Approximately Interpolating Between Uniformly and Non-Uniformly Polynomial Kernels

Akanksha Agrawal
Indian Institute of Technology Madras, India

M. S. Ramanujan
University of Warwick, Coventry, UK

Abstract

The problem of computing a minimum set of vertices intersecting a finite set of forbidden minors in a given graph is a fundamental graph problem in the area of kernelization with numerous well-studied special cases.

A major breakthrough in this line of research was made by Fomin et al. [FOCS 2012], who showed that the ρ-Treewidth Modulator problem (delete minimum number of vertices to ensure that treewidth is at most ρ) has a polynomial kernel of size $k^{g(\rho)}$ for some function g. A second standout result in this line is that of Giannapoulou et al. [ACM TALG 2017], who obtained an $f(\eta)k^{\mathcal{O}(1)}$-size kernel (for some function f) for the η-Treedepth Modulator problem (delete fewest number of vertices to make treedepth at most η) and showed that some dependence of the exponent of k on ρ in the result of Fomin et al. for the ρ-Treewidth Modulator problem is unavoidable under reasonable complexity hypotheses.

In this work, we provide an approximate interpolation between these two results by giving, for every $\epsilon > 0$, a $(1+\epsilon)$-approximate kernel of size $f'(\eta, \rho, 1/\varepsilon) \cdot k^{g'(\rho)}$ (for some functions f' and g') for the problem of deciding whether k vertices can be deleted from a given graph to obtain a graph that has elimination distance at most η to the class of graphs that have treewidth at most ρ.

Graphs of treedepth η are precisely the graphs with elimination distance at most $\eta - 1$ to the graphs of treewidth 0 and graphs of treewidth ρ are simply graphs with elimination distance 0 to graphs of treewidth ρ. Consequently, our result "approximately" interpolates between these two major results in this active line of research.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Lossy Kernelization, Treewidth Modulator, Vertex Deletion Problems

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.36

Funding Akanksha Agrawal is partly supported by Science and Engineering Research Board, Startup Grant (SRG/2022/000962), and Veena and Induprakas Keri Faculty Fellowship. M. S. Ramanujan is supported by Engineering and Physical Sciences Research Council (EPSRC) grants EP/V007793/1 and EP/V044621/1.

1 Introduction

Polynomial-time preprocessing is one of the widely used methods to tackle NP-hardness in practice, and the area of *kernelization* has been extremely successful in laying down a mathematical framework for the design and rigorous analysis of preprocessing algorithms for decision problems. The central notion in this area is that of a *kernelization* algorithm (whose output is often called a *kernel*), which is a preprocessing algorithm that runs in polynomial time and transforms a "large" instance of a decision problem into a significantly smaller, but equivalent instance. Over the last decade, the area of kernelization has seen the development of a wide range of tools to design preprocessing algorithms and lower bounds techniques. The reader may find an introduction to the field in [16, 18, 6, 8]. An "efficient

preprocessing algorithm" in this setting is referred to as a *polynomial kernelization* and is simply a kernelization algorithm whose output has size bounded polynomially in a parameter of the input. The central classification task in the area of kernelization is to identify NP-hard problems and associated parameters for which polynomial kernels exist.

One of the most intensively investigated problems in the literature on kernelization is the $\mathcal{F}$-Deletion problem and its special cases, which are well-studied NP-complete problems in their own right. In this problem, $\mathcal{F}$ is a fixed finite family of graphs and one is given a graph G and an integer k as input. The objective is to determine whether at most k vertices can be deleted from G so that the resulting graph is $\mathcal{F}$-minor free (does not contain a minor isomorphic to a graph in $\mathcal{F}$). Well-studied special cases of this problem include Vertex Cover ($\mathcal{F} = \{K_2\}$), Feedback Vertex Set ($\mathcal{F} = \{K_3\}$), Planarization ($\mathcal{F} = \{K_{3,3}, K_5\}$) [21], Diamond Hitting Set ($\mathcal{F} = \{\theta_3\}$) [9], Pathwidth One Vertex Deletion ($\mathcal{F} = \{K_3, T_2\}$) [22], and θ_c-Deletion [14, 10].

A common feature shared by many such well explored special cases of this problem is that $\mathcal{F}$ contains at least one planar graph. Motivated by this, Fomin et al. [11] in an influential work, investigated this restriction of the problem (when the family $\mathcal{F}$ contains at least one planar graph) and demonstrated the existence of a polynomial kernel for every such $\mathcal{F}$. This particular variant of $\mathcal{F}$-Deletion is known in the literature as the Planar $\mathcal{F}$-Deletion problem. More precisely, Fomin et al. showed that for every such family $\mathcal{F}$, there is a function g depending only on $\mathcal{F}$ such that Planar $\mathcal{F}$-Deletion has a polynomial kernel of size bounded by $k^{g(\mathcal{F})}$. This raised the question of designing a *uniformly-polynomial kernel*, i.e., a kernel of size at most $f(\mathcal{F}) \cdot k^c$ for some universal constant c that does not depend on $\mathcal{F}$. Subsequently, Giannopoulou [12] addressed this question and proved that: (i) even when the $\mathcal{F}$-minor free graphs under consideration are the graphs of treewidth at most a constant ρ, one does not expect a uniformly-polynomial kernel for Planar $\mathcal{F}$-Deletion (this problem is called ρ-Treewidth Modulator) unless $\mathsf{NP} \subseteq$ co-NP.), and (ii) if the $\mathcal{F}$-minor free graphs correspond to the class of graphs of *treedepth* at most some constant η (i.e., for the η-Treedepth Modulator problem), then there is indeed a kernel of size $f(\eta) \cdot k^c$ for some constant c independent of η, i.e., there is a uniformly-polynomial kernel. In fact, they obtain a kernel of size $2^{\mathcal{O}(\eta^2)} \cdot k^c$. Note that the treedepth of a graph expresses the number of rounds needed to obtain an empty graph by removing one vertex from every connected component in each round and this parameter upper bounds treewidth.

1.1 Our work

The aforementioned results of Fomin et al. [11] and Giannopoulou et al. [12] are the starting point of this work. Our main goal is to obtain a unifying result "interpolating" between the positive results of both these papers. That is, we would like to obtain an algorithm where, by instantiating various parameters appropriately we can derive both their results as special cases (at least in the qualitative sense contrasting uniformly vs non-uniformly polynomial kernels, i.e., ignoring the precise growth rate of the functions f and g). However, at first glance, it is not even clear what problem such an algorithm could be required to solve. Our first contribution therefore is to introduce a new problem (which we call Deletion (η/ρ)-Elimination) as the ideal candidate for this task. This problem is built on the recently popular notion of elimination distance [3, 4, 1, 17, 13] and is formally described as follows: The input is a graph G, and integer k and the goal is to decide whether there is a set $S \subseteq V(G)$ of size at most k such that $G - S$ has elimination distance at most η to graphs of treewidth ρ?

The notion of elimination distance of a graph G to a graph class $\mathcal{H}$ was introduced by Bulian and Dawar [3] and roughly speaking, it expresses the number of rounds needed to obtain a graph in $\mathcal{H}$ by removing one vertex from every connected component in each round. Notice how elimination distance naturally generalizes treedepth. We refer the reader to Section 2 for formal definitions of these notions. At this point, it is sufficient to know that:

- Graphs of treedepth η are precisely the graphs with elimination distance at most $\eta - 1$ to the graphs of treewidth 0. Therefore, when $\rho = 0$, Deletion (η/ρ)-Elimination is nothing but the $(\eta+1)$-Treedepth Modulator problem.
- Graphs of treewidth ρ are simply graphs with elimination distance 0 to graphs of treewidth ρ. Therefore, when $\eta = 0$, Deletion (η/ρ)-Elimination is simply the ρ-Treewidth Modulator problem.

Thus, if one were to obtain a kernel of size $f(\eta, \rho) \cdot k^{g(\rho)}$ (where the function g is independent of η) for the Deletion (η/ρ)-Elimination problem, then by setting $\eta = 0$ one would obtain the non-uniformly polynomial kernel result of Fomin et al., whereas setting $\rho = 0$ gives the uniformly-polynomial kernel result of Giannopolou et al. [12] (again, ignoring the precise growth rates of f and g). Although we have not settled this question completely in this paper, we provide an *approximate polynomial kernel* [20] for this problem whose error can be made arbitrarily small thus providing an arbitrarily refined interpolation between these two results. Stated formally, we prove the following:

▶ **Theorem 1.** *For every $\rho, \eta \in \mathbb{N}$ and $0 < \varepsilon < 1$, (ρ/η)-Modulator has a $(1+\varepsilon)$-approximate kernel of size $f(\eta, \rho, 1/\varepsilon) \cdot k^{g(\rho)}$ for some functions f and g.*

Approximate kernels are a useful relaxation of the standard notion of kernels and combine well with approximation algorithms. Informally speaking, an α-approximate kernel is a polynomial-time algorithm that given as input a pair (I, k) where I is the problem instance and k is the parameter, outputs an instance (I', k') of the same problem such that $|I'| + k' \leq g(k)$ for some computable function g and any c-approximate solution for the instance I' can be turned in polynomial time into a $(c \cdot \alpha)$-approximate solution for the original instance I. We refer the reader to Section 2 for a formal definition of all related terms.

By instantiating $\eta = 0$ and $\rho = 0$ respectively in Theorem 1, we obtain, for every $0 < \varepsilon < 1$, a $(1+\varepsilon)$-approximate kernel of size at most $f_1(\rho, 1/\varepsilon)k^{g(\rho)}$ for ρ-Treewidth Modulator for some functions f_1 and g, and a $(1+\varepsilon)$-approximate kernel of size at most $f_2(\eta, 1/\varepsilon)k^c$ for ρ-Treewidth Modulator for some function f_2 and a universal constant c independent of η and ε.

2 Preliminaries

We refer to the book of Diestel [7] for standard graph terminology. Whenever the context is clear, we use n and m to denote the number of vertices and the number of edges in the input graph, respectively. The set $\mathcal{C}(G)$ denotes the set of connected components of G. For sets $X, Y \subseteq V(G)$, an X-Y *separator* in G is a set $S \subseteq V(G)$, such that $G - S$ has no $x - y$ path, where $x \in X \setminus S$ and $y \in Y \setminus S$. By $\mathsf{sep}_G(X, Y)$ we denote the size of a minimum sized X-Y separator in G. For a tree T and vertices $u, v \in V(T)$, we denote the unique path between u and v by $\mathsf{Pth}_T(u, v)$. A *rooted tree* is a tree with a special vertex called the *root* of the tree. Consider a rooted tree T with root r. A vertex $t \in V(T) \setminus \{r\}$ is a *leaf* of T if it is a vertex of degree exactly one in T. Moreover, if $V(T) = \{r\}$, then r is the leaf (as well as the root) of T. A vertex which is not a leaf, is a *non-leaf* vertex. By $\mathsf{desc}_T(t)$, we denote the set of all descendants of t in T. The subscript T is dropped when clear from the context.

A *rooted forest* is a forest where each of its connected component is a rooted tree. For a rooted forest F, a vertex $v \in V(F)$ that is not a root of any of its rooted trees is a *leaf* if it is of degree exactly one in F. We denote the set of leaves in a rooted forest by $\mathsf{Lf}(F)$. The *depth*, denoted by $\mathsf{depth}(T)$ of a rooted tree T is the maximum number vertices in a root to leaf path in T. The *depth*, denoted by $\mathsf{depth}(F)$ of a rooted forest is the maximum over the depths of its rooted trees.

Least Common Ancestor-Closure of Sets in Trees. For a rooted tree T and vertex set $M \subseteq V(T)$ the least common ancestor-closure (*LCA-closure*), denoted by $\mathsf{LCA\text{-}closure}_T(M)$, is obtained by the following process. Initially, set $M' = M$. Then, as long as there are vertices x and y in M' whose least common ancestor w is not in M', add w to M'. When the process terminates, output M' as the LCA-closure of M. The following folklore result summarizes the properties of LCA-closures which we will use in this paper.

▶ **Observation 2.** *Let T be a rooted tree, $M \subseteq V(T)$ and $M' = \mathsf{LCA\text{-}closure}_T(M)$. Then $|M'| \leq 2|M|$ and for every connected component C of $T - M'$, $|N(C)| \leq 2$. Moreover, the number of components of $T - M'$ that have exactly 2 neighbors in M' is at most $|M'| - 1$.*

Tree decompositions. A *tree decomposition* of a graph G is a pair (T, β), where T is a tree rooted at r and $\beta : V(T) \to 2^{V(G)}$ that satisfies the following properties: (i) $\bigcup_{t \in V(T)} \beta(t) = V(G)$, (ii) for every edge $\{u, v\} \in E(G)$ there is a node $t \in V(T)$, such that $u, v \in \beta(t)$, and (iii) for every $v \in V(G)$, the graph $T[X_v]$ is a subtree of T, where $X_v = \{t \in V(T) \mid v \in \beta(t)\}$. For $t \in V(T)$, we call $\beta(t)$ the *bag* of t. The sets in $\{\beta(t) \mid t \in V(T)\}$ are called *bags* of (T, β). We refer to the vertices in $V(T)$ as *nodes*. The *width* of the tree decomposition (T, β) is $\max_{t \in V(T)} |\beta(t)| - 1$. The *treewidth* of G, denoted by $\mathsf{tw}(G)$, is the minimum over the widths over all possible tree decompositions of G. For $\rho \in \mathbb{N}$, by $\mathcal{T}_\rho$, we denote the family of graphs of treewidth bounded by ρ.

▶ **Proposition 3.** *Consider a graph G, a tree decomposition (T, β) of G, and a clique $S \subseteq V(G)$ in G. Then, there is $t \in V(T)$, such that $S \subseteq \beta(t)$.*

▶ **Proposition 4.** *For a graph G and distinct non-adjacent vertices $u, v \in V(G)$ such that $\mathsf{sep}_G(\{u\}, \{v\}) \geq \mathsf{tw}(G) + 2$, we have $\mathsf{tw}(G) = \mathsf{tw}(G + \{(u, v)\})$. That is, adding the edge (u, v) does not increase the treewidth of G.*

▶ **Proposition 5.** *Consider an integer ρ and two graphs H_1 and H_2, where:*
1. *$Z = V(H_1) \cap V(H_2)$ is a clique in both H_1 and H_2, and*
2. *$\mathsf{tw}(H_1), \mathsf{tw}(H_2) \leq \rho$.*

Then, $\mathsf{tw}(H_1 \cup H_2) \leq \rho$, where $H_1 \cup H_2$ is the graph with $V(H_1 \cup H_2) = V(H_1) \cup V(H_2)$ and $E(H_1 \cup H_2) = E(H_1) \cup E(H_2)$.

▶ **Proposition 6** ([15])**.** *For a graph G, in time $2^{\mathcal{O}(\mathsf{tw}(G))}|V(G)|$, we can compute a tree decomposition of G of width at most $2 \cdot \mathsf{tw}(G) + 1$.*

Forest embeddings and $(\eta, \mathcal{H})$-decompositions. We define some useful definitions and give some preliminary results regarding forest embeddings and $(\eta, \mathcal{H})$-decompositions.

▶ **Definition 7** (Forest embedding)**.** A *forest embedding* of a graph G is a pair (F, f), where F is a rooted forest and $f : V(G) \to V(F)$ is a bijective function, such that for each $\{u, v\} \in E(G)$, either $f(u)$ is a descendant of $f(v)$, or $f(v)$ is a descendant of $f(u)$. The

depth of the forest embedding (F, f) is the depth of the rooted forest F.[1] The *treedepth* of a graph G, denoted by $\mathsf{td}(G)$, is the minimum over the depths over all possible forest embeddings of G.

The next observation states that for every clique S in G and every forest embedding of G, there is a root-to-leaf path in this forest embedding that contains all the vertices of S.

▶ **Observation 8.** *Consider a graph G, a forest embedding $f : V(G) \to V(F)$ of G into the rooted forest F, and a clique $S \subseteq V(G)$ in G. Then, there is a rooted tree $T \in \mathcal{C}(F)$[2] with root r and a leaf $t \in V(T)$, such that for each $s \in S$, we have $f(s) \in V(\mathsf{Pth}_T(r, t))$. That is, the vertices in S must be mapped to vertices in one root-to-leaf path in a tree of F.*

Next, we recall the notion of *elimination distance* introduced by Bulian and Dawar [3]. We rephrase their definition and introduce notation that will facilitate our presentation in this paper.

▶ **Definition 9** (Elimination Distance and $(\eta, \mathcal{H})$-decompositions)**.** Consider a family $\mathcal{H}$ of graphs and an integer $\eta \in \mathbb{N}$. An $(\eta, \mathcal{H})$-decomposition of a graph G is a tuple $(X, Y, F, f : X \to V(F), g : \mathcal{C}(G[Y]) \to \mathsf{Lf}(F) \cup \{\perp\})$, where (X, Y) is a partition of $V(G)$ and F is a rooted forest of (at most) depth η, such that the following conditions are satisfied:

1. (F, f) is a forest embedding of $G[X]$,
2. each connected component of $G[Y]$ belongs to $\mathcal{H}$, and
3. for a connected component C of $G[Y]$, a vertex $v \in V(C)$, and an edge $\{u, v\} \in E(G)$, either $u \in Y$ or $f(u)$ is a vertex in the unique path in F from r to $g(C)$, where r is the root of the connected component in F containing the vertex $g(C)$.[3]

We say that G admits an $(\eta', \mathcal{H})$-decomposition if there is some $\eta \leq \eta'$, for which there is an $(\eta, \mathcal{H})$-decomposition of G. The *elimination distance* of G to $\mathcal{H}$ (or the $\mathcal{H}$-*elimination distance* of G) is the smallest integer η^* for which G admits an $(\eta^*, \mathcal{H})$-decomposition.

Note that when $\mathcal{H}$ is the class of edgeless graphs, then a graph has elimination distance η to $\mathcal{H}$ precisely when it has treedepth at most η. On the other hand, for every class $\mathcal{H}$, a graph G has elimination distance 0 to $\mathcal{H}$ precisely when $G \in \mathcal{H}$. A graph has elimination distance 0 to graphs of treewidth at most ρ if and only if it has treewidth at most ρ.

Consider an $(\eta, \mathcal{H})$-decomposition $\mathbb{D} = (X, Y, F, f, g)$ of a graph G. We say that X is the *interior part* of $\mathbb{D}$ and Y is the *exterior part* of $\mathbb{D}$. For a leaf $u \in \mathsf{Lf}(F)$, by $\widehat{P}_u^{\mathbb{D}}$ we denote the path from u to r in the tree T, where T is the tree rooted at r in F, containing u. Moreover, by $P_u^{\mathbb{D}}$, we denote the graph $G[\{f^{-1}(w) \mid w \in V(\widehat{P}_u^{\mathbb{D}})\}]$. For a connected component $C \in \mathcal{C}(G[Y])$, by $C_{\mathsf{ext}}^{\mathbb{D}}$ we denote the graph $G[V(C) \cup \{f^{-1}(w) \mid w \in V(\mathsf{Pth}_F(g(C), r))\}]$, where r is the root of the component of F containing $g(C)$. (For the above notations we drop the superscript $\mathbb{D}$, when it is clear from the context.)

▶ **Observation 10.** *If (X, Y, F, f, g) is an $(\eta, \mathcal{H})$-decomposition of a graph G such that $|V(G)| \geq 2$, then there is an $(\eta', \mathcal{H})$-decomposition (X', Y', F', f', g') such that (1) F' is a rooted tree with at least two vertices, (2) $f' : X \to V(F')$ is a bijective function, (3) $g' : \mathcal{C}(G[Y]) \to \mathsf{Lf}(F')$, and (4) $\eta' = \mathsf{depth}(F') \leq \eta + 2$.*

[1] Sometimes we slightly abuse the notation for simplicity, and say that, for every $\beta \geq \alpha$, (F, f) is a forest embedding of depth β, where α is the depth of F.
[2] Recall that $\mathcal{C}(F)$ denotes the set of connected components of F,
[3] If $g(C) = \perp$, then u must belong to Y.

For a graph G and integers $\eta, \rho \in \mathbb{N}$, a set $S \subseteq V(G)$ is a (ρ/η)-modulator, if $G - S$ admits an $(\eta, \mathcal{T}_\rho)$-decomposition. Next we state a pair of results regarding the computation of $(\eta, \mathcal{T}_\rho)$-decompositions in bounded treewidth graphs that use Courcelle's theorem [5] and the Counting Monadic Second Order Logic (CMSO)-expressibility of graphs of treewidth ρ as well as that of elimination distance to CMSO-expressible graph classes.

▶ **Observation 11.** *There is an algorithm that, given a graph G and non-negative integers ρ and η, runs in time $f(\rho, \eta, \mathsf{tw}(G))n^{\mathcal{O}(1)}$ for some computable function f and computes a minimum (ρ/η)-modulator S of G and an $(\eta, \mathcal{T}_\rho)$-decomposition of $G - S$.*

We note that in order to obtain an $(\eta, \mathcal{T}_\rho)$-decomposition of $G - S$ given S, one cannot use a direct invocation of Courcelle's theorem on $G - S$. However, one can use standard self-reducibility arguments using repeated invocations of Courcelle's theorem. In particular, if we can identify a vertex v (which must exist) such that $G - S - \{v\}$ has a $(\eta - 1, \mathcal{T}_\rho)$-decomposition, then we can place v at the "top" of the decomposition, delete it and recurse into each connected component.

▶ **Observation 12.** *Given a graph H, a vertex set Q in H, and integers η' and ρ', in time $\widehat{h}(\eta', \rho', \mathsf{tw}(H)) \cdot |V(H)|^{\mathcal{O}(1)}$, we can correctly do one of the following: i) determine the existence of an $(\eta', \mathcal{T}_{\rho'})$-decomposition $\mathbb{D} = (X, Y, F, f, g)$ of H, such that $Q \subseteq Y$, or ii) conclude that such a decomposition does not exists.*

Parameterized problems and (approximate) kernels. A parameterized problem Π is a subset of $\Gamma^* \times \mathbb{N}$ for some finite alphabet Γ. An instance of a parameterized problem consists of a pair (x, k), where k is called the *parameter*. We assume that k is *given in unary* and hence $k \leq |x|$.

▶ **Definition 13** (Kernelization)**.** Let $\Pi \subseteq \Gamma^* \times \mathbb{N}$ be a parameterized problem and g be a computable function. We say that Π *admits a kernel of size g* if there exists an algorithm referred to as a *kernelization* algorithm (or a *kernel*) that, given $(x, k) \in \Gamma^* \times \mathbb{N}$, outputs in time polynomial in $|x| + k$, a pair $(x', k') \in \Gamma^* \times \mathbb{N}$ such that (a) $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$, and (b) $|x'| + k' \leq g(k)$.

▶ **Definition 14** ([20])**.** A parameterized optimization (minimization or maximization) problem is a computable function $\Pi : \Sigma^* \times \mathbb{N} \times \Sigma^* \to \mathbb{R} \cup \{\pm\infty\}$.

The *instances* of a parameterized optimization problem Π are pairs $(I, k) \in \Sigma^* \times \mathbb{N}$, and a *solution* to (I, k) is simply a string $s \in \Sigma^*$, such that $|s| \leq |I| + k$. The *value* of the solution s is $\Pi(I, k, s)$. Since we only deal with a minimization problem in this work, we state some of the definitions only in terms of minimization problems when the definition for maximization problems is analogous. The parameterized optimization version of Deletion (η/ρ)-Elimination is a minimization problem with the optimization function ρ/η-$\mathsf{Del} : \Sigma^* \times \mathbb{N} \times \Sigma^* \to \mathbb{R} \cup \{\infty\}$ defined as follows.

$$\rho/\eta\text{-}\mathsf{Del}(G, k, S) = \begin{cases} \infty & \text{if } S \text{ is not a solution for } G, \\ \min\{|S|, k+1\} & \text{otherwise.} \end{cases}$$

▶ **Definition 15** ([20])**.** For a parameterized minimization problem Π, the *optimum value* of an instance $(I, k) \in \Sigma^* \times \mathbb{N}$ is $\mathsf{OPT}_\Pi(I, k) = \min_{\substack{s \in \Sigma^* \\ |s| \leq |I| + k}} \Pi(I, k, s)$.

Consequently, for Deletion (η/ρ)-Elimination, we define $\mathsf{OPT}(G, k) = \min_{S \subseteq V(G)} \rho/\eta\text{-}\mathsf{Del}\ (G, k, S)$.

In the above definition, k is effectively a threshold; for solutions of size at most k we care about what their size is, while all solutions of size larger than k are *equally bad*, and are consequently assigned value $k+1$. We point the interested reader to Section 2.1, [20] and Section 3.2, [19] for an in-depth discussion of these definitions and their motivations.

We now recall the other relevant definitions from [20] regarding *approximate kernels*.

▶ **Definition 16** ([20]). Let $\alpha \geq 1$ be a real number and Π be a parameterized minimization problem. An *α-approximate polynomial time preprocessing algorithm* $\mathcal{A}$ for Π is a pair of polynomial-time algorithms. The first one is called the *reduction algorithm*, and computes a map $\mathcal{R}_{\mathcal{A}} : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$. Given as input an instance (I, k) of Π, the reduction algorithm outputs another instance $(I', k') = \mathcal{R}_{\mathcal{A}}(I, k)$.

The second algorithm is called the *solution-lifting algorithm*. This algorithm takes as input an instance $(I, k) \in \Sigma^* \times \mathbb{N}$ of Π, the output instance (I', k') of the reduction algorithm, and a solution s' to the instance (I', k'). The solution-lifting algorithm works in time polynomial in $|I|$,k,$|I'|$,k' and s', and outputs a solution s to (I, k) such that $\frac{\Pi(I,k,s)}{\mathsf{OPT}(I,k)} \leq \alpha \cdot \frac{\Pi(I',k',s')}{\mathsf{OPT}(I',k')}$.

The *size* of a polynomial time preprocessing algorithm $\mathcal{A}$ is a function $\mathsf{size}_{\mathcal{A}} : \mathbb{N} \to \mathbb{N}$ defined as $\mathsf{size}_{\mathcal{A}}(k) = \sup\{|I'| + k' : (I', k') = \mathcal{R}_{\mathcal{A}}(I, k), I \in \Sigma^*\}$.

▶ **Definition 17** (α-approximate kernelization). An *α-approximate kernelization* (or *α-approximate kernel*) for a parameterized optimization problem Π, and real $\alpha \geq 1$, is an α-approximate polynomial time preprocessing algorithm $\mathcal{A}$ for Π such that $\mathsf{size}_{\mathcal{A}}$ is upper bounded by a computable function $g : \mathbb{N} \to \mathbb{N}$. We say that $\mathcal{A}$ is an α-approximate polynomial kernelization if g is a polynomial function.

▶ **Definition 18** ([20]). Let $\alpha \geq 1$ be a real number, and Π be a parameterized minimization problem. An α-approximate polynomial time preprocessing algorithm for Π is said to be *strict* if, for every instance (I, k), reduced instance $(I', k') = \mathcal{R}_{\mathcal{A}}(I, k)$ and solution s' to (I', k'), the solution s to (I, k) output by the solution-lifting algorithm when given s' as input satisfies the following.

$$\frac{\Pi(I, k, s)}{OPT(I, k)} \leq \max\left\{\frac{\Pi(I', k', s')}{OPT(I', k')}, \alpha\right\}$$

The notion of *strictness* in the above direction allows one to "chain" multiple α-approximate preprocessing algorithms to obtain a single α-approximate preprocessing algorithm.

▶ **Definition 19.** A *reduction rule* is simply the reduction algorithm of a polynomial time preprocessing algorithm. The reduction rule *applies* if the output instance of the reduction algorithm is not the same as the input instance.

▶ **Definition 20** ([20]). A reduction rule is said to be *α-safe for* Π if it is the reduction algorithm of a strict α-approximate polynomial time preprocessing algorithm for Π. A reduction rule is *safe* if it is 1-safe.

In the context of our work, it is sufficient for us to argue that all our reduction rules are $(1+\varepsilon)$-safe.

▶ **Definition 21** ([20]). A *polynomial-size approximate kernelization scheme* (PSAKS) for a parameterized optimization problem Π is a family of α-approximate polynomial kernelization algorithms, with one such algorithm for every $\alpha > 1$.

3 The PSAKS for Deletion (η/ρ)-Elimination

3.1 Overview of our algorithm

We begin with an overview of our algorithm. Let (G, k) be the given instance of Deletion (η/ρ)-Elimination. We begin by computing a factor-$\mathcal{O}(1)$-approximate solution S for G (see Lemma 22 in Section 3.2). That is, there is a constant γ such that S is a (ρ/η)-modulator of size at most $\gamma \cdot \mathsf{OPT}$. If $|S| > \gamma k$, then we conclude that the input instance is a no-instance and so, we return a trivial no-instance of size $\mathcal{O}(k)$. Henceforth, we assume that $|S| = \mathcal{O}(k)$. We next compute an $(\eta, \mathcal{T}_\rho)$-decomposition (X, Y, F, f, g) of $G - S$ in polynomial time using Observation 11. By Observation 10, we may assume without loss of generality that (X, Y, F, f, g) is an $(\eta', \mathcal{T}_\rho)$-decomposition of $G - S$ where F is a tree and $\eta' \leq \eta + 2$.

We next execute a series of reduction rules that either delete vertices or add edges between non-adjacent vertices. Each such rule reduces our instance complexity by a polynomial (in n) amount after each application, where the rules are applied in the order in which we state them. In particular, assuming n denotes $|V(G)|$, our algorithm will apply a total of at most $\binom{n}{2} + n$ reduction rules. Moreover each of these rules is either 1-safe or $(1 + \varepsilon)$-safe (see Definition 20), implying that by combining them, we get the required PSAKS.

We now proceed to sketch each reduction rule and give intuitive reasonings behind their correctness. The formal description of the rules and correctness arguments can be found in Section 3.2.

3.1.1 Overview of the reduction rules

Rule 1: Identifying highly-connected pairs. Our first reduction rule adds edges between non-adjacent pairs of vertices in G, that have sufficiently high flow between them. This reduction rule is based on the fact that, roughly speaking, for such pairs of non-adjacent vertices u, v one of the following holds with respect to some solution S^* (i.e., a (ρ/η)-modulator of size at most k) and an $(\eta, \mathcal{T}_\rho)$-decomposition of $G - S^*$, call it $\mathbb{D}$:

- **(i)** At least one of them belongs to S^*.
- **(ii)** Both belong to the interior of $\mathbb{D}$ and are mapped to vertices in a single root-to-leaf path in the associated forest.
- **(iii)** Exactly one of them belongs to a connected component C in the exterior of $\mathbb{D}$ while the other belongs to the interior of the same decomposition and is mapped to a vertex of the associated forest, which is an ancestor of the leaf-vertex with which C is associated.
- **(iv)** Both belong to a bag in any optimal tree decomposition of the exterior part of $\mathbb{D}$.

In any of these cases, the addition of such an edge (u, v) maintains equivalence with the given instance. A similar observation was crucially used in the work of Fomin et al. [11] and this rule (adding edges between highly connected pairs of non-adjacent vertices) was also explicitly used by Giannapolou et al. [12] in their work. The reduction in the instance complexity given by this rule is captured by the fact that the number of non-edges strictly decreases with each application.

Rule 2: Bounding the degree of F (and hence, the size of the interior X) and the number of connected components in $G - (S \cup X \cup \mathsf{Mark}_1)$ where X and F are given by the current decomposition (X, Y, F, f, g) and Mark_1 is a marked set of vertices to be defined. Note that since the depth of F is bounded by η', if we have a bound on the degree of F, then we can obtain a bound on the number of vertices in X. This part of our work is also inspired by arguments used by Fomin et al. [11] and Giannopolou et al. [12] (the latter in the setting of treedepth modulators).

Towards designing rules that help us achieve the required degree bound, we begin by employing the *near-protrusion decomposition* technique of Fomin et al. [11] as follows. We first create a marked set of vertices, denoted by Mark_1, in $G-S$. We do this by adding a minimum separator to our marked set, for every pair of non-adjacent vertices in S. We remark that inapplicability of our first reduction rule ensures that the sizes of such separators are "small". For technical reasons, we will in fact add more vertices to Mark_1 besides what we discussed above and we defer the precise description of these vertices to the next subsection. After we remove $S \cup \mathsf{Mark}_1$ from G, we obtain a set of connected components. We observe that the first reduction rule ensures that the neighborhood of each such connected component inside S is a clique. Moreover, we observe that the neighborhood of each such connected component disjoint from S has size bounded by $\mathcal{O}(\rho+\eta)$. We next analyze these connected components and group them into different classes based on an appropriate notion of equivalence, which depends on the following for each such connected component C:

- Firstly, we have a set $\widehat{S}'$ of at most $\rho+1$ vertices from $S \cap N(C)$. Intuitively speaking $\widehat{S}'$ represents the vertices from S that are not deleted in some hypothetical optimal solution and instead, lie in the exterior part of the associated optimal decomposition (since $S \cap N(C)$ is shown to be a clique it follows that these vertices must be contained in a single bag of any tree-decomposition of the exterior part).
- The second element determining the equivalence class of C will be a set $B \subseteq X \cap N(C)$ of size at most $\eta+1$. This set denotes the vertices in $X \cap N(C)$ that are disjoint from an optimal solution.
- We will further partition the above set B into two sets, B_1 and B_2, that will correspond to the subsets that go to the interior and exterior of an optimal decomposition, respectively.
- Finally we have an integer $\widehat{\eta}$ which will denote the minimum integer for which an $(\widehat{\eta}, \mathcal{T}_\rho)$-decomposition for the connected component exists, such that $\widehat{S}' \cup B_2$ belong to the exterior of this decomposition.

Once we formalize the above equivalence, we show that (a) the number of possible equivalence classes to which C can belong is bounded polynomially in k and (b) if, for the equivalence class containing C we have sufficiently many other connected components, then we can safely reduce the size of the instance by deleting C. Once this is done, we obtain bounds on: i) the number of vertices in X, and ii) the number of connected components in $G-(S \cup X \cup \mathsf{Mark}_1)$.

From this point onwards, in order to obtain our approximate kernel, it is enough to bound the sizes of the connected components in $G-(S \cup X \cup \mathsf{Mark}_1)$. To achieve this, we will in fact bound the sizes of the connected components in $G-(S \cup \mathsf{Mark}_1)$. Towards this, we design our next two reduction rules, out of which the next rule is the only "lossy" reduction rule in our algorithm.

Rule 3: Converting near protrusions to protrusions. Using the terminology of Fomin et al., the connected components of $G-(S \cup \mathsf{Mark}_1)$ are *near protrusions*. Essentially, for each connected component C of $G-(S \cup \mathsf{Mark}_1)$, the graph induced on C has bounded treewidth (in this case, treewidth $O(\rho+\eta)$) and moreover, any (ρ,η)-modulator of size at most k contains all but at most $\kappa(\rho+\eta)$ neighbors of C for some constant κ. We note that in standard terminology, such a connected component C would be a $\mathcal{O}(\rho+\eta)$-protrusion if the neighborhood of C is also bounded by $\mathcal{O}(\rho+\eta)$, in which case one can rely on several "protrusion-reduction" techniques in the literature. However, dealing with near protrusions requires a more careful approach because there could be $k^{\mathcal{O}(\rho+\eta)}$ possible ways in which an optimal solution intersects $N(C)$. As our main aim is to avoid the dependence on η in the exponent of k, this obstacle cannot be overcome only by using techniques from literature

and requires a new approach. The key insight in this part of our work lies in a reduction rule that enables us to delete vertices in such a way that we eliminate near protrusions that are not already $\mathcal{O}(1/\varepsilon(\rho+\eta))$-protrusions. However, this rule is not lossless although the loss ε can be made arbitrarily close to 0 (at a cost of larger kernel size eventually). We next describe the key insight that facilitates this.

Consider a connected component C of $G-(S\cup\mathsf{Mark}_1)$ such that $N(C) > (\kappa/\varepsilon+1)\cdot(\rho+\eta)$ where κ is the constant described in the previous paragraph. Then, we have that by greedily adding $N(C)$ to any solution we eventually compute (i.e., by deleting $N(C)$ from the graph and modifying the budget k accordingly), we will correctly pick at least $(\kappa/\varepsilon)\cdot(\rho+\eta)$ vertices of an optimal solution and incorrectly pick at most an ε fraction of $|N(C)|$ to the solution that is eventually returned. This is the informal reasoning behind the $(1+\varepsilon)$-safeness of this rule. By repeating this rule exhaustively, we arrive at an instance where every component of $G-(S\cup\mathsf{Mark}_1)$ is a $(\kappa/\varepsilon+1)\cdot(\rho+\eta)$-protrusion, i.e., it induces a graph with treewidth at most $(\kappa/\varepsilon+1)\cdot(\rho+\eta)$ and has a boundary of size at most $(\kappa/\varepsilon+1)\cdot(\rho+\eta)$ through which it interacts with the rest of the graph.

Rule 4: Bounding the size of components in $G-(S\cup\mathsf{Mark}_1)$, i.e., protrusion reduction. Here, we use the "lossless protrusion replacer" of Fomin et al. [11] to reduce all remaining $(\kappa/\varepsilon+1)\cdot(\rho+\eta)$-protrusions to a size that depends only on ε,ρ and η. Essentially, this is a subroutine that replaces any sufficiently large $(\kappa/\varepsilon+1)\cdot(\rho+\eta)$-protrusion in $G-(S\cup\mathsf{Mark}_1)$ with a strictly smaller one, repeatedly. In the process, one is guaranteed that any feasible solution for the reduced instance can be used to efficiently compute a feasible solution for the original instance without changing the gap between the feasible solution and the optimum. Since every connected component of $G-(S\cup\mathsf{Mark}_1)$ is a $(\kappa/\varepsilon+1)\cdot(\rho+\eta)$-protrusion following the application of Rule 3, this gives a bound of $f(1/\varepsilon,\rho,\eta)$ (for some function f) on the size of each component of $G'-(S\cup\mathsf{Mark}_1)$ where G' is the graph obtained after exhaustively applying Rule 4.

3.2 Formal description of the algorithm

We now formally describe our kernelization algorithm. The algorithm begins by computing an approximate solution of size at most $\mathcal{O}(k)$, using the following lemma.

▶ **Lemma 22.** *There is an algorithm that, given a graph G and non-negative integers k,η,ρ, runs in time bounded by $\widetilde{h}(\eta+\rho)\cdot n^{\mathcal{O}(1)}$ for a computable function $\widetilde{h}$ and either correctly concludes that (G,k) is a no-instance of* Deletion (η/ρ)-Elimination*, or outputs a (ρ,η)-modulator of G of size at most $\widehat{c}\cdot\log(\rho+\eta)\cdot k$ for some constant $\widehat{c}$.*

If Lemma 22 returns that (G,k) is a no-instance of Deletion (η/ρ)-Elimination, then we return a trivial instance with $\mathcal{O}(k+\eta+\rho)$ vertices and an optimal solution value of at least $k+1$, as our output. Hereafter we assume that the output of Lemma 22 is a (ρ,η)-modulator S for G, of size at most $\widehat{c}\cdot\log(\rho+\eta)\cdot k$. We also assume that $|V(G-S)|\geq 2$, as otherwise, we can return G itself as a kernel with at most $\widehat{c}\cdot\log(\rho+\eta)\cdot k+2$ vertices. Using Observation 11, we compute an $(\eta,\mathcal{T}_\rho)$-decomposition, $\widetilde{\mathbb{D}}=(\widetilde{X},\widetilde{Y},\widetilde{T},\widetilde{f},\widetilde{g})$, of $G-S$ in time $\widehat{\ell}(\rho+\eta)n^{\mathcal{O}(1)}$ for some computable function $\widehat{\ell}$. Notice that using the above $(\eta,\mathcal{T}_\rho)$-decomposition and our assumption that $|V(G-S)|\geq 2$, we can compute an $(\eta',\mathcal{T}_\rho)$-decomposition (Observation 10), $\mathbb{D}=(X,Y,T,f,g)$, of $G-S$, where the following properties are satisfied:

- T is a rooted tree with at least two vertices,
- $f:X\to V(T)$ is a bijective function,
- $g:\mathcal{C}(G[Y])\to\mathsf{Lf}(T)$ and
- $\eta'=\mathsf{depth}(T)\leq\eta+2$.

We let r denote the root of the tree T. For $a \in V(T)$, we let $X_a = \{f^{-1}(a') \mid a' \in V(\mathsf{Pth}_T(a,r))\}$. Furthermore, we let G_a be the graph that appears "below" a, i.e., $V(G_a) = \{v \in X \mid f(v) \in \mathsf{desc}_T(a)\} \cup \{v \in Y \mid v \in V(C),$ where $C \in \mathcal{C}(G[Y])$ and $g(C) \in \mathsf{desc}_T(a)\}$, and $G_a = G[V(G_a)]$.

In the following we formalize the "monotonicity" of the problem in a way that will be useful in our proofs.

▶ **Observation 23.** *For every $\rho, \eta \in \mathbb{N}$, graph H, (ρ,η)-modulator $\widehat{S} \subseteq V(H)$ for H and every (not necessarily induced) subgraph $\widehat{H}$ of H, $\widehat{S} \cap V(\widehat{H})$ is a (ρ,η)-modulator for $\widehat{H}$.*

3.2.1 Identifying highly connected pairs

Let $\boxed{\phi_1 = k + \eta + 6\rho + 1}$. The first reduction rule adds an edge between a pair of (distinct) non-adjacent vertices $u, v \in V(G)$, where $\mathsf{sep}_G(u,v) \geq \phi_1 + 1$. Recall that $\mathsf{sep}_G(u,v)$ denotes the maximum number of internally vertex-disjoint u-v paths in G.

▶ **Reduction Rule 1.** *Let the input be (G,k). If there are distinct non-adjacent vertices $u, v \in V(G)$ such that $\mathsf{sep}_G(u,v) \geq \phi_1 + 1$, then return $(G + \{u,v\}, k)$.*

Notice that this rule can be applied exhaustively in polynomial time. In the next lemma, we analyze the effect of this reduction rule on the input instance.

▶ **Lemma 24.** *Let $u, v \in V(G)$ be non-adjacent vertices such that $\mathsf{sep}_G(u,v) \geq \phi_1 + 1$, and let $G' = G + \{(u,v)\}$. Then, the following hold:*

1. *Every (ρ,η)-modulator for G' is also a (ρ,η)-modulator for G.*
2. *Every (ρ,η)-modulator for G of size at most k is also a (ρ,η)-modulator for G'.*

The main consequence of Lemma 24 is that we can exhaustively apply Reduction Rule 1 without affecting solutions of size at most k for our instance.

In the context of our approximate kernel result, this is formalized below.

▶ **Lemma 25.** *Reduction Rule 1 is 1-safe.*

3.2.2 Bounding the degree of T

In what follows, we assume that Reduction Rule 1 is not applicable and continue to work with the (ρ,η)-modulator S and the $(\eta', \mathcal{T}_\rho)$-decomposition $\mathbb{D} = (X, Y, T, f, g)$, of $G - S$ where r denotes the root of the tree T. For each $C \in \mathcal{C}(G[Y])$, in time $2^{\mathcal{O}(\rho)}n$ we compute a tree decomposition of C, denoted by $\mathbb{T}_C = (T_C, \beta_C)$ of width at most 3ρ, where T_C is a rooted tree, using Proposition 6. To formalize our next marking scheme we recall the notion of "upward closure" for a vertex in $G - S$, which has also been used in previous related work [12, 1].

▶ **Definition 26.** Consider a vertex $v \in V(G - S)$. The *upward closure* of v in $G - S$ is the set $\mathsf{Up\text{-}Cls}(v)$, defined as follows: If $v \in X$, then $\mathsf{Up\text{-}Cls}(v) = X_{f(v)}$ and if $v \in Y$, then $\mathsf{Up\text{-}Cls}(v) = X_{f(g(C))} \cup \beta(b_v^*)$, where b_v^* is an (arbitrarily selected) node from T_C, such that $v \in \beta_C(b_v^*)$ and $\mathbb{T}_C = (T_C, \beta_C)$ is the tree decomposition computed above for the component $C \in \mathcal{C}(G[Y])$ that contains v.

We are now ready to present our first marking scheme.

▶ **Marking Scheme 1.** *Initialize $\mathsf{Mark}_1 = \emptyset$, and perform the following two steps (in the given order).*

1. *For every distinct $u, v \in S$, such that $\{u,v\} \notin E(G)$, compute a minimum $\{u\} - \{v\}$ separator $Z_{u,v}$ in G (in polynomial time), and for each $w \in Z_{u,v}$, add $\mathsf{Up\text{-}Cls}(w)$ to Mark_1.*

2. *For $C \in \mathcal{C}(G[Y])$, i) if $\mathsf{Mark}_1 \cap V(C) \neq \emptyset$, then let $M^*_C = \emptyset$, and ii) otherwise, let $\widehat{M}_C = \{b^*_v \in V(T_C) \mid v \in \mathsf{Mark}_1 \cap V(C) \neq \emptyset\}$, and $M^*_C = \mathsf{LCA\text{-}closure}_{T_C}(\widehat{M}_C)$. For each $C \in \mathcal{C}(G[Y])$ and $b \in M^*_C$, add $\beta_C(b)$ to Mark_1.*

As Reduction Rule 1 is not applicable, we obtain the following bound on $|\mathsf{Mark}_1|$.

▶ **Observation 27.** *We have $|\mathsf{Mark}_1| \leq 2 \cdot \binom{|S|}{2} \cdot \phi_1 \cdot (\eta' + 1) \cdot (\rho + 1)$.*

We define a few notations and give some results that will be useful in designing our next reduction rule. For a graph H and vertex set Q in $V(H)$, we let $\mathsf{elim\text{-}dist}_Q(H)$ the smallest integer λ for which H admits a $(\lambda, \mathcal{T}_\rho)$ decomposition, $\widetilde{\mathbb{D}} = (\widehat{X}, \widehat{Y}, \widehat{F}, \widehat{f}, \widehat{g})$, such that $Q \subseteq \widehat{Y}$, that is, Q appears in the *exterior* of this decomposition. If $Q = \emptyset$, then $\mathsf{elim\text{-}dist}_Q(H)$ is simply the elimination distance of H to $\mathcal{T}_\rho$. Notice that due to Observation 12, we can compute $\mathsf{elim\text{-}dist}_Q(H)$ in time bounded by $\widehat{h}(\eta, \rho, \mathsf{tw}(H)) \cdot |V(H)|^{\mathcal{O}(1)}$. In all our instantiations of H, we will have $\mathsf{tw}(H) = \mathcal{O}(\eta + \rho)$, and hence, the time required to compute $\mathsf{elim\text{-}dist}_Q(H)$ in these cases will be bounded by $\widehat{h}(\eta, \rho) \cdot n^{\mathcal{O}(1)}$

For every set $Z \subseteq V(G) \setminus S$, we define $\widehat{\mathcal{D}}_Z$ to be the set of connected components from $\mathcal{C}(G - (S \cup Z))$ that do not contain a vertex from Mark_1, i.e., $\widehat{\mathcal{D}}_Z = \{D \in \mathcal{C}(G - (S \cup Z)) \mid V(D) \cap \mathsf{Mark}_1 = \emptyset\}$. For such a Z and S, we now define a table $\mathcal{D}$ indexed by tuples comprising an element $s \in S \cup \{\iota\}$, a subset of S of size at most $\rho + 1$, a pair of disjoint subsets of Z plus a number.

▶ **Definition 28.** Fix $Z \subseteq V(G) \setminus S$ and a subset $\mathcal{D} \subseteq \widehat{\mathcal{D}}_Z$. For every $s \in S \cup \{\iota\}$, $S' \subseteq S \setminus \{s\}$ of size at most $\rho + 1$, disjoint subsets B_1, B_2 of Z, and $\widehat{\eta} \in [\eta]_0 \cup \{\infty\}$, $\mathcal{D}[s, S', B_1, B_2, \widehat{\eta}]$ denotes the set of all connected components $D \in \mathcal{D} \subseteq \widehat{\mathcal{D}}_Z$ for which the following hold:

1. if $s \neq \iota$, then $s \in N(V(D))$,
2. $S' \cup B_1 \cup B_2 \subseteq N(V(D))$, and
3. if $\widehat{\eta} \neq \infty$ then $\mathsf{elim\text{-}dist}_{S' \cup B_2}(D) = \widehat{\eta}$, and otherwise, $\mathsf{elim\text{-}dist}_{S' \cup B_2}(D) > \widehat{\eta}$.

The following regarding the elements of $\widehat{\mathcal{D}}_Z$ will be handy.

▶ **Observation 29.** *Consider a set $Z \subseteq V(G) \setminus S$. For each $D \in \widehat{\mathcal{D}}_Z$, the following properties hold:*

1. *$N(V(D)) \cap S$ is a clique and*
2. *$\mathsf{elim\text{-}dist}_\emptyset(D) \leq \eta$.*

We will now describe a marking scheme, where, for a given $Z \subseteq V(G) \setminus S$ and a subset $\mathcal{D} \subseteq \widehat{\mathcal{D}}_B$, will create a subset of $\mathcal{D}$. We remark that for the actual execution of this marking scheme, we select Z and $\mathcal{D}$ later.

▶ **Marking Scheme 2.** *Given $Z \subseteq V(G) \setminus S$ and a subset $\mathcal{D} \subseteq \widehat{\mathcal{D}}_Z$ as input, create a set $\mathsf{Mark}_2[Z, \mathcal{D}]$ as follows:*

1. *Initialize $\mathsf{Mark}_2[Z, \mathcal{D}] = \emptyset$.*
2. *For each $s \in S \cup \{\iota\}$, $S' \subseteq S \setminus \{s\}$ of size at most $\rho + 1$, disjoint subsets B_1, B_2 of Z, and $\widehat{\eta} \in [\eta]_0 \cup \{\infty\}$, add $\min\{|\mathcal{D}[s, S', B_{\mathsf{del}}, B_{\mathsf{int}}, B_{\mathsf{ext}}, \widehat{\eta}]|, \phi_1\}$-many elements from $\mathcal{D}[s, S', B_1, B_2, \widehat{\eta}]$ to $\mathsf{Mark}_2[Z, \mathcal{D}]$.*

Using the above marking scheme, we design our next reduction rule.

▶ **Reduction Rule 2.** *Given $Z \subseteq V(G) \setminus S$ and a subset $\mathcal{D} \subseteq \widehat{\mathcal{D}}_Z$, if $\mathcal{D} \setminus \mathsf{Mark}_2[Z, \mathcal{D}]$ is non-empty, then select $D \in \mathcal{D} \setminus \mathsf{Mark}_2[Z, \mathcal{D}]$ and return $(G - V(D), k)$.*

We say that for a given Z and $\mathcal{D}$, Reduction Rule 2 is *applicable* if $\mathcal{D} \setminus \mathsf{Mark}_2[Z, \mathcal{D}]$ is non-empty. Notice that given $Z \subseteq V(G) \setminus S$ and a subset $\mathcal{D} \subseteq \widehat{\mathcal{D}}_Z$, $|\mathsf{Mark}_2[Z, \mathcal{D}]|$ can be bounded by $(|S|+1)^{\rho+3} \cdot 3^{|Z|} \cdot (\eta+3) \cdot \phi_1$. In all our applications of Reduction Rule 2, we will have that $|Z| \leq \eta' + 1 = \eta + 3$, which implies an upper bound of $\boxed{\phi_2 = (|S|+1)^{\rho+3} \cdot 3^{\eta+3} \cdot (\eta+3) \cdot \phi_1}$ on the size of $|\mathsf{Mark}_2[Z, \mathcal{D}]|$ in all our instantiations of Z. Moreover, it is straightforward to see that for all such Z, $\mathsf{Mark}_2[Z, \mathcal{D}]$ can be computed in time $\mathcal{O}(\phi_2 \cdot n^{\mathcal{O}(1)})$.

The exhaustive application of Reduction Rule 2 essentially says that any element in $\mathcal{D}$ that is "untouched" by Marking Scheme 2 must be deleted. The intuition is that for any component that is deleted by this rule, sufficiently many "similar representatives" are preserved, thus also preserving solutions of size at most k. This is formalized in the following lemma.

▶ **Lemma 30.** *Given $Z \subseteq V(G) \setminus S$ and $\mathcal{D} \subseteq \widehat{\mathcal{D}}_Z$, for which there is a $D \in \mathcal{D} \setminus \mathsf{Mark}_2[Z, \mathcal{D}]$, let $G' = G - V(D)$ be the graph obtained by an application of Reduction Rule 2. Then, a set $S^* \subseteq V(G)$ of size at most k is a (ρ, η)-modulator for G if and only if $S^* \cap V(G')$ is a (ρ, η)-modulator for G'.*

We now argue that Reduction Rule 2 can be used in our approximate kernel.

▶ **Lemma 31.** *Reduction Rule 2 is 1-safe.*

Recall the $(\eta', \mathcal{T}_\rho)$-decomposition $\mathbb{D} = (X, Y, T, f, g)$ of $G - S$ that we currently have. Using Reduction Rule 2, we are able to reduce $|V(T)|$, $|\mathcal{C}(G[Y])|$, and for each $C \in \mathcal{C}(G[Y])$, bound $|\mathcal{C}(C - \mathsf{Mark}_1)|$ by appropriately choosing (several values for) Z and $\mathcal{D}$. This is argued as follows.

Reducing $|V(T)|$ and $|\mathcal{C}(G[Y])|$

Recall that $\mathsf{depth}(T) \leq \eta' = \eta + 2$. Thus, the number of vertices in T is at most $\eta + 2$ times the maximum degree of a vertex in T. We will next focus on bounding degree of vertices in T, in particular, we will bound the maximum number of children for a vertex in T. Consider a vertex $a \in V(T)$. Recall that $X_a = \{f^{-1}(a') \mid a' \in V(\mathsf{Pth}_T(a, r))\}$, where r is the root of T.

We apply Reduction Rule 2, by choosing $Z = X_a$ with the subset $\mathcal{D} = \mathcal{D}_a \subseteq \widehat{\mathcal{D}}_{X_a}$ defined as follows:

- If a is a non-leaf vertex in T, then we set $\mathcal{D}_a = \{D \in \mathcal{C}(G_{a'}) \mid V(D) \cap \mathsf{Mark}_1 = \emptyset, a' \in V(T) \text{ and } a = \mathsf{par}_T(a')\}$.
- Otherwise, a is a leaf of T and we set $\mathcal{D}_a = \{C \in \mathcal{C}(G[Y]) \mid V(D) \cap \mathsf{Mark}_1 = \emptyset \ , \ g(C) = a\}$.

Intuitively speaking, we have defined $\mathcal{D}_a$ as the subset of $\widehat{\mathcal{D}}_{X_a}$ that comprises of those connected components that appear "below" the children of a in T. Moreover, when a is a leaf, $\mathcal{D}_a$ will be those connected components $C \in G[Y]$, for which $g(C) = a$.[4]

Note that $\mathcal{D}_a \subseteq \widehat{\mathcal{D}}_{X_a}$ (see Definition 28 to recall the definition of $\widehat{\mathcal{D}}_Z$ for our choice of $Z = X_a$). Also, notice that the number of children of a in T is bounded by $|\mathcal{D}_a| + |\mathsf{Mark}_1|$. Recall that for $a \in V(a)$, $|X_a| \leq \eta$. This together with Lemma 22 and Lemma 30 implies the following result.

[4] Recall our assumption that for each $C \in G[Y]$, $g(C) \in \mathsf{Lf}(T)$.

▶ **Observation 32.** *For each $a \in V(T)$, in time bounded by $(k \cdot \log(\eta+\rho))^{\mathcal{O}(\rho)} \cdot 2^{\mathcal{O}(\eta)} \cdot n^{\mathcal{O}(1)}$ we can either apply Reduction Rule 2 for $Z = X_a$ and $\mathcal{D} = \mathcal{D}_a$, or correctly conclude that it is not applicable. Moreover, if for no $a \in V(T)$, the reduction rule is applicable, then $|V(T)| \leq \eta' \cdot (\phi_2 + |\mathsf{Mark}_1|) = (k \cdot \log(\eta+\rho))^{\mathcal{O}(\rho)} \cdot 2^{\mathcal{O}(\eta)}$ and $|\mathcal{C}(G[Y])| \leq |V(T)| \cdot (\phi_2 + |\mathsf{Mark}_1|) = (k \cdot \log(\eta+\rho))^{\mathcal{O}(\rho)} \cdot 2^{\mathcal{O}(\eta)}$.*

Note that to obtain our kernel, it is enough to bound the number of vertices appearing in the connected components in $G[Y]$ by our stated bound. To this end, we will first bound the number of connected components in $G[Y] - \mathsf{Mark}_1$ by this bound. Due to Observation 32, we have already bounded the number of connected components in $\mathcal{C}(G[Y])$ and therefore, to achieve the above, it is enough to bound, for each $C \in \mathcal{C}(G[Y])$, the number of connected components in $C - \mathsf{Mark}_1$.

Bounding $|\mathcal{C}(C - \mathsf{Mark}_1)|$, for $C \in G[Y]$

Consider a connected component $C \in G[Y]$. Recall that we have already computed a tree decomposition $\mathbb{T}_C$ of width at most $6 \cdot \rho$ for C. If $V(C) \cap \mathsf{Mark}_1 = \emptyset$, then $|\mathcal{C}(C - \mathsf{Mark}_1)| = 1$. Thus we hereafter assume that $V(C) \cap \mathsf{Mark}_1 \neq \emptyset$. The above implies that $M^*_C \neq \emptyset$ (see Marking Scheme 1). By the construction of Mark_1, notice that $\mathsf{Mark}_1 \cap V(C) = \cup_{b \in M^*_C} \beta_C(b)$. Let $\mathcal{H}_C$ be the set of connected components in $T_C - M^*_C$, i.e., $\mathcal{H}_C = \mathcal{C}(T_C - M^*_C)$. For each $H \in \mathcal{H}_C$, let $\widetilde{\mathcal{D}}_H$ be the set of connected components of $G[V_H]$, where $V_H = (\cup_{b \in V(H)} \beta_C(b)) \setminus \mathsf{Mark}_1$. For $H \in \mathcal{H}_C$, (intuitively speaking) we let $Z[H]$ be set of vertices that appear in the bags that neighbor H in T_C and the vertices from X, that appear in the root-to-$g(C)$ path in F (recall that $g(C) \in \mathsf{Lf}(F)$). Formally, for $H \in \mathcal{H}_C$, let $Z[H] = (\cup_{b \in N_T(V(H))} \beta_C(b)) \cup X_{g(C)}$. We next summarize some properties of $\mathcal{H}_C$ and the sets we constructed above, which follows from their definitions and Observation 2.

▶ **Observation 33.** *We have $|\mathcal{H}_C| \leq 2|\mathsf{Mark}_1|$, and for each $H \in \mathcal{H}_C$, the following holds:*
1. $|N_T(V(H))| \leq 2$,
2. $Z[H] \subseteq \mathsf{Mark}_1 \cup X \subseteq V(G) \setminus S$, *where* $|Z[H]| \leq 2 \cdot (6\rho+1) + \eta' = 2 \cdot (6\rho+1) + \eta + 2$,
3. *for each* $D \in \widetilde{\mathcal{D}}_H$, $N(V(D)) \subseteq S \cup Z[H]$,
4. $\widetilde{\mathcal{D}}_H \subseteq \widehat{\mathcal{D}}_{Z[H]}$, *and*
5. *for each* $D \in \mathcal{C}(C - \mathsf{Mark}_1)$, *there is some* $H \in \mathcal{H}_C$, *such that* $D \in \widetilde{\mathcal{D}}_C$.

From the above observation, to bound the size of $\mathcal{C}(C - \mathsf{Mark}_1)$, it is enough to bound $|\widetilde{\mathcal{D}}_H|$, for each $H \in \mathcal{H}_C$. For each $H \in \mathcal{H}_C$, we use Reduction Rule 2 (if applicable), for $Z = Z[H]$ and $\mathcal{D} = \widetilde{\mathcal{D}}_H$. The above, together with Lemma 30 implies the following.

▶ **Observation 34.** *For each $C \in \mathcal{C}(G[Y])$ and $H \in \mathcal{H}_C$, in time bounded by $(k \cdot \log(\eta+\rho))^{\mathcal{O}(\rho)} \cdot 2^{\mathcal{O}(\eta)} \cdot n^{\mathcal{O}(1)}$, we can either apply Reduction Rule 2 for $Z = Z[H]$ and $\mathcal{D} = \widetilde{\mathcal{D}}_H$, or correctly conclude that it cannot be applied. Moreover, if for no $C \in \mathcal{C}(G[Y])$ and $H \in \mathcal{H}_C$, the reduction rule is applicable, $|\mathcal{C}(G[Y] - \mathsf{Mark}_1)|$ can be bounded by $(k \cdot \log(\eta+\rho))^{\mathcal{O}(\rho)} \cdot 2^{\mathcal{O}(\eta)}$.*

That is, if we have exhaustively applied the first two reduction rules, then we will have bounded $|X|$ and the number of connected components in $G[Y] - \mathsf{Mark}_1$ by $f(\eta, \rho) \cdot k^{\mathcal{O}(\rho)}$ for some comptuable function f. In the rest of the paper, our goal is to bound the size of the components in $G[Y] - \mathsf{Mark}_1$.

3.2.3 Converting components in $G - (S \cup \mathsf{Mark}_1)$ to protrusions

We begin by arguing that for any component C of $G - (S \cup \mathsf{Mark}_1)$, any (ρ, η)-modulator contains "almost all" neighbors of C inside S.

▶ **Lemma 35.** *For every $C \in (G - (S \cup \mathsf{Mark}_1))$, any (ρ, η)-modulator for G must contain at least $|N(C) \cap S| - \eta - \rho - 2|$ vertices of $N(C) \cap S$.*

We have already noted that every component in $(G-(S\cup\mathsf{Mark}_1)$ has at most $2\rho+\eta'+1 = 2\rho+\eta+3$ neighbors in Mark_1 (see discussion following Observation 27). This fact, plus Lemma 35 motivate the following reduction rule.

▶ **Reduction Rule 3.** *If there is a $C \in (G - (S \cup \mathsf{Mark}_1))$ such that $|N(C)| \geq (\lceil 1/\varepsilon \rceil + 1) \cdot (3\rho + 2\eta + 5)$, then return $(G - (N(C), k - (|N(C)| - 3\rho - 2\eta - 5))$. That is, delete $N(C)$ and reduce the parameter by $|N(C)| - 3\rho - 2\eta - 5|$.*

We note that if Reduction Rule 3 is not applicable, then every connected component in $(G-(S\cup\mathsf{Mark}_1)$ has a neighborhood of size at most $\xi = \mathcal{O}(1/\varepsilon(\rho+\eta))$. The intuition behind the this reduction rule is that if $N(C)$ is sufficiently large compared to $3\rho + 2\eta + 5$, then, any (ρ, η)-modulator for G contains "most" vertices of $N(C)$ and we can add all vertices of $N(C)$ to the solution while charging the few error vertices to the many "correct" vertices.

▶ **Lemma 36.** *Reduction Rule 3 is $(1 + \varepsilon)$-safe.*

3.2.4 Reducing protrusions in $G - (S \cup \mathsf{Mark}_1)$

At this point, we have that every component of $G - (S \cup \mathsf{Mark}_1)$ has treewidth at most $\eta + \rho + 1 \leq \xi$ and moreover, has neighborhood at most ξ. In other words, every component of $G - (S \cup \mathsf{Mark}_1)$ is a ξ-protrusion, where ξ is a constant depending only on $1/\varepsilon, \rho, \eta$. In what follows, we use tools from protrusion reduction and replacement to reduce the size of each component of $G - (S \cup \mathsf{Mark}_1)$. In particular, we use the *lossless protrusion replacer* of Fomin et al. [11].

▶ **Definition 37** ([11]; Lossless Protrusion Replacer)**.** A lossless protrusion replacer for a min-CMSO vertex subset problem Π is a family of algorithms, with one algorithm for every constant r. The r'th algorithm has the following specifications. There exists a constant r' (which depends on r) such that given an instance G and an r-protrusion X in G of size at least r', the algorithm runs in time $\mathcal{O}(|X|)$ and outputs an instance G' with the following properties:

1. G' is obtained from G by replacing X by a r-boundaried graph X' with fewer than r' vertices and thus $|V(G')| < |V(G)|$.
2. $\mathsf{OPT}(G') \leq \mathsf{OPT}(G)$.
3. There is an algorithm that runs in $\mathcal{O}(|X|)$ time and given a feasible solution S' to G' outputs a set $X^* \subseteq X$ such that $S = (S' \setminus X') \cup X^*$ is a feasible solution to G and $|S| \leq |S'| + \mathsf{OPT}(G) - \mathsf{OPT}(G')$.

Since graphs admitting an $(\eta, \mathcal{T}_\rho)$-decomposition are minor-closed, CMSO-expressible and exclude at least one planar graph as a minor (e.g., an $(\eta+\rho+2) \times (\eta+\rho+2)$ grid), it follows from [11] that the Deletion (η/ρ)-Elimination problem has a lossless protrusion replacer. Moreover, using reasoning similar to that in Lemmas 25, 31 and 36 (with the solution-lifting algorithm given by the third property listed in Definition 37), we conclude that this reduction is 1-safe. Hence, we infer that by exhaustively invoking this subroutine with $r = \chi$, we are able to reduce the size of every connected component in $G - (S \cup \mathsf{Mark}_1)$ to some r' which is a function of r, which in turn is only a function of $1/\varepsilon, \rho$ and η. This gives us a bound of $f(\eta, \rho, 1/\varepsilon) \cdot k^{g(\rho)}$ on the size of the final reduced instance (for some functions f and g), proving Theorem 1.

4 Conclusion and future work

The central follow-up question to our work specifically on Deletion (η/ρ)-Elimination is whether the approximation in our result can be avoided and instead a "standard" polynomial kernel of size $f(\eta,\rho)k^{g(\rho)}$ can be obtained for the problem.

On a broader level, our work motivates the interpolation between other results centered around treedepth and treewidth in parameterized complexity. For example, consider a graph problem that has an $f(td)\cdot n^{\mathcal{O}(1)}$ algorithm parameterized by the treedepth td for some function f, but has a best-possible (or best-known) algorithm with running time $n^{g(tw)}$ for some function h when parameterizing by the treewidth tw (e.g., Grundy Coloring [2]). Our work naturally motivates the study of these problems with the aim of obtaining algorithms with running time $f(\eta)\cdot n^{g(\rho)}$ on graphs that have elimination distance η to the class $\mathcal{T}_\rho$. Such algorithms would provide interpolations between known results that are based on these well-studied width measures.

References

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

2 Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, and Yota Otachi. Grundy distinguishes treewidth from pathwidth. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPIcs*, pages 14:1–14:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ESA.2020.14.

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

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

5 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

6 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

7 Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 3rd edition, 2005.

8 Rodney G Downey and Michael Ralph Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012.

9 Samuel Fiorini, Gwenaël Joret, and Ugo Pietropaoli. Hitting diamonds and growing cacti. In *Integer Programming and Combinatorial Optimization, 14th International Conference, IPCO 2010, Lausanne, Switzerland, June 9-11, 2010. Proceedings*, pages 191–204, 2010. doi:10.1007/978-3-642-13036-6_15.

10 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, Geevarghese Philip, and Saket Saurabh. Hitting forbidden minors: Approximation and kernelization. *SIAM J. Discrete Math.*, 30(1):383–410, 2016. doi:10.1137/140997889.

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

12 Archontia C. Giannopoulou, P. Jansen Bart M. Daniel Lokshtanov, and Saket Saurabh. Uniform kernelization complexity of hitting forbidden minors. *ACM Trans. Algorithms*, 13(3):35:1–35:35, 2017. doi:10.1145/3029051.

13 Bart M. P. Jansen, Jari J. H. de Kroon, and Michal Wlodarczyk. Vertex deletion parameterized by elimination distance and even less. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1757–1769. ACM, 2021. doi:10.1145/3406325.3451068.

14 Gwenaël Joret, Christophe Paul, Ignasi Sau, Saket Saurabh, and Stéphan Thomassé. Hitting and harvesting pumpkins. *SIAM J. Discrete Math.*, 28(3):1363–1390, 2014. doi:10.1137/120883736.

15 Tuukka Korhonen. Single-exponential time 2-approximation algorithm for treewidth. In *FOCS*, 2021.

16 Stefan Kratsch. Recent developments in kernelization: A survey. *Bulletin of the EATCS*, 113, 2014.

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

18 Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Kernelization – Preprocessing with a guarantee. In *The Multivariate Algorithmic Revolution and Beyond*, pages 129–161. Springer, 2012.

19 Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Lossy kernelization. *CoRR*, abs/1604.04111, 2016. arXiv:1604.04111.

20 Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Lossy kernelization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 224–237, 2017. doi:10.1145/3055399.3055456.

21 Dániel Marx and Ildikó Schlotter. Obtaining a planar graph by vertex deletion. *Algorithmica*, 62(3-4):807–822, 2012. doi:10.1007/s00453-010-9484-z.

22 Geevarghese Philip, Venkatesh Raman, and Yngve Villanger. A quartic kernel for pathwidth-one vertex deletion. In *Graph Theoretic Concepts in Computer Science - 36th International Workshop, WG 2010, Zarós, Crete, Greece, June 28-30, 2010 Revised Papers*, pages 196–207, 2010. doi:10.1007/978-3-642-16926-7_19.