A Polynomial Kernel for Deletion to the Scattered Class of Cliques and Trees

Ashwin Jacob
National Institute of Technology Calicut, Kozhikode, India

Diptapriyo Majumdar
Indraprastha Institute of Information Technology Delhi, New Delhi, India

Meirav Zehavi
Ben-Gurion University of The Negev, Beersheba, Israel

Abstract

The class of graph deletion problems has been extensively studied in theoretical computer science, particularly in the field of parameterized complexity. Recently, a new notion of graph deletion problems was introduced, called *deletion to scattered graph classes*, where after deletion, each connected component of the graph should belong to at least one of the given graph classes. While fixed-parameter algorithms were given for a wide variety of problems, little progress has been made on the kernelization complexity of any of them. Here, we present the first non-trivial polynomial kernel for one such deletion problem, where, after deletion, each connected component should be a clique or a tree - that is, as dense as possible or as sparse as possible (while being connected). We develop a kernel of $\mathcal{O}(k^5)$ vertices for the same.

2012 ACM Subject Classification Mathematics of computing → Graph theory; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Parameterized Complexity, Kernelization, Scattered Graph Classes, New Expansion Lemma, Cliques or Trees Vertex Deletion

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2024.41

Related Version *Full Version*: `https://arxiv.org/abs/2409.14209`

Funding *Diptapriyo Majumdar*: Supported by Science and Engineering Research Board (SERB) grant SRG/2023/001592.
Meirav Zehavi: Supported by European Research Council (ERC) starting grant titled PARAPATH.

1 Introduction

Graph modification problems form one of the most fundamental problem classes in algorithms and graph theory. The input instance of a graph modification problem consists of an undirected/directed graph, and a non-negative integer k, and the objective is to decide if there exists a set of at most k vertices/edges/non-edges whose deletion/addition yields in a graph belonging to some special graph class. A specific graph modification problem allows to perform a specific graph operation, usually being *vertex deletion* or *edge deletion* or *edge addition* or *edge editing*. Each of these operations, and vertex-deletion in particular, have been extensively studied from the perspective of classical and parameterized complexity. For example, some vertex deletion graph problems that have received intense attention in the past three decades include VERTEX COVER, FEEDBACK VERTEX SET, CLUSTER VERTEX DELETION SET, INTERVAL VERTEX DELETION SET, CHORDAL VERTEX DELETION SET, and more (see [3, 6, 7, 8, 9, 14, 17, 20, 32, 28, 38]).

A graph class Π is said to be *hereditary* if Π is closed under taking induced subgraphs. There are several hereditary graph classes Π such that the corresponding Π-VERTEX DELETION problem is well-studied. Such examples include all of the problems listed above.

Formally, the Π-Vertex Deletion is defined as follows: given a graph G and a non-negative integer k, we ask whether G contains at most k vertices whose deletion results in a graph belonging to class Π. Lewis and Yannakakis [35] proved that for every non-trivial Π, the Π-Vertex Deletion problem is NP-complete. Later, Cai [5] has proved that if a hereditary graph class Π can be described by a finite set of forbidden subgraphs containing all minimal forbidden subgraphs in the class, then vertex deletion to Π becomes fixed-parameter tractable (FPT).

Most of the computational problems that are NP-hard in general graphs can often be solved in polynomial time when restricted to special graph classes. For example, Vertex Cover is NP-hard on general graphs, but can be solved in polynomial time in forests, bipartite graphs, interval graphs, chordal graphs, claw-free graphs, and bounded treewidth graphs (see [12, 13, 23, 37]). Additionally, several other graph theoretic problems have also been studied in special graph classes (see [2, 4, 11, 18, 22, 24, 30, 31, 33, 36] for some examples). If your input graph is such that each of its connected components belong to one of those special graph classes where the problem is solvable, then the problem can be solved by solving it over each component of the graph. Therefore, such a graph where each of the components belong to different graph classes are interesting. We say that such graphs belong to a scattered graph class. Vertex deletion problems are useful to find a set of few vertices whose removal results in a graph class where the problem of our interest is tractable. Since the same problem is tractable in scattered graph classes (i.e. tractable in each of the graph class), vertex deletion to scattered graph classes are interesting to look at as well.

Many of the graph classes can be characterized by a set of forbidden graphs [15, 34, 23, 10]. Vertex deletion problems for such graph classes boils down to hitting such forbidden subgraphs occuring as induced subgraphs of the input graph. Unlike this, for deletion to a scattered graph class, the deletion set X might separate the vertices of the union of the forbidden subgraphs for each of the graph classes (instead of hitting them) so that all such graphs do not occur in any of the connected components of the graph $G - X$. This ramps up the difficulty for coming up with FPT, approximation and kernelization algorithms for deletion to scattered graph classes. A naive approach of finding the solutions (or kernels) for each of the deletion problems separately and "combining" them is unlikely to work.

Ganian et al. [21] studied backdoors to scattered classes of CSP problems. Subsequently, Jacob et al. [25, 26] built on the works by Ganian et al. [21] and initiated a systematic study of vertex deletion to scattered graph classes. They considered the $(\Pi_1, \ldots, \Pi_d)$-Deletion problem where the input instance is a graph G a parameter k with respect to d (constant) hereditary graph classes $\Pi_1, \ldots, \Pi_d$. The objective is to decide if there is a set of at most k vertices S such that every connected component of $G - S$ is in Π_i for some $i \in [d]$. After that, Jacob et al. [26] considered specific pairs of hereditary graph classes Π_1 and Π_2 and have provided singly exponential-time fixed-parameter tractable (FPT) algorithms and approximation algorithms for (Π_1, Π_2)-Deletion problems. Very recently, Jansen et al. [29] conducted a follow-up work on $(\Pi_1, \ldots, \Pi_d)$-Deletion problems and have improved the results appearing in [25]. A common theme for the FPT algorithms for deletion to scattered graph classes is a non-trivial "unification" of the techniques used in the deletion problems of each of the graph classes.

Our Problem and Results. To the best of our knowledge, vertex deletion to scattered graph classes is essentially unexplored from the perspective of polynomial kernelization that is a central subfield of parameterized complexity. The only folklore result that follows from Jacob et al. [26] states that if there are two hereditary graph classes Π_1 and Π_2 such that both Π_1 and Π_2 can be described by finite forbidden families and P_d (the induced path of d vertices) is a forbidden induced subgraph for Π_1 for some fixed constant d, then the problem (Π_1, Π_2)-Deletion can be formulated as a d-Hitting Set problem and hence admits a polynomial kernel. This folklore result is very restrictive and does not capture any hereditary graph class whose forbidden sets are not bounded by a fixed constant.

In this paper, we initiate the study of vertex deletion to scattered graph classes from the perspective of polynomial kernelization. We consider the problem Cliques or Trees Vertex Deletion where given a graph G and a non-negative integer k, we ask if G contains a set S of at most k vertices, such that $G - S$ is a simple graph and every connected component of it is either a clique or a tree – that is, as dense as possible or as sparse as possible (while being connected). Naturally, we are specifically interested in the case where the input graph is already a simple graph. However, our preprocessing algorithm can produce intermediate multigraphs. Hence, we directly consider this more general formulation. Formally, we define our problem as follows.

Cliques or Trees Vertex Deletion (CTVD) **Parameter:** k
Input: An undirected (multi)graph $G = (V, E)$ and a non-negative integer k.
Question: Does G contain a set S of at most k vertices such that $G - S$ is a simple graph and every connected component of $G - S$ is either a clique or a tree?

This problem is particularly noteworthy as it captures the essence of scattered classes: allowing the connected components to belong to vastly different graph classes and ideally the simplest ones where various computational problems are polynomial-time solvable. Here, we indeed consider the extremes: the simplest densest graph (cliques) and the most natural class of sparsest connected graphs (trees). If X is a feasible solution to Cliques or Trees Vertex Deletion for a graph G, then we call X a *(clique, tree)-deletion set* of G. We consider the (upper bound on the) solution size k as the most natural parameter. Jacob et al. [26] proved that Cliques or Trees Vertex Deletion is in FPT - specifically, that it admits an algorithm that runs in $\mathcal{O}^*(4^k)$-time. In this paper, we prove the following result on the polynomial kernelization for this problem.

▶ **Theorem 1.1.** Cliques or Trees Vertex Deletion *(CTVD) admits a kernel with $\mathcal{O}(k^5)$ vertices.*

Our theorem is the first non-trivial result on a polynomial kernel for vertex deletion to pairs of graph classes. The proof of this kernelization upper bound is based on several non-trivial insights, problem specific reduction rules, and structural properties of the solutions.

Organization of the Paper. In Section 2, we introduce basic terminologies and notations. Section 3 is devoted to the proof of our main result (Theorem 1.1). Finally, in Section 4, we conclude with some future research directions.

2 Preliminaries

We use standard graph theoretic terminologies from Diestel's book [15]. For a vertex v in G, let $d_G(v)$ denote the degree of v in G, which is the number of edges in G incident to v. When we look at the number of edges incident on v, we take the multiplicity of every edge into account. A *pendant vertex* in a graph G is a vertex having degree one in G. A *pendant edge* in a graph G is the edge incident to a pendant vertex in G. A *path* P in a graph is a sequence of distinct vertices $(v_1, \ldots, v_r)$ such that for every $1 \leq i \leq r - 1$, $v_i v_{i+1}$ is an edge.

A *degree-2-path* in G is a path P such that all its internal vertices have degree exactly 2 in G. If a graph G has a degree-2-path $P = (v_1, \ldots, v_r)$ such that v_1 is a pendant vertex, for every $i \in \{2, \ldots, r-1\}$, $d_G(v_i) = 2$ and $d_G(v_r) > 2$, then we call P a *degree-2-tail of length* r. If G has a degree-2-path $P = (v_1, \ldots, v_r)$ such that for all $i \in \{2, \ldots, r-1\}$, $d_G(v_i) = 2$, and $d_G(v_1), d_G(v_r) > 2$, then we call P a *degree-2-overbridge of length* r. Sometimes, for simplicity, we use $P = v_1 - v_2 - \ldots - v_r$ to denote the same path P of r vertices. The graph K_t for integer $t \geq 1$ is the clique of t vertices. For a non-negative integer c, the graph cK_1 is the collection of c isolated vertices. The graphs C_t and P_t for integer $t \geq 1$ are the cycle and path of t vertices respectively. We define a *paw graph* as the graph with four vertices u_1, u_2, u_3 and u_4 where u_1, u_2, u_3 form a triangle, and u_1 alone is adjacent to u_4 (thus, u_4 is a pendant vertex). We define a *diamond graph* as the graph with four vertices u_1, u_2, u_3 and u_4 where u_1, u_2, u_3 form a triangle, and u_1, u_2 are adjacent to u_4. Note that both paw and diamond contain a triangle as well as a $2K_1$ as induced subgraphs. We say that a vertex $x \in V(G)$ is *adjacent* to a subgraph $G[Y]$ for some $x \notin Y$ if x has a neighbor in Y in the graph G. A *cut* of G is a bipartition (X, Y) of $V(G)$ into nonempty subsets X and Y. The set $E_G(X, Y)$ is denoted as the edges *crossing the cut.* We omit the subscript when the graph is clear from the context. Let C be a cycle having r vertices We use $C = v_1 - v_2 - \ldots - v_r - v_1$ to denote the cycle with edges $v_i v_{i+1}$ for every $1 \leq i \leq r-1$ and the edge $v_r v_1$.

▶ **Definition 2.1** (v-flower). *For a graph G and a vertex v in G, a* v-flower *is the structure formed by a family of ℓ cycles $C_1, C_2, \ldots C_\ell$ in G all containing v and no two distinct cycles C_i and C_j sharing any vertex except v. We refer to the C_is' as the* petals *and to v as the* core*. The number of cycles ℓ is the* order *of the v-flower.*

▶ **Proposition 2.2** ([13], Lemma 9.6). *Given a graph G with $v \in V(G)$ and an integer k, there exists a polynomial-time algorithm that either provides a v-flower of order $k+1$ or compute a set $Z \subseteq V(G) \setminus \{v\}$ with at most $2k$ vertices satisfying the following properties: Z intersects every cycle of G that passes through v, and there are at most $2k$ edges incident to v and with second endpoint in Z.*

Let q be a positive integer and G be a bipartite graph with vertex bipartition (A, B). For $\widehat{A} \subseteq A$ and $\widehat{B} \subseteq B$, a set $M \subseteq E(G)$ of edges is called a *q-expansion* of $\widehat{A}$ into $\widehat{B}$ if

(i) every vertex of $\widehat{A}$ is incident to exactly q edges of M, and

(ii) exactly $q|\widehat{A}|$ vertices of $\widehat{B}$ are incident to the edges in M.

The vertices of $\widehat{A}$ and $\widehat{B}$ that are the endpoints of the edges of M are said to be *saturated* by the q-expansion M. We would like to clarify that by definition of q-expansion M of $\widehat{A}$ into $\widehat{B}$, all vertices of $\widehat{A}$ are saturated by M, and $|\widehat{B}| \geq q|\widehat{A}|$. But not all vertices of $\widehat{B}$ are guaranteed to be saturated by M.

▶ **Lemma 2.3** (q-Expansion Lemma [39, 13]). *Let $q \in \mathbb{N}$ and G be a bipartite graph with vertex bipartition (A, B) such that $|B| \geq q|A|$, and there is no isolated vertex in B. Then, there exist non-empty vertex sets $X \subseteq A$ and $Y \subseteq B$ such that*

(i) *there is a q-expansion M of X into Y, and*

(ii) *no vertex in Y has a neighbor outside X, that is, $N(Y) \subseteq X$.*

Furthermore, the sets X and Y can be found in time $\mathcal{O}(mn^{1.5})$.

Recently, Fomin et al. [19] have designed the following generalization of the Lemma 2.3 (q-Expansion Lemma) as follows.

▶ **Lemma 2.4** (New q-Expansion Lemma [19, 1, 27]). *Let q be a positive integer and G be a bipartite graph with bipartition (A, B). Then there exists $\widehat{A} \subseteq A$ and $\widehat{B} \subseteq B$ such that there is a q-expansion M of $\widehat{A}$ into $\widehat{B}$ in G such that*

(i) *$N(\widehat{B}) \subseteq \widehat{A}$, and*
(ii) *$|B \setminus \widehat{B}| \leq q|A \setminus \widehat{A}|$.*

Furthermore, the sets $\widehat{A}$, $\widehat{B}$ and the q-expansion M can be computed in polynomial-time.

Observe that the Lemma 2.4 statement does not require the two conditions that B has no isolated vertex and $|B| \geq q|A|$ that were required for the Lemma 2.3. In particular, if $|B| > q|A|$, then it must be that $|\widehat{B}| > q|\widehat{A}|$ and $\widehat{B}$ will contain some vertex that is not saturated by the q-expansion M.

Forbidden Subgraph Characterization. Given a graph class $\mathcal{G}$, any (induced) subgraph that is not allowed to appear in any graph of $\mathcal{G}$ is called an *obstruction* for $\mathcal{G}$ (also known as forbidden subgraphs or forbidden induced subgraphs). We first identify the obstructions for Cliques or Trees Vertex Deletion. Clearly, on simple graphs, we cannot have an obstruction for both a tree and a clique in the same connected component. If $\mathcal{G}$ is the class of all cliques, then the obstruction for $\mathcal{G}$ is $2K_1$ and if $\mathcal{G}$ is the class of all forests, then any cycle C_t with $t \geq 3$ is an obstruction for $\mathcal{G}$. Note that a cycle C_t with $t \geq 4$ contains $2K_1$ as an induced subgraph. Throughout the paper, we sometimes abuse the notation where an obstruction (or a forbidden induced subgraph) is viewed as a set and sometimes it is viewed as an (induced) subgraph.

▶ **Observation 2.5.** *For every integer $t \geq 4$, the cycles C_t contains $2K_1$ as induced subgraph.*

Thus, we can conclude that the obstructions for Cliques or Trees Vertex Deletion are cycles C_t with $t \geq 4$ and connected graphs with both $2K_1$ and C_3 as induced subgraphs. For multigraphs, a vertex with a self-loop and two vertices with two (or more) edges are obstructions as well. If a connected graph has both $2K_1$ and C_3 as induced subgraphs, a (clique, tree)-deletion set either intersects the union of the vertex sets of these subgraphs or contains a subset separating them. The following lemma claims that if a connected graph contains both $2K_1$ and C_3 as induced subgraphs, then it contains a paw or a diamond.

▶ **Lemma 2.6** ($\star$).[1] *A connected graph G with both $2K_1$ and C_3 as induced subgraphs contains either a paw or a diamond as an induced subgraph.*

From Lemma 2.6, we get a forbidden subgraph characterization for the class of graphs where each connected component is a clique or a tree.

▶ **Lemma 2.7** ($\star$). *Let $\mathcal{G}$ be the class of all simple graphs where each connected component is a clique or a tree. Then, a simple graph G belongs to $\mathcal{G}$ if and only if G does not contain any paw, diamond or cycle C_i with $i \geq 4$ as an induced subgraph.*

Parameterized Complexity and Kernelization. A *parameterized problem L* is a set of instances $(x, k) \in \Sigma^* \times \mathbb{N}$ where Σ is a finite alphabet and $k \in \mathbb{N}$ is a parameter. The notion of "tractability" in parameterized complexity is defined as follows.

▶ **Definition 2.8** (Fixed-Parameter Tractability). *A parameterized problem L is said to be* fixed-parameter tractable *(or FPT) if given $(x, k) \in \Sigma^* \times \mathbb{N}$, there is an algorithm $\mathcal{A}$ that correctly decides if $(x, k) \in L$ in $f(k)|x|^{\mathcal{O}(1)}$-time for some computable function $f : \mathbb{N} \to \mathbb{N}$. This algorithm $\mathcal{A}$ is called* fixed-parameter algorithm *(or* FPT algorithm*) for the problem L.*

[1] Due to lack of space, the proofs that are omitted or marked $\star$ can be found in the full version.

Observe in the above definition that we allow combinatorial explosion with respect to the parameter k while the algorithm runs in polynomial-time with respect to $|x|$. We say that two instances (x, k) of L and (x', k') of L are *equivalent* if $(x, k) \in L$ if and only if $(x', k') \in L$. The notion of kernelization (also known as parameterized preprocessing) is defined as follows.

▶ **Definition 2.9** (Kernelization). *A* kernelization *for a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that given an instance (x, k) of L, outputs an equivalent instance (x', k') (called* kernel*) of L in time polynomial in $|x| + k$ such that $|x'| + k' \leq g(k)$ for some function $g : \mathbb{N} \to \mathbb{N}$. If $g(k)$ is $k^{\mathcal{O}(1)}$, then L is said to admit a* polynomial kernel.

A kernelization algorithm usually consists of a collection of *reduction rules* that have to be applied exhaustively in sequence. A reduction rule is *safe* if given an instance (x, k) of L, one application of the reduction rule outputs an equivalent instance (x', k') of L. It is well known that "a decidable parameterized problem is FPT if and only if it admits a kernelization". For more details, we refer to [13, 16, 19] for more formal definitions about parameterized complexity and kernelization.

3 A Polynomial Kernel for Cliques and Trees

This section is devoted to a polynomial kernel for Cliques or Trees Vertex Deletion. As the first step, we invoke the following proposition by Jacob et al. [26] that computes a (clique, tree)-deletion set $S \subseteq V(G)$ with at most $4k$ vertices.

▶ **Proposition 3.1** ([26], Theorem 6). Cliques or Trees Vertex Deletion *admits a 4-approximation algorithm.*

We begin with the following observation, whose proof is trivial.

▶ **Observation 3.2.** *For any subset $Z \subseteq V(G)$, if (G, k) is a yes-instance for* Cliques or Trees Vertex Deletion *with solution X, then $(G - Z, k)$ is a yes-instance for* Cliques or Trees Vertex Deletion *with solution $X \setminus Z$.*

Overview of the Kernelization Algorithm. We start with invoking the 4-approximation algorithm for Cliques or Trees Vertex Deletion(Proposition 3.1) to get a (clique, tree)-deletion set S of size at most $4k$. In Section 3.1, we provide some reduction rules that guarantee that every connected component of $G - S$ has a neighbor in S, and the graph has no degree-2-path of G has more than four vertices. Subsequently, in Section 3.2, we provide some reduction rules and prove that the number of vertices in the connected components of $G - S$ that are cliques is $\mathcal{O}(k^5)$. Finally, in Section 3.3, we reduce the number of vertices in the connected components of $G - S$ that are trees to $\mathcal{O}(k^2)$. For this, we prove a variant of Proposition 2.2 and use reduction rules related to that using New q-Expansion Lemma (i.e. Lemma 2.4).

3.1 Initial Preprocessing Rules

Let S be a 4-approximate (clique, tree)-deletion set of G obtained from Proposition 3.1. If $|S| > 4k$, we conclude that (G, k) is a no-instance and return a trivial constant sized no-instance. Hence, we can assume without loss of generality that $|S| \leq 4k$. We also assume without loss of generality that G has no connected component that is a clique or a tree. If such components are there, we can delete those components. Hence, we can naturally assume from now onwards, that every connected component of $G - S$ (that is either a clique or a tree) has some neighbor in S. But some of our subsequent reduction rules can create some component in $G - S$ that is neither a clique nor a tree. So, we state the following reduction rule for the sake of completeness. In this following reduction rule, we delete isolated connected components C in $G - S$, whose safeness easily follows. All the obstructions for Cliques or Trees Vertex Deletion are connected graphs and intersect with S. Thus, no obstruction can be part of isolated component C, and also S.

▶ **Reduction Rule 3.3.** *If there exists a connected component C in $G - S$ such that no vertex in C has a neighbor in S, then remove C from G. The new instance is $(G - C, k)$.*

Since one of our subsequent reduction rules can create parallel edges. As parallel edges are also obstructions, we state the following reduction rule whose safeness is also trivial.

▶ **Reduction Rule 3.4.** *If there is an edge with multiplicity more than two, reduce the multiplicity of that edge to exactly two.*

We also have the following reduction rule that helps us to bound the number of pendant vertices attached to any vertex.

▶ **Reduction Rule 3.5.** *If there exists a vertex u in G adjacent to vertices v and v' that are pendants in G, then remove v from G. The new instance is $(G - v, k)$.*

▶ **Lemma 3.6** ($\star$)**.** *Reduction Rule 3.5 is safe.*

We can conclude that if Reduction Rule 3.5 is not applicable, then every vertex in G is adjacent to at most one pendant vertex in G. From now, we assume that every vertex in G is adjacent to at most one pendant vertex. Our next two reduction rules help us to reduce the length (the number of vertices) of a degree-2-path in G. Note that a degree-2-path can be of two types, either a degree-2-tail or a degree-2-overbridge. The following two reduction rules handle both such types.

▶ **Reduction Rule 3.7.** *Let $P = (v_1, v_2, \dots, v_\ell)$ be degree-2-tail of length ℓ such that $d_G(v_1) > 2$, $d_G(v_\ell) = 1$ and $Z = \{v_3, v_4, \dots, v_\ell\}$. Then, remove Z from G. The new instance is $(G - Z, k)$.*

▶ **Lemma 3.8** ($\star$)**.** *Reduction Rule 3.7 is safe.*

Our previous reduction rule has illustrated that we can shorten a long degree-2-tail to length at most two. Now, we consider a degree-2-overbridge P of length ℓ. Our next lemma gives us a structural characterization that if we delete all but a few vertices of P, then the set of all paws and diamonds remain the same even after deleting those vertices.

▶ **Lemma 3.9** ($\star$)**.** *Let $P = (v_1, v_2, \dots, v_\ell)$ be a degree-2-overbridge of length ℓ in G and $Z = \{v_3, v_4, \dots, v_{\ell-2}\}$. Consider the graph G' obtained from G by deleting the vertices of Z and then adding the edge $v_2v_{\ell-1}$. Then the following statements hold true.*

(i) *Every paw and every diamond of G is disjoint from Z.*
(ii) *Every paw and every diamond of G' is disjoint from $\{v_2, v_{\ell-1}\}$.*
(iii) *The set of paws and diamonds in both G and $G - Z$ are the same.*

Our next reduction rule exploits the above lemma and reduces the length of a degree-2-overbridge to at most four.

▶ **Reduction Rule 3.10.** *Let $P = (v_1, v_2, \dots, v_\ell)$ be a degree-2-overbridge of length ℓ in G and $Z = \{v_3, v_4, \dots, v_{\ell-2}\}$. Let G' be the graph obtained from G by removing Z and adding the edge $v_2v_{\ell-1}$. The new instance is (G', k). We refer to Figure 1 for an illustration.*

▶ **Lemma 3.11** ($\star$)**.** *Reduction Rule 3.10 is safe.*

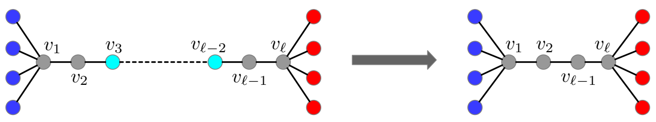


Figure 1 An illustration of applying Reduction Rule 3.10.

3.2 Bounding the Clique Vertices in $G - S$

Let $V_1 \subseteq V(G) \setminus S$ denote the set of vertices of the connected components of $G - S$ that form cliques of size at least 3. We now bound the number of connected components in $G[V_1]$ (which are cliques). Let us create an auxiliary bipartite graph $H = (S, \mathcal{C})$ with S on one side and $\mathcal{C}$ having a vertex set corresponding to each of the clique connected components in V_1 on the other side. We add an edge (s, C) with $s \in S$ and $C \in \mathcal{C}$ if s is adjacent to at least one vertex in C. We now show how to ensure that $|\mathcal{C}| \leq 2|S|$. Note that by Reduction Rule 3.3, no component in $\mathcal{C}$ is an isolated vertex in H. So, we have the following reduction rule, where we rely on the Expansion Lemma.

▶ **Reduction Rule 3.12.** *If $|\mathcal{C}| \geq 2|S|$, then call the algorithm provided by the q-Expansion Lemma with $q = 2$ (Lemma 2.3) to compute sets $X \subseteq S$ and $\mathcal{Y} \subseteq \mathcal{C}$ such that there is a 2-expansion M of X into $\mathcal{Y}$ in H and $N_H(\mathcal{Y}) \subseteq X$. The new instance is $(G - X, k - |X|)$.*

▶ **Lemma 3.13** ($\star$). *Reduction Rule 3.12 is safe.*

Thus, we have the following observation.

▶ **Observation 3.14.** *After exhaustive applications of Reduction Rules 3.3- 3.12, $|\mathcal{C}| \leq 8k$.*

We now give one of the most crucial reduction rules that gives us an upper bound the size of every clique in $G[V_1]$. We have the following marking scheme for each of the cliques in $G[V_1]$.

Procedure 1 Mark-Clique-K.

For every non-empty subset Z of size at most 3 of S, for every function $f : Z \to \{0, 1\}$, let $K_{Z,f}$ be the set of vertices v in K such that for each $z \in Z$,
- if $f(z) = 1$, then v is adjacent to z.
- if $f(z) = 0$, then v is not adjacent to z.

We arbitrarily mark $\min\{|K_{Z,f}|, k + 4\}$ vertices of $K_{Z,f}$. Note that we have marked at most $\varepsilon(k) = (2^3\binom{4k}{3} + 2^2\binom{4k}{2} + 2\binom{4k}{1})(k + 4)$ vertices in K. Let $v \in K$ be a vertex that is not marked by the above procedure Mark-Clique-K. The following set of lemmas illustrate that v is an irrelevant vertex of G.

▶ **Lemma 3.15** ($\star$). *Let S be a (clique, tree)-deletion set of at most $4k$ vertices and K be a connected component of $G - S$ that is a clique. Moreover, let $v \in K$ be a vertex that is not marked by the procedure* Mark-Clique-K *and $X \subseteq V(G) \setminus \{v\}$ be a set of at most k vertices. If $G - X$ has a vertex subset O and $G[O]$ is isomorphic to C_4, or a diamond or a paw then $G - (X \cup \{v\})$ also contains a C_4, or a diamond, or a paw as an induced subgraph.*

Our previous lemma has illustrated that if $G - X$ has a paw or a diamond or a C_4 as an induced subgraph, then $G - (X \cup \{v\})$ also has a paw or a diamond or a C_4 respectively. We will now illustrate and prove an analogous statement when $G - X$ has an induced cycle of length larger than 4. We begin with the following observation.

▶ **Observation 3.16** (⋆). *Let $C = v - u - u_1 - u_2 - \ldots - u' - v$ be a cycle of length at least 5 in G where the path $P = u - u_1 - u_2 - \ldots - u'$ is an induced path in G. Then there exists a cycle of length at least 4 or a diamond as an induced subgraph in G.*

Using the above observation, we can prove the following lemma.

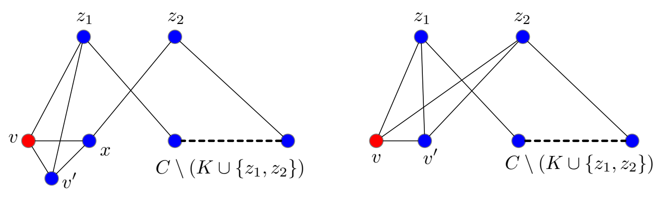
▶ **Lemma 3.17.** *Let S be a (clique, tree)-deletion set set of at most $4k$ vertices and K be a connected component of $G - S$ that is a clique. Moreover, let $v \in K$ be a vertex that is not marked by the procedure* Mark-Clique-K *and $X \subseteq V(G) \setminus \{v\}$ be a set of at most k vertices. If $G - X$ has an induced cycle of length at least 5, then there exists a cycle of length at least 4 or a diamond as an induced subgraph in $G - (X \cup \{v\})$.*

Proof. Suppose that the premise of the statement is true but for the sake of contradiction, we assume that $G - (X \cup \{v\})$ does not have cycles of length at least 4 and diamonds as induced subgraphs. Since v is the only vertex that is in G but not in $G - \{v\}$, it follows that there is an induced cycle C of length at least 5 in $G - X$ such that $v \in C$. Note that C has at most two vertices from K including v as K is a clique. Furthermore, it must have two non-adjacent vertices from S as otherwise C contains a triangle or an induced C_4, contradicting that C is an induced cycle of length at least 5. Let $z_1, z_2 \in C \cap S$ that are non-adjacent. There are two cases.

Case (i): The first case is $|C \cap K| = 1$ and let $C \cap K = \{v\}$. Then, both z_1 and z_2 are adjacent to $v \in C$. Let us define a function $f : \{z_1, z_2\} \to \{0, 1\}$ with $f(z_1) = 1$ and $f(z_2) = 1$. For the set $\{z_1, z_2\}$ and the function f, the vertex v is unmarked by the procedure Mark-Clique-K. Hence, there are $k+4$ vertices that are adjacent to both z_1 and z_2 and are marked by the procedure. All the marked vertices are in $K \setminus \{v\}$ out of which at most k vertices are in X. Hence, there is $v' \in K \setminus X$ such that v' is adjacent to both z_1 and z_2 and is marked by the procedure. Let us look at the cycle $C' = (C \setminus \{v\}) \cup \{v'\}$. Note that C' is a cycle where the path from z_1 to z_2 is an induced path in $G - X$. There could be edges from v' to other vertices of C'. By Observation 3.16, C' is either an induced cycle of length at least 4 in G or it induces a diamond. Since $C' \cap (X \cup \{v\}) = \emptyset$, this contradicts our initial assumption that $G - (X \cup \{v\})$ does not have cycles of length at least 4 and diamonds as induced subgraphs.

Case (ii): The second and last case is $|C \cap K| = 2$. Let $v, x \in C \cap K$. Observe that v and x are two consecutive vertices in C. Since $vx \in E(G)$, it must be that v is adjacent to z_1 and x is adjacent to z_2. As C is an induced cycle of length at least 5, it must be that z_1 is not adjacent to x and z_2 is not adjacent to v.
Let us define a function $f : \{z_1, z_2\} \to \{0, 1\}$ with $f(z_1) = 1$ and $f(z_2) = 0$. For the set $\{z_1, z_2\}$ and the function f, the vertex v is unmarked by the procedure Mark-Clique-K. Hence, there are $k + 4$ vertices that is adjacent to z_1 and not adjacent to z_2 that are marked by the procedure. All the marked vertices are in $K \setminus \{v\}$ out of which at most k vertices are in X. Hence, there is $v' \in K \setminus X$ such that v' is adjacent to z_1 and not adjacent to z_2 that is marked by the procedure.
We replace v in C by v' to get a new cycle C' that has the same number of vertices as C (see Figure 2 for an illustration). Note that C' is a cycle where the path from z_1 to x is an induced path in $G - X$. There could be edges from v' to other vertices of C'. By Observation 3.16, C' is either an induced cycle of length at least 4 in G or it induces a diamond. This contradicts our initial assumption that $G - (X \cup \{v\})$ does not have cycles of length at least 4 and diamonds as induced subgraphs.

Since the above cases are mutually exhaustive, this completes the proof. ◀



z_1
z_2
v
x
v'
$C \setminus (K \cup \{z_1, z_2\})$
z_1
z_2
v
v'
$C \setminus (K \cup \{z_1, z_2\})$

Figure 2 An illustration of C_5.

Consider a connected component K of $G - S$ such that S is a (clique, tree)-deletion set of G with at most $4k$ vertices and K is a clique. Lemma 3.15 and Lemma 3.17 illustrate that if a vertex $v \in K$ is not marked by the procedure Mark-Clique-K, then deleting v from the graph is safe. As a consequence of this, we have the following reduction rule the safeness of which follows from the above two lemmas.

▶ **Reduction Rule 3.18.** *Let K be a connected component in $G[V_1]$. If $v \in K$ is an unmarked vertex after invoking the procedure* Mark-Clique-K*, then remove v from G. The new instance is $(G - v, k)$.*

▶ **Lemma 3.19.** *Reduction Rule 3.18 is safe.*

Proof. The forward direction ($\Rightarrow$) is trivial as if X is a solution of size at most k in G, then by Observation 3.2, $X \setminus \{v\}$ is a solution of size at most k in $G - v$.

For the backward direction ($\Leftarrow$), let X be a solution of size at most k in $G - v$. Targeting a contradiction, suppose X is not a solution in G. Then, there exists an obstruction O of Cliques or Trees Vertex Deletion in $G - X$ containing v that is a diamond, or a paw, or C_i where $i \geq 4$ due to Lemma 2.7. If O is isomorphic to a C_4, or a diamond, or a paw, then by Lemma 3.15, $G - (X \cup \{v\})$ also contains a C_4, or a diamond, or a paw as induced subgraph. It contradicts Lemma 2.7 that X is a solution to $G - \{v\}$. Else, O is isomorphic to C_i where $i \geq 5$. By Lemma 3.17, $G - (X \cup \{v\})$ contains a C_j, where $j \geq 4$, or a diamond as induced subgraph. This contradicts Lemma 2.7 as X is a solution to $G - \{v\}$. Hence X is a solution to G. ◀

We have the following lemma that bounds $|V_1|$, i.e. the number of vertices that are part of cliques of size at least 3 in $G - S$.

▶ **Lemma 3.20.** *Let G be the graph obtained after exhaustive application of Reduction Rules 3.3 to 3.18. Then $|V_1| \leq 8k\varepsilon(k)$ where $\varepsilon(k) = (k+4)(8\binom{4k}{3} + 4\binom{4k}{2} + 2\binom{4k}{1})$.*

Proof. Since Reduction Rules 3.3-3.12 are not applicable, it follows from Observation 3.14 that the number of connected components in $G[V_1]$ is at most $8k$. Every connected component of $G[V_1]$ is a clique. For every connected component K of $G[V_1]$, observe that the procedure Mark-Clique-K marks at most $\varepsilon(k)$ vertices from K. Since Reduction Rule 3.18 is not applicable, $G[V_1]$ has no unmarked vertices. As $G[V_1]$ has at most $8k$ connected components, it follows that $|V_1| \leq 8k\varepsilon(k)$. ◀

3.3 Bounding the Tree Vertices in $G - S$

In this section, we describe the set of reduction rules that we use to reduce the number of vertices that participate in the forests of $G - S$. Let $V_2 = V \setminus (S \cup V_1)$. Note that $G[V_2]$ is the collection of trees in $G - S$.

▶ **Reduction Rule 3.21.** *Let v be a leaf in a connected component C of $G[V_2]$ such that neither v nor its neighbor in C is adjacent to any vertex of S. Then, delete v from G and the new instance is $(G - v, k')$ with $k' = k$.*

▶ **Lemma 3.22** ($\star$). *Reduction Rule 3.21 is safe.*

Let C be a connected component of $G[V_2]$. We say that C is a *pendant tree* of $G[V_2]$ if either C consists of pendant vertex of G or has a unique vertex u that has a unique neighbour in S and no other vertex of C has any neighbor in S. We have the following observation.

▶ **Observation 3.23** ($\star$). *Let C be a connected component of $G[V_2]$. Then, C is a pendant tree if and only if $E(C, V(G) \setminus C)$ contains a single edge.*

Given a pendant tree C of $G[V_2]$, we call $x \in S$ the *unique S-neighbor* of C if $N_G(C) = \{x\}$ and $|N_G(x) \cap C| = 1$. Furthermore, given a vertex $x \in S$, we call C a *pendant tree-neighbor* of x if C is a pendant tree of $G[V_2]$ and x is the *unique S-neighbor* of C.

▶ **Reduction Rule 3.24.** *Let C be a pendant tree in $G[V_2]$ such that $x \in S$ is a unique S-neighbor of C. Then, delete all the vertices of C except for the vertex u that has the unique S-neighbor $x \in S$ in C to obtain the graph G'. Let (G', k') be the output instance such that $k' = k$.*

▶ **Lemma 3.25** ($\star$). *Reduction Rule 3.24 is safe.*

▶ **Lemma 3.26** ($\star$). *If Reduction Rule 3.24 is not applicable to the input instance (G, k), then any pendant tree of $G[V_2]$ is a pendant vertex of G.*

Recall from Definition 2.1 that a v-flower of order r is a collection of r cycles that pairwise intersect at v and are pairwise disjoint otherwise. Our next reduction rule uses the concept of v-flower and Proposition 2.2 as follows.

▶ **Reduction Rule 3.27.** *For $v \in S$, we invoke Proposition 2.2 in $G[V_2 \cup \{v\}]$. If this gives a v-flower of order $3k + 2$, then delete v from G and the new instance is $(G - v, k - 1)$.*

▶ **Lemma 3.28** ($\star$). *Reduction Rule 3.27 is safe.*

Since Reduction Rule 3.27 is not applicable, invoking Proposition 2.2 of order $(3k + 2)$ at $G[\{v\} \cup V_2]$ gives us a set $H_v \subseteq V_2$ of at most $6k + 4$ vertices that intersects all cycles of $G[\{v\} \cup V_2]$ that passes through v. Our following lemma proves that the same vertex subset H_v also intersects all paws and diamonds of $G[\{v\} \cup V_2]$ passing through v.

▶ **Lemma 3.29** ($\star$). *If Reduction Rule 3.27 is not applicable, then polynomial time, we can obtain a vertex subset $H_v \subseteq V_2$ with $|H_v| \leq 6k + 4$ such that H_v intersects every cycle, every paw, and every diamond in $G[\{v\} \cup V_2]$ that passes through v.*

When the above mentioned reduction rules are not applicable, we have the following lemma which bounds the number of connected components of the graph $G[V_2 \setminus H_v]$ that is adjacent to only v.

Construction of Auxiliary Bipartite Graph. If none of the above reduction rules are applicable, we exploit the structural properties of the graph using the above mentioned lemmas and construct an auxiliary bipartite graph that we use in some reduction rules later. Let $\mathcal{C}$ denote the connected components of $G[V_2 \setminus (H_v \cup \{v\})]$ that are adjacent to v. In other words, if $D \in \mathcal{C}$, then D is a connected component of $G[V_2 \setminus (H_v \cup \{v\})]$ such that v is adjacent to D.

▶ **Definition 3.30.** *Given $v \in S$, let H_v denote the set of at most $6k+4$ vertices obtained by Lemma 3.29 for the graph $G[\{v\} \cup V_2]$. Consider the graph $G[V_2 \setminus (H_v \cup \{v\})]$ that is a forest. We define an auxiliary bipartite graph $\mathcal{H} = (H_v \cup (S \setminus \{v\}), \mathcal{C})$ where $H_v \cup (S \setminus \{v\})$ is on one side, and $\mathcal{C}$ on the other side. The set $\mathcal{C}$ contains a vertex for each connected component C of $G[V_2 \setminus (H_v \cup \{v\})]$ that has a vertex adjacent to v. We add an edge between $h \in H_v \cup (S \setminus \{v\})$, and connected component $C \in \mathcal{C}$ if h is adjacent to a vertex in component $C \in \mathcal{C}$.*

We prove the following observation that is crucial to the next reduction rule which reduces the number of edges incident to a vertex $v \in S$ with other endpoint being V_2.

▶ **Observation 3.31 ($\star$).** *Let $\mathcal{H} = (H_v \cup (S \setminus \{v\}), \mathcal{C})$ be the auxiliary bipartite graph as defined in Definition 3.30. If v has degree more than $60(k+1)$ in $G[\{v\} \cup V_2]$, then $\mathcal{C}$ has more than $4(|S| + |H_v|)$ components.*

Applying the New Expansion Lemma. If there is a vertex $v \in S$ such that there are at least $60(k+1)$ edges incident to v with the other endpoints being in V_2, then Observation 3.31 implies that $|\mathcal{C}| > 4(|S| + |H_v|)$. Suppose that we apply new 4-expansion lemma (Lemma 2.4 with $q = 4$) on $\mathcal{H}$ to obtain $A \subseteq (S \setminus \{v\}) \cup H_v, \mathcal{B} \subseteq \mathcal{C}$ with a 4-expansion $\widehat{M}$ of A into $\mathcal{B}$. Then, it satisfies that (i) $|\mathcal{C} \setminus \mathcal{B}| \leq 4|(S \cup H_v) \setminus A|$ and $N_{\mathcal{H}}(\mathcal{B}) \subseteq A$. As $|\mathcal{C} \setminus \mathcal{B}| \leq 4|(S \cup H_v) \setminus A|$ and $|\mathcal{C}| > 4(|S| + |H_v|)$, it must be that $|\mathcal{B}| > 4|A|$. Then, there must be a component $C^* \in \mathcal{B}$ such that C^* is not an endpoint of $\widehat{M}$ (or not saturated by $\widehat{M}$). Let $\widehat{\mathcal{B}} \subseteq \mathcal{B}$ denote the components of $\mathcal{B}$ that are saturated by $\widehat{M}$. As some component of $\mathcal{B}$ is not in $\widehat{\mathcal{B}}$, it must be that $\widehat{\mathcal{B}} \subset \mathcal{B}$. We use these characteristics crucially to prove that our next reduction rule is safe.

▶ **Reduction Rule 3.32.** *Let $v \in S$ be a vertex with degree at least $60(k+1)$ in $G[\{v\} \cup V_2]$ and let $\mathcal{H}$ be the auxiliary bipartite graph as illustrated in Definition 3.30. We invoke the algorithm provided by Lemma 2.4 (i.e. new q-expansion lemma with $q = 4$) to compute sets $A \subseteq H_v \cup (S \setminus \{v\})$ and $\mathcal{B} \subseteq \mathcal{C}$ such that A has a 4-expansion $\widehat{M}$ into $\mathcal{B}$ in $\mathcal{H}$ and $N_{\mathcal{H}}(\mathcal{B}) \subseteq A$. Let $\widehat{\mathcal{B}} \subseteq \mathcal{B}$ denotes the vertices of $\mathcal{B}$ that are saturated by $\widehat{M}$ (endpoints of $\widehat{M}$ in $\mathcal{B}$). Remove the edges between v and the connected components in $\widehat{\mathcal{B}}$ in G and create a double edge between v and every vertex in A to obtain the graph G'. The new instance is (G', k') with $k = k'$. We refer to the Figure 3 for an illustration.*

Before we prove the safeness of the above reduction rule, we prove the following lemma.

▶ **Lemma 3.33.** *Let X be an optimal (clique, tree)-deletion set of G of size at most k and $A, \mathcal{B}, \widehat{\mathcal{B}}$ denote the vertex subsets obtained from Reduction Rule 3.32. Then, $v \in X$ or $A \subseteq X$.*

Proof. Let X be an optimal (clique, tree)-deletion set of G of size at most k. As the Lemma 2.4 (new q-expansion lemma) with $q = 4$ has been already applied in Reduction Rule 3.32 and the obtained sets are $A \subseteq (S \cup H_v) \setminus \{v\}$ and $\mathcal{B} \subseteq \mathcal{C}$ such that $N_{\mathcal{H}}(\mathcal{B}) \subseteq A$ (due to item (ii) of Lemma 2.4), any connected component $C \in \mathcal{B}$ can have neighbors only in $A \cup H_v \cup \{v\}$ in G. By Definition 3.30, every connected component $C \in \mathcal{C}$ has a vertex that is adjacent to v. If some component $C^* \in \mathcal{C}$ has two vertices adjacent to v, then there is a cycle that passes through v and the vertices of C^* but avoids H_v. This contradicts with Lemma 3.29 that H_v intersects all cycles passing through v. Hence, for every connected component $C \in \mathcal{C}$, there is exactly one vertex that is adjacent to v.

Suppose for the sake of contradiction that there is an optimal (clique, tree)-deletion set X^* of G of size at most k such that $v \notin X^*$ and $A \not\subseteq X^*$. Let $X^*_{\mathcal{B}}$ denotes the intersection of X^* with the vertices that are in the connected components in $\mathcal{B}$. We set

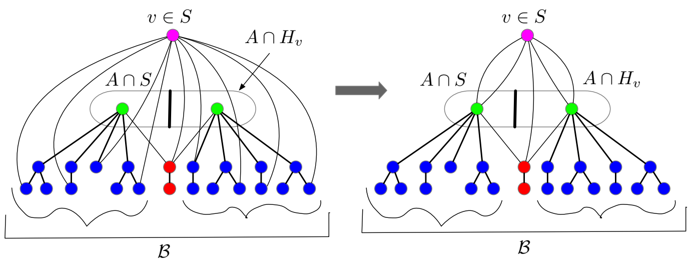


Figure 3 An illustration of Reduction Rule 3.32. The blue components are the components of $\mathcal{B}$ and the red component is the chosen component C for which the edge between u and C is not deleted. The blue components of $\mathcal{B}$ are the ones that are the endpoints of expansion $\widehat{M}$.

$\widehat{X} = (X^* \setminus X^*_{\mathcal{B}}) \cup (A \setminus X^*) \cup \{v\}$. We claim that $\widehat{X}$ is a (clique, tree)-deletion set of G and $|\widehat{X}| < |X^*|$. For the first part, note that the cycles hit by X^* but not $\widehat{X}$ must contain a vertex from $X^*_{\mathcal{B}}$. Such cycles must contain a vertex in $N_{\mathcal{H}}(\mathcal{B})$ as each component of $\mathcal{B}$ is a forest. It follows from the item (i) of Lemma 2.4 that $N_{\mathcal{H}}(\mathcal{B}) \subseteq A$ and $A \subseteq H_v \cup (S \setminus \{v\})$. Therefore, the neighbors of all vertices spanned by the connected components of $\mathcal{B}$ are contained in $A \cup \{v\}$. By construction of $\widehat{X}$, as $A \cup \{v\} \subseteq \widehat{X}$, it follows that $\widehat{X}$ is a (clique, tree)-deletion set of G.

Now, we claim that $|\widehat{X}| < |X^*|$. Since $A \not\subset X^*$ and $v \notin X^*$, there is $x \in A \setminus X^*$. Due to Lemma 2.4, there are four connected components C_1, C_2, C_3, C_4 in $\widehat{\mathcal{B}}$ such that v is adjacent to one vertex from each of C_1, C_2, C_3, C_4 and x is adjacent to some vertex in each of C_1, C_2, C_3, C_4. If v is adjacent to x, then $G[\{v, x\} \cup C_1 \cup C_2 \cup C_3 \cup C_4]$ has several cycles with a chord (or subdivision of diamonds) that contains $\{x, v\}$ and vertices from exactly two connected components from $\{C_1, C_2, C_3, C_4\}$. In particular for every pair $i, j \in \{1, 2, 3, 4\}$, there is a cycle with a chord containing v, x and vertices from C_i and C_j. Since $v, x \notin X^*$, it must be that X^* must have at least one vertex from at least three of these connected components $\{C_1, C_2, C_3, C_4\}$. As our updating procedure removes the vertices of $\mathcal{B}$ from X^* and adding vertices of $A \setminus X^*$, it follows that for every $x \in A \setminus X^*$, one vertex is added and at least three additional vertices appearing in the components of $\{C_1, C_2, C_3, C_4\}$ are removed from X^*. This ensures that $\widehat{X}$ has strictly lesser vertices than X^* contradicting the optimality of X^*. This completes the proof. ◀

We use the above lemma to prove that Reduction Rule 3.32 is safe.

▶ **Lemma 3.34.** *Reduction Rule 3.32 is safe.*

Proof. For the forward direction ($\Rightarrow$), let X be a (clique, tree)-deletion set of G with at most k vertices. We assume without loss of generality that X is an optimal (clique, tree)-deletion set of G. Due to Lemma 3.33, it follows that either $v \in X$ or $A \subseteq X$. For both the cases, observe that $G' - X$ is a subgraph of $G - X$. Hence, X is a (clique, tree)-deletion set of G' of size at most k'.

For the backward direction ($\Leftarrow$), let X' be a (clique, tree)-deletion set of G' of size at most $k'(= k)$. Note that in G', we have double edge between v and every vertex in A. Thus, $v \in X'$ or $A \subseteq X'$ to hit the cycles formed by these double edges. In case $v \in X'$, then the graphs $G - X'$ and $G' - X'$ are precisely the same. Therefore, X' is a (clique, tree)-deletion set of G as well. For the other case, we have that $A \subseteq X'$ but $v \notin X'$. Suppose for the sake of contradiction that some component of $G - X'$ is neither a clique, nor a tree. Then, $G - X'$ has an obstruction O. Since for any connected component $C' \in \mathcal{B}$, it must be that $N_{\mathcal{H}}(C') \subseteq A$, it follows that $N_G(C') \subseteq A \cup \{v\}$. By construction of G', an edge $uv \in E(G)$

is not an edge of G' if $u \in C'$ for some connected component $C' \in \widehat{\mathcal{B}}$. The obstruction O must contain an edge $uv \in E(G)$ such that $uv \notin E(G')$. Such an edge is possible only for some $C'' \in \widehat{\mathcal{B}}$. As v can have at most one neighbor in every component $C'' \in \mathcal{B}$ and the vertices of C'' are adjacent to only $A \cup \{v\}$ with $A \subseteq X'$, the only possible way uv edge can be part of an obstruction in $G - X'$ is a paw. Then, this obstruction O is a paw containing u and v. Since $|\mathcal{C}| > 4(|S| + |H_v|)$, and due to the condition (ii) of Proposition 2.4, $|\mathcal{C} \setminus \mathcal{B}| \leq 4|(S \cup H_v) \setminus A|$, it must be that $|\mathcal{B}| > 4|A|$. Therefore, there is at least one connected component $C \in \mathcal{B} \setminus \widehat{\mathcal{B}}$. Furthermore, Reduction Rule 3.32 has chosen not to delete the edge vu^* such that $u^* \in C$ and $vu^* \in E(G)$. But by construction of G', $vu^* \in E(G')$. Observe that $O^* = (O \setminus \{u\}) \cup \{u^*\})$ also induces a paw in the graph G. Then, u^* must be in the set X' as otherwise it would contradict that X' is a (clique, tree)-deletion set of X'. So, we set $X^* = (X' \setminus \{u^*\}) \cup \{v\}$ and clearly by construction $|X^*| = |X'|$. Since the neighrborhood of any connected component of $\mathcal{B}$ is contained in $A \cup \{v\}$, this ensures us that X^* is a (clique, tree)-deletion set of G. This completes the proof of the lemma. ◀

Observe that the above mentioned reduction rules do not increase the degree of v. When none of the above mentioned reduction rules are applicable, no connected component C (with at least three vertices) of $G[V_2]$ can have two leaves u and v both of which are pendant vertices in G. But, it is possible that C has one leaf u that is a pendant vertex of the whole graph G.

▶ **Lemma 3.35.** *Let G be the graph obtained after applying Reduction Rules 3.3-3.32 exhaustively. Then the number of vertices in V_2 is at most $1525k|S|$.*

Proof. We use N to denote the vertices of V_2 that are adjacent to some vertex of S. As Reduction Rule 3.32 is not applicable, for every $v \in S$, there are at most $60(k+1)(\leq 61k)$ edges incident to v with the other endpoint being in V_2 (hence in N). Hence, the number of vertices of N is at most $61k|S|$.

Let us bound the number of leaves in the forest $G[V_2]$ that is not in N. Since Reduction Rule 3.21 is exhaustively applied, such a leaf is adjacent to a vertex that is adjacent to some vertex in S (therefore, such vertices are in N). Since Reduction Rule 3.5 is exhaustively applied, two such leaves are not adjacent to the same vertex. Hence, we can define an injective function from the leaves of $G[V_2 \setminus N]$ to the internal vertices in the forest $G[V_2]$ that is in N.

Thus, the total number leaves in $G[V_2]$ is at most $2|N|$. Since, the number of vertices with degree at least 3 in $G[V_2]$ is at most the number of leaves in $G[V_2]$, the number of vertices of $G[V_2]$ with degree at least three is at most $2|N|$. Therefore, the sum of the number leaves in $G[V_2]$, and the number of vertices with degree at least 3 of $G[V_2]$ is at most $4|N|$. Additionally, there are some vertices of N that are neither counted as a leaf of $G[V_2]$ nor is counted as a vertex of degree at least three in $G[V_2]$. Number of such vertices is at most $|N|$.

It remains to bound the number of degree 2 vertices in the forest $G[V_2]$ that is not in N. Note that such vertices are also degree 2 vertices in G. Since Reduction Rules 3.7 and 3.10 are exhaustively applied, any degree-2-overbridge of $G[V_2]$ has size at most four. Each such degree-2-overbridge connects two vertices of $G[V_2]$ such that those two vertices are leaves, or vertices from N. We replace all such degree-2-overbridges by edges to get a forest H with at most $5|N|$ vertices, and thus edges. Since each edge of H corresponds to at most 4 vertices, we have at most $20|N|$ vertices of V_2 each of which have degree exactly two in G.

Thus the total number of vertices in $G[V_2]$ is bounded by $25|N|$. Since $|N| \leq 61k|S|$, the the total number of vertices in the forest $G[V_2]$ is bounded by $1525k|S|$. ◀

Combining Lemma 3.20 and Lemma 3.35, we are ready to prove our final result that we restate below.

▶ **Theorem 1.1.** Cliques or Trees Vertex Deletion *(CTVD) admits a kernel with* $\mathcal{O}(k^5)$ *vertices.*

Proof. Given the input instance (G, k), the kernelization algorithm invokes Reduction Rules 3.3-3.32 exhaustively. Let (G', k') denotes the output instance such that S' is a (clique, tree)-deletion set of G' with at most $4k$ vertices. Suppose that $V_1 \subseteq G' - S'$ denotes the vertices such that every connected component of $G[V_1]$ is a clique and $V_2 \subseteq G' - S'$ denotes the vertices such that every connected component of $G[V_2]$ is a tree. Since Reduction Rules 3.3-3.18 are not applicable, it follows from Lemma 3.20 that $|V_1|$ is $\mathcal{O}(k^5)$. Additionally, as Reduction Rules 3.3-3.32 are not applicable, it follows from Lemma 3.35 that $|V_2| \leq 1525k|S'| = 6100k^2$. Therefore, the total number of vertices in G' is $|S'| + |V_1| + |V_2|$ which is $\mathcal{O}(k^5)$. ◀

4 Conclusions and Future Research

Our paper initiates a study of polynomial kernelization for vertex deletion to pairs of scattered graph classes. One natural open question is to improve the size of our kernel, e.g. to $\mathcal{O}(k^3)$ vertices. We believe that such a result is possible to achieve, but we suspect that it would require new techniques to develop such results. Jacob et al. [26] have provided an $\mathcal{O}^*(4^k)$-time algorithm for Cliques or Trees Vertex Deletion. It would also be interesting to design an FPT algorithm where the base of the exponent is (substantially) improved from 4. On a broader level, it would be interesting to explore the possibility of getting a polynomial kernel for problems where the objective is to delete a set of at most k vertices so that the connected components would belong to other interesting pairs of graph classes, such as (interval graph, trees), and (chordal graph, bipartite permutation). In addition, vertex/edge deletion to scattered graph classes are also interesting from approximation algorithms perspective. In fact, it would be interesting to improve the approximation guarantee of Proposition 3.1 that is also an open problem. Additionally, the dual version of this problem, i.e. "packing vertex-disjoint obstructions to the scattered class of cliques and trees" is also interesting from the perspective of parameterized complexity. The same problem can be considered as packing vertex-disjoint induced subgraphs that are paws or diamonds or cycles of length at least 4. A natural approach to solve this problem requires to design an Erdos-Posa style theorem for packing obstructions for scattered class of cliques and trees. Finally, a more general open problem is to identify pairs of graph classes (Π_1, Π_2) for which vertex deletion to Π_1 as well as vertex deletion to Π_2 admits polynomial sized kernels, but (Π_1, Π_2)-Deletion does not admit a polynomial kernel.

References

1 Jasine Babu, R. Krithika, and Deepak Rajendraprasad. Packing arc-disjoint 4-cycles in oriented graphs. In Anuj Dawar and Venkatesan Guruswami, editors, *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2022, December 18-20, 2022, IIT Madras, Chennai, India*, volume 250 of *LIPIcs*, pages 5:1–5:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.FSTTCS.2022.5.

2 Marthe Bonamy, Konrad K. Dabrowski, Carl Feghali, Matthew Johnson, and Daniël Paulusma. Independent feedback vertex set for P_5-free graphs. *Algorithmica*, 81(4):1342–1369, 2019. doi:10.1007/S00453-018-0474-X.

3 Anudhyan Boral, Marek Cygan, Tomasz Kociumaka, and Marcin Pilipczuk. A fast branching algorithm for cluster vertex deletion. *Theory Comput. Syst.*, 58(2):357–376, 2016. doi:10.1007/S00224-015-9631-7.

4 Hajo Broersma, Jirí Fiala, Petr A. Golovach, Tomás Kaiser, Daniël Paulusma, and Andrzej Proskurowski. Linear-time algorithms for scattering number and hamilton-connectivity of interval graphs. *J. Graph Theory*, 79(4):282–299, 2015. doi:10.1002/JGT.21832.

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

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

7 Yixin Cao and Dániel Marx. Chordal editing is fixed-parameter tractable. *Algorithmica*, 75(1):118–137, 2016. doi:10.1007/S00453-015-0014-X.

8 Jianer Chen. Vertex cover kernelization. In *Encyclopedia of Algorithms*, pages 2327–2330. Springer, 2016. doi:10.1007/978-1-4939-2864-4_460.

9 Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010. doi:10.1016/J.TCS.2010.06.026.

10 Maria Chudnovsky, Neil Robertson, Paul Seymour, and Robin Thomas. The strong perfect graph theorem. *Annals of mathematics*, pages 51–229, 2006.

11 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition.* MIT Press, 2009.

12 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.

13 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer, 2015. doi:10.1007/978-3-319-21275-3.

14 Jan Derbisz, Lawqueen Kanesh, Jayakrishnan Madathil, Abhishek Sahu, Saket Saurabh, and Shaily Verma. A polynomial kernel for bipartite permutation vertex deletion. *Algorithmica*, 84(11):3246–3275, 2022. doi:10.1007/S00453-022-01040-9.

15 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

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

17 Maël Dumas and Anthony Perez. An improved kernelization algorithm for trivially perfect editing. In Neeldhara Misra and Magnus Wahlström, editors, *18th International Symposium on Parameterized and Exact Computation, IPEC 2023, September 6-8, 2023, Amsterdam, The Netherlands*, volume 285 of *LIPIcs*, pages 15:1–15:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.IPEC.2023.15.

18 Bruno Escoffier, Laurent Gourvès, and Jérôme Monnot. Complexity and approximation results for the connected vertex cover problem in graphs and hypergraphs. *J. Discrete Algorithms*, 8(1):36–49, 2010. doi:10.1016/J.JDA.2009.01.005.

19 Fedor V. Fomin, Tien-Nam Le, Daniel Lokshtanov, Saket Saurabh, Stéphan Thomassé, and Meirav Zehavi. Subquadratic kernels for implicit 3-hitting set and 3-set packing problems. *ACM Trans. Algorithms*, 15(1):13:1–13:44, 2019. doi:10.1145/3293466.

20 Fedor V Fomin, Saket Saurabh, and Yngve Villanger. A polynomial kernel for proper interval vertex deletion. *SIAM Journal on Discrete Mathematics*, 27(4):1964–1976, 2013. doi:10.1137/12089051X.

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

22 Petr A. Golovach, Daniël Paulusma, and Erik Jan van Leeuwen. Induced disjoint paths in claw-free graphs. *SIAM J. Discret. Math.*, 29(1):348–375, 2015. doi:10.1137/140963200.

23 Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57. Elsevier, 2004.

24 Pinar Heggernes, Dieter Kratsch, and Daniel Meister. Bandwidth of bipartite permutation graphs in polynomial time. *J. Discrete Algorithms*, 7(4):533–544, 2009. `doi:10.1016/J.JDA.2008.11.001`.

25 Ashwin Jacob, Jari J. H. de Kroon, Diptapriyo Majumdar, and Venkatesh Raman. Deletion to scattered graph classes I - case of finite number of graph classes. *J. Comput. Syst. Sci.*, 138:103460, 2023. `doi:10.1016/J.JCSS.2023.05.005`.

26 Ashwin Jacob, Diptapriyo Majumdar, and Venkatesh Raman. Deletion to scattered graph classes II - improved FPT algorithms for deletion to pairs of graph classes. *J. Comput. Syst. Sci.*, 136:280–301, 2023. `doi:10.1016/J.JCSS.2023.03.004`.

27 Ashwin Jacob, Diptapriyo Majumdar, and Venkatesh Raman. Expansion lemma - variations and applications to polynomial-time preprocessing. *Algorithms*, 16(3):144, 2023. `doi:10.3390/A16030144`.

28 Hugo Jacob, Thomas Bellitto, Oscar Defrain, and Marcin Pilipczuk. Close relatives (of feedback vertex set), revisited. In Petr A. Golovach and Meirav Zehavi, editors, *16th International Symposium on Parameterized and Exact Computation, IPEC 2021, September 8-10, 2021, Lisbon, Portugal*, volume 214 of *LIPIcs*, pages 21:1–21:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPICS.IPEC.2021.21`.

29 Bart M. P. Jansen, Jari J. H. de Kroon, and Michal Wlodarczyk. Single-exponential FPT algorithms for enumerating secluded f-free subgraphs and deleting to scattered graph classes. In Satoru Iwata and Naonori Kakimura, editors, *34th International Symposium on Algorithms and Computation, ISAAC 2023, December 3-6, 2023, Kyoto, Japan*, volume 283 of *LIPIcs*, pages 42:1–42:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. `doi:10.4230/LIPICS.ISAAC.2023.42`.

30 Matthew Johnson, Giacomo Paesani, and Daniël Paulusma. Connected vertex cover for $(sP_1 + P_5)$-free graphs. *Algorithmica*, 82(1):20–40, 2020. `doi:10.1007/S00453-019-00601-9`.

31 Tereza Klimosová, Josef Malík, Tomás Masarík, Jana Novotná, Daniël Paulusma, and Veronika Slívová. Colouring $(P_r + P_s)$-free graphs. *Algorithmica*, 82(7):1833–1858, 2020. `doi:10.1007/S00453-020-00675-W`.

32 Tomasz Kociumaka and Marcin Pilipczuk. Faster deterministic feedback vertex set. *Inf. Process. Lett.*, 114(10):556–560, 2014. `doi:10.1016/J.IPL.2014.05.001`.

33 Dieter Kratsch, Haiko Müller, and Ioan Todinca. Feedback vertex set on at-free graphs. *Discret. Appl. Math.*, 156(10):1936–1947, 2008. `doi:10.1016/J.DAM.2007.10.006`.

34 Cornelis Lekkeikerker and Johan Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962.

35 John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is np-complete. *J. Comput. Syst. Sci.*, 20(2):219–230, 1980. `doi:10.1016/0022-0000(80)90060-4`.

36 Barnaby Martin, Daniël Paulusma, and Erik Jan van Leeuwen. Disconnected cuts in claw-free graphs. *J. Comput. Syst. Sci.*, 113:60–75, 2020. `doi:10.1016/J.JCSS.2020.04.005`.

37 George J Minty. On maximal independent sets of vertices in claw-free graphs. *Journal of Combinatorial Theory, Series B*, 28(3):284–304, 1980. `doi:10.1016/0095-8956(80)90074-X`.

38 Venkatesh Raman, Saket Saurabh, and C. R. Subramanian. Faster fixed parameter tractable algorithms for finding feedback vertex sets. *ACM Trans. Algorithms*, 2(3):403–415, 2006. `doi:10.1145/1159892.1159898`.

39 Stéphan Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Trans. Algorithms*, 6(2):32:1–32:8, 2010. `doi:10.1145/1721837.1721848`.