$. Given two distinct vertices u and v we define $\lambda_G(u, v)$ as the maximum cardinality of a set of pairwise internally vertex-disjoint uv-paths in G.

2.1.1 Treedepth

A *rooted tree* T is a tree with one distinguished vertex $r \in V(T)$, called the *root* of T. A *rooted forest* is a disjoint union of rooted trees. The roots introduce natural parent-child and ancestor-descendant relations between vertices in forest. Let x, y be vertices of a rooted forest F. The vertex x is an *ancestor* of y if x belongs to the path linking y to the root of the tree to which y belongs. It is a *proper ancestor* if, in addition, it is not equal to y. We denote by $\mathbf{anc}_F(x)$ the proper ancestors of x. Observe that this set is empty if x is the root of a tree. We may omit the index F if it is clear from the context. Vertex y is a *descendant* of x, if x is an ancestor of y. A *proper descendant* of x is a descendant that is not x itself. We denote by $\pi(x)$ the parent of x in F. The parent of the root of the tree is $\perp$. Vertices whose parent is x are called the *children* of y.

For a rooted forest F and a vertex $v \in V(F)$, we denote by F_v the subtree rooted at v that contains all v's descendants, including v itself. The *depth* of a vertex x in a rooted forest F is the number of vertices on the unique simple path from x to the root of the tree to which x belongs. It is denoted by $\mathbf{depth}(x, F)$. The *height* of v is the maximum number of vertices on a simple path from v to a leaf in F_v. The height of F is the maximum height of a vertex of F and is denoted by $\mathbf{height}(F)$. Given a rooted forest F and a vertex $v \in V(F)$ we define the *reach* of v in F as

$$\mathbf{reach}(v, F) := \mathbf{height}(F) - \mathbf{depth}(v, F).$$

Intuitively, the reach of v shows the maximum height of a subtree that we can attach as a child of v without increasing the total height of the decomposition. Two vertices x and y are in *ancestor-descendant* relation if x is an ancestor of y or vice versa.

Definition 1 (Treedepth). *A treedepth decomposition of a graph G is a rooted forest F on the vertex set $V(G)$ (i.e., $V(G) = V(F)$) such that for every edge $\{u, v\}$ of G, the endpoints u and v are in ancestor-descendant relation. The* treedepth *of G, denoted $\mathbf{td}(G)$, is the least $d \in \mathbb{N}$ such that there exists a treedepth decomposition F of G with $\mathbf{height}(F) = d$.*

We say that an edge $\{p, q\}$ is represented in a treedepth decomposition if p and q are in ancestor-descendant relation. Observe that the treedepth of a disconnected graph is the maximum treedepth of its connected components.

Observation 2.1. *Let G be a graph and $S \subseteq V(G)$ such that $G[S]$ is clique. If F is a treedepth decomposition of G, then all the vertices of S belong to a root-to-leaf path of a tree T of F.*

Observation 2.2. *Let G be a graph with treedepth decomposition F and let H be a connected subgraph of G. All vertices of G belong to the same tree T in F. If $u, v \in V(H)$ are not in ancestor-descendant relation in T, then some vertex of H is a common ancestor of u and v.*

Observation 2.3. *If F is a treedepth decomposition of $G - S$ for some $S \subseteq V(G)$ and v is a node in a tree T of F, then all vertices of $N_G(T_v)$ are ancestors of v or belong to $N_G(T_v) \cap S$.*

We will work with the notion of a *nice treedepth decomposition*. A treedepth decomposition F of a graph G is a nice treedepth decomposition if, for every $v \in V(F)$, the subgraph of G induced by the vertices in F_v is connected.

Lemma 2.1 ([34]). *For every fixed η there is a polynomial-time algorithm that, given a graph G, either determines that $\mathbf{td}(G) > \eta$ or computes a nice treedepth decomposition F of G of depth $\mathbf{td}(G)$.*

Proposition 2.1. *For any treedepth decomposition F of a graph G, there exists a nice treedepth decomposition F^* of G whose height does not exceed the height of F, such that no vertex has greater depth in F^* than in F.*

Proof. Let F be a treedepth decomposition of G. While there is a node $u \in F$ that is not a root and no vertex of F_u is adjacent in G to $\pi(u)$, do the following. If $\pi(u)$ is not a root, then remove the edge in F from u to $\pi(u)$ and make u a child of $\pi(\pi(u))$. If $\pi(u)$ is a root, then remove the edge from u to $\pi(u)$ and make u the root of the resulting tree in F. It is easy to see that, since no vertex in F_u was adjacent to $\pi(u)$, we do not violate the validity of the treedepth decomposition. It is also easy to see that the depth of vertices does not increase.

If the operation cannot be applied anymore, then for every $u \in F$ that is not a root, the subtree F_u contains a vertex adjacent to $\pi(u)$. A simple induction on the height of u then shows that $G[F_u]$ is connected for every u, implying that F is a nice treedepth decomposition. $\square$

We use a known approximation algorithm for Treedepth-η Deletion in the kernelization.

Lemma 2.2 ([22, Lemma 2])**.** *Fix $\eta \in \mathbb{N}$. Given a graph G, one can in polynomial time compute a subset $S \subseteq V(G)$ such that $\mathbf{td}(G - S) \leq \eta$ and $|S|$ is at most 2^η times the size of a minimum treedepth-η modulator of G.*

2.2 Parameterized complexity and kernelization

In this section we define the concepts needed to prove lower bounds on kernelization. It will be convenient to consider lower bounds against compressions into small instances of arbitrary problems, rather than merely compressions of one parameterized problem into itself (which is a kernelization).

Definition 2.1 (Compression)**.** *Let $\mathcal{Q}, \mathcal{Q}' \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems and let $f \colon \mathbb{N} \to \mathbb{N}$ be a function. A* size-f compression of $\mathcal{Q}$ into $\mathcal{Q}'$ *is an algorithm that, given an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, takes time polynomial in $|x| + k$ and outputs an instance $(x', k') \in \Sigma^* \times \mathbb{N}$ such that:*

1. *$(x, k) \in \mathcal{Q}$ if and only if $(x', k') \in \mathcal{Q}'$, and*
2. *both $|x'|$ and k' are bounded by $f(k)$.*

We say that $\mathcal{Q}$ has a compression of size f if there is a parameterized problem $\mathcal{Q}'$ for which there exists a size-f compression of $\mathcal{Q}$ into $\mathcal{Q}'$. A kernelization of size f *for a parameterized problem $\mathcal{Q}$ is simply a compression of $\mathcal{Q}$ into $\mathcal{Q}$.*

To transfer lower bounds from one problem to another, we use the following type of reducibility.

Definition 2.2 (Polynomial-parameter transformation)**.** *Let $\mathcal{Q}, \mathcal{Q}' \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems and let $d \in \mathbb{N}$. A* degree-d polynomial-parameter transformation *from $\mathcal{Q}$ to $\mathcal{Q}'$ is an algorithm that, given an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, takes time polynomial in $|x| + k$ and outputs an instance $(x', k') \in \Sigma^* \times \mathbb{N}$ such that:*

1. *$(x, k) \in \mathcal{Q}$ if and only if $(x', k') \in \mathcal{Q}'$, and*
2. *$k' \in \mathcal{O}(k^d)$.*

Proposition 2.2 shows how to obtain a compression from a polynomial-parameter transformation.

Proposition 2.2. *Let $\mathcal{Q}$ and $\mathcal{Q}'$ be parameterized problems, and let $c, d \in \mathbb{N}$. If there is a degree-d polynomial-parameter transformation from $\mathcal{Q}$ to $\mathcal{Q}'$, and problem $\mathcal{Q}'$ has a compression of size $\mathcal{O}(k^c)$, then $\mathcal{Q}$ has a compression of size $\mathcal{O}(k^{c \cdot d})$.*

Proof. The compression algorithm for $\mathcal{Q}$ works as follows. On input (x, k), it first applies the polynomial-parameter transformation to compute an equivalent instance (x', k') of $\mathcal{Q}'$ whose parameter value k' is $\mathcal{O}(k^d)$. It then applies the compression for $\mathcal{Q}'$ to the instance (x', k'), resulting in an instance (x^*, k^*). By the guarantee of the compression, the size and parameter of the compressed instance are bounded by $\mathcal{O}((k^d)^c) \in \mathcal{O}(k^{c \cdot d})$. Since both steps preserve the answer and run in polynomial time, this is a valid compression for $\mathcal{Q}$. $\square$

For further background on parameterized complexity and kernelization we refer to one of the textbooks [14, 16, 32] or recent surveys [5, 27, 30].

3 Kernelization Lower Bounds

We turn our attention to kernelization and compression lower bounds. To prove that $\mathcal{F}$-Minor-Free Deletion does not have uniformly polynomial kernels for suitable families $\mathcal{F}$, we give a polynomial-parameter transformation from a problem for which a compression lower bound is known. The following problem is the starting point for our transformation.

Exact d-Uniform Set Cover **Parameter:** The universe size n.
Input: A finite set U of size n, an integer k, and a set family $\mathcal{F} \subseteq 2^U$ of size-d subsets of U.
Question: Is there a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ consisting of at most k sets such that every element of U is contained in exactly one subset of $\mathcal{F}'$?

Observe that since all subsets in $\mathcal{F}$ have size exactly d, the requirement that each universe element is contained in exactly one subset in $\mathcal{F}'$ implies that a set $\mathcal{F}'$ can only be a solution if it consists of n/d subsets. This implies that $k = n/d$ for all nontrivial instances of the problem. Hermelin and Wu [23] obtained a compression lower bound for Exact d-Uniform Set Cover. The same problem was also studied by Dell and Marx [12] under the name Perfect d-Set Matching. They obtained a slightly stronger compression lower bound, which forms the starting point for our reduction.

Theorem 3 ([12, Theorem 1.2])**.** *For every fixed $d \geq 3$ and $\epsilon > 0$, there is no compression of size $\mathcal{O}(k^{d-\epsilon})$ for* Exact d-Uniform Set Cover *unless* NP $\subseteq$ coNP/poly.

We remark that, while Dell and Marx stated their main theorem in terms of kernelizations, the same lower bounds indeed hold for compressions. This follows from the fact that the lower bound machinery on which their result is based holds for arbitrary compressions, rather than just kernelizations (see [13]). Hermelin and Wu explicitly mention that their (slightly weaker) lower bound also holds against compressions [23, §1.1].

3.1 The construction

We present the construction that will be used to prove that various families of $\mathcal{F}$-Minor-Free Deletion problems do not admit uniformly polynomial kernels. We start by giving some simple propositions that will be used in the correctness proof of the construction. Recall that a vertex v of a graph G is *simplicial* if $N_G(v)$ is a clique. We use $\mathbf{tw}(G)$ to denote the treewidth of a graph G.

Proposition 3.1 (cf. [8, Rule 3.1])**.** *If G is a graph and v is a simplicial vertex of G of degree at most $d-1$, then $\mathbf{tw}(G) \leq d-1$ if and only if $\mathbf{tw}(G - \{v\}) \leq d-1$.*

Since a d-vertex graph has treewidth at most $d-1$, Proposition 3.1 implies the following.

Proposition 3.2. *If G is a graph and $Z \subseteq V(G)$ is a set of size d such that all vertices of $V(G) \setminus Z$ are simplicial and have degree at most $d-1$, then $\mathbf{tw}(G) \leq d-1$.*

Now we give the construction and prove its correctness.

Lemma 3.1. *For every fixed d there is a polynomial-time algorithm that, given a set U of size n, an integer k, and a d-uniform set family $\mathcal{F} \subseteq \binom{U}{d}$, computes a graph G' with vertex cover number $\mathcal{O}(k^2)$ and an integer $k' \in \mathcal{O}(k^2)$, such that:*

1. *If there is a set $S' \subseteq V(G')$ of size at most k' such that $G' - S'$ is K_{d+1}-minor-free, then there is an exact set cover of U consisting of k sets from $\mathcal{F}$.*
2. *If there is an exact set cover of U consisting of k sets from $\mathcal{F}$, then there is a set $S' \subseteq V(G')$ of size at most k' such that $G' - S'$ is K_{d+1}-minor-free, P_{4d}-minor-free, and has treewidth at most $d-1$.*

Proof. Given U of size n, the integer k, and the d-uniform set family $\mathcal{F}$ the algorithm proceeds as follows. If $k \neq n/d$ then, since precisely n/d different d-size sets are needed to exactly cover U, no exact set cover with k sets exists. We may then output $G' := K_{d+1}$ and $k' := 0$, so we focus on the case that $k = n/d$. The main idea behind the construction is to create an $n \times k$ matrix with one vertex per cell. Each one of the k columns contains n vertices that correspond to the n universe elements. By turning columns into cliques and adding small gadgets, we will ensure that solutions to the vertex deletion problem must take the following form: they delete all vertices of the matrix except for exactly d per column. By enforcing that from each row, all vertices but one are deleted, and that the d surviving vertices in a column form a subset in $\mathcal{F}$, we relate the minor-free deletion sets to solutions of the exact covering problem.

The formal construction proceeds as follows. Without loss of generality we can assume that the universe U consists of $[n] = \{1, 2, \ldots, n\}$, which simplifies the exposition.

1. Initialize G' as the graph consisting of $n \times k$ vertices $v_{i,j}$ for $i \in [n]$ and $j \in [k]$. For each column index $j \in [k]$ turn the vertex set $\{v_{i,j} \mid i \in [n]\}$ into a clique. We refer to $M := \{v_{i,j} \mid i \in [n], j \in [k]\}$ as the *matrix vertices.*
2. For every row index $i \in [n]$ add a dummy clique D_i consisting of $d-1$ vertices to G'. Make all vertices in D_i adjacent to vertices $\{v_{i,j} \mid j \in [k]\}$ of the i-th row.
3. As the last step we encode the set family $\mathcal{F}$ into the graph. For every set $X \in \binom{U}{d} \setminus \mathcal{F}$, which is a size-$d$ subset of $[n]$ that is not in the set family $\mathcal{F}$, we do the following. For each column index $j \in [k]$, we create an *enforcer vertex* $f_{j,X}$ for the set X into column j. The neighborhood of $f_{j,X}$ consists of the d vertices $\{v_{i,j} \mid i \in X\}$, i.e., the vertices in column j corresponding to set X.

Observation 3.1. *All vertices of $V(G') \setminus M$ are simplicial in G': their neighborhood is a clique.*

Observation 3.2. *The set $M \cup (\bigcup_{i \in n} D_i)$ is a vertex cover of G' of size $n(k+d) \in \mathcal{O}(k^2)$.*

This concludes the construction of G'. It is easy to see that it can be performed in polynomial time for fixed d, since G' has $\mathcal{O}(n^{d+1})$ vertices. Define $k' := k(n-d)$. Since d is fixed we may absorb it into the $\mathcal{O}$-notation. As $n = kd$ this implies $k' \in \mathcal{O}(k^2)$. We prove that this choice of G' and k' satisfies the two statements in the lemma.

(1) Suppose that there is a set $S' \subseteq V(G')$ of size at most k' such that $G' - S'$ is K_{d+1}-minor-free. The following claim shows that S' intersects M in a very specific way.

Claim 3.1. *The set S' is a subset of M that contains exactly $k-1$ vertices from each row and $n-d$ vertices from each column.*

Proof. Since each of the k columns of M induces a clique in G, S' avoids at most d vertices in each column. Since each column contains n vertices, this implies that S' contains at least $n-d$ vertices from each column, so at least $k(n-d) = k'$ vertices from M. Hence the set S' of size k' cannot contain any other vertex and must be a subset of M. If there is a column from which S' contains more than $n-d$ vertices, then together with the $n-d$ vertices from each of the remaining $k-1$ columns the size of S' is at least $n-d+1+(k-1)(n-d) = k'+1$, contradicting the choice of S'. Hence S' contains exactly $n-d$ vertices from each column of M.

The argumentation for rows is similar, but here we also use the dummy cliques D_i for $i \in [n]$. If S' contains less than $k-1$ vertices from the i-th row of M (i.e., of $\{v_{i,j} \mid j \in [k]\}$), then two remaining vertices in row i together with D_i of size $d-1$ form a clique of size $d+1$, contradicting our choice of S'. Suppose there is a row from which S' contains more than $k-1$ vertices. Since S' also contains at least $k-1$ vertices from each of the other $n-1$ rows, this implies that the total size of S' is at least $k+(n-1)(k-1) = k+nk-n-k+1 = nk-dk+1 = k'+1$, where the last step uses the fact that $dk = n$ as observed at the beginning of the proof. Hence the set S' of size k' contains exactly one vertex from each row. ⌟

Claim 3.2. *Let $j \in [k]$ and let $X_j := \{i \mid i \in [n] \wedge v_{i,j} \notin S'\}$. Then $X_j \in \mathcal{F}$.*

Proof. By the previous claim, the set X_j has size exactly d. To see that the set family $\mathcal{F}$ indeed has a set containing the universe elements corresponding to the vertices in column j that are avoided by the deletion set S', observe the following. If $X_j \notin \mathcal{F}$ then during the construction we created an enforcer vertex f_{j,X_j} for set X_j into column j. But then vertex f_{j,X_j}, which is not contained in M and therefore not in S', forms a clique together with its d neighbors in column j. As the size of this clique is $d+1$, this contradicts the choice of S'. ⌟

Using the two claims it is easy to finish the proof. For each $j \in [k]$, define the set X_j as in the second claim. It follows that the subsets $X_1, \ldots, X_k$ are contained in $\mathcal{F}$. Since S' avoids exactly one element in each row by Claim 3.1, no universe element is contained in two different sets $X_j, X_{j'}$. To see that every universe element is contained in at least one set X_j, note that the previous argument shows that the k size-d sets $X_1, \ldots, X_k$ are pairwise disjoint. Their union therefore has size $dk = n$, which proves that all universe elements are covered.

(2) It remains to prove the second statement in the lemma. Suppose that $\mathcal{F}' \subseteq \mathcal{F}$ is an exact set cover of U. As observed above, $\mathcal{F}'$ consists of k distinct subsets $X_1, \ldots, X_k \subseteq [n]$, each of size d. We construct a deletion set $S' \subseteq V(G')$ as follows. For each column index $j \in [k]$, add the vertices $\{v_{i,j} \mid i \notin X_j\}$ to S'. Clearly the resulting set S' has size exactly $k' = (n-d)k$. Since a graph of treewidth at most $d-1$ does not contain K_{d+1} as a minor [4], it suffices to prove that $G' - S'$ has treewidth at most $d-1$ and avoids P_{4d} as a minor. Before proving these two claims, we consider the structure of the connected components of $G' - S'$. For each column $j \in [k]$ define $Z_j := \{v_{i,j} \mid i \in X_j\}$, which are precisely the vertices in column j not contained in S'. By the construction of G' they induce a clique in G', and are therefore contained in a single connected component of $G' - S'$.

Claim 3.3. *Let C_j be the connected component of $G' - S'$ containing Z_j. Then $C_j \cap M = Z_j$.*

Proof. Assume for a contradiction that C_j contains a vertex $v \in M \setminus Z_j$. Let P be a shortest path from a member of Z_j to v through C_j. Then P is an induced path. Suppose that P contains a vertex of $V(G') \setminus M$. Then P has at least three vertices and contains a vertex u of $V(G') \setminus M$ as an interior vertex. But by Observation 3.1 vertex u is simplicial in G' and therefore in $G' - S'$; this contradicts the fact that P is an induced path. We may conclude that P consists entirely of vertices of M.

Observe that by construction of G', the vertices in $N_{G'}(Z_j) \cap M$ are those in $M \setminus Z_j$ that share a row or column with a member of Z_j. By our choice of Z_j, all vertices of $M \setminus Z_j$ that are in column j together with Z_j, are contained in S' and therefore do not occur in the component C_j of $G' - S'$. Now consider vertices that share a row with Z_j. Since we constructed S' from an exact set cover of U, every element of U is contained in exactly one of the sets $X_1, \ldots, X_k$. This implies that for each $i \in [n]$ such that vertex $v_{i,j} \in Z_j$, we have $v_{i,j'} \in S'$ for $j' \neq j$. Hence all vertices not in Z_j that share a row with a member of Z_j, are contained in S' and do not occur in the connected component C_j of $G' - S'$. It follows that $N_{G'-S'}(Z_i) \cap M = \emptyset$ and therefore that P cannot be an induced path in $(G' - S')[M]$ connecting a vertex of Z_i and a vertex of $M \setminus Z_i$. Hence $C_j \cap M = Z_j$. ⌟

Claim 3.4. *Let C_j be the connected component of $G' - S'$ containing Z_j. Then all vertices of $C_j \setminus Z_j$ are simplicial in G (and therefore in C_j) and have degree less than d in C_j.*

Proof. By the previous claim, the component C_j consists of Z_j together with vertices in $V(G') \setminus M$. By Observation 3.1, all vertices of $C_j \setminus M$ are simplicial in G' and therefore in C_j. Consider a simplicial vertex $v \in C_j \setminus M$. If v is a vertex in the dummy clique for a row i ($i \in [n]$), then the neighborhood of v in C_j consists of $D_i \setminus \{v\}$ together with the single vertex in row i that is not in S'. Since D_i has size $d-1$, the degree of v is $((d-1)-1)+1 < d$. If v is not in a dummy clique, then v is an enforcer vertex for some set $X \in \binom{U}{d} \setminus \mathcal{F}$ into column j. Recall that Z_j contains the vertices of the j-th column corresponding to the set $X_j \in \mathcal{F}$ in the exact cover. Since $X_j \in \mathcal{F}$ while $X \notin \mathcal{F}$, it follows that at least one neighbor of v in G is not contained in Z_j. Hence the degree of v in C_j is strictly less than d. ⌟

The following claim summarizes our insights into the structure of $G' - S'$.

Claim 3.5. *Every connected component of $G' - S'$ is either a singleton enforcer vertex, or a component C_j for $j \in [k]$ consisting of the vertices Z_j, the dummy vertices $\bigcup_{i:v_{i,j} \in Z_j} D_i$, and the enforcer vertices into column j for sets X that intersect X_j.*

Proof. Consider a connected component C of $G'-S'$ that is not a single enforcer vertex. Then C contains at least one vertex from M. If C contains an enforcer vertex then this follows from the fact that the only neighbors of enforcer vertices are in M. If C contains a dummy vertex of clique D_i, then let j be the index of the set covering i (such that $i \in X_j$); it follows that $v_{i,j} \notin S'$ is a neighbor of the dummy vertex that is contained in M and belongs to the same connected component. Hence every C contains at least one vertex from M; let j be a column containing such a vertex. Then all of Z_j is contained in C, since Z_j forms a clique. Each vertex from Z_j is adjacent to the dummy clique in its row. Since no vertices from dummy cliques are contained in S', this shows that $\bigcup_{i:v_{i,j} \in Z_j} D_i$ is also contained in the component C of $G' - S'$. If $f_{j,X}$ is an enforcer vertex for a set X with $X \cap X_j \neq \emptyset$, then $f_{j,X}$ is a neighbor of any vertex representing a vertex in $X \cap X_j$, showing that $f_{j,X}$ is in component C. Hence C includes all the vertices mentioned in the claim. To see that C cannot include any other vertex, observe that by Claim 3.3 component C contains no other vertex of M. Since the dummy vertices for the remaining rows are only adjacent to vertices of $M \setminus Z_j$, they are not contained in component C. The same holds for enforcer vertices into columns that are not j. Finally, it is easy to verify that enforcer vertices into column j for sets X that are disjoint from X_j form singleton connected components of $G' - S'$ and are not included in C. This proves the claim. ⌟

Let us prove that $G' - S'$ has treewidth at most $d-1$. Since singleton graphs have treewidth zero, by the previous claim it suffices to bound the treewidth of components C_j of the form described in the claim. By Claim 3.4 all vertices of $C_j \setminus Z_j$ are simplicial in C_j and have degree less than d in C_j. Since the set Z_j has size d, by Proposition 3.2 we now find that $\mathbf{tw}(C_j) \leq d-1$. As this bounds the treewidth of all nontrivial components of $G' - S'$, this proves that $\mathbf{tw}(G' - S') \leq d-1$ and therefore that $G' - S'$ is K_{d+1}-minor-free. It remains to prove that G' has no path minor on $4d$ vertices or more.

Claim 3.6. *No connected component of $G' - S'$ contains a simple path of $4d$ vertices.*

Proof. As the claim trivially holds for singleton components, it suffices to consider components C_j for $j \in [k]$ that contain one of the sets Z_j, as described in Claim 3.5. Consider a simple path P in C_j. Since the only neighbors of the enforcer vertices in C_j are the d vertices in Z_j, path P must visit a vertex of Z_j between visiting different enforcer vertices. Since no vertex of Z_j can be visited twice, P contains at most $d+1$ enforcer vertices. To see that P contains at most $2(d-1)$ dummy vertices, we will prove that P contains vertices from at most two different dummy cliques $D_i, D_{i'}$. To see this, observe that the vertex $v_{i,j}$ is a cut-vertex in C_j that separates the dummy clique D_i from the rest of the component C_j. Hence the path cannot enter a dummy clique, visit some vertices, and then exit to the rest of the component. Hence at most two dummy cliques can be visited by P, one containing the starting point of P and one containing the endpoint of P. So indeed P contains at most $2(d-1)$ dummy vertices. By Claim 3.5 the only vertices in C_j are those of dummy cliques, the enforcer vertices, and the set Z_j. Since P contains at most $2(d-1)$ dummies, $d+1$ enforcers, and the set Z_j has size d, it follows that the simple path P contains at most $2(d-1)+(d+1)+d = 4d-1$ vertices. ⌟

Since G has a path on d vertices as a minor if and only if it contains a path on d vertices as a subgraph, Claim 3.6 proves that $G' - S'$ is P_{4d}-minor-free. This concludes the proof of Lemma 3.1. □

3.2 Kernelization lower bounds

By combining the construction of Lemma 3.1 with the tools of Section 2.2 we now derive several kernelization lower bounds for $\mathcal{F}$-Minor-Free Deletion problems. Concretely, we prove Theorem 1.

Theorem. *Let $d \geq 3$ be a fixed integer and $\epsilon > 0$. If the parameterization by solution size k of one of the problems*

1. *$\{K_{d+1}\}$-Minor-Free Deletion,*
2. *$\{K_{d+1}, P_{4d}\}$-Minor-Free Deletion, and*
3. *Treewidth-$(d-1)$ Deletion*

admits a compression of bitsize $\mathcal{O}(k^{\frac{d}{2}-\epsilon})$, or a kernel with $\mathcal{O}(k^{\frac{d}{4}-\epsilon})$ vertices, then $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. In fact, even if the parameterization by the size x of a vertex cover of the input graph admits a compression of bitsize $\mathcal{O}(x^{\frac{d}{2}-\epsilon})$ or a kernel with $\mathcal{O}(x^{\frac{d}{4}-\epsilon})$ vertices, then $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

Proof. We first prove the statement about $\{K_{d+1}\}$-Minor-Free Deletion. Observe that for every fixed d, the transformation of Lemma 3.1 forms a degree-two polynomial-parameter transformation from Exact d-Uniform Set Cover to $\{K_{d+1}\}$-Minor-Free Deletion: it is a polynomial-time algorithm that maps an instance $(U, \mathcal{F}, k)$ of Exact d-Uniform Set Cover to an instance (G', k') of $\{K_{d+1}\}$-Minor-Free Deletion with $k' \in \mathcal{O}(k^2)$, and the two statements in the lemma ensure that $(U, \mathcal{F}, k)$ is a yes-instance if and only if (G', k') is a yes-instance.

Now assume that $\{K_{d+1}\}$-Minor-Free Deletion parameterized by k has a compression of size $\mathcal{O}(k^{\frac{d}{2}-\epsilon})$ for some $d \geq 3$ and $\epsilon > 0$. By Proposition 2.2, it follows that Exact d-Uniform Set Cover has a compression of size $\mathcal{O}(k^{d-2\epsilon})$. By Theorem 3 this implies $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. Observe that, since the graph constructed in Lemma 3.1 has a vertex cover of size $\mathcal{O}(k^2)$, the construction also serves as a degree-two polynomial parameter transformation from Exact d-Uniform Set Cover to the parameterization of $\{K_{d+1}\}$-Minor-Free Deletion by vertex cover. Hence the existence of a compression with bitsize $\mathcal{O}(x^{\frac{d}{2}-\epsilon})$ implies $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ by the same argument as above.

Concerning the existence of kernels with few vertices, observe that a kernelized instance with $\mathcal{O}(k^{\frac{d}{4}-\epsilon})$ vertices can be encoded in $\mathcal{O}(k^{\frac{d}{2}-2\epsilon})$ bits, by writing down the adjacency matrix of the graph and target value (which does not exceed the order of the graph) in binary. Hence a kernel with $\mathcal{O}(k^{\frac{d}{4}-\epsilon})$ or $\mathcal{O}(x^{\frac{d}{4}-\epsilon})$ vertices yields a compression that implies $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

Now consider the other two problems mentioned in the theorem. By exactly the same argumentation, it suffices to argue that the construction of Lemma 3.1 is a valid degree-two polynomial parameter transformation of Exact d-Uniform Set Cover into these problems. Let $(U, \mathcal{F}, k)$ be an instance of Exact d-Uniform Set Cover and consider the pair (G', k') constructed in the lemma. If $(U, \mathcal{F}, k)$ is a yes-instance, then by the second statement of Lemma 3.1 we can delete k' vertices from G' to make it both K_{d+1} and P_{4d} minor-free, hence (G', k') is a yes-instance of $\{K_{d+1}, P_{4d}\}$-Minor-Free Deletion. For the reverse direction, if G' can be made both K_{d+1} and P_{4d} minor-free by k' vertex deletions, then in particular it can be made K_{d+1}-minor-free by k' deletions so the first item of Lemma 3.1 proves that $(U, \mathcal{F}, k)$

is a YES-instance. So the construction is a degree-two polynomial parameter transformation to $\{K_{d+1}, P_{4d}\}$-MINOR-FREE DELETION.

Finally, consider the TREEWIDTH-$(d-1)$ DELETION problem. If $(U, \mathcal{F}, k)$ is a YES-instance then, by the second item of Lemma 3.1, the treewidth of the constructed graph G' can be reduced to at most $d-1$ by k' deletions, implying that (G', k') is a YES-instance of TREEWIDTH-$(d-1)$ DELETION. For the reverse direction, if the treewidth of G' can be reduced to at most $d-1$ by k' deletions, then since a graph of treewidth at most $d-1$ does not contain K_{d+1} as a minor (cf. [4]), it follows that G' can be made K_{d+1}-minor-free by k' deletions. By the first item of Lemma 3.1, this implies that $(U, \mathcal{F}, k)$ is a YES-instance. The construction is therefore also a degree-two polynomial parameter transformation to TREEWIDTH-$(d-1)$ DELETION, which proves the theorem. □

4 Structural results about treedepth

In this section we derive several properties of treedepth decompositions that will be needed to analyze the effect of the graph reduction steps. We start by proving some general facts about treedepth in Section 4.1. In Section 4.2 we start analyzing properties of instances of TREEDEPTH-η DELETION, introducing the notion of nearly clique separated sets to show that minimum solutions intersect certain parts of the graph in only few vertices. Finally, in Section 4.3 we present the lemmata discussed in the introduction concerning three types of graph transformations (edge additions, edge removals, and vertex removals) and derive conditions under which these do not change the answer to an instance of TREEDEPTH-η DELETION.

4.1 Properties of treedepth

The following lemma shows that either the reach of a vertex is large, or the height of the decomposition is large. It will be used to argue that, in treedepth-η decompositions of a reduced graph, the reach of a vertex is large enough to allow a deleted component of the graph to be embedded below it without increasing the total decomposition height.

Lemma 4.1. *Let d and t be positive integers, G be a connected graph, and let $H_1, H_2, \ldots, H_t$ be vertex-disjoint connected subgraphs of G with $\mathbf{td}(H_i) \geq d$. Let T be a treedepth decomposition of G. If $v \in V(G)$ such that $v \in N_G(H_i)$ for every $i \in [t]$, and $\mathbf{reach}(v, T) < d$, then $\mathbf{height}(T) \geq t + 1$.*

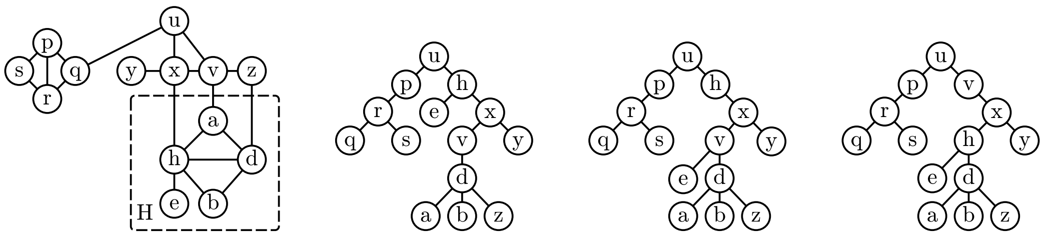
Proof. Suppose that $\mathbf{reach}(v, T) = \mathbf{height}(T) - \mathbf{depth}(v, T) < d$. It follows that the subtree T_v rooted at v has height at most d, otherwise the path from the root of T to v, and then to a deepest leaf in T_v, would contain at least $\mathbf{depth}(v, T) + (d + 1) - 1$ vertices (we subtract one because v is counted twice). Since $\mathbf{height}(T) = \mathbf{reach}(v, T) + \mathbf{depth}(v, T) < d + \mathbf{depth}(v, T)$, this would give a contradiction.

Claim 4.1. *For every $i \in [t]$ some vertex of H_i is not in T_v.*

Proof. If $H_i \subseteq T_v$ then the rooted subtree T_v is a treedepth decomposition of $G[T_v]$, a supergraph of H_i, of height d. Since vertex v is not in H_i, the rooted forest obtained by removing v is a treedepth decomposition of $G[T_v \setminus \{v\}]$ of height less than d. But then the supergraph $G[T_v \setminus \{v\}]$ of H_i has treedepth less than d, contradicting $\mathbf{td}(H_i) \geq d$. ⌟

Claim 4.2. *For every $i \in [t]$ some vertex of H_i is an ancestor of v.*

Proof. Consider some $i \in [t]$. Since $v \in N_G(H_i)$, there is a vertex u in H_i that is adjacent to v and therefore u is an ancestor of v in T, or u is in T_v. In the first case we are done. In the second case, let w be a vertex of H_i that is not in T_v, which exists by the previous claim.



(a) Graph G, $H \subseteq G$. (b) $T = T^*$. (c) $\hat{T}$. (d) $\widetilde{T}$.

Figure 1: Illustration for the proof of Lemma 4.3. 1(a) The graph G and connected subgraph H consisting of $\{a, b, h, d, e\}$ are shown. As $N_G(H) = \{x, v, z\} \subseteq N_G[v] = \{u, v, x, z, a\}$, the lemma applies. 1(b) A treedepth decomposition T for G. As it is a nice decomposition, it is also used as T^*. Vertex h is the first member of H on the path from v to the root. Vertex $z \in N_G(H)$ is a neighbor of v that is not an ancestor of v. The path P consists of (x, h). 1(c) The result of cutting off the (singleton) subtree T^*_e and attaching an minimum-height decomposition $\hat{T}^e$ for that subgraph at v. 1(d) The final decomposition T' used to invoke the induction hypothesis.

If w is an ancestor of v we are again done. If not, then u and w are vertices that are not in ancestor-descendant relation that belong to the same connected subgraph H_i of G. Hence, by Observation 2.2 there is a vertex x in H_i that is a common ancestor of u and w, and x does not belong to T_v since $w \notin T_v$. Since all ancestors of $u \in T_v$, which do not belong do T_v, are also ancestors of v, the claim follows. (Note that $x \neq v$ since $x \in H_i$.) ⌟

The claim shows that v has t ancestors unequal to v itself. Hence $\mathbf{height}(T) \geq t + 1$. □

The next lemma will be used to argue that an edge must be represented in sufficiently shallow decompositions.

Lemma 4.2. *Let q be a positive integer and G be a graph. If $u, v \in V(G)$ are joined by q internally vertex-disjoint paths and F is a treedepth decomposition of G in which u and v are not in ancestor-descendant relation, then* $\mathbf{height}(F) > q$.

Proof. Let P_u and P_v be the paths from u and v to the root, and let P be the intersection of the two paths. Since u and v are not in ancestor-descendant relation we have $u, v \notin P$. As P contains all common ancestors of u and v in F, in graph G the vertices u and v are separated by $V(P)$. From Menger's Theorem, as u and v are connected by q internally vertex-disjoint paths, it follows that $|V(P)| \geq q$. This implies that $\mathbf{height}(T) > q$. □

The following technical lemma gives conditions under which a treedepth decomposition can be modified to ensure that a vertex set $V(H)$ is embedded in the subtree below a distinguished vertex v. The updated decomposition therefore represents all possible edges between v and $V(H)$. The lemma will be crucial in the correctness proof of Lemma 4.7, which gives conditions under which edges can safely be deleted from an instance.

Lemma 4.3. *Let G be a connected graph, let $H \subseteq G$ be a connected subgraph of G, and let $v \in V(G) \setminus V(H)$ be a vertex such that $N_G(H) \subseteq N_G[v]$. For any treedepth decomposition T of G, there exists a treedepth decomposition T' of G such that:*

1. $\mathbf{height}(T') \leq \max(\mathbf{height}(T), \mathbf{depth}(v, T) + \mathbf{td}(G[V(H)]))$.
2. *All vertices of $V(H)$ belong to T'_v, the subtree of T' rooted at v.*

Proof. Let G, H, and v be as stated. We use induction on $\mathbf{depth}(v,T)$. If $\mathbf{depth}(v,T) = 1$ then v is the root of T. Since G is connected all its vertices belong to the same decomposition tree and are therefore contained in T_v. So $T' = T$ trivially satisfies the requirements. For the induction step, assume that $\mathbf{depth}(v,T) > 1$.

By Proposition 2.1 there exists a nice treedepth decomposition T^* of G whose height does not exceed the height of T, such that no vertex has greater depth in T^* than in T. If $V(H) \subseteq T^*_v$ then the lemma holds, as we may take T' equal to T^*. Assume for the remainder that $V(H) \setminus T^*_v \neq \emptyset$. Since G is connected and $V(H) \subsetneq V(G)$ (as $v \in V(G) \setminus V(H)$), the set $N_G(H)$ is not empty.

First consider the case that all vertices of $N_G(H)$ lie on the path from v to the root in T^*. Then it is easy to find a decomposition as described in the lemma: we form T' by restricting the decomposition T^* to the vertices of $V(G) \setminus V(H)$, then we take a minimum-height treedepth decomposition T^H of the graph $G[V(H)]$ and attach the root of T^H to vertex v to ensure that $H \subseteq T'_v$. From this construction it easily follows that $\mathbf{height}(T') \leq \max(\mathbf{height}(T), \mathbf{depth}(v,T) + \mathbf{td}(G[V(H)]))$. To see that all edges are represented in the model, observe that (i) all edges of $G - V(H)$ are represented because T' contains the restriction of T^* to $V(G) \setminus V(H)$, (ii) all edges of $G[V(H)]$ are represented because a valid decomposition of $G[V(H)]$ is inserted into T', while finally all edges between $V(H)$ and $V(G) \setminus V(H)$ are represented because all vertices of $N_G(H)$ are ancestors of v and therefore ancestors of every vertex in $H \subseteq T'_v$.

In the remainder we therefore focus on the case that some vertex $z \in N_G(H)$ is not an ancestor of v. Since v is an ancestor of itself, we have $z \neq v$. Observe that z is adjacent in G to v, since $z \in N_G(H) \subseteq N_G[v]$ and $z \neq v$. Since T^* is a valid treedepth decomposition, if z is not an ancestor of v, then $z \in T^*_v$.

Claim 4.3. *The path in T^* from v to the root contains a vertex of H.*

Proof. Assume for a contradiction that the path from v to the root contains no vertex of H, i.e., no ancestor of v is contained in H. Let $h \in V(H) \setminus T^*_v$, which exists by our assumption above. Let $h' \in V(H)$ be adjacent in G to z; such a vertex exists since $z \in N_G(H)$. If one of h or h' is an ancestor of v in T^* then we are done. If this is not the case, then observe that $h' \in T^*_v$: to realize its edge to $z \in T^*_v$ without being an ancestor of v, it must lie in T^*_v. Since $h \notin T^*_v$ but $h' \in T^*_v$, the only common ancestors of h and h' are ancestors of v, which are not contained in H by assumption. But by Observation 2.2, the common ancestors of h and h' separate h and h' in G. But then these common ancestors are a vertex subset of $V(G) \setminus V(H)$ that separate h and h' in G, contradicting the fact that H is a connected subgraph of G. The claim follows. ⌟

Let $h \in V(H)$ be the first vertex from H on the path from v to the root of T^*. Let P be the path from $\pi(v)$ to h in T^*. This choice of h and the fact that T^* is nice has a useful consequence. Let $c_1, \ldots, c_t$ be vertices of T^* that are unequal to v, whose parent belongs to P, and for which the subtree rooted there contains at least one vertex of $V(H)$. (It may be that there are no such vertices.)

Claim 4.4. *For each vertex c_i with $i \in [t]$, the subtree $T^*_{c_i}$ contains no vertex of $N_G(H)$.*

Proof. Suppose that $T^*_{c_i}$ contains a vertex $x \in N_G(H)$. By our definition of c_i we have $x \neq v$ and vertex c_i is not in an ancestor-descendant relation with v in T^*. Hence no descendant of c_i is in ancestor-descendant relation with v either. Since $N_G(H) \subseteq N_G[v]$ and $x \neq v$ we have $\{x, v\} \in E(G)$. However, since $x \in T^*_{c_i}$ is not in ancestor-descendant relation with v, this edge is not realized in the decomposition T^*; a contradiction. ⌟

Claim 4.5. *For each vertex c_i with $i \in [t]$ we have $T^*_{c_i} \subseteq V(H)$.*

Proof. Assume for a contradiction that $T^*_{c_i} \not\subseteq V(H)$. Since $T^*_{c_i}$ contains at least one vertex of $V(H)$ by definition of c_i, it follows that $T^*_{c_i}$ contains both a vertex $h' \in V(H)$ and a vertex $z \notin V(H)$. By the definition of a nice treedepth decomposition, the graph $G[T^*_{c_i}]$ is connected and contains a path P' from h' to z. Since h' is in $V(H)$ but z is not, it follows that $P' \subseteq T^*_{c_i}$ contains a vertex of $N_G(H)$; a contradiction to Claim 4.4. ⌟

For each $i \in [t]$ let $\hat{T}^i$ be a minimum-height treedepth decomposition of the graph $G[T^*_{c_i}]$. Since $T^*_{c_i} \subseteq V(H)$ for all $i \in [t]$ it follows that $\mathbf{height}(\hat{T}^i) \leq \mathbf{td}(G[V(H)])$ for all $i \in [t]$. We now obtain a new decomposition tree $\hat{T}$ from T^* as follows. Remove the subtrees $T^*_{c_1}, \ldots, T^*_{c_t}$ from T^*. Then add the trees $\hat{T}^1, \ldots, \hat{T}^t$ and connect the root of each of these trees to v. This results in a valid decomposition of G of height at most $\max(\mathbf{height}(T^*), \mathbf{depth}(v, T^*) + \mathbf{td}(G[V(H)]))$. To see that the decomposition is valid, observe that all edges of $G[T^*_{c_i}]$ for $i \in [t]$ are represented in the subtrees that we inserted. Since we attach the replacement trees to the descendant v of the vertex of the path P that they were originally attached to, the edges to the rest of the graph are represented as well.

As the next step, we swap the labels of h and v in $\hat{T}$ and use the resulting decomposition as $\widetilde{T}$. It is easy to see that moving v to the location of its ancestor h maintains the fact that all edges incident on v are represented. It remains to prove that all edges incident on h are still represented after the swap. For this it suffices to observe that the only vertices that were in ancestor-descendant relation with h in $\hat{T}$, but are not in ancestor-descendant relation with h in $\widetilde{T}$, are those contained in subtrees of $\hat{T}$ attached to the path P at vertices other than $c_1, \ldots, c_t$. But by our choice of $c_1, \ldots, c_t$, such subtrees contain no vertices of H. By the same argumentation as in Claim 4.4, such subtrees contain no vertex of $N_G(H)$ either. So the vertices to which ancestor-descendant relation is lost are not neighbors of h in G, which implies that $\widetilde{T}$ is indeed a valid decomposition of G.

We finish the proof by applying the induction hypothesis. Since $\widetilde{T}$ was obtained from $\hat{T}$ by swapping the labels of v and h in the tree, while $\mathbf{depth}(v, \hat{T}) > \mathbf{depth}(h, \hat{T})$ since h is an ancestor of v in $\hat{T}$, it follows that $\mathbf{depth}(v, \widetilde{T}) < \mathbf{depth}(v, \hat{T}) \leq \mathbf{depth}(v, T^*) \leq \mathbf{depth}(v, T)$. This implies that we may apply induction to G, H, v, and the decomposition $\widetilde{T}$, to conclude that there is a treedepth decomposition T' such that $V(H) \subseteq T'_v$ and $\mathbf{height}(T')$ is bounded by

$$\max(\mathbf{height}(\widetilde{T}), \mathbf{depth}(v, \widetilde{T}) + \mathbf{td}(G[V(H)])) \leq \max(\mathbf{height}(T), \mathbf{depth}(v, T) + \mathbf{td}(G[V(H)])).$$

This concludes the proof. □

4.2 Nearly clique-separated sets

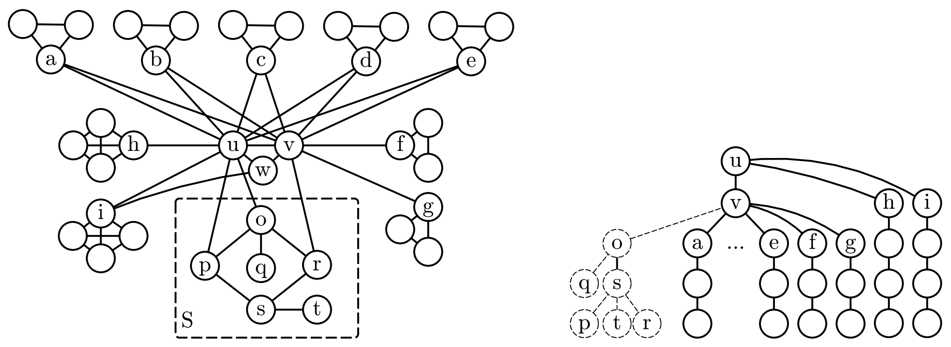
The purpose of this section is to introduce the following notion.

Definition 4.1. *Let G be a graph, let $S \subseteq V(G)$, and let ℓ be an integer. The set S is ℓ-*nearly clique separated *if there is a set $Q \subseteq N_G(S)$ of size at most ℓ such that $N_G(S) \setminus Q$ is a clique.*

Nearly clique separated sets are important because minimum treedepth modulators contain only few of their vertices, implying that the structure of a nearly clique separated subgraph does not change too much when a minimum modulator is removed.

Lemma 4.4. *Let G be a graph, let ℓ be an integer, and let $S \subseteq V(G)$ with $\mathbf{td}(G[S]) \leq \eta$. If S is ℓ-nearly clique separated, then $|Z \cap S| \leq \eta + \ell$ for any minimum treedepth-η modulator Z of G.*

Proof. Assume for a contradiction that Z' is a minimum treedepth-η modulator of G such that $|Z' \cap S| > \eta + \ell$. Let $Q \subseteq N_G(S)$ be a set of size at most ℓ such that $N_G(S) \setminus Q$ is a clique, which exists by Definition 4.1. Let $K := N_G(S) \setminus Q$. Then $G[K]$ is a clique, and therefore $|K \setminus Z'| \leq \eta$ (otherwise the $\eta + 1$ remaining vertices of the clique would cause the treedepth of $G - Z'$ to



(a) Graph G, set $S \subseteq V(G)$.

(b) Decompositions F and F'.

Figure 2: Illustration for the proof of Lemma 4.5. 2(a) Graph G with vertex set $S \subseteq V(G)$ whose neighborhood $N_G(S) = \{u, v\}$ is a clique. This is a YES-instance for treedepth-5 transversal with $k = 1$, since $\{w\}$ is a solution. Lemma 4.5 is applicable with $\ell = 2$ by choosing $X_1^v, \ldots, X_7^v$ to be the triangles containing $a, \ldots, e, f, g$ respectively. The sets $X_1^u, \ldots, X_7^u$ are the triangles containing $a, \ldots, e$ and the 4-cliques containing h and i, respectively; note that sets X_i^u may intersect sets X_j^v, but that, e.g., X_i^u and X_j^u must be disjoint. 2(b) The solid lines show a treedepth-5 decomposition for the graph $(G - S) - \{z\}$, where $\{z\}$ is a solution to the reduced instance $G - S$. The dotted lines show how to augment to a decomposition for the graph $G - \{w\}$ by attaching a minimum-width decomposition for $G[S]$ to the lowest neighbor of S.

exceed η). We claim that $Z = (Z' \setminus S) \cup (K \setminus Z') \cup Q$ is a treedepth-η modulator of G of size less than $|Z'|$.

The fact that the treedepth of $G - Z$ is at most η can be verified as follows. Observe that $\mathbf{td}(G[S]) \leq \eta$ and $N_G(S) \subseteq Z$, implying that no vertex of S is adjacent to a vertex outside of S in $G - Z$. Hence all connected components of $G - Z$ that contain a vertex of S, have treedepth at most η. On the other hand, for every connected component H of $G - Z$ with $V(H) \cap S = \emptyset$ there exists a connected component H' of $G - Z'$ such that $H \subseteq H'$ and thus, $\mathbf{td}(H) \leq \mathbf{td}(H') \leq \eta$. To see that Z is smaller than Z', observe that we remove $|Z' \cap S| > \eta + \ell$ vertices from Z', while we add $|K \setminus Z'| \leq \eta$ plus $|Q| \leq \ell$ vertices. Hence Z is a smaller treedepth-η modulator than Z', contradicting optimality of Z'. $\square$

4.3 Safe Transformations

In this section we analyze three different operations in a graph: removing vertices, removing edges, and adding edges. For each type of operation we give conditions under which the transformation provably does not change the answer to an instance (G, k) of TREEDEPTH-η DELETION. We say that instances (G, k) and (G', k) are *equivalent* if one is a YES-instance if and only if the other is. When proving that two instances are equivalent, we frequently use the fact that if G' is a minor of G and (G, k) is a YES-instance of TREEDEPTH-η DELETION, then (G', k) is a YES-instance as well. This follows from the fact that treedepth does not increase when taking minors, so that if $\mathbf{td}(G - Z) \leq \eta$ we must have $\mathbf{td}(G' - Z) \leq \eta$ since $G' - Z$ is a minor of $G - Z$.

Lemma 4.5 (Vertex removal)**.** *Let (G, k) be an instance of* TREEDEPTH-η DELETION *and let ℓ be an integer. Let $S \subseteq V(G)$ be such that $N_G(S)$ is a clique and $\mathbf{td}(G[S]) \leq \eta$. For every $v \in N_G(S)$, let $X_1^v, \ldots, X_{\ell+\eta}^v \subseteq V(G)$ induce connected subgraphs of G such that:*

1. *$\forall v \in N_G(S), \forall i \in [\ell + \eta]$: $\mathbf{td}(G[X_i^v]) \geq \mathbf{td}(G[S])$ and $v \in N_G(X_i^v)$,*
2. *$\forall v \in N_G(S)$, the sets $X_1^v, \ldots, X_{\ell+\eta}^v$ are pairwise disjoint and disjoint from S, and*

3. *$G - S$ has a minimum treedepth-η modulator containing $\leq \ell$ vertices of $\mathcal{X}$,*

where $\mathcal{X} := \bigcup_{v \in N_G(S)} \bigcup_{i \in [\ell+\eta]} X_i^v$. Then (G,k) is equivalent to the instance $(G - S, k)$.

Proof. If (G,k) is a YES-instance then $(G-S,k)$ trivially is a YES-instance as well. For the reverse direction, let Z be a minimum treedepth-η modulator of $G - S$ with $|Z \cap \mathcal{X}| \leq \ell$, which exists by assumption and has size at most k. Let F be a minimum-height treedepth decomposition of $G - Z$, which has height at most η. If $N_G(S) \setminus Z = \emptyset$ then S forms an isolated component of treedepth at most η in the graph $G - Z$, implying that $\mathbf{td}(G - Z) = \max(\mathbf{td}((G - S) - Z), \mathbf{td}(G[S])) \leq \eta$. Assume then that $N_G(S) \setminus Z$ is not empty. Since the set $N_G(S)$ is a clique in G, it follows that $N_G(S) \setminus Z$ is a clique in $G - Z$. Hence all vertices of $N_G(S) \setminus Z$ appear on one root-to-leaf path in F. Let v be the vertex of $N_G(S) \setminus Z$ of greatest depth in F.

Claim 4.6. *We have* $\mathbf{reach}(v, F) \geq \mathbf{td}(G[S])$.

Proof. Since the sets $X_1^v, \ldots, X_{\ell+\eta}^v \subseteq \mathcal{X}$ are pairwise disjoint, each of the at most ℓ vertices in $Z \cap \mathcal{X}$ intersects at most one such subset. Hence there are at least η such subsets $X_{j_1}^v, \ldots, X_{j_\eta}^v$ that are not intersected by Z. Each $X_{j_t}^v$ therefore induces a connected subgraph of $(G-S)-Z$ of treedepth at least $\mathbf{td}(G[S])$ by (1) that is adjacent in $(G - S) - Z$ to v. Now apply Lemma 4.1 to the connected component of $(G - S) - Z$ containing v and the tree T in F representing that component, using $X_{j_1}^v, \ldots, X_{j_\eta}^v$ as connected subgraphs of treedepth at least $\mathbf{td}(G[S])$. The lemma shows that if $\mathbf{reach}(v, T) < d$, then $\mathbf{height}(T) \geq \eta + 1$, contradicting the fact that $\mathbf{height}(T) \leq \mathbf{height}(F) \leq \eta$. Hence $\mathbf{reach}(v, T) \geq d$. Since F is at least as high as T, this implies $\mathbf{reach}(v, F) \geq \mathbf{td}(G[S])$. ⌟

Starting from the decomposition F of $(G - S) - Z$, we now obtain a decomposition F' of $G - S$ as follows. Take a minimum-height decomposition forest of $G[S]$, add it to F and connect the roots of all vertices in the decomposition forest to v. Since $N_G(S) \setminus Z$ appears on the path from v to the root of its tree, this results in a valid treedepth decomposition of $G - S$. Since $\mathbf{reach}(v, F) \geq \mathbf{td}(G[S])$, the height of F' equals the height of F. Hence $G - Z$ has treedepth at most η, showing that (G,k) is a YES-instance. □

Lemma 4.5 was inspired by earlier work [7, Rule 6] on PATHWIDTH. The next lemma concerns edge addition.

Lemma 4.6 (Edge addition). *Let (G,k) be an instance of* TREEDEPTH-η DELETION *and let ℓ be an integer. Let $X \subseteq V(G)$ and let $\{u,v\} \in \binom{V(G)}{2} \setminus E(G)$. If the following conditions hold:*

1. *the graph $G[X \cup \{u,v\}]$ contains at least $\ell + \eta$ internally vertex-disjoint paths between u and v, and*
2. *G has a minimum treedepth-η modulator containing $\leq \ell$ vertices of X,*

then (G,k) is equivalent to the instance $(G + \{u,v\}, k)$ obtained by adding the edge $\{u,v\}$.

Proof. If $(G + \{u,v\}, k)$ is a YES-instance then (G,k) is as well. In the other direction, suppose that (G,k) is a YES-instance and let Z be a minimum treedepth-η modulator of G with $|Z \cap X| \leq \ell$, which exists by assumption. Let F be a minimum-height treedepth decomposition of $G - Z$. If $Z \cap \{u,v\} \neq \emptyset$ then $G - Z = (G + \{u,v\}) - Z$ and so Z is a solution for $G + \{u,v\}$, proving it to be a YES-instance.

Assume then that $Z \cap \{u,v\} = \emptyset$. Since there are $\ell + \eta$ internally vertex-disjoint paths between u and v in $G[X \cup \{u,v\}]$, while Z intersects at most ℓ of them as $|Z \cap X| \leq \ell$, it follows that there are η internally vertex-disjoint paths between u and v in $G - Z$. By Lemma 4.2, these η paths prove that if u and v are not in ancestor-descendant relation in F then $\mathbf{height}(F) > \eta$, a contradiction. So u and v are in ancestor-descendant relation which shows that F is also a valid treedepth decomposition of $(G + \{u,v\}) - Z$, proving $(G + \{u,v\}, k)$ to be a YES-instance. □

Finally, we consider edge removal.

Lemma 4.7 (Edge removal)**.** *Let (G,k) be an instance of* Treedepth-η Deletion *and let ℓ be an integer. Let $S \subseteq V(G)$ and let $v \in V(G) \setminus S$ such that $N_G(S) \subseteq N_G[v]$. Let $X_1, \ldots, X_{\ell+\eta} \subseteq V(G)$ be connected subgraphs of G such that:*

1. *$\forall i \in [\ell+\eta]\colon \mathbf{td}(G[X_i]) \geq \mathbf{td}(G[S])$ and $v \in N_G(X_i)$,*
2. *the sets $X_1, \ldots, X_{\ell+\eta}$ are pairwise disjoint and disjoint from S, and*
3. *any graph obtained from G by removing edges between v and S has a minimum treedepth-η modulator containing $\leq \ell$ vertices of $\mathcal{X}$,*

where $\mathcal{X} := \bigcup_{i \in [\ell+\eta]} X_i$. Then (G,k) is equivalent to the instance (G',k), where G' is obtained from G by removing all edges between v and S.

Proof. If (G,k) is a yes-instance then its minor (G',k) is as well. In the other direction, assume that (G',k) is a yes-instance. Assume additionally that (G,k) is a no-instance; we shall argue for a contradiction. Consider the set of edges $Y \subseteq E(G)$ that were removed from G to obtain $G' = G - Y$, and observe that all edges in Y are incident on v. Let $Y' \subseteq Y$ be a *minimal* set such that $G^* := G - Y'$ is a yes-instance. Clearly Y' is not empty, as G is a no-instance. Let $\{u,v\}$ be an arbitrary edge of Y', which must have v as an endpoint. By minimality of Y' we know that $G^* + \{u,v\} = G - (Y' \setminus \{u,v\})$ is a no-instance. We will derive a contradiction by proving that $G^* + \{u,v\}$ is actually a yes-instance. By (3), graph G^* has a minimum treedepth-η modulator Z containing at most ℓ vertices of $\mathcal{X}$. Since (G^*,k) is a yes-instance, the size of Z is at most k. Let F be a minimum-height, nice treedepth decomposition of $G^* - Z$, of height at most η. If $\{u,v\} \cap Z \neq \emptyset$ then the graphs $G^* - Z$ and $(G^* + \{u,v\}) - Z$ are identical, so $\mathbf{td}(G^* + \{u,v\}) - Z \leq \eta$, proving that $G^* + \{u,v\}$ is a yes-instance. In the remainder we consider the case that $\{u,v\} \cap Z = \emptyset$.

Claim 4.7. *We have $\mathbf{reach}(v,F) \geq \mathbf{td}(G[S])$.*

Proof. The proof is similar to that of Claim 4.6. Since $X_1, \ldots, X_{\ell+\eta}$ are pairwise disjoint subsets of $\mathcal{X}$, the set Z intersects at most ℓ of them. Hence at least η of them, say $X_{j_1}, \ldots, X_{j_\eta}$, are disjoint from Z and therefore induce connected subgraphs of $G^* - Z$ of treedepth at least $\mathbf{td}(G[S])$. Since these sets are disjoint from S and they were adjacent to v in G, we have not removed their edges to v when constructing G^* and therefore $v \in N_{G^*-Z}(X_{j_t})$ for all $t \in [\eta]$. Applying Lemma 4.1 to the connected component of $G^* - Z$ containing v and the tree T in F representing that component, we find that if $\mathbf{reach}(v,T) < d$, then $\mathbf{height}(T) \geq \eta + 1$, a contradiction. Hence $\mathbf{reach}(v,F) \geq \mathbf{reach}(v,T) \geq d$. ⌟

We use the lower bound on $\mathbf{reach}(v,F)$ in the following arguments. Let S_u be the connected component of $G^*[S] - Z$ that contains u, the other endpoint of $\{u,v\}$. We first deal with an easy case.

Claim 4.8. *If $N_{G^*-Z}(S_u) = \emptyset$, then $(G^* + \{u,v\}, k)$ is a* yes*-instance.*

Proof. If $N_{G^*-Z}(S_u) = \emptyset$, then S_u forms a connected component of the graph $G^* - Z$, since this connected set has no neighbors. From the definition of a nice treedepth decomposition, there is a single tree T_u in F whose vertices are S_u. Since S_u is a subgraph of $G[S]$ we have $\mathbf{td}(G[S_u]) \leq \mathbf{td}(G[S])$. Now obtain a treedepth decomposition F' of $(G^* + \{u,v\}) - Z$ as follows. Remove the tree T_u from F, let T'_u be a minimum-height treedepth decomposition of $G[S_u]$, add this tree to F and make the root of T'_u a child of v. Since the height of T'_u is at most $\mathbf{td}(G[S])$, while $\mathbf{reach}(v,F)$ is at least $\mathbf{td}(G[S])$, it follows that F' is not higher than F. Since $u \in S_u$ is a descendant of v in F', it follows that F' represents the edge $\{u,v\}$ and is therefore a valid treedepth decomposition of $(G^* + \{u,v\}) - Z$ of height at most η. Hence Z is a treedepth-η modulator of $G' + \{u,v\}$ of size at most k, showing that $(G^* + \{u,v\}, k)$ is a yes-instance. ⌟

The claim shows that if $N_{G^*-Z}(S_u) = \emptyset$ then $(G^* + \{u, v\}, k)$ is a YES-instance, a contradiction to our starting assumption. Hence in the remainder it suffices to deal with the case that $N_{G^*-Z}(S_u) \neq \emptyset$. Let x be a vertex in $N_{G^*-Z}(S_u)$ and let y be a neighbor of x in S_u. Since $x \in N_{G^*-Z}(S_u)$ and S_u is a connected component of $G^*[S] - Z$, it follows that $x \notin S$. Hence $x \in N_{G^*}(S)$.

Claim 4.9. *Vertices u and v belong to the same connected component of $G^* - Z$.*

Proof. Observe that there exists a path from u to y in S_u, since S_u is a connected component of $G^* - Z$. If $x = v$ then combining this path from u to y in S_u with an edge from y to $x = v$, we obtain a uv path in $G^* - Z$, proving the claim. Assume then that $x \neq v$. Since $x \in N_{G^*}(S)$ and $G^* \subseteq G$, it follows that $x \in N_G(S)$. Since we did not remove edges between v and vertices outside S when forming G^*, while $N_G(S) \subseteq N_G[v]$, it follows that $x \in N_G(v)$. Now we obtain a path from u to v in $G^* - Z$ as follows: start with the path from u to y in S_u, follow the edge to x, and finally follow the edge to v. ⌟

Using the claim we can finish the proof. Let G_{uv} be the connected component of $G^* - Z$ containing u and v. Let T be the tree in F representing G_{uv}. Since $N_G(S) \subseteq N_G[v]$, while S_u is a connected component of $G^* - Z$, it follows that $N_{G^*-Z}(S_u) \subseteq N_{G^*-Z}[v]$. We may therefore apply Lemma 4.3 to the connected graph G_{uv}, the vertex v, and the connected subgraph $H := S_u$ of G_{uv}, along with the decomposition T of G_{uv}. The lemma guarantees that there is a decomposition T' of G_{uv} such that all vertices of S_u are in the subtree of T' rooted at v, and $\textbf{height}(T') \leq \max(\textbf{height}(T), \textbf{depth}(v, T) + \textbf{td}(G_{uv}[S_u]))$. It is easy to see that if we replace T by T' in the decomposition F, we obtain a valid treedepth decomposition F' of $G^* - Z$. Since u is in the subtree rooted at v, the decomposition represents the edge $\{u, v\}$ and is therefore also a decomposition of $(G^* + \{u, v\}) - Z$. It remains to bound the height of F, for which it suffices to bound the height of T'.

By Claim 4.7 we have $\textbf{reach}(v, F) \geq \textbf{td}(G[S])$. Using the definition of reach this implies that $\textbf{height}(F) \geq \textbf{depth}(v, F) + \textbf{td}(G[S])$. Since $\textbf{td}(G_{uv}[S_u]) \leq \textbf{td}(G[S])$ this implies that $\textbf{height}(T') \leq \max(\textbf{height}(T), \textbf{height}(F)) \leq \textbf{height}(F)$ by the expression above. Hence T' does not increase the height of F' beyond η, showing that F' is a treedepth decomposition of $G^* + \{u, v\}$ of height at most η. Hence Z is a solution for $G^* + \{u, v\}$ of size at most k, proving that $(G^* + \{u, v\}, k)$ is a YES-instance. As this contradicts our starting assumption, this concludes the proof of Lemma 4.7. □

5 Uniformly polynomial kernelization for Treedepth-η Deletion

In this section we develop the kernelization for TREEDEPTH-η DELETION. As described in the introduction, the two main ingredients are a decomposition algorithm (Section 5.1) and a reduction algorithm (Section 5.2) that will be applied to each piece of the decomposition. These will be combined into the final kernelization algorithm in Section 5.3.

5.1 Structural decomposition of the input graph

We present the algorithm that decomposes an instance (G, k) into a small number of pieces that each have a constant-size intersection with any minimum solution. The procedure is given as Algorithm 1. In the following lemma we analyze its behavior. Let us point out that the sets S and Y computed by the algorithm decompose the graph into η-nearly clique separated components C of $G - (S \cup Y)$: the neighborhood of each component C in S is a clique, and its neighborhood in the rest of the graph is contained on one root-to-leaf path in the decomposition F and therefore has size at most η. The intersection size of minimum treedepth-η modulators with such components is therefore at most 2η by Lemma 4.4.

Algorithm 1 Decompose(Graph G, $\eta \in \mathbb{N}$, $k \in \mathbb{N}$)

```
1:  while ∃ distinct $p, q \in G$ such that $\{p, q\} \notin E(G)$ and $\lambda_G(p, q) \geq k + \eta$ do
2:      Add the edge $\{p, q\}$ to $G$
    {All non-adjacent pairs $\{p, q\}$ at this point satisfy $\lambda_G(p, q) < k + \eta$}
3:  Apply Lemma 2.2 on the current graph to compute an approximate treedepth-$\eta$ modulator $S$
4:  if $|S| > 2^\eta \cdot k$ then
5:      Report that the original input graph does not have a treedepth-$\eta$ modulator of size $\leq k$
6:  Initialize $Y_0$ and $Y_1$ as empty vertex sets
7:  for each $\{p, q\} \in \binom{S}{2} \setminus E(G)$ do
8:      Let $Y_{p,q} \subseteq V(G) \setminus \{p, q\}$ be a minimum $pq$-separator {Menger's theorem: $|Y_{p,q}| < k + \eta$}
9:      Add $Y_{p,q}$ to $Y_0$
10: Compute a minimum-height nice treedepth decomposition $F$ of $G - S$ using Lemma 2.1
11: for each $v \in Y_0$ do
12:     Add the proper ancestors $\mathbf{anc}_F(v)$ of $v$ in $F$ to $Y_1$ {Since $F$ has height $\leq \eta$, $|\mathbf{anc}_F(v)| < \eta$}
13: Let $Y$ be $Y_0 \cup Y_1$
14: Define $\mathcal{T} := \{u \in V(F) \setminus Y \mid u \text{ is a root or } \pi(u) \in Y\}$
15: while there is a node $u_0$ in $\mathcal{T}$ such that:
        1. $G[N_G(F_{u_0})]$ is a clique, and
        2. for each $v \in N_G(F_{u_0})$ there are distinct nodes $u_1^v, \ldots, u_{\eta+k}^v \in \mathcal{T} \setminus \{u\}$ such that:
           $\forall i \in [\eta + k] \colon v \in N_G(F_{u_i}) \wedge \mathbf{td}(G[F_{u_i}]) \geq \mathbf{td}(G[F_{u_0}])$
    do
16:     Remove the vertices of $F_{u_0}$ from $G$ and $F$ and remove $u$ from $\mathcal{T}$ {For $v \neq v' \in N_G(F_{u_0})$
        we may have $u_i^v = u_j^{v'}$}
17: Output the updated graph and the decomposition $F$, the modulator $S$, and the separator $Y$
```

Lemma 5.1. *Let (G, k) be an instance of* Treedepth-η Deletion. *Then in polynomial time we can either conclude that (G, k) is a* NO*-instance, or find a graph G' with $V(G') \subseteq V(G)$, a treedepth-η modulator S of G', a treedepth decomposition F' of $G' - S$ of height at most η, and a set $Y \subseteq V(G') \setminus S$ satisfying the following properties.*

(1) (G, k) is equivalent to (G', k).

(2) $|S| \leq 2^\eta \cdot k$.

(3) $|Y| \leq \eta(2^\eta \cdot k)^2 \cdot (k + \eta)$.

(4) For every $u \in V(F') \setminus Y$ the graph $G'[F'_u]$ is connected.

(5) Let $\mathcal{T} := \{u \in F' - Y \mid u \text{ is a root or } \pi(u) \in Y\}$. The vertex sets of the connected components of $G' - (S \cup Y)$ are exactly the vertex sets of the subtrees of F' rooted at members of $\mathcal{T}$.

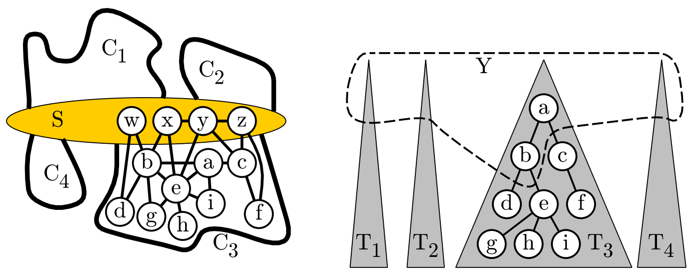
(6) For every connected component C of $G' - (S \cup Y)$, the set $N_{G'}(C) \cap S$ is a clique.

(7) The number of connected components of $G' - (S \cup Y)$ is at most

$$(|S| + |Y| + |S|^2 + |S| \cdot |Y| + \eta \cdot |Y|) \cdot (\eta + k).$$

Proof. We prove that the Decompose algorithm has the desired properties. Let us go through the algorithm line by line to analyze its effect. Along the way we will establish that the requirements from the lemma are satisfied. Consider an execution of Decompose(G, η, k) and let us denote by G^0 the state of the graph before the execution starts.

Let us first consider the effect of the edges that are added in Line 2.



(a) Graph G', modulator S. (b) Decomposition of $G' - S$.

Figure 3: Schematic illustration of an instance that has been decomposed using Lemma 5.1. 3(a) The resulting graph G' and the suboptimal treedepth-4 modulator S in G' used when decomposing. Graph $G' - S$ has four connected components, of which the third is drawn in detail. 3(b) Illustration of the treedepth-4 decomposition F' of $G' - S$. The forest F' contains four decomposition trees $T_1, \ldots, T_4$, one for each component of $G' - S$. By the properties of a treedepth decomposition, for any vertex $v \in V(G') \setminus S$, each neighbor $u \in N_{G'}(v)$ is an ancestor of v in F', descendant of v in F', or contained in S. Lemma 5.1 ensures that for each connected component C of $G - (S \cup Y)$, the set $N_G(C) \cap S$ is a clique. This is illustrated for the connected component consisting of $\{e, g, h, i\}$, whose neighbors among S are $\{x, y\}$, a 2-clique. As the set Y is closed under taking ancestors, it consists of the top parts of decomposition trees in F'.

Claim 5.1. *If the algorithm transforms G into $G + \{p, q\}$ by adding an edge in Line 2, then the* Treedepth-η Deletion *instance (G, k) is equivalent to instance $(G + \{p, q\}, k)$.*

Proof. If $(G + \{p, q\}, k)$ is a YES-instance then clearly its minor (G, k) is as well. So assume that (G, k) is a YES-instance, implying that minimum treedepth-η modulators in G have size at most k. Define $\ell := k$ and let $X := V(G)$. Since the algorithm ensures that $\lambda_G(p, q) \geq k + \eta$, the graph $G[X \cup \{p, q\}] = G$ contains at least $k + \eta$ internally vertex-disjoint pq-paths. Since a minimum solution in G contains at most k vertices, trivially any minimum treedepth-η modulator intersects X in at most $k = \ell$ vertices. But then these choices of X and ℓ satisfy the conditions of Lemma 4.6, which proves that (G, k) is equivalent to $(G + \{p, q\}, k)$. ⌟

The claim shows that every edge addition preserves the answer to the instance. Hence for the graph G obtained after finishing the first **while**-loop, instance (G, k) is equivalent to (G^0, k). Once the **while**-loop terminates, for each non-adjacent vertex pair $\{p, q\}$ that remains, the number of internally disjoint pq-paths must be less than $k + \eta$. Let us consider the set S that is computed, and denote by G^1 the status of the graph in Line 3. By Lemma 2.2 the treedepth of $G^1 - S$ is at most η. If $|S| > 2^\eta \cdot k$, since Lemma 2.2 guarantees a factor 2^η-approximation, the minimum solution size for (G^1, k) exceeds k. By the equivalence of the instance to (G^0, k) the algorithm is therefore correct if it outputs that the original input G^0 does not have a size-k solution. If a set S is returned it must therefore satisfy (2). We continue by analyzing the computed set Y.

Claim 5.2. $|Y_0| \leq |S|^2 \cdot (k + \eta) \leq (2^\eta \cdot k)^2 \cdot (k + \eta)$.

Proof. Since $|S| \leq 2^\eta \cdot k$, there are at most $(2^\eta \cdot k)^2$ pairs of nonadjacent vertices among S. For every nonadjacent pair of vertices $\{p, q\}$, the termination condition of the **while**-loop ensures that $\lambda_{G^1}(p, q) \leq \eta + k$. By Menger's Theorem the value $\lambda_{G^1}(p, q)$ equals the minimum size of a vertex pq-separator that avoids p and q; such a set can be computed efficiently (cf. [38, Chapter 9]). Hence $|Y_{pq}| < k + \eta$ for all considered pairs. The claim follows. ⌟

Claim 5.3. $|Y| \leq \eta \cdot (2^\eta \cdot k)^2 \cdot (k + \eta)$.

Proof. Since Y_1 contains the proper ancestors of every member of Y_0, while every vertex has at most $\eta - 1$ proper ancestors in a treedepth decomposition of height at most η, it follows that $|Y_1| \leq (\eta - 1)|Y_0|$. Using the previous claim, we find that $Y = Y_0 \cup Y_1$ has size at most $\eta \cdot (2^\eta \cdot k)^2 \cdot (k + \eta)$. ⌟

The claim shows that requirement (3) is satisfied. Before we analyze last **while**-loop, we consider the structure of the computed set $\mathcal{T}$ of *topmost* vertices in the forest that do not belong to Y.

Claim 5.4. *If u, u' are distinct vertices in $\mathcal{T}$ then the subtrees $F_u, F_{u'}$ are disjoint.*

Proof. Assume for a contradiction that the claim is false. By symmetry, we may assume that u' is in the subtree of F rooted at u. Then u' is not the root of a tree. By definition of the set $\mathcal{T}$ this implies that $\pi(u')$ is a vertex in Y. But since we added the ancestor of every vertex in Y_0 to Y_1, this implies that the ancestor u of u' must be contained in Y. By definition of $\mathcal{T}$ this contradicts that $u \in \mathcal{T}$. ⌟

We observe that, by updating the set $\mathcal{T}$ in Line 16, the algorithm ensures that at any point of its execution of the last **while**-loop, even though the graph might have changed after the point that $\mathcal{T}$ was defined and computed, the set $\mathcal{T}$ still satisfies that definition.

Using Claim 5.4 we analyze what happens in the **while**-loop of Line 15. In Line 16 we consider the vertices contained in the subtree of F rooted at a node u_0 that satisfies the conditions of the **while**-loop, and we remove them from the graph.

Claim 5.5. *If the algorithm transforms G into $G - F_{u_0}$ in Line 16, then the* Treedepth-η Deletion *instance (G, k) is equivalent to instance $(G - F_{u_0}, k)$.*

Proof. Define A as the vertices in the subtree of F rooted at u_0. If (G, k) is a yes-instance then its minor $(G - A, k)$ is as well. For the reverse direction, suppose that $(G - A, k)$ is a yes-instance. We aim at applying Lemma 4.5. By the preconditions to the loop, we know that $N_G(A)$ is a clique in G. Define $\ell := k$. Then, clearly, for any $\mathcal{X} \subseteq V(G)$ the graph $G - A$ has a minimum treedepth-η modulator intersecting $\mathcal{X}$ in at most ℓ vertices, showing that the third requirement of Lemma 4.5 is satisfied. Let us verify the first two requirements are satisfied as well. For each $v \in N_G(A)$ and $i \in [\eta + k]$, define X_i^v as the vertices in the subtree of F rooted at the node u_i^v identified in the algorithm. Then the test in the algorithm ensures the first condition of Lemma 4.5 is satisfied. The fact that, for each $v \in N_G(A)$, the vertex sets X_i^v are pairwise disjoint and disjoint from A, follows from Claim 5.4. It remains to check that all sets X_i^v induce connected subgraphs of G. If the treedepth decomposition F is still nice when the statement is executed, then this follows from the definition of a nice treedepth decomposition. While earlier removals may have caused F to no longer be a nice decomposition, since the sets X_i^v correspond to subtrees of the nice treedepth forest originally computed in Line 10, and other iterations of the loop do not affect the graphs they induce, all sets X_i^v indeed induce connected subgraphs of G at the time the statement is executed. Hence all requirements are met and Lemma 4.5 implies the claim. ⌟

The combination of Claims 5.1 and 5.5 proves that for the state G' of the graph upon termination, instance (G^0, k) is equivalent to (G', k). Hence (1) holds.

Claim 5.6. *The **while**-loop of Line 15 can be evaluated in polynomial time for every fixed η.*

Proof. As each iteration removes a vertex, the number of iterations is bounded by the order of the input graph. Let us prove that each iteration can be done in polynomial time. To test the loop condition, it suffices to do the following. For each $u \in \mathcal{T}$ we consider the subtree F_u rooted at F and compute a minimum-height treedepth decomposition of $G[F_u]$. Since $F_u \subseteq V(G) \setminus S$ the treedepth is at most η, this can be done in polynomial time for fixed η by Lemma 2.1. For each possible choice of u_0 we can then test whether the conditions hold for u_0 by checking whether, for each $v \in N_G(F_u)$, there are sufficiently many components also adjacent to v whose treedepth is at least that of F_u. ⌟

Let G' and F' denote the graph and treedepth decomposition upon termination. The following claim proves (4).

Claim 5.7. *For every $u \in V(F') \setminus Y$, the graph $G'[F'_u]$ is connected.*

Proof. Let G^2 and F^2 denote the status of the graph and decomposition after Line 10. By definition of a nice treedepth decomposition, for all $u \in V(F^2)\setminus Y$, the graph $G^2[F^2_u]$ is connected. To see that this still holds once the **while**-loop of Line 6 has removed parts of the decomposition and the graph, it suffices to observe the following. For every node $u \in V(F') \setminus Y$ that has survived, no removals were made in the subtree F^2_u: if any removal would have been made, then since we remove entire subtrees rooted at topmost vertices in $\mathcal{T}$, vertex u itself would have been removed. ⌟

Let $\mathcal{T}' := \{u \in F' - Y \mid u \text{ is a root or } \pi(u) \in Y\}$ as in the lemma statement. We now establish (5).

Claim 5.8. *The vertex sets of the connected components of $G' - (S \cup Y)$ are exactly the vertex sets of the subtrees of F' rooted at members of $\mathcal{T}'$.*

Proof. In one direction, let $u \in \mathcal{T}'$ and consider the subtree F_u of F rooted at u. No descendant of u is contained in Y, otherwise u itself would have been included in Y_1 and therefore in Y. By Claim 5.7 the graph $G'[F'_u]$ is connected. Assume that the set F'_u has a neighbor x in G that does not belong to S. Since F' is a valid treedepth decomposition of $G' - S$, vertex x is an ancestor or descendant of a member of F'_u. Since all descendants of F'_u are contained in F'_u, it follows that x is a proper ancestor of u. But by definition of $\mathcal{T}'$, either u is a root or $\pi(u) \in Y$, implying that all proper ancestors of u are in Y. So all vertices in $N_{G'}(F'_u)$ belong to S or to Y, proving that each member of $\mathcal{T}'$ yields a connected component of $G' - (S \cup Y)$.

For the reverse direction, consider some connected component C of $G' - (S \cup Y)$. By Observation 2.2, all vertices of C belong to one tree T' of F'. Consider the least common ancestor u of the vertices in C in tree T'. If $u \in \mathcal{T}'$ then we are done, since by Claim 5.7 the graph $G'[T'_u]$ is connected and is disjoint from S and Y; therefore C must equal $G'[T'_u]$. Assume for a contradiction that $u \notin \mathcal{T}'$.

If $u \notin Y$, then since $u \notin \mathcal{T}'$ it follows that u is not the root of T and the parent of u does not belong to Y. But by Claim 5.7 the graph $G'[T'_{\pi(u)}]$ is connected. It is disjoint from S and disjoint from Y, since all ancestors of Y are in Y. Hence C is not a connected component of $G' - (S \cup Y)$ because there is a connected supergraph of C in $G' - (S \cup Y)$.

Finally, consider the case that $u \in Y$. Since $u \notin C$ is the least common ancestor of vertices of C, at least two different children c_1, c_2 of u contain members x_1, x_2 of C. But by Observation 2.2, a common ancestor of x_1 and x_2 is contained in H. But then this must be an ancestor of u. However, all ancestors of u (including u itself) are contained in Y, proving that C intersects Y and is not a connected component of $G' - (S \cup Y)$. ⌟

The following claim proves (6).

Claim 5.9. *For every connected component C of $G' - (S \cup Y)$, the set $N_{G'}(C) \cap S$ is a clique in G'.*

Proof. Let C be a connected component of $G' - (S \cup Y)$ and assume for a contradiction that $N_{G'}(C) \cap S$ is not a clique. Let $\{p, q\} \in N_{G'}(C) \cap S$ be non-adjacent in G'. Then we added a pq-separator $Y_{p,q}$ disjoint from p and q to the set Y_0, and it was even a separator in the supergraph of G' that we considered during Line 9. Consequently, no connected component of $G' - Y_0$ can be simultaneously adjacent to both p and q. Since $Y \supseteq Y_{p,q}$, it follows that no connected component of $G' - Y$ can be adjacent to both p and q; a contradiction. ⌟

Finally, we bound the number of connected components of $G' - (S \cup Y)$ to establish (7). By Claim 5.8 it suffices to bound $|\mathcal{T}'|$. We partition $\mathcal{T}'$ into two sets. Let $\mathcal{T}'_S$ contain the nodes $u \in \mathcal{T}'$ such that $N_{G'}(F'_u)$ is a clique; we call these the *simplicial components*. Let $\mathcal{T}'_N$ be the remaining nodes in $\mathcal{T}'$, corresponding to *non-simplicial components*.

Claim 5.10. *$|\mathcal{T}'_S| \leq (|S| + |Y|)(\eta + k)$.*

Proof. Consider a node $u_0 \in \mathcal{T}'_S$. Since $N_{G'}(F'_{u_0})$ is a clique, it satisfies the first requirement of the **while**-loop in Line 15. Hence if it was not removed by the algorithm, the second requirement cannot be met. So there is some $v \in N_{G'}(F_{u_0})$ for which there are no $\eta + k$ other nodes u_i in $\mathcal{T}$ with $v \in N_{G'}(F_{u_i})$ and $\mathbf{td}(G'[F'_{u_i}]) \geq \mathbf{td}(G'[F'_{u_0}])$. Charge u_0 to such a neighbor v.

Since F' is a treedepth decomposition of $G' - S$, all neighbors of F'_{u_0} in G are either contained in S or are proper ancestors of u_0. Hence all neighbors of F'_{u_0} are contained in $S \cup Y$. Now assume for a contradiction that we charge more than $\eta + k$ nodes of $\mathcal{T}'_S$ to the same member x of $S \cup Y$. Letting u_0 be a node charged to x that minimizes $\mathbf{td}(G'[F'_{u_0}])$, we now find that the other $\eta + k$ nodes charged to x also have subtrees adjacent to x that have treedepth at least that of $G'[F'_{u_0}]$; but then u_0 cannot be charged to x. It follows that we charge at most $k + \eta$ times to each member of $S \cup Y$, proving the size bound. ⌟

Finally, we bound the number of non-simplicial components.

Claim 5.11. *$|\mathcal{T}'_N| \leq (|S|^2 + |S| \cdot |Y| + \eta \cdot |Y|) \cdot (\eta + k)$.*

Proof. Consider some $u \in \mathcal{T}'_N$. By definition of the non-simplicial nodes, there is a pair of vertices $\{p, q\} \subseteq N_{G'}(F'_u)$ that is not adjacent in G. As observed above, all vertices in $N_{G'}(F'_u)$ are members of S or proper ancestors of u in F', and are therefore contained in Y. Note that the connected subgraph F'_u contains the interior vertices of a path between p and q. By Claim 5.4, these paths are pairwise internally vertex-disjoint for different members of $\mathcal{T}'_N$. Charge every $u \in \mathcal{T}'_N$ to a pair of non-adjacent vertices in $N_{G'}(F'_u)$. Since the **while**-loop of Line 2 adds edges between pairs that are connected by $\eta + k$ pairwise internally vertex-disjoint paths, we can charge at most $\eta + k$ times to each pair. To prove the claim, it suffices to bound the number of possible pairs. Now observe that every pair $\{p, q\}$ to which we charge consists of vertices of $S \cup Y$. The number of pairs where both ends are from S, or exactly one end is from S, is clearly at most $|S|^2$ and $|S| \cdot |Y|$, respectively. Finally, observe that for pairs where both members are from Y, these members are in ancestor-descendant relation in F' since both endpoints are ancestors of the nodes u that charge to them. Since the height of F' is at most η, each node in F' has less than η ancestors. If we thus count, for each node in Y, the number of pairs where the other node is higher in the forest, we count at most η incident pairs per vertex of Y, for a total of at most $\eta \cdot |Y|$. Hence the total number of pairs to which we charge is at most $|S|^2 + |S| \cdot |Y| + \eta \cdot |Y|$. As we charge at most $\eta + k$ nodes of $\mathcal{T}'_N$ to each pair, the claim follows. ⌟

Algorithm 2 Reduce(Graph G, treedepth-η modulator S, treedepth-η decomposition F of $G - S$, node v of F, $k \in \mathbb{N}$)

```
1: Let $T$ be the tree in $F$ containing $v$
2: while $\exists$ distinct $p, q \in N_G(T_v) \cup \{v\}$ with $\{p,q\} \notin E(G)$ and $\lambda_{G[\{p,q\}\cup T_v]}(p,q) \geq 3\eta$ do
3:     Add the edge $\{p,q\}$ to $G$
4: while $\exists$ distinct children $c_0, c_1, \ldots, c_{3\eta}$ of $v$ such that $c_0$ has a neighbor $s \in S$, $N_G(T_{c_0}) \subseteq N_G[s]$, and for $i \in [3\eta]$ we have $\mathbf{td}(G[T_{c_i}]) \geq \mathbf{td}(G[T_{c_0}])$ and $s \in N_G(T_{c_i})$ do
5:     Remove the edges between $s$ and members of $T_{c_0}$ from graph $G$
6: while $\exists$ a child $c^*$ of $v$ such that $N_G(T_{c^*})$ is a clique, and for every $w \in N_G(T_{c^*})$ there are $3\eta$ distinct children $c_1^w, \ldots, c_{3\eta}^w \neq c^*$ of $v$ such that for all $i \in [3\eta]$ we have $\mathbf{td}(G[T_{c_i^w}]) \geq \mathbf{td}(G[T_{c^*}])$ and $w \in N_G(T_{c_i^w})$ do
7:     Remove the vertices in $T_{c^*}$ from $F$ and from $G$
8: for each remaining child $c$ of $v$ in $T$ do
9:     Reduce($G$, $S$, $F$, $c$, $k$)
```

Since $|\mathcal{T}| = |\mathcal{T}_N| + |\mathcal{T}_S|$, by combining Claims 5.10, 5.11 and 5.8 we establish (7). This concludes the proof of Lemma 5.1. $\square$

5.2 Reduction algorithm

The reduction algorithm that will be applied to each piece of the decomposition is given as Algorithm 2. To prove that it works correctly, we will prove that it maintains a set of concrete invariants.

Definition 5.1 (Invariants). *Consider an execution of Reduce(G, S, F, v, k). Let T be the tree of F containing v and let G^0, F^0, T^0 be the status of G, F and T at the start of the iteration. We define the following invariants of Algorithm 2.*

(I) F is a treedepth decomposition of $G - S$ of height at most η.
(II) For every vertex $u \in T_v$ the graph $G[T_u]$ is connected.
(III) The set $N_G(T_v) \cap S = N_G(T_v) \setminus \mathbf{anc}_T(v)$ is a clique in G.
(IV) The graph G can be obtained from G^0 by

- *adding edges whose endpoints belong to $N_{G^0}(T_v^0) \cup \{v\}$,*
- *removing edges between $N_{G^0}(T_v^0) \cap S$ and proper descendants of v in T^0,*
- *removing vertex sets of subtrees rooted at children of v.*

(V) F is a rooted subforest of F^0.
(VI) For every $u \in T_v$ we have $N_G(T_u) \cap S \subseteq N_{G^0}(T_u^0)$.
(VII) The instance (G, k) is equivalent to the instance (G^0, k).

Lemma 5.2. *The Reduce algorithm preserves its invariants.*

Proof. We will prove that if the invariants hold, then any step taken by the algorithm preserves the invariants. For concreteness, we denote by G^0 and F^0 the state of G and F at the time the procedure is called. During the execution of the algorithm, the structures G and F change. Let T^0 be the tree in F^0 containing v. The proof is by induction on the height of T_v^0, which is at least one. Assume that the invariants hold before some step of the algorithm and let G, F, T denote the status of the structures before the step. We use G', F', T' for the status after the step. We make a distinction based on the action taken by the algorithm.

Adding an edge. Suppose that the algorithm adds an edge $\{p,q\}$ in Line 3 so that $G' := G + \{p,q\}$. To see that F is still a valid treedepth decomposition of $G' - S$, it suffices to observe that the added edge either has an endpoint in S, or both its endpoints are ancestors of v, implying that the edge is represented by the decomposition. Hence Invariant (I) is preserved. To see that invariant (VI) is preserved, note that both endpoints of the added edge are contained in $N_G(T_v) \cup \{v\}$, and so the only vertex in T_v that can be incident on the added edge is v itself. If an edge was added from v to a vertex $s \in S$ then some member of T_v was already adjacent to s. The only other invariant that is not trivially maintained is Invariant (VII). To see that it is maintained as well, observe the following.

By Invariant (III), the set $N_G(T_v) \setminus \mathbf{anc}_T(v)$ is a clique. Since the edge $\{p, q\}$ we add either has an endpoint in $\mathbf{anc}_T(v)$, or is an edge between v and a member of $N_G(T_v) \setminus \mathbf{anc}_T(v) = N_G(T_v) \cap S$, it follows that $N_{G'}(T'_v) \setminus \mathbf{anc}_{T'}(v)$ is also a clique (the decomposition tree does not change). Hence the set T'_v is η-nearly clique separated in G', since v has at most η ancestors. By Lemma 4.4, any minimum treedepth-η modulator of G' intersects T'_v in at most 2η vertices. We may therefore apply Lemma 4.7 where the set T'_v is used as X and $\ell = 2\eta$, to establish that $(G' = G + \{p, q\}, k)$ is equivalent to (G, k) and therefore, using the invariant applied to G, to (G^0, k). This proves that Invariant (VII) is maintained.

Removing a set of edges. Now consider what happens when the algorithm removes a set of edges in Line 5. Since the edges we remove have exactly one endpoint in S, all invariants except Invariant (VII) are easily seen to be preserved. To prove that (VII) is also preserved, we will apply Lemma 4.7. Let us consider the requirements for the lemma. Define $\ell := 2\eta$ and let $X_1, \ldots, X_{\ell+\eta}$ be the vertex sets of $T_{c_1}, \ldots, T_{c_{3\eta}}$ identified in the algorithm. By Invariant (II), for every $i \in [3\eta]$ the graph $G[T_{c_i}]$ is connected. The condition in the **while** loop ensures that $\mathbf{td}(G[T_{c_i}]) \geq \mathbf{td}(G[T_{c_0}])$ for all $i \in [3\eta]$. Let S be the vertices of T_{c_0}. It follows that our choice of S and the X_i satisfy the first two conditions of Lemma 4.7, when using the vertex s in the algorithm as v in the lemma statement. To see that the third condition is also valid, observe that $\mathcal{X} := \bigcup_{i \in [3\eta]} X_i$ is contained in T_v and that in any graph obtained from G by removing edges between s and T_{c_0}, the set T_v is η-nearly clique separated by Invariant (III). Hence, by Lemma 4.4, any minimum treedepth modulator in a graph obtained from G by removing edges between s and T_{c_0} contains at most $2\eta = \ell$ vertices from T_v. Together with the fact that $N_G(T_{c_0}) \subseteq N_G[s]$ we find that all conditions of Lemma 4.7 are satisfied, which proves that instance (G', k) is equivalent to (G, k). By Invariant (VII) and transitivity, instance (G', k) is equivalent to (G^0, k) and therefore said invariant is preserved.

Removing the vertices of a child subtree. As the next operation, suppose that the Reduce algorithm removes the vertices in a subtree rooted at a child c^* of v, in Line 7. Then $N_G(T_{c^*})$ is a clique and for every $w \in N_G(T_{c^*})$ there are 3η distinct children $c^w_1, \ldots, c^w_{3\eta}$ unequal to c^* such that the treedepth of the subgraphs they represent is at least $\mathbf{td}(G[T_{c^*}])$, and they each contain a neighbor of w. Observe that by Invariant (II), for any $i \in [3\eta]$ and $w \in N_G(T_{c^*})$ the graph $G[T_{c^w_i}]$ is connected. Set $\ell := 2\eta$. We will prove that the conditions of Lemma 4.5 are satisfied for this choice of ℓ, using T_{c^*} as S. Observe that the height of T_{c^*} is at most η by Invariant (I). The previous observations ensure that, when choosing X^w_i as $T_{c^w_i}$ for all $w \in N_G(T_{c^*})$ and $i \in [3\eta]$, the first condition of Lemma 4.5 is satisfied. The fact that for each choice of w, for all i the defined sets X^w_i are pairwise disjoint (note that sets for different choices of w may overlap) follows from the fact that the X^w_i come from different children of v. Hence the second condition of the lemma is also satisfied. To see that the last condition is satisfied, note that $N_G(T_v) \setminus \mathbf{anc}_T(v)$ is a clique by Invariant (III), and hence T_v is η-nearly clique separated in G. Therefore, using Lemma 4.4, any minimum treedepth-η modulator in G intersects T_v in at most $\ell = 2\eta$ vertices. Hence the final condition of Lemma 4.5 is satisfied, proving that (G', k) is equivalent to (G, k) and therefore to (G^0, k). This implies that Invariant (VII) is satisfied.

Let us now consider the other invariants. The only remaining invariant that is not trivially maintained is Invariant (II), which says that the graph $G[T_u]$ is connected for any vertex u

in T_v. Let T' denote the status of T after the deletion of T_{c^*}. Observe that since we removed an entire subtree rooted at a child of v, the only vertex u in T'_v for which T'_u differs from T_u, is vertex v itself. To see that $G'[T'_v]$ is connected, observe that if T'_v is not a single vertex, then by the properties of treedepth decompositions, v is a cutvertex in $G[T_v]$ that separates the vertices in T_{c^*} from the remaining vertices. Hence any simple path that existed between two vertices of $T_v \setminus T_{c^*}$ in the graph $G[T_v]$, still exists in $G[T'_v]$. It follows that Invariant (II) is maintained.

Executing a recursive call. The last case is when the operation that the algorithm performs is making a recursive call. This is where we use induction. If the algorithm makes a recursive call, then this is for children of v and therefore the height of T_v is larger than one. Since the height of T_u is smaller than the height of T_v for all children u of v, by induction we find that the recursive call maintains the invariants. This concludes the proof of Lemma 5.2. □

Having established the invariants of the algorithm, we know that it preserves the answer to an instance of Treedepth-η Deletion. For the purposes of obtaining a kernel, we also need to prove that it achieves a provable size reduction. We do this in the following lemma.

Lemma 5.3. *Let Reduce be called for the input (G^0, S^0, F^0, v, k), and let T^0 be the tree in F^0 containing v. Let G', F' be the graph and decomposition once the procedure has finished. Let*

$$\phi(u) := (2 \cdot 3\eta \cdot 2^{\eta})^{\mathbf{height}(T^0_v)} \cdot (|N_{G^0}(T^0_v) \cap S| + 1),$$

for any vertex $u \in T^0_v$. Then the number of leaves in F'_v is at most $\phi(v)$.

Proof. We will prove the lemma by induction on $\mathbf{height}(T^0_v)$. Before doing so, however, we establish the structure of the graph once the Reduce algorithm has reached Line 8. Let G be the state of the graph once reaching Line 8, let F be the state of the forest, and let T be the tree of F containing v.

Claim 5.12. *Let u be a child of v in T. Then for every pair of distinct vertices $\{p, q\} \subseteq N_G(T_u)$ there is a pq-path in the graph $G[\{p, q\} \cup T_u]$.*

Proof. If $p, q \in N_G(T_u)$ then there is a neighbor p' of p in T_u, and a neighbor q' of q in T_u. By Invariant (II), the graph $G[T_u]$ is connected and contains a path P connecting p' and q'. By adding the edges to p and q we obtain the desired pq path in $G[\{p, q\} \cup T_u]$. ⌟

We now derive bounds on the number of children of v once the execution has reached Line 8. Let $\mathcal{C}^+$ denote the set of children u of v for which $N_G(T_u) \cap S \neq \emptyset$. Let $\mathcal{C}^-$ denote the remaining children of v.

Claim 5.13. $|\mathcal{C}^-| \leq 3\eta \cdot 2^{\mathbf{depth}(v, T^0)}$.

Proof. Consider a child u of v such that $N_G(T_u) \cap S = \emptyset$. By Observation 2.3, the only possible vertices of $N_G(T_u)$ are proper ancestors of u, which are exactly the ancestors Y of v (since v is its own ancestor). There are exactly $\mathbf{depth}(v, T) = \mathbf{depth}(v, T^0)$ of them. For $Y' \subseteq Y$ let $\mathcal{C}^-_{Y'}$ contain the children u of v for which $N_G(T_u) = Y'$. Since there are $2^{|Y|} = 2^{\mathbf{depth}(v, T^0)}$ possible groups, to establish the claim it suffices to bound the size of each group by 3η.

Fix some $Y' \subseteq Y$ and the group $\mathcal{C}^-_{Y'}$. Assume for a contradiction that $|\mathcal{C}^-_{Y'}| > 3\eta$. For any pair of distinct vertices $\{p, q\} \in Y'$ there are at least 3η internally vertex-disjoint pq-paths in the graph $G[\{p, q\} \cup T_v]$, since we get one such path through each set T_c with $c \in \mathcal{C}^-_{Y'}$, by Claim 5.12. If these paths exist in G, then surely they must have existed in the state of the graph when the **while**-loop of Line 2 terminated, since we only delete vertices and edges afterward. So the paths were detected in that loop, causing the edge $\{p, q\}$ to be added to G. Since such edges are not removed in the **while**-loop of Line 4, during the **while**-loop of Line 6 for each $c \in \mathcal{C}^-_{Y'}$ the set $N_G(T_c)$ is a clique. But if $|\mathcal{C}^-_{Y'}| > 3\eta$, then letting c^* be a member of $\mathcal{C}^-_{Y'}$ minimizing $\mathbf{td}(G[T_{c^*}])$ and letting $c_1, \dots, c_{3\eta}$ be 3η arbitrary other members of $\mathcal{C}^-_{Y'}$, since all subtrees rooted at $\mathcal{C}^-_{Y'}$ have the same G-neighborhood we now find that this choice of c^* and using the same $c_1, \dots, c_{3\eta}$ for all $w \in N_G(T_c)$, the conditions of the **while**-loop are satisfied, causing c^* to be deleted. Hence if the algorithm was correctly executed, $|\mathcal{C}^-_{Y'}| \le 3\eta$. The claim follows. ⌟

Claim 5.14. $\sum_{u \in \mathcal{C}^+} |N_G(T_u) \cap S| \le 3\eta \cdot 2^{\mathbf{depth}(v,T^0)} \cdot |N_{G^0}(T^0_v) \cap S|$.

Proof. For each child u of v, by Observation 2.3 we know that $N_G(T_u) \setminus S$ consists of ancestors of v. As in the proof of the previous claim, let Y be the ancestors of v and partition the set $\mathcal{C}^+$ into subsets $\mathcal{C}^+_{Y'}$ for $Y' \subseteq Y$ such that for all $u \in \mathcal{C}^+_{Y'}$ we have $N_G(T_u) \setminus S = Y'$.

Fix an arbitrary $Y' \subseteq Y$. We start by showing that for every vertex s of S, there are at most 3η members u of $\mathcal{C}^+_{Y'}$ such that $s \in N_G(T_u)$. Assume for a contradiction that some $s \in S$ is adjacent to more than 3η of the subtrees rooted at $\mathcal{C}^+_{Y'}$. For each subtree $u \in \mathcal{C}^+_{Y'}$, by Claim 5.12 for any vertex $y \in Y'$ there is an su-path in subgraph $G[\{s,y\} \cup T_u]$. Since all subtrees T_u are contained in T_v, this implies that $\lambda_{G[\{s,y\} \cup T_v]}(s,y) \ge 3\eta$ for every $y \in Y'$. Hence the edge addition rule triggered in Line 3 for s and every member of Y', showing that s is adjacent to all members of Y'. Now let $c_0 \in \mathcal{C}^+_{Y'}$ minimize $\mathbf{td}(G[T_{c_0}])$ among all members of $\mathcal{C}^+_{Y'}$ for which $s \in N_G(T_{c_0})$, and let $c_1, \dots, c_{3\eta}$ be 3η members u of $\mathcal{C}^+_{Y'}$ for which $s \in N_G(T_u)$. Then $s \in N_G(T_{c_0})$ and for all $i \in [3\eta]$ we have $s \in N_G(T_{c_i})$. By choice of c_0 we know $\mathbf{td}(G[T_{c_i}]) \ge \mathbf{td}(G[T_{c_0}])$ for all $i \in [3\eta]$. Finally, since s is adjacent to all members of Y' and $N_G(T_{c_0}) \setminus Y' \subseteq S$ is a clique containing s, it follows that $N_G(T_{c_0}) \subseteq N_G[s]$. But then all conditions of the **while**-loop of Line 4 are satisfied, showing that the algorithm would have removed the edges from s to T_{c_0}; a contradiction. Hence we established that for every $s \in S$ there are at most 3η members u in $\mathcal{C}^+_{Y'}$ with $s \in N_G(T_u)$. Let us finish the proof of the claim with this knowledge. Observe the following double-counting equality:

$$\sum_{u \in \mathcal{C}^+_{Y'}} |N_G(T_u) \cap S| = \sum_{s \in S} \left| \{u \in \mathcal{C}^+_{Y'} \mid s \in N_G(T_u)\} \right|.$$

By the previous argument, every $s \in S$ is a neighbor of at most 3η subtrees rooted in $\mathcal{C}^+_{Y'}$. In addition, observe that every $s \in S$ that has a neighbor in T_u for some $u \in \mathcal{C}^+_{Y'}$ trivially also has a neighbor in $T_v \supset T_u$. Hence vertices $s \in S \setminus N_G(T_v)$ have no neighbors among such subtrees T_u. This proves that:

$$\sum_{s \in S} \left| \{u \in \mathcal{C}^+_{Y'} \mid s \in N_G(T_u)\} \right| = \sum_{s \in N_G(T_v) \cap S} \left| \{u \in \mathcal{C}^+_{Y'} \mid s \in N_G(T_u)\} \right| \le |N_G(T_v) \cap S| \cdot 3\eta.$$

Combining these two inequalities, and summing over all possible Y', we find that:

$$\begin{aligned} \sum_{u \in \mathcal{C}^+} |N_G(T_u) \cap S| &\le \sum_{Y' \subseteq Y} \sum_{u \in \mathcal{C}^+_{Y'}} |N_G(T_u) \cap S| \\ &= \sum_{Y' \subseteq Y} \sum_{s \in N_G(T_v) \cap S} \left| \{u \in \mathcal{C}^+_{Y'} \mid s \in N_G(T_u)\} \right| \\ &\le 2^{|Y|} \cdot |N_G(T_v) \cap S| \cdot 3\eta \end{aligned}$$

Observe that $|Y| = \mathbf{depth}(v, T^0)$. To establish the claim, observe that by Invariant (VI) we have $|N_G(T_v) \cap S| \le |N_{G^0}(T^0_v) \cap S|$. ⌟

Using the previous claims we now bound the number of leaves that remain in the subtree T_v after the algorithm has terminated. Observe that once the algorithm recurses on a child u, the height of T_u is less than the height of T_v and therefore we may apply induction to bound the number of leaves in the subtrees resulting from recursive calls.

If there are no children of v left to recurse on, then v is the only leaf in T_v that remains. It is easy to see that the bound of Lemma 5.3 ensures that $\phi(u) \geq 1$ and therefore the base case of the induction holds. If v has at least one child left, then v is not a leaf. The number of leaves in T_v is obtained by summing the bounds for its children. Let $f(v)$ denote the number of leaves of the subtree T_v after the procedure terminated and observe the following derivation. We count the leaves in children rooted at $\mathcal{C}^+$ and the leaves in children rooted at $\mathcal{C}^-$ separately.

$$\begin{aligned}
\sum_{u \in \mathcal{C}^-} f(u) \leq \sum_{u \in \mathcal{C}^-} \phi(u) && \text{By induction.}\\
\leq \sum_{u \in \mathcal{C}^-} (3\eta \cdot 2^{\eta+1})^{\mathbf{height}(T_u^0)} \cdot (|N_{G^0}(T_u^0) \cap S| + 1) && \\
\leq \sum_{u \in \mathcal{C}^-} (3\eta \cdot 2^{\eta+1})^{\mathbf{height}(T_u^0)} \cdot (0 + 1) && \text{Definition of } \mathcal{C}^-.\\
\leq \sum_{u \in \mathcal{C}^-} (3\eta \cdot 2^{\eta+1})^{\mathbf{height}(T_v^0)-1} && u \text{ is below } v.\\
\leq |\mathcal{C}^-| \cdot (3\eta \cdot 2^{\eta+1})^{\mathbf{height}(T_v^0)-1} && \\
\leq 3\eta \cdot 2^{\mathbf{depth}(v,T^0)} \cdot (3\eta \cdot 2^{\eta+1})^{\mathbf{height}(T_v^0)-1} && \text{Claim 5.13.}\\
\leq (3\eta \cdot 2^{\eta+1})^{\mathbf{height}(T_v^0)} && \mathbf{depth}(v, T^0) \leq \eta.
\end{aligned}$$

Now we consider $\mathcal{C}^+$.

$$\begin{aligned}
\sum_{u \in \mathcal{C}^+} f(u) \leq \sum_{u \in \mathcal{C}^+} \phi(u) && \text{By induction.}\\
\leq \sum_{u \in \mathcal{C}^+} (3\eta \cdot 2^{\eta+1})^{\mathbf{height}(T_u^0)} \cdot (|N_{G^0}(T_u^0) \cap S| + 1) && \\
\leq \sum_{u \in \mathcal{C}^+} (3\eta \cdot 2^{\eta+1})^{\mathbf{height}(T_v^0)-1} \cdot (|N_{G^0}(T_u^0) \cap S| + 1) && u \text{ is below } v.\\
\leq (3\eta \cdot 2^{\eta+1})^{\mathbf{height}(T_v^0)-1} \cdot \sum_{u \in \mathcal{C}^+} (|N_{G^0}(T_u^0) \cap S| + 1) && \text{Rearranging.}\\
\leq (3\eta \cdot 2^{\eta+1})^{\mathbf{height}(T_v^0)-1} \cdot 2 \sum_{u \in \mathcal{C}^+} |N_{G^0}(T_u^0) \cap S| && N_{G^0}(T_u^0) \cap S| > 0 \text{ by def. } \mathcal{C}^+.\\
\leq (3\eta \cdot 2^{\eta+1})^{\mathbf{height}(T_v^0)-1} \cdot 2 \cdot 3\eta \cdot 2^{\mathbf{depth}(v,T^0)} \cdot |N_{G^0}(T_v^0) \cap S| && \text{Claim 5.14.}\\
\leq (3\eta \cdot 2^{\eta+1})^{\mathbf{height}(T_v^0)} \cdot |N_{G^0}(T_v^0) \cap S| && \mathbf{depth}(v, T^0) \leq \eta.
\end{aligned}$$

Since $f(v) = (\sum_{u \in \mathcal{C}^+} f(u)) + (\sum_{u \in \mathcal{C}^-} f(u))$, combining the bounds for $\mathcal{C}^+$ and $\mathcal{C}^-$ to bound $f(v)$ proves Lemma 5.3 by simple formula manipulation. □

Finally, since a kernelization is an *efficient* preprocessing algorithm, we have to show that the Reduce algorithm can be implemented efficiently.

Lemma 5.4. *The Reduce algorithm can be implemented to run in polynomial time for every fixed η.*

Proof. The nontrivial algorithm tasks that have to be done for one iteration of the algorithm are determining the treedepth of the subgraph induced by a subtree of F, and finding the maximum number of internally vertex-disjoint paths in some subgraph. For every fixed η, the first can be done in polynomial (even linear) time, using for example the FPT algorithm of Reidl et al. [34]. It is well known that the number of internally vertex-disjoint paths can be computed in polynomial time using flow techniques; see, for example, Schrijver [38, Chapter 9]. Since the Reduce algorithm recurses at most once on each child, it follows that the overall runtime is polynomial in the size of the input graph. □

5.3 Final kernelization algorithm

Armed with the decomposition and the reduction algorithm, we can formulate the complete kernelization algorithm for TREEDEPTH-η DELETION.

Theorem. *For each fixed η,* TREEDEPTH-η DELETION *has a polynomial kernel with $\mathcal{O}(k^6)$ vertices: an instance (G, k) can be efficiently reduced to an equivalent instance (G', k) with $2^{\mathcal{O}(\eta^2)}k^6$ vertices.*

Proof. Fix some $\eta \geq 1$. When presented with an input (G, k), the kernelization algorithm proceeds as follows. It first applies Lemma 5.1. If the lemma reports that G has no treedepth-η modulator of size at most k then we output a constant-size NO-instance and terminate. Otherwise we obtain an equivalent instance (G', k) along with a treedepth-η modulator S, a set $Y \subseteq V(G') \setminus S$, a treedepth decomposition F' of G' of height at most η satisfying the conditions outlined in Lemma 5.1. The graph G' consists of S, Y, and the vertices of $G' - (S \cup Y)$. The lemma guarantees that the vertex sets of connected components of $G' - (S \cup Y)$ correspond to the vertex sets of subtrees of F' rooted at members of $\mathcal{T}'$. It also gives a bound on the cardinality of $|\mathcal{T}'|$, by bounding the number of connected components of $G' - (S \cup Y)$. Since $|S|$ and $|Y|$ are already small, to bound the total size of the instance it suffices to shrink each component of $G' - (S \cup Y)$ to size polynomial in k.

To achieve this, we do the following. For each $v \in \mathcal{T}'$ we call Reduce(G', S, F', v, k). From the guarantees of Lemma 5.1, it follows that the invariants outlined in Section 5.2 are satisfied for the first call of Reduce. Since the changes that are made to the graph by the reduction algorithm are local, as formalized in the invariant, when we call Reduce on the next member of $\mathcal{T}'$ the invariants are still initially satisfied. As Invariant (VII) ensures that each transformation yields an instance equivalent to the one we started with, after executing Reduce for each $v \in \mathcal{T}'$ the resulting instance (G', k) is equivalent to the original input. Since Decompose and Reduce both run in polynomial time for fixed η, the entire procedure runs in polynomial time. The resulting instance (G', k) is given as the output of the kernelization. It remains to bound the number of vertices in G'.

The vertex set of the final graph G' consists of S, Y, and whatever is left of the subtrees rooted at $\mathcal{T}'$. By Lemma 5.3, after Reduce has finished processing for $v \in \mathcal{T}'$, the number of leaves in the subtree of F' rooted at v has been reduced to at most $\phi(v) = (3\eta \cdot 2^{\eta+1})^{\textbf{height}(T_v)} \cdot (|S| + 1)$. As the height of any tree in F' is at most η, any leaf has at most $(\eta - 1)$ proper ancestors. Hence the number of vertices in the reduced subtree F'_v is at most η times the number of leaves, so at most

$$\eta \cdot (3\eta \cdot 2^{\eta+1})^{\eta} \cdot (|S| + 1).$$

By combining this bound with the number of connected components of $G' - (S \cup Y)$, which equals $|\mathcal{T}'|$ by Lemma 5.1, we can now obtain a final size bound for G'. Observe that, for fixed η, Lemma 5.1 shows that $|S| \in \mathcal{O}(k)$ and $|Y| \in \mathcal{O}(k^3)$, implying that $|\mathcal{T}'| \in \mathcal{O}(k^5)$. Hence we find:

$$\begin{aligned} |V(G')| &\leq |S| + |Y| + |\mathcal{T}'| \cdot \eta \cdot (3\eta \cdot 2^{\eta+1})^{\eta} \cdot (|S| + 1) \\ &\in \mathcal{O}(k + k^3 + k^5 \cdot k) \in \mathcal{O}(k^6). \end{aligned}$$

A straight-forward computation shows that the number of vertices in the kernel is bounded by $2^{c \cdot \eta^2} \cdot k^6$ for an explicit, small constant c that can be extracted from our arguments. We omit the computations for ease of presentation. This concludes the proof. □

6 Conclusion

In this paper we (re-)studied the Planar $\mathcal{F}$-Minor-Free Deletion problem from the perspective of (uniform) kernelization. We answered the question whether all Planar $\mathcal{F}$-Minor-Free Deletion problems have uniformly polynomial kernels negatively, but showed that the special case Treedepth-η Deletion (which is a Planar $\mathcal{F}$-Minor-Free Deletion problem for every η, where every $\mathcal{F}$ contains a path) has uniformly polynomial kernels.

In a recent paper, Fellows and Jansen [15] analyzed the connection between kernelization algorithms and minor order obstruction sets, suggesting that the sizes of kernels and the sizes of the graphs in related obstruction sets are closely linked. Our results in this paper give another example of this connection. For every k and d, the graphs in which at most k vertices can be deleted to obtain a graph of treewidth at most d are a minor-closed family, and are therefore characterized by a finite construction set $\mathcal{O}_k$. How do the members of this obstruction set relate to kernelization? We proved that the number of vertices in kernels for Treewidth-$(d-1)$ Deletion must be $\Omega(k^{\frac{d}{4}-\epsilon})$ unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. By a simple construction (although different than that of Lemma 3.1), we can prove that there are minor-minimal obstructions that have $\Omega(k^d)$ vertices. For upper bounds, consider the obstruction set $\mathcal{O}'_k$ for the class of graphs whose treedepth can be reduced to η by at most k deletions. By adapting arguments from the kernelization—which is needed to circumvent the need to add edges to the graph—we can prove an upper bound of $2^{\mathcal{O}(\eta^2)}k^6$ on the number of vertices of members of the obstruction set $\mathcal{O}'_k$. Hence in the positive case we also see the connection between kernel sizes and obstruction sizes.

The distinction between uniformly versus non-uniformly polynomial kernels is similar to the distinction between algorithms whose parameter dependence is fixed-parameter tractable (FPT) versus slicewise-polynomial (XP), and opens up a similarly broad area of investigation. The kernelization complexity of $\mathcal{F}$-Minor-Free Deletion is still wide open. Some notable open problems in this direction are:

- Does $\mathcal{F}$-Minor-Free Deletion admit a polynomial kernel for any fixed set $\mathcal{F}$, even when $\mathcal{F}$ contains no planar graphs? Even for the special case of deleting k vertices to get a planar graph (Vertex Planarization), we do not know the answer.
- Consider the graphs that can be made planar by at most k vertex deletions, and the corresponding obstruction set $\mathcal{O}^*_k$ for this family. Can the size of the members of $\mathcal{O}^*_k$ be bounded polynomially in k? By the suggested connection between kernel sizes and obstruction sizes, this may shed light on the kernelization complexity of Vertex Planarization.
- Is it possible to obtain a dichotomy theorem, characterizing the families $\mathcal{F}$ for which Planar $\mathcal{F}$-Minor-Free Deletion admits uniformly polynomial kernels?

These questions are part of a large research program into the complexity of $\mathcal{F}$-Minor-Free Deletion problems, whose importance was recognized by its listing in the *Research Horizons* section of the recent textbook by Downey and Fellows [14, Chapter 33.2].

References

[1] I. Adler, M. Grohe, and S. Kreutzer, *Computing excluded minors*, in Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008), ACM-SIAM, 2008, pp. 641–650. 1

[2] N. Alon, G. Gutin, E. J. Kim, S. Szeider, and A. Yeo, *Solving MAX-r-SAT above a tight lower bound*, Algorithmica, 61 (2011), pp. 638–655. 1

[3] H. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos, *(Meta) Kernelization*, in Proc. 50th FOCS, IEEE, 2009, pp. 629–638. 1, 3

[4] H. L. Bodlaender, *A partial k-arboretum of graphs with bounded treewidth*, Theor. Comput. Sci., 209 (1998), pp. 1–45. 9, 12

[5] ——, *Kernelization: New upper and lower bound techniques*, in Proc. 4th IWPEC, 2009, pp. 17–37. 6

[6] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin, *On problems without polynomial kernels*, J. Comput. Syst. Sci., 75 (2009), pp. 423–434. 1

[7] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch, *Kernel bounds for structural parameterizations of pathwidth*, in Proc. 13th SWAT, 2012, pp. 352–363. 3, 17

[8] ——, *Preprocessing for treewidth: A combinatorial analysis through kernelization*, SIAM J. Discrete Math., 27 (2013), pp. 2108–2142. 1, 7

[9] C. Chekuri and J. Chuzhoy, *Polynomial bounds for the grid-minor theorem*, in Proc. 46th STOC, 2014, pp. 60–69. 2

[10] B. Courcelle, *The monadic second-order logic of graphs III: tree-decompositions, minor and complexity issues*, ITA, 26 (1992), pp. 257–286. 1

[11] M. Cygan, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *On the hardness of losing width*, in Proc. 6th IPEC, 2011, pp. 159–168. 2, 3, 4

[12] H. Dell and D. Marx, *Kernelization of packing problems*, in Proc. 23rd SODA, 2012, pp. 68–81. 3, 7

[13] H. Dell and D. van Melkebeek, *Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses*, J. ACM, 61 (2014), p. 23. 1, 7

[14] R. G. Downey and M. R. Fellows, *Fundamentals of Parameterized Complexity*, Texts in Computer Science, Springer, 2013. 6, 31

[15] M. R. Fellows and B. M. P. Jansen, *FPT is characterized by useful obstruction sets: Connecting algorithms, kernels, and quasi-orders*, TOCT, 6 (2014), p. 16. 31

[16] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Springer-Verlag New York, Inc., 2006. 6

[17] F. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos, *Bidimensionality and kernels*, in Proc. 21st SODA, 2010, pp. 503–510. 1, 3

[18] F. V. Fomin, B. M. P. Jansen, and M. Pilipczuk, *Preprocessing subgraph and minor problems: When does a small vertex cover help?*, J. Comput. System Sci., 80 (2014), pp. 468–495. 4

[19] F. V. Fomin, D. Lokshtanov, N. Misra, G. Philip, and S. Saurabh, *Hitting forbidden minors: Approximation and kernelization*, in Proc. 28th STACS, 2011, pp. 189–200. 3

[20] F. V. Fomin, D. Lokshtanov, N. Misra, and S. Saurabh, *Planar $\mathcal{F}$-Deletion: Approximation, kernelization and optimal FPT algorithms*, in Proc. 53rd FOCS, 2012, pp. 470–479. 1, 2, 3, 4

[21] L. Fortnow and R. Santhanam, *Infeasibility of instance compression and succinct PCPs for NP*, J. Comput. Syst. Sci., 77 (2011), pp. 91–106. 1

[22] J. Gajarský, P. Hlinený, J. Obdrzálek, S. Ordyniak, F. Reidl, P. Rossmanith, F. S. Villaamil, and S. Sikdar, *Kernelization using structural parameters on sparse graph classes*, in Proc. 21st ESA, 2013, pp. 529–540. 3, 4, 6

[23] D. Hermelin and X. Wu, *Weak compositions and their applications to polynomial lower bounds for kernelization*, in Proc. 23rd SODA, 2012, pp. 104–113. 3, 7

[24] B. M. P. Jansen, *Turing kernelization for finding long paths and cycles in restricted graph classes*, Proc. 22nd ESA, (2014). 1

[25] E. J. Kim, A. Langer, C. Paul, F. Reidl, P. Rossmanith, I. Sau, and S. Sikdar, *Linear kernels and single-exponential algorithms via protrusion decompositions*, in Proc. 40th ICALP, 2013, pp. 613–624. 3, 4

[26] S. Kratsch, *Polynomial kernelizations for MIN $F^+\Pi_1$ and MAX NP*, in Proc. 26th STACS, 2009. 1

[27] S. Kratsch, *Recent developments in kernelization: A survey*, Bulletin of the EATCS, 113 (2014), pp. 58–97. 6

[28] S. Kratsch and M. Wahlström, *Compression via matroids: a randomized polynomial kernel for odd cycle transversal*, in SODA, 2012, pp. 94–103. 1

[29] ——, *Representative sets and irrelevant vertices: New tools for kernelization*, in Proc. 53rd FOCS, 2012, pp. 450–459. 1

[30] D. Lokshtanov, N. Misra, and S. Saurabh, *Kernelization - Preprocessing with a guarantee*, in The Multivariate Algorithmic Revolution and Beyond, 2012, pp. 129–161. 6

[31] J. Nesetril and P. Ossona de Mendez, *Tree-depth, subgraph coloring and homomorphism bounds*, Eur. J. Comb., 27 (2006), pp. 1022–1041. 2

[32] R. Niedermeier, *Invitation to Fixed-Parameter Algorithms*, Oxford University Press, 2006. 6

[33] M. Pilipczuk, M. Pilipczuk, P. Sankowski, and E. J. van Leeuwen, *Network sparsification for steiner problems on planar and bounded-genus graphs*, Proc. 55th FOCS, abs/1306.6593 (2013). 1

[34] F. Reidl, P. Rossmanith, F. S. Villaamil, and S. Sikdar, *A faster parameterized algorithm for treedepth*, in Proc. 41st ICALP, 2014, pp. 931–942. 3, 4, 5, 30

[35] N. Robertson and P. D. Seymour, *Graph minors. I. Excluding a forest*, J. Comb. Theory, Ser. B, 35 (1983), pp. 39–61. 2

[36] N. Robertson and P. D. Seymour, *Graph minors. V. Excluding a planar graph*, J. Combin. Theory Ser. B, 41 (1986), pp. 92–114. 2

[37] ——, *Graph minors. XIII. The disjoint paths problem*, J. Combin. Theory Ser. B, 63 (1995), pp. 65–110. 1

[38] A. Schrijver, *Combinatorial Optimization. Polyhedra and Efficiency*, Springer, Berlin, 2003. 21, 30