Modifying a Graph Using Vertex Elimination*

Petr A. Golovach[1], Pinar Heggernes[1], Pim van 't Hof[1], Fredrik Manne[1], Daniël Paulusma[2], and Michał Pilipczuk[1]

[1] Department of Informatics, University of Bergen, Norway
{petr.golovach,pinar.heggernes,pim.vanthof,fredrik.manne,michal.pilipczuk}@ii.uib.no
[2] School of Engineering and Computing Sciences, Durham University, UK
daniel.paulusma@durham.ac.uk

Abstract. Vertex elimination is a graph operation that turns the neighborhood of a vertex into a clique and removes the vertex itself. It has widely known applications within sparse matrix computations. We define the Elimination problem as follows: given two graphs G and H, decide whether H can be obtained from G by $|V(G)| - |V(H)|$ vertex eliminations. We show that Elimination is W[1]-hard when parameterized by $|V(H)|$, even if both input graphs are split graphs, and W[2]-hard when parameterized by $|V(G)| - |V(H)|$, even if H is a complete graph. On the positive side, we show that Elimination admits a kernel with at most $5|V(H)|$ vertices in the case when G is connected and H is a complete graph, which is in sharp contrast to the W[1]-hardness of the related Clique problem. We also study the case when either G or H is tree. The computational complexity of the problem depends on which graph is assumed to be a tree: we show that Elimination can be solved in polynomial time when H is a tree, whereas it remains NP-complete when G is a tree.

1 Introduction

Consider the problem of choosing a set S of resilient communication hubs in a network, such that if any subset of the hubs should stop functioning then all the remaining hubs in S can still communicate. Such a set is attractive if the probability of a hub failure is high, or if the network is dynamic and hubs can leave the network. We can formulate this as a graph problem in the following way: Given a graph G and an integer k, is there a set S of k vertices, such that if any subset of S is removed from G, then every pair of remaining vertices in S are still connected via paths in the modified graph? Obviously, choosing S to be a clique of size k would solve the problem, but only allowing for cliques is overly restrictive. A necessary and sufficient condition on S is that for each pair $u, v \in S$, either u and v are adjacent or there is a path between u and v in G not containing any vertex of S except for u and v. Thus we can view the described problem as a relaxation of the well-known Clique problem.

The above problem can be restated using a well-known graph operation related to Gaussian elimination: vertex elimination [18]. The *elimination* of a vertex v from a graph G is the operation that adds edges to G such that the neighbors of v form a clique, and then removes v from the resulting graph. With this operation, the above problem can be defined as follows: find a set S of size k such that eliminating all vertices of $V(G) \setminus S$ leaves S as a clique. In fact, we state a more general problem: the Elimination problem takes as input two graphs G

* This work is supported by EPSRC (EP/G043434/1) and Royal Society (JP100692), by the Research Council of Norway (197548/F20), and by the ERC grant "Rigorous Theory of Preprocessing", reference 267959. An extended abstract of this paper was presented at the 38th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2012) [9].

and H, and asks whether a graph isomorphic to H can be obtained by the elimination of $|V(G)| - |V(H)|$ vertices from G. If this is possible, then we say that H is an *elimination* of G.

The vertex elimination operation described above has long known applications within linear algebra, and it simulates in graphs the elimination of a variable from subsequent rows during Gaussian elimination of symmetric matrices [18]. The resulting *Elimination Game* [18] is an algorithm that takes as input a graph G and an ordering α of its vertices, and eliminates the vertices of G one by one, in the order defined by α, until the graph becomes empty. Finding an ordering α that minimizes the amount of edges added during this process, called the *fill-in*, is crucial for sparse matrix computations, and a vast amount of results have appeared on this subject during the last 40 years [8, 10, 18, 21]. Our problem Elimination is equivalent to stopping Elimination Game after $|V(G)| - |V(H)|$ steps to see whether the resulting graph at that point is isomorphic to H. A crucial aspect of Elimination Game is the order in which the vertices are chosen, as this influences the fill-in. Note however that, for our problem, only the set of $|V(G)| - |V(H)|$ vertices chosen to be eliminated is important, and not the order in which they are eliminated.

Graph modification problems resulting from operations like vertex deletion, edge deletion, edge contraction, and local complementation are well studied, especially within parameterized complexity [1, 3, 6, 11, 13, 15–17, 19, 24]. Given the wide use of the vertex elimination operation, we find it surprising that this operation has not been studied in the context of graph modification problems before. The only related study we are aware of is by Samdal [22], who generated all eliminations of the $n \times n$ grids for $n \leq 7$.

Our Contribution. In this paper we study the computational complexity of Elimination. In particular, we show that Elimination is W[1]-hard when parameterized by $|V(H)|$ even when both input graphs are split graphs, and W[2]-hard when parameterized by $|V(G)|-|V(H)|$ even when H is a complete graph. On the positive side, for the case when H is complete, we show that Elimination is fixed-parameter tractable when parameterized by $|V(H)|$, and admits a kernel with at most $5|V(H)|$ vertices on connected graphs, which contrasts the hardness of the related Clique problem. We also study the cases when one of the input graphs is a tree. It turns out that the complexity of the problem changes completely depending on which input graph is a tree; we show that if G is a tree then the problem remains NP-complete, whereas if H is a tree then it can be solved in polynomial time. The mentioned kernel result is obtained by proving a combinatorial theorem (Theorem 4) on the maximum number of leaves in a spanning tree of a graph, similar to a proof by Kleitman and West [14]. This result might be of independent interest.

Notation and Terminology. All graphs in this paper are undirected, finite, and simple. Let $G = (V, E)$ be a graph. In case V and E are not specified, we use $V(G)$ and $E(G)$ to denote the vertex set and edge set of a graph G, respectively. The *neighborhood* of a vertex $v \in V$ is the set of its neighbors $N_G(v) = \{w \in V \mid vw \in E\}$, and the *closed neighborhood* of v is the set $N_G[v] = N_G(v) \cup \{v\}$. The *degree* of v is $d_G(v) = |N_G(v)|$. For any subset $A \subseteq V$, we define $N_G[A] = \bigcup_{a \in A} N_G[a]$, $N_G(A) = N_G[A] \setminus A$, and $d_G(A) = |N_G(A)|$. For any subset $A \subseteq V$, $G[A]$ denotes the subgraph of G induced by A. For a subgraph H of G, we write $G \setminus H$ to denote the graph obtained from G by deleting all the vertices of H from G, i.e., $G \setminus H = G[V(G) \setminus V(H)]$.

A *clique* is a set of vertices that are pairwise adjacent. A vertex v is *simplicial* if $N_G(v)$ is a clique. A graph G is *complete* if $V(G)$ is a clique. The complete graph on k vertices is denoted by K_k. An *independent* set is a set of vertices that are pairwise non-adjacent. If G is a bipartite graph, where (A, B) is a partition of V into two independent sets, then we denote it as $G = (A, B, E)$ and we call (A, B) a *bipartition* of G. A graph is a *split graph* it its vertex set can be partitioned into a clique and an independent set.

A *tree* is a connected graph without cycles. A connected subgraph of a tree T is called a *subtree* of T. For a tree T with at least two vertices, we denote by $L(T)$ the set of leaves of T. The remaining set of vertices is denoted by $I(T) = V(T) \setminus L(T)$, and the vertices in $I(T)$ are called the *inner vertices* of T. A vertex is a *cut-vertex* if the removal of the vertex leaves the graph with more connected components than before. For a graph G, by $C(G)$ we denote the set of cut-vertices of G. A connected graph is 2-*connected* if it does not contain a cut-vertex. A maximal 2-connected subgraph of G is called a *biconnected component* (*bicomp* for short), and we denote by $\mathcal{B}(G)$ the set of bicomps of G. Consider the bipartite graph $\mathcal{T}_G$ with the vertex set $C(G) \cup \mathcal{B}(G)$, where $(C(G), \mathcal{B}(G))$ is the bipartition, such that $c \in C(G)$ and $B \in \mathcal{B}(G)$ are adjacent if and only if $c \in V(B)$. This graph $\mathcal{T}_G$ is a tree if G is connected, and is called the *bicomp-tree* of G.

A parameterized problem Q belongs to the class XP if each instance (I, k) can be solved in $|I|^{g(k)}$ time for some function g that depends only on the parameter k, and $|I|$ denotes the size of I. If a problem belongs to XP, then it can be solved in polynomial time for every fixed k. If a parameterized problem can be solved by an algorithm with running time $f(k)\,|I|^{O(1)}$, then we say the problem is *fixed-parameter tractable*. The class of all fixed-parameter tractable problems is denoted by FPT. Between FPT and XP is a hierarchy of parameterized complexity classes $\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{W}[\mathsf{P}] \subseteq \mathsf{XP}$, where hardness for one of the W-classes is considered to be strong evidence for the problem not being contained in FPT. Two instances (I, k) and (I', k') of a parameterized problem are *equivalent* if they are either both yes-instances or both no-instances. A parameterized problem is said to admit a *kernel* if there is a polynomial-time algorithm (called a *kernelization algorithm*) that transforms each instance (I, k) of the problem into an equivalent instance (I', k') of the same problem such that $k' \leq k$ and $|I'| \leq h(k)$ for some function h. If h is a polynomial or a linear function in k, then we say that the problem admits a *polynomial kernel* or a *linear kernel*, respectively. We refer to the textbook by Downey and Fellows [6] for formal background on parameterized complexity.

2 Preliminaries and Hardness of the Elimination Problem

We start this section with an observation that provides a characterization of graphs that have some fixed graph H as an elimination. Our proofs heavily rely on this observation.

Observation 1 ([21]) *Let G and H be two graphs, where $V(H) = \{u_1, \ldots, u_h\}$. Then H is an elimination of G if and only if there exists a set $S = \{v_1, \ldots, v_h\}$ of h vertices in G that satisfies the following: $u_iu_j \in E(H)$ if and only if $v_iv_j \in E(G)$ or there is a path in G between v_i and v_j whose internal vertices are all in $V(G) \setminus S$, for $1 \leq i < j \leq h$.*

For two input graphs G and H that form an instance of Elimination, we let n denote the number of vertices in G. If G and H form a yes-instance, we say that a subset $X \subseteq V(G)$ is a *solution* if H is the resulting graph when all vertices in X are eliminated. By Observation 1, the vertices in X can be eliminated in any order. A vertex which is not eliminated is said to be *saved*. The set $S = V(G) \setminus X$ of saved vertices is called a *witness*.

Since we can check in polynomial time whether a set $S \subseteq V(G)$ of $|V(H)|$ vertices is a witness, Observation 1 immediately implies the following result.

Corollary 1. Elimination *is in* XP *when parameterized by* $|V(H)|$.

Corollary 1 naturally raises the question whether Elimination is fixed-parameter tractable when parameterized by $|V(H)|$. The following theorem shows that this is highly unlikely.

Theorem 1. Elimination *is* W[1]*-hard when parameterized by* $|V(H)|$*, even if both* G *and* H *are split graphs.*

Proof. We give a reduction from the Clique problem, which takes as input a graph G and an integer k, and asks whether G contains a clique on at least k vertices. This problem is known to be W[1]-complete when parameterized by k [6]. We assume that G is a connected graph on at least four vertices, and that $k \geq 4$. From an instance (G, k) of Clique, where $G = (V, E)$, we construct an instance (G^*, H) of Elimination as follows.

We construct a new set of vertices $V_E = \{v_{uw} \mid uw \in E\}$. Our new graph G^* has vertex set $V \cup V_E$, where each vertex v_{uw} in V_E is made adjacent to exactly two vertices in V: u and w. After this, we make V into a clique by adding all possible edges between the vertices of V. This completes the construction of G^*, which is a split graph. Observe that every vertex v_{uw} in V_E has degree 2, and that these are the only vertices of degree 2 in G^*, since we assumed that $|V| \geq 4$. Let H be the split graph with vertex set $C_H \cup I_H$, where $C_H = \{x_1, \ldots, x_k\}$ is a clique and $I_H = \{y_{ij} \mid 1 \leq i < j \leq k\}$ is an independent set, and where every vertex y_{ij} is (only) adjacent to x_i and x_j. We claim that H is an elimination of G^* if and only if G has a clique of size at least k.

Suppose G has a clique $C \subseteq V$ of size at least k. Let $E' \subseteq E$ be the set of edges in $G[C]$, and let $V_{E'} = \{v_{uw} \mid uw \in E'\}$ be the corresponding subset of V_E. Note that, in G^*, the vertices of $V_{E'}$ have no neighbor in $V \setminus C$. Hence the vertices of C, together with the $|C|(|C|-1)/2$ vertices in $V_{E'}$, induce a subgraph in G^* that is isomorphic to H. In order to obtain H from G^*, we first eliminate all the vertices in $V_E \setminus V_{E'}$ in arbitrary order, and then eliminate all the vertices in $V \setminus C$ in arbitrary order. Note that during this procedure we only eliminate vertices that are simplicial in the current graph, which is equivalent to deleting those vertices from the graph. Hence we can obtain H from G^* by eliminating all the vertices in $V(G^*) \setminus (C \cup V_{E'})$, which means that H is an elimination of G^*.

For the reverse direction, suppose H is an elimination of G^*, and let $X \subseteq V(G)$ be a solution. We consider how the graph under consideration changes each time we eliminate a vertex of X. By Observation 1, we may eliminate the vertices of X in arbitrary order, so let us first eliminate the vertices of $X \cap V_E$ before eliminating the vertices of $X \cap V$. Then eliminating a vertex $x \in X \cap V_E$ is equivalent to deleting x from the graph. After we have eliminated all the vertices in $X \cap V_E$, we have obtained a graph G'. Note that G' is a split graph, that some vertices in the maximum clique of G' might be simplicial, and that every vertex of V_E that had degree 2 in G^* still has degree 2 in G'. Every time a vertex $x \in X \cap V$ is eliminated, one of two cases can occur. If x is simplicial, then x is simply deleted from the graph, and the size of the maximum clique decreases by 1. Otherwise, the neighbors of x in V_E become adjacent to all the remaining vertices in V; since we assumed that $k \geq 4$, this means they will get degree at least 3 in the final graph. By assumption, we obtain the graph H after eliminating all vertices in X. Note that the vertices of V that were not eliminated have exactly the same two neighbors in H as they had in G^*. Let C be this set of neighbors. By the construction of H, we conclude that the vertices of C form a clique of size k in G. □

Since Elimination is unlikely to be FPT in general as a result of Theorem 1, it is natural to ask whether certain restrictions on G or H make the problem tractable. In Section 3, we restrict H to be a complete graph; note that due to Theorem 1, restricting H to be a split graph does not suffice to guarantee tractability. In Section 4, we study the variant where either G or H is a tree.

Another possible way of achieving tractability is to investigate a different parameterization of the problem. For instance, instead of choosing the size of the witness as the parameter, we can parameterize Elimination by the size of the solution, i.e., the number of eliminated vertices. The next theorem shows that the problem remains intractable with this parameter.

Theorem 2. Elimination *is* W[2]*-hard when parameterized by* $|V(G)| - |V(H)|$*, even if* H *is a complete graph.*

Proof. We reduce from the Colorful Red-Blue Dominating Set problem. This problem takes as input a bipartite graph $G = (R, B, E)$ with partition classes R and B, an integer k, and a coloring function $c : R \to \{1, \ldots, k\}$. For $1 \leq i \leq k$, let R_i denote the subset of vertices of R with color i. The task is to decide if there exists a set $D \subseteq R$ of k distinctly colored vertices such that D dominates B, i.e., such that $B \subseteq N_G[D]$. This problem is known to be W[2]-hard when parameterized by k (Lemma 39 in [5]). From the reduction in [5] it is clear that we may assume, without loss of generality, that none of the sets R_i is empty and hence $|R| \geq k$; we make this assumption below. Given an instance (G, k, c) of Colorful Red-Blue Dominating Set, where $G = (R, B, E)$, we create an instance $(G^*, K_{|V(G^*)|-k-1})$ of Elimination as follows.

To construct G^*, we start with a copy of G. For each R_i, we add two vertices x_i, y_i and make each of them adjacent to all the vertices in R_i. We also add a vertex z and make it adjacent to all the vertices in R, as well as a vertex z' that is made adjacent to z only. This finishes the construction of G^*; see also Figure 1. We claim that (G, k, c) is a yes-instance of Colorful Red-Blue Dominating Set if and only if $(G^*, K_{|V(G^*)|-k-1})$ is a yes-instance of Elimination, or equivalently, if and only if we can transform G^* into a complete graph by eliminating $k + 1$ vertices.

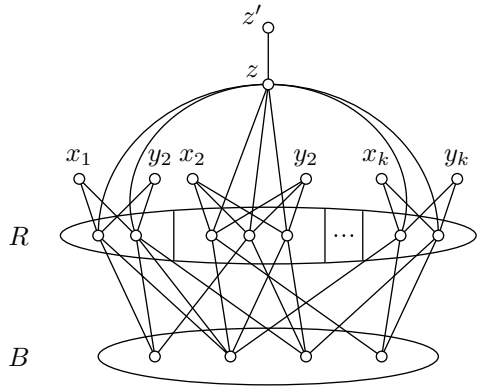

Fig. 1. The graph G^* constructed in the proof of Theorem 2.

Suppose there exists a set $D \subseteq R$ of k distinctly colored vertices such that D is a dominating set of B. For $1 \leq i \leq k$, let d_i be the vertex in D with color i. Graph G^* can be transformed into a complete graph by eliminating the following $k + 1$ vertices. We first eliminate z. This turns the vertices of $R \cup \{z'\}$ into a big clique. We then eliminate the vertices of D one by one. Each time we eliminate a vertex d_i, the vertices x_i and y_i both become adjacent to each other, and to all the (remaining) vertices of the big clique. The same holds for all the vertices of B that are adjacent to d_i. Since D dominates B, the resulting graph is complete.

For the reverse direction, suppose there is a subset $X \subseteq V(G^*)$ of $k+1$ vertices in G^* whose elimination results in a complete graph. In order to ensure that the vertices of $(R_i \cup \{x_i, y_i\}) \setminus X$ are pairwise adjacent in the final complete graph, X must contain at least one vertex from each of the sets $R_i \cup \{x_i, y_i\}$. Since none of the k sets R_i is empty and $|X| = k + 1$, this means that X contains at most one vertex from $B \cup \{z, z'\}$. In order to ensure that z' is adjacent to all other vertices in the final graph, X must contain either z or z'. If X contains neither z nor z',

or if $X \cap (B \cup \{z, z'\}) = \{z'\}$, then there will not be any edge between any two distinct sets R_i and R_j in the final graph. Hence we must have $z \in X$. Eliminating z turns $R \cup \{z'\}$ into a big clique. In order to ensure that all the vertices x_i and y_i are adjacent to the vertices of B in the final graph, X contains exactly one vertex, say d_i, from each of the sets R_i. We claim that the set $D = \{d_1, \ldots, d_k\}$ dominates B. For contradiction, suppose there is a vertex $b \in B$ that is not adjacent to any vertex in D. Then b will not be adjacent to any of the vertices x_i, y_i in the final graph. This contradiction proves that D dominates B, and thus (G, k, c) is a yes-instance of Colorful Red-Blue Dominating Set. □

We point out that the reductions used in the proofs of Theorems 1 and 2 immediately imply that the unparameterized version of Elimination is NP-complete, even if both G and H are split graphs, or if H is a complete graph.

3 Eliminating to a Complete Graph

In this section, we consider a special case of the Elimination problem when H is a complete graph. This corresponds exactly to the problem described in the first paragraph of Section 1. We define the problem Clique Elimination, which takes as input a graph G on n vertices and a positive integer k, and asks whether the complete graph K_k is an elimination of G. Recall that Clique Elimination is W[2]-hard when parameterized by $|V(G)| - k$ due to Theorem 2. In this section, we study the parameterized complexity of Clique Elimination when we choose k as the parameter.

If G contains a tree T with k leaves as a subgraph, then K_k is an elimination of G, as the leaves of T can serve as a witness. It is easy to observe that G contains a tree with k leaves as a subgraph if and only if G contains $K_{1,k}$, i.e., a star with k leaves, as a minor. Moreover, by Observation 1, for any fixed graph H, the property that H is an elimination of a graph G can be expressed in monadic second-order logic. Since graphs that exclude $K_{1,k}$ as a minor have bounded treewidth [20], Courcelle's Theorem [4] implies that Clique Elimination fixed-parameter tractable when parameterized by k.

Even though fixed-parameter tractability of Clique Elimination is already established, two interesting questions remain. Does the problem admit a polynomial kernel? Does there exist an algorithm for the problem with single-exponential dependence on k? We provide an affirmative answer to both questions below. In particular, we prove the following result.

Theorem 3. Clique Elimination *on connected graphs admits a kernel with at most* $5k$ *vertices that can be computed in* $O(n^3)$ *time.*

We would like to remark that the assumption that the input graph is connected is probably necessary: Clique Elimination on general graphs admits a simple composition algorithm that takes the disjoint union of instances, so the existence of a polynomial kernel in the general setting would imply that NP $\subseteq$ coNP/poly. We refer an interested reader to the work of Bodlaender et al. [2] for an introduction to the methods of proving implausibility of polynomial kernelization algorithms.

As a result of Theorem 3, we can obtain an algorithm with single-exponential dependence on k, thus outperforming the algorithm obtained from the aforementioned combination of meta-theorems.

Corollary 2. Clique Elimination *can be solved in* $O(12.21^k + n^3)$ *time and polynomial space.*

Proof. We first observe that Clique Elimination can be solved by in time $O\left(\binom{n}{k} n^2\right)$ on any instance (G,k) with $|V(G)| = n$ as follows: for each subset $S \subseteq V(G)$ such that $|S| = k$, we test whether eliminating all vertices in $V(G) \setminus S$ yields the graph K_k. The latter test can be performed in time $O(n^2)$ by using an algorithm due to Tarjan and Yannakakis [23].

Let (G,k) be an instance of Clique Elimination. The pair (G,k) is a yes-instance if and only if there is a connected component G_i of G such that (G_i,k) is a yes-instance due to the fact that for any witness S, all the vertices of S must belong to a single connected component of G. Hence we may assume that G is connected. We apply the $O(n^3)$-time kernelization algorithm of Theorem 3 to obtain an equivalent instance (G',k') such that $k' \leq k$ and G' is a connected graph on at most $5k$ vertices. We then solve the problem on this instance in time

$$O\left(\binom{5k}{k}(5k)^2\right) = O\left(\binom{5k}{k}k^2\right)$$

using the brute-force approach described above. It follows from Sterling's formula that

$$\binom{5k}{k} = O\left(\frac{\left(\frac{5k}{e}\right)^{5k}}{\left(\frac{4k}{e}\right)^{4k} \cdot \left(\frac{k}{e}\right)^{k}}\right) = O\left(\left(\frac{5^5}{4^4}\right)^k\right) = O(12.21^k)\,.$$

Since $\frac{5^5}{4^4} < 12.21$, it holds that $O\left(\binom{5k}{k}k^2\right) = O(12.21^k)$, yielding the claimed overall running time of $O(12.21^k + n^3)$. □

The remainder of this section is devoted to the proof of Theorem 3. Before presenting the formal proof, we give some intuition behind our approach. Our kernelization algorithm is based on the observation that the max-leaf number of a graph, i.e., the maximum number of leaves a spanning tree of the graph can have, is a lower bound on the size of a complete graph that can be obtained as an elimination. Kleitman and West [14] showed that a connected graph G with minimum degree at least 3 admits a spanning tree with at least $|V(G)|/4+2$ leaves. Their result immediately leads to a linear kernel for Clique Elimination provided that the input graph G has minimum degree at least 3. Unfortunately, we are unable to get rid of all vertices of degree at most 2 in our setting. However, we can modify our input graph in polynomial time such that we either can solve the problem directly, or obtain a new graph G^* with no vertices of degree 1 and with no edge between any two vertices of degree 2. We then prove a modified version of the aforementioned result by Kleitman and West [14], namely that such graphs G^* admit a spanning tree with at least $|V(G^*)|/5 + 2$ leaves. This leads to Theorem 3.

We now proceed with the formal proof of Theorem 3. Following Observation 1, we will be looking for a set S that is a witness of cardinality k, i.e., every two non-adjacent vertices of S can be connected by a path of which all internal vertices are outside S.

We start by providing four reduction rules, i.e., polynomial-time algorithms that, given an instance (G,k) of Clique Elimination, output an instance (G',k') of the same problem. A reduction rule is *safe* if the instances (G,k) and (G',k') are equivalent. Since Theorem 3 states that Clique Elimination admits a linear kernel for *connected* graphs, we assume the graph G in every input instance (G,k) to be connected. Moreover, since we only eliminate vertices in each of the reduction rules and the vertex elimination operation preserves connectivity, the graph G' in every output instance (G',k') is also connected.

Reduction Rule 1 *If $k \leq 3$, then output a trivial yes-instance*

- *if $k = 1$ and G contains at least one vertex, or*
- *if $k = 2$ and G contains at least one edge, or*

– *if $k = 3$ and either G contains a vertex of degree at least 3 or G is a cycle,*

and output a trivial no-instance otherwise.

Lemma 1. *Reduction Rule 1 is safe.*

Proof. Reduction Rule 1 is trivially safe in case $k \leq 2$. Suppose $k = 3$. If G contains a vertex u of degree at least 3, then K_3 is an elimination of G, as three neighbors of u can serve as a witness. Recall that G is a connected graph. If G contains no vertex of degree at least 3, then G is either a cycle or a path. If G is a cycle, then eliminating all but three vertices of G yields K_3, showing that K_3 is also an elimination of G in this case. If G is a path, then K_3 is not an elimination of G, since eliminating any vertex in a path results in a shorter path. □

Reduction Rule 2 *If $k > 3$ and G contains a vertex v of degree 1, then eliminate its sole neighbor v' to obtain a graph G'. Output the instance (G', k).*

Lemma 2. *Reduction Rule 2 is safe.*

Proof. We need to argue that if we can find a witness S, then we can also find a witness S' of the same size that does not contain v'. If $v' \notin S$, then we set $S' = S$. If $v' \in S$, then we claim that $v \notin S$. For contradiction, suppose $v \in S$. Since v is adjacent only to v' and all the vertices of S form a clique in the graph obtained by eliminating all the vertices in $V(G) \setminus S$, we must have $k \leq 2$. This contradicts the assumption that $k > 3$. We now set $S' = (S \setminus \{v'\}) \cup \{v\}$ to obtain a witness set of the same cardinality that does not contain v'. □

Reduction Rule 3 *If $k > 3$ and G contains a triangle v', v_1, v_2 such that v_1, v_2 are of degree 2, then eliminate v' to obtain a graph G'. Output the instance (G', k).*

Lemma 3. *Reduction Rule 3 is safe.*

Proof. Again, we need to argue that if we can find a witness S, then we can also find a witness S' of the same size that does not contain v'. If $v' \notin S$, then we set $S' = S$. Suppose that $v' \in S$. Since $k > 3$, neither v_1 nor v_2 belongs to S. We set $S' = (S \setminus \{v'\}) \cup \{v_1\}$ to obtain a witness set of the same cardinality that does not contain v'. □

Reduction Rule 4 *If $k > 3$ and G has a path v_0, v_1, v_2, v_3 such that v_1, v_2 are of degree 2, then eliminate v_0 to obtain a graph G'. Output the instance (G', k).*

Lemma 4. *Reduction Rule 4 is safe.*

Proof. We need to argue that if we can find a witness S, then we can also find a witness S' of the same size that does not contain v_0. If $v_0 \notin S$, then we set $S' = S$. Suppose that $v_0 \in S$. Since $k > 3$, the set S contains at most one vertex from the set $\{v_1, v_2, v_3\}$, as otherwise one of them could be connected to at most two other vertices from S via paths avoiding other vertices from S. If $|S \cap \{v_1, v_2, v_3\}| = 0$, then we take $S' = (S \setminus \{v_0\}) \cup \{v_1\}$, while if $|S \cap \{v_1, v_2, v_3\}| = 1$, then we take $S' = (S \setminus \{v_0, v_1, v_2, v_3\}) \cup \{v_1, v_2\}$. It is easy to check that S' defined in this manner is a witness of the same cardinality that does not contain v_0. □

We are now ready to describe our kernelization algorithm in detail and argue why it runs in time $O(n^3)$. Let (G, k) be an instance of Clique Elimination that is given as input to our kernelization algorithm. If $k \leq 3$, then the algorithm applies Reduction Rule 1 and terminates. If $k > 3$, then we exhaustively apply Reduction Rules 2, 3 and 4 as follows; note that the parameter does not change during the execution of any of these three rules. First, we check

in linear time whether G has a vertex of degree 1, and apply Reduction Rule 2 if this is the case. If G has no vertex of degree 1, then we check, again in linear time, whether G has two adjacent vertices v_1, v_2 of degree 2. If so, then we apply Reduction Rule 3 if v_1 and v_2 have a common neighbor v', and we apply Reduction Rule 4 otherwise. Suppose G has no vertices of degree 1 and no edge between any two vertices of degree 2. If G has at most $5k - 11$ vertices, then we output the current instance (G, k) as the obtained kernel. Otherwise, i.e., if G has at least $5k - 10$ vertices, then we can safely return a trivial yes-instance due to Theorem 4 below, which is our modified version of the aforementioned result by Kleitman and West [14].

From the description of the algorithm it is clear that deciding which reduction rule to apply can be done in linear time. Applying any of the reduction rules takes $O(n^2)$ time, which is the time it takes to eliminate a vertex. Since the number of vertices strictly decreases at every step, the kernelization algorithm has an overall running time of $O(n^3)$. This concludes the proof of Theorem 3.

The proof of Theorem 4 below closely follows the proof of the aforementioned result by Kleitman and West [14], but requires a more extensive case analysis when picking the initial tree T that is grown into a spanning tree with the required number of leaves, and we add a fourth "growing rule" to the three rules that were used by Kleitman and West. Note that we cannot drop the condition that no two vertices of degree 2 are adjacent in the theorem below, as no cycle admits a spanning tree with more than two leaves.

Theorem 4. *Let G be a connected graph with minimum degree at least 2 such that no two vertices of degree 2 are adjacent. Then G admits a spanning tree with at least $|V(G)|/5 + 2$ leaves, and this bound is best possible.*

Proof. We gradually grow a tree T in G keeping track of three parameters:

- n, the number of vertices in T;
- l, the number of leaves in T;
- m, the number of *dead* leaves in T, i.e., leaves of T that have no neighbor in $G \setminus T$.

The tree will be grown via a number of operations called *expansions*: by an expansion of a vertex $x \in V(T)$ we mean the adding of all the vertices $v \in V(G) \setminus V(T)$ with $xv \in E(G)$ and all the edges $xv \in E(G)$ with $v \notin V(T)$ to the tree T. We start with a tree T such that only leaves of T have neighbors in $G \setminus T$. Therefore, if we only use expansions to grow the tree, at each step of the growth process only the leaves of T are adjacent to $G \setminus T$. A leaf that is not dead, is called *alive*.

For a tree T, we consider the potential $\phi(T)$ defined as $\phi(T) = 4l + m - n$. The goal is to:

(a) find a starting tree T with $\phi(T) \geq 9$;
(b) provide a set of growing rules, such that there is always a rule applicable unless T is a spanning tree, and $\phi(T)$ does not decrease during the application of any rule;
(c) prove that during the whole process the potential increases by at least 1.

If goals (a), (b) and (c) are accomplished, then we can grow T using the rules until it becomes a spanning tree; in this situation we have $l = m$ and $n = |V(G)|$. As the potential increased by at least 1 during the whole process, we infer that $5l \geq |V(G)| + 10$, and hence $l \geq |V(G)|/5 + 2$, as claimed.

Goal (a) can be achieved by a careful case study, the full description of which we give at the end of the proof.

Having chosen the starting tree T, we can proceed with the growing rules. In order to grow the tree we always choose the rule that has the lowest number among the applicable ones, i.e., when applying a rule, we can always assume that the ones with lower numbers are not

applicable. The four growing rules are illustrated in Figure 2. We would like to point out that the first three rules were already used in the original proof of Kleitman and West [14].

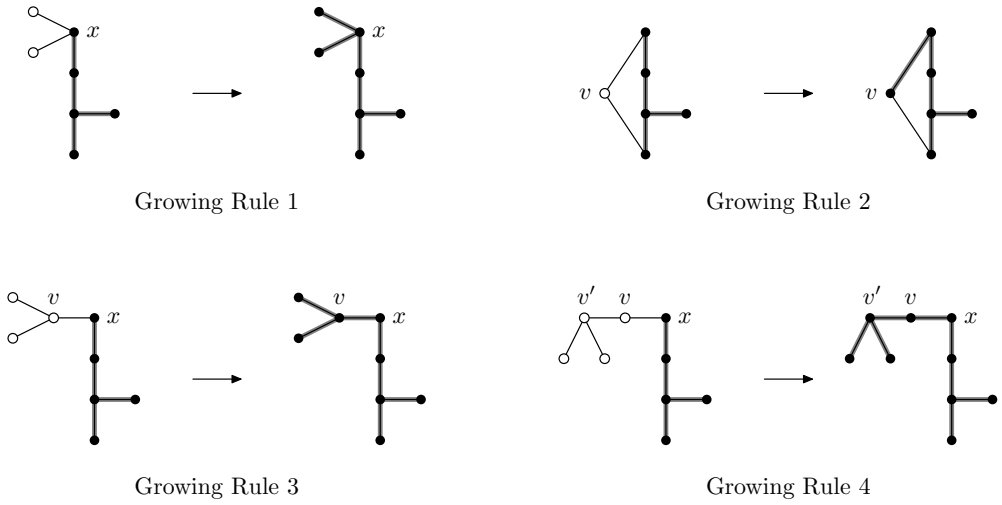

Fig. 2. An illustration of the four growing rules. Black vertices and bold edges belong to the tree T.

Growing Rule 1 If some leaf x of T has at least two neighbors from $G \setminus T$, expand x. The potential $\phi(T)$ increases by at least $4 \cdot (d-1) - d = 3d - 4 \geq 2$, where $d \geq 2$ is the number of the aforementioned neighbors from $G \setminus T$.

Growing Rule 2 If some vertex $v \in V(G \setminus T)$ is adjacent to at least two leaves of T, expand one of these leaves. Observe that, as Rule 1 was not applicable and only leaves of T are adjacent to $G \setminus T$, this expansion results in adding only v to T. Moreover, all the remaining leaves adjacent to v were alive but become dead, so the potential $\phi(T)$ increases by at least $1 - 1 = 0$.

Growing Rule 3 If there is a vertex $v \in V(G \setminus T)$ of degree at least 3 in G that is adjacent to a leaf x of T, expand x (which results in adding only v to T, as Rule 1 was not applicable) and then expand v. The potential increases by at least $4 \cdot (d-2) - d = 3d - 8 \geq 1$, where $d \geq 3$ is the degree of v, as all the other neighbors of v are added to T as leaves, due to Rule 2 not being applicable.

Growing Rule 4 If there is a vertex $v \in V(G \setminus T)$ of degree 2 in G that is adjacent to a leaf x of T, expand x (which results only in adding v as a leaf, as Rule 1 was not applicable), then expand v, and then expand the second neighbor v' of v that became a leaf in T during the previous expansion. Note that v' could not be already in T, as otherwise Rule 2 would be triggered on vertex v. Since we assumed that no vertices of degree 2 are adjacent in G, the degree of v' is at least 3 and, as Rule 3 was not applicable, none of the neighbors of v' was in T. Denote by d the degree of v'; therefore, we have added to the tree T exactly $d+1$ vertices (v, v' and $d-1$ other neighbors of v') and increased the number of leaves by exactly $d-2$. Hence, the increase of the potential is $4(d-2) - (d+1) = 3d - 9 \geq 0$, as $d \geq 3$.

Let us now argue why goal (c) is achieved. It is clear that if Growing Rule 1 or 3 is applied at least once, then the potential increases by at least 1. Suppose only Growing Rules 2 and 4 are applied during the whole process. Let x be a vertex of G that was added to T as a leaf during the very last rule application. Then x is a dead leaf. Since this was not taken into account when we determined a lower bound of 0 on the increase of the potential, the potential increases by at least 1. Thus, from the previously described analysis we conclude that using the presented method we are able to grow a tree with at least $|V(G)|/5 + 2$ leaves.

We now show that goal (a) can be achieved, i.e., that we can always find a starting tree T in G such that $\phi(T) \geq 9$. Recall that G is a connected graph with minimum degree at least 2 such that no two vertices of degree 2 are adjacent. This implies that the maximum degree of G is at least 3. We distinguish several cases below, and argue how we can find a starting tree T with $\phi(T) \geq 9$ in each of these cases (see Figure 3 for guidance along the proof):

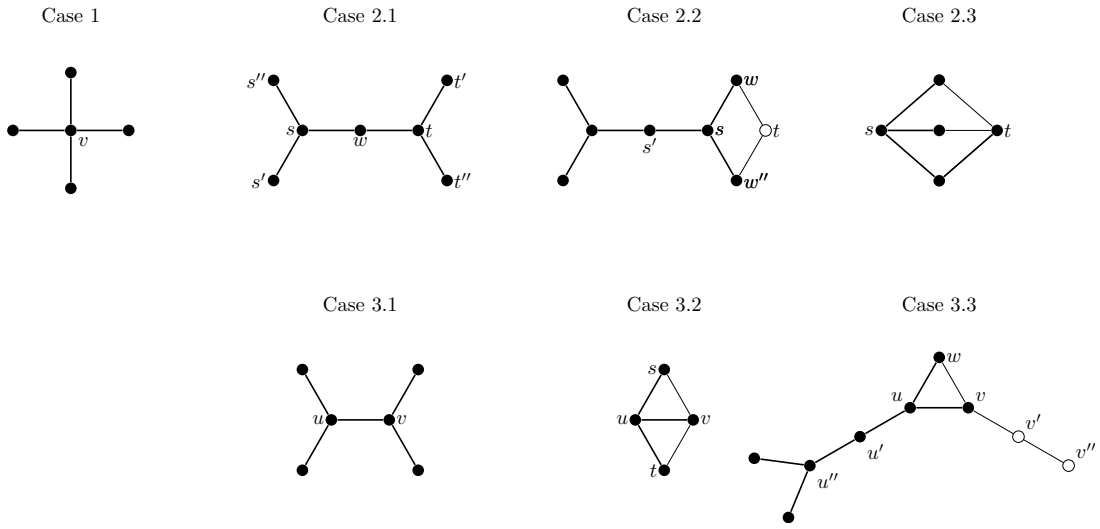

Fig. 3. The trees T chosen to start the growing process, for each of the cases.

1. If the maximum degree of G is at least 4, we start with T being a star consisting of a vertex v of degree at least 4 as the center and all its neighbors attached as pendants. The potential of this tree T is equal to at least $4d - (d+1) = 3d - 1 \geq 9$, where $d \geq 4$ is the degree of v. From now on, we assume that the maximum degree of G is equal to 3.
2. Assume that no vertices of degree 3 are adjacent in G. Take any vertex w of degree 2 and let s, t be his neighbors. Note that both s and t have degree 3.
   - **2.1.** Assume that s and t have exactly one common neighbor, namely w. Denote the remaining two neighbors of s by s', s'' and the remaining two neighbors of t by t', t''. Obviously, s', s'', t', t'', w are pairwise distinct. We now take as T the tree consisting of 7 vertices: $s, s', s'', t, t', t'', w$, obtained by attaching s', s'', t', t'' to the path $s - w - t$ as leaves. This tree T has potential at least $4 \cdot 4 - 7 = 9$.
   - **2.2.** Assume that s and t have exactly two common neighbors w and w'. Let s' be the remaining neighbor of s. Since we assumed that no vertices of degree 3 are adjacent, the degree of s' is equal to 2. Observe that the neighbors of s' can have only s' as their common neighbor, because the remaining neighbors of s are already adjacent to t,

which is distinct from the second neighbor of s'. Thus, we can follow the same choice of T as in the Case 2.1, but starting with s' instead of w.

2.3. Assume that s and t have exactly three common neighbors. Then, the whole graph G is just s and t connected via three internally vertex-disjoint paths of length 2. It is clear that Theorem 4 holds in this case.

3. Now we can safely assume that G contains two adjacent vertices of degree 3; denote them by u, v.

3.1. Assume that u and v have no common neighbors. Then we take as T the tree on 6 vertices consisting of the edge uv and all neighbors of u, v attached to either u or v as leaves. This tree has 4 leaves, so its the potential is equal to at least $4 \cdot 4 - 6 = 10 \geq 9$.

3.2. Assume that u and v have exactly two common neighbors s and t. Observe that then $N(u) = \{v, s, t\}$ and $N(v) = \{u, s, t\}$. Take as T the star having u as the center and v, s, t as three leaves. Observe that the potential of this tree is at least $4 \cdot 3 + 1 - 4 = 9$, as v is a dead leaf.

3.3. Finally, assume that u and v have exactly one common neighbor w. Observe that if u and w had any common neighbor apart from v, or v and w had any common neighbor apart from u, then we would be able to proceed with the pair (u, w) or with the pair (v, w) as in Case 3.2. Therefore, assume that $N(u) \cap N(w) = \{v\}$ and $N(v) \cap N(w) = \{u\}$. Let u' be the remaining neighbor of u and v' be the remaining neighbor of v. By what we assumed so far, we know that $u' \neq v'$ and neither u' nor v' is adjacent to w. If any of them had degree at least 3, then we would be able to apply Case 3.1 with the pair (u, u') or with the pair (v, v'). Therefore, assume that both u' and v' have degree 2. Then u' and v' are not adjacent, since we assumed that no vertices of degree 2 are adjacent in G. Denote their second neighbors, different from u and v, by u'' and v'', respectively. As the maximum degree in G is equal to 3, at least one of them is not adjacent to w; assume without loss of generality that u'' is not adjacent to w. Of course, u'' is also not adjacent to v, as $u'' \notin \{u, w, v'\}$ and has degree 3. Therefore, we can use the same reasoning as in Case 2.1, starting with the vertex u' as the degree-2 vertex.

It remains to show that the bound $|V(G)|/5 + 2$ in Theorem 4 is best possible. An infinite family of graphs G satisfying the premises of the theorem and having a spanning tree with exactly $|V(G)|/5 + 2$ leaves can be obtained by connecting a number of diamonds in the way as shown in Figure 4. This completes the proof of Theorem 4. □

4 The Elimination Problem on Trees

In this section, we study Elimination when G or H is a tree. When H is a tree, we show in Section 4.1 that the problem can be solved in polynomial time. We then show in Section 4.2 that when G is a tree, the problem is NP-complete.

4.1 Eliminating to a Tree in Polynomial Time

In this subsection, we present a polynomial-time algorithm for solving Elimination on input pairs (G, H) where H is a tree. We first prove a sequence of lemmas that show that if a graph G contains a tree H as an elimination, then there exists a witness that satisfies some nice structural properties. These properties will then be exploited in the proof of Theorem 5, where our algorithm uses a dynamic programming approach on the bicomp-tree of G to find a

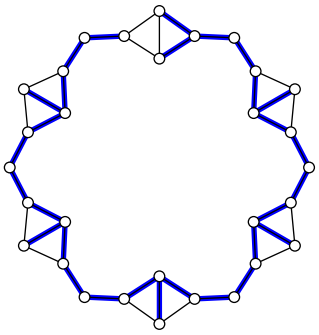

Fig. 4. A graph on 30 vertices for which the maximum possible number of leaves in a spanning tree is exactly $30/5 + 2 = 8$; the bold (blue) edges indicate a spanning tree with eight leaves. Generalizing this example yields an infinite family of graphs showing that the bound in Theorem 4 is tight.

witness with the aforementioned nice properties, or to conclude that H is not an elimination of G.

Let (G, H) be an instance of Elimination where G is a connected graph and H is a tree on at least 3 vertices. This implies in particular that $L(H) \neq \emptyset$ and $I(H) \neq \emptyset$; recall that $L(H)$ and $I(H)$ denote the sets of leaves and inner vertices of H, respectively. Also recall that $C(G)$ and $\mathcal{B}(G)$ denote the sets of cut-vertices and bicomps of G, respectively, and that $\mathcal{T}_G$ is the bicomp-tree of G.

Throughout this subsection, up to the statement of Theorem 5, we assume that (G, H) is a yes-instance, i.e., that H is an elimination of G. Let $S = \{v_x \mid x \in V(H)\}$ be a witness, where v_x is the vertex of G that corresponds to the vertex x of H, and let $X = V(G) \setminus S$ be the corresponding solution yielding H. The witness S satisfies the structural properties given in the two following lemmas.

Lemma 5. *For any bicomp $B \in \mathcal{B}(G)$ it holds that $|V(B) \cap S| \leq 2$, and if $v_x, v_y \in V(B) \cap S$ for $x \neq y$, then $xy \in E(H)$.*

Proof. To obtain a contradiction, assume that there is a bicomp $B \in \mathcal{B}(G)$ that contains three vertices $v_x, v_y, v_z \in S$. Since any bicomp with at least three vertices is a 2-connected graph, B has two vertex-disjoint v_x, v_y-paths. Suppose, for contradiction, that at least one internal vertex v_q of one of these paths belongs to S, i.e., is not eliminated. Then x, y and q belong to a cycle of length at least 3 in H, contradicting the assumption that H is a tree. Hence, all internal vertices of these two v_x, v_y-paths are eliminated, and consequently v_x and v_y are adjacent in the graph obtained from G by the elimination of X. By the same arguments, we conclude that v_x, v_z and v_y, v_z are adjacent in this graph, i.e., it has a triangle; a contradiction. To prove the second claim of the lemma, it is sufficient to observe that $V(B) \cap S = \{v_x, v_y\}$ and, therefore, B contains a v_x, v_y-path that avoids other vertices of S. Hence, v_x and v_y are adjacent in the graph obtained from G by the elimination of the vertices of X, and $xy \in E(H)$. □

Lemma 6. *For any $x \in I(H)$, $v_x \in C(G)$.*

Proof. To obtain a contradiction, assume that there is a vertex $x \in I(H)$ such that v_x is not a cut-vertex of G. Let B be the bicomp of G that contains v_x. Since x is an inner vertex of H, x is adjacent to at least two vertices y_1, y_2. For $i = 1, 2$, G has a v_x, v_{y_i}-path P_i that avoids the

vertices of $S \setminus \{v_x, v_{y_i}\}$. Let u_1 and u_2 be the vertices adjacent to v_x in P_1 and P_2 respectively. Observe that $u_1, u_2 \in V(B)$, because v_x is not a cut-vertex. The 2-connected graph B contains a u_1, u_2-path P that avoids v_x. Suppose that some vertex $v_z \in S$ is an inner vertex of P. By Lemma 5, v_z is the unique vertex of S in P. By concatenating the v_z, u_i-subpath of P and the u_i, v_{y_i}-subpath of P_i, we obtain a v_z, v_{y_i}-walk in G that avoids other vertices of S for $i = 1, 2$. It follows that z is adjacent to y_1 and y_2, which would imply a cycle with vertices z, y_1, x, y_2 in H; a contradiction. Therefore, the set of inner vertices of P does not include any vertex of S. Then the concatenation of the $v_{y_1}u_1$-subpath of P_1, P, and the u_2, v_{y_2}-subpath of P_2 gives a v_{y_1}, v_{y_2}-walk in G that avoids $S \setminus \{v_{y_1}, v_{y_2}\}$. This means that y_1 and y_2 are adjacent in H, yielding the desired contradiction. □

Lemmas 5 and 6 exhibit some properties of the witness S, and these properties hold for any witness. In particular, Lemma 6 shows that for every inner vertex x of H, the corresponding vertex v_x in S is a cut-vertex in G. Lemmas 7 and 8 below imply that there exists a witness S' such that a similar property holds for the leaves of H. To be more precise, Lemma 8 implies that there exists a witness S' such that for every leaf x of H, the corresponding vertex v'_x in S' is either a cut-vertex of G, or a vertex of a bicomp B of G that contains exactly one cut-vertex of G. Lemma 7 shows that in the latter case, replacing v'_x in S' by any other vertex of $V(B) \setminus C(G)$ yields another witness.

Lemma 7. *Let $x \in L(H)$ and suppose that $v_x \in V(B) \setminus C(G)$ for a bicomp B. Let v'_x be an arbitrary vertex of $V(B) \setminus C(G)$ and $S' = (S \setminus \{v_x\}) \cup \{v'_x\}$. Then the graph obtained from G by the elimination of $X' = V(G) \setminus S'$ is isomorphic to H, where the isomorphism maps any vertex $y \in V(H) \setminus \{x\}$ to v_y and maps x to v'_x.*

Proof. Since the lemma trivially holds when $v'_x = x_x$, we assume that $v'_x \neq v_x$. First, we observe that $v'_x \notin S$. Otherwise, $v'_x = v_z$ for some leaf z of T, since $v_z \in C(G)$ for any inner vertex z due to Lemma 6. Then by Lemma 5, the leaves z and x are adjacent in H; a contradiction.

Let y be the unique inner vertex in H that is adjacent to x. The graph G has a v_x, v_y-path P that avoids the vertices of $S \setminus \{v_x, v_y\}$. By Lemma 6, $v_y \in C(G)$. The bicomp B contains a v'_x, v_x-path P'. If P' has a vertex $v_z \in S$ distinct from v_x, then by Lemma 5, z is adjacent to x in H. Hence, $z = y$ and the v'_x, v_y-subpath of P' avoids other vertices of S'. If P' has no vertices from S except v_x, then the v'_x, v_y-walk obtained by the concatenation of P' and P avoids other vertices of S'. In both cases we conclude that v'_x and v_y are adjacent in the graph obtained from G by the elimination of X'.

Now suppose that there is a v'_x, v_z-path P in G that avoids the vertices of $S' \setminus \{v'_x, v_z\}$ for a vertex $z \in V(H)$ such that $z \neq y$. If P contains a vertex $u \in S \setminus \{v_z\}$, then $u = v_x$, since all other vertices of S' are also vertices of S. Then the v_x, v_z-subpath of P avoids the vertices of $S \setminus \{v_x, v_z\}$. This implies that v_x and v_z are adjacent in the graph obtained from G by eliminating X; a contradiction, as v_x is only adjacent to v_y in this graph. Hence, P avoids the vertices of $S \setminus \{v_z\}$. The 2-connected graph B has a v_x, v'_x-path P' that avoids v_y. Since P' is a path in B, P' cannot contain any vertex of S except v_x. Then the v_x, v_z-walk obtained by the concatenation of the path P' and P avoids the vertices of $S \setminus \{v_x, v_z\}$; a contradiction.

We conclude that there is a v'_x, v_z-path in G that avoids $S' \setminus \{v'_x, v_z\}$ if and only if $z = y$.

It remains to prove that replacing x by x' in the witness S does not influence the adjacencies between any other vertices in H. Assume that for two vertices $z_1, z_2 \in V(H) \setminus \{x\}$, there is a v_{z_1}, v_{z_2}-path P in G that avoids the vertices of $S' \setminus \{v_{z_1}, v_{z_2}\}$ but includes a vertex from $S \setminus \{v_{z_1}, v_{z_2}\}$. Then P contains v_x, and it follows that v_x is adjacent to v_{z_1} and v_{z_2} in the graph obtained from G by eliminating X; a contradiction. Finally, suppose that for two vertices $z_1, z_2 \in V(H) \setminus \{x\}$, there is a v_{z_1}, v_{z_2}-path P in G that avoids the vertices of $S \setminus \{v_{z_1}, v_{z_2}\}$

but includes a vertex from $S' \setminus \{v_{z_1}, v_{z_2}\}$. Then P contains v'_x, and the v'_x, v_{z_1}- and v'_x, v_{z_2}-subpaths of P avoid the vertices of $S' \setminus \{v'_x, v_{z_1}\}$ and of $S' \setminus \{v'_x, v_{z_2}\}$, respectively. Hence, $y = z_1 = z_2$; a contradiction. □

Lemma 8. *Let $x \in L(H)$ and suppose that $v_x \in V(B) \setminus C(G)$ for a bicomp B where $|V(B) \cap C(G)| \geq 2$. Then there is a vertex $v'_x \in V(B) \cap C(G)$ such that the graph obtained from G by the elimination of $X' = V(G) \setminus S'$, where $S' = (S \setminus \{v_x\}) \cup \{v'_x\}$, is isomorphic to H, where the isomorphism maps any vertex $y \in V(H) \setminus \{x\}$ to v_y and maps x to v'_x.*

Proof. For a cut-vertex $c \in V(B) \cap C(G)$, denote by G^c the subgraph of G obtained by the removal of the vertices of the connected components of $G - c$ that do not contain the vertex v_x. We claim that if $|V(B) \cap C(G)| \geq 2$, then there is $c \in V(B) \cap C(G)$ such that $S \subseteq V(G^c)$.

To prove the claim, assume for contradiction that for each $c \in V(B) \cap C(G)$, there is $v_z \in S$ such that $v_z \notin V(G^c)$. We choose $c_1, c_2 \in V(B) \cap C(G)$ as follows. By Lemma 5, B contains at most two vertices of S, and if B contains $v_y \neq v_x$ for $y \in V(H)$, then y is adjacent to x in H. By Lemma 6, $v_y \in V(B) \cap C(G)$ and we set $c_1 = v_y$, the vertex $c_2 \in V(B) \cap C(G) \setminus \{c_1\}$ is chosen arbitrary. If B has no other vertices of S except v_x, then c_1, c_2 are arbitrary distinct vertices of $V(B) \cap C(G)$. The 2-connected graph B has a v_x, c_1-path P_1 and a v_x, c_2-path P_2 that avoid c_2 and c_1 respectively. By our assumption, for each $i \in \{1, 2\}$, there is a vertex $v_{z_i} \in S$ such that $v_{z_i} \notin V(G^{c_i})$. Then there is a c_i, v_{z_i}-path P'_i for each $i \in \{1, 2\}$. Let v_{y_i} be the first vertex from $S \setminus \{v_x\}$ on the path Q_i obtained by the concatenation of P_i and P'_i, if we are looking from v_x. Then the v_x, v_{y_i}-subpath of Q_i avoids the vertices of $S \setminus \{v_x, v_{y_i}\}$ for each $i \in \{1, 2\}$. It means that v_x is adjacent to v_{y_1} and v_{y_2} in the graph obtained from G by the elimination of X, contradicting the assumption that x is a leaf of H.

We now use this claim as follows. Let $c \in V(B) \cap C(G)$ be a cut-vertex such that $S \subseteq V(G^c)$, and let $Y = V(G) \setminus V(G^c)$. By our claim, $Y \subseteq X$, and the elimination of X can be seen as the consecutive elimination of Y and $X \setminus Y$. Observe that the graph obtained from G by the elimination of Y is the graph G^c, and c is not a cut-vertex of G^c. By Lemma 7, we can replace v_x by c in S. □

Lemmas 6, 7 and 8 together imply that there exists a witness S'' that is a subset of $C(G) \cup U$ for any subset $U \subseteq V(G) \setminus C(G)$ containing exactly one vertex of $V(B) \setminus C(G)$ for every block B with $|V(B) \cap C(G)| = 1$ and no vertices of any other block of G; note that a block B satisfies $|V(B) \cap C(G)| = 1$ if and only if B corresponds to a leaf of $\mathcal{T}_G$. The existence of such a witness S'' will be used in the proof of Theorem 5, where our algorithm for Elimination will use a dynamic programming approach to find such a witness S'', or to conclude that no such witness, and hence no witness at all, exists. In order to be able to explain this in more detail, we need to introduce some additional terminology and prove two more lemmas.

We choose an arbitrary inner vertex z of H and say that it is the *root* of H. The root defines the parent-child relation between any two adjacent vertices of H. For any two vertices $x, y \in V(H)$, we say that y is a *descendant* of x if x lies on the unique path in H from y to the root z. If y is a descendant of x and $xy \in E(H)$, then y is a *child* of x, and x is the *parent* of y. By definition, every vertex $x \in V(H)$ is a descendant of itself. For a vertex $x \in V(H)$, H_x denotes the subtree of H induced by the descendants of x, and for a vertex $x \in V(H)$ with a child y, H_{xy} is the subtree of H induced by x and the descendants of y.

Consider $r = v_z \in V(G)$. We choose r to be the *root* of the bicomp-tree $\mathcal{T}_G$ of G. By Lemma 6, r is a cut-vertex in G. The root r defines the parent-child relation on $\mathcal{T}_G$. Each bicomp B is a child of some inner vertex c in $\mathcal{T}_G$, and we say that the vertices of B are *children* of the corresponding cut-vertex c in G. A vertex $v \in V(G)$ is a *descendant* of a cut-vertex c if v is a child of some descendant of c in $\mathcal{T}_G$. For a cut-vertex c, we write G_c to denote the subgraph of G induced by the descendants of c. For a cut-vertex c and a bicomp B such that B is a child

of c in $\mathcal{T}_G$, G_{cB} is the subgraph of G induced by the vertices of B and the descendants of all cut-vertices $c' \in V(B) \setminus \{c\}$.

Now consider two vertices p and q in H, such that neither is a descendant of the other, and let x be their lowest common ancestor. A crucial observation in our algorithm is that v_p and v_q are descendants of v_x in G, and there are exactly two bicomps B' and B'' such that B' and B'' are children of v_x and such that v_p belongs to $G_{v_x B'}$ and v_q belongs to $G_{v_x B''}$. In particular, there does not exist a single bicomp B such that B is a child of v_x and both v_p and v_q belong to the graph $G_{v_x B}$. The following lemma formalizes and generalizes this idea.

Lemma 9. *For any inner vertex $x \in V(H)$, if $y \in V(H)$ is a descendant of x in H, then v_y is a descendant of v_x in G. Moreover, if $y_1, \ldots, y_l$ are the children of x in H, then there are distinct children $B_1, \ldots, B_l$ of v_x in the bicomp-tree for which the following holds: for each $i \in \{1, \ldots, l\}$, if $y \in V(H_{xy_i})$, then $v_y \in G_{v_x B_i}$.*

Proof. We prove the first claim of the lemma by induction with respect to the structure of H. Clearly, for each $y \in V(H)$, y is a descendant of z in H and v_y is a descendant of $r = v_z$ in G. Suppose that x is an inner vertex of H and let y be the parent of x. We assume that for any descendant y' of y in H, $v_{y'}$ is a descendant of v_y in G, and we prove that for any descendant y' of x in H, $v_{y'}$ is a descendant of v_x in G. Assume that, contrary to the claim, x has descendants for which it is not so. Denote by B the bicomp of G such that B is the parent of the cut-vertex v_x in the bicomp-tree. If there is a descendant y' of the vertex x such that $v_{y'} \in V(B)$, then $v_{y'}$ is not a descendant of v_x and we consider this vertex. Notice that in this case y' is adjacent to x in H and $V(B) \cap S = \{v_x, v_{y'}\}$ by Lemma 5. Otherwise, we choose an arbitrary descendant y' of the vertex x in H such that $v_{y'}$ is not a descendant of v_x in G. In this case either v_x is the unique vertex of S in B or $v_y \in V(B)$. The graph G has a v_x, v_y-path P and a $v_x, v_{y'}$-path P' avoiding the vertices of $S \setminus \{v_x, v_y\}$ and $S \setminus \{v_x, v_{y'}\}$, respectively. If P and P' have a common vertex except v_x, then v_y and $v_{y'}$ are adjacent in the graph obtained from G by the elimination of X. This yields a contradiction, since y is the parent of x in the tree H, and y' is a descendant of x. Hence, v_x is the only common vertex of P and P'. Let u and u' be the vertices adjacent to v_x in P and P' respectively. Since v_x is a descendant of v_y, $u \in V(B)$, and $u' \in V(B)$ because $v_{y'}$ is not a descendant of v_x. The 2-connected graph B has a u, u'-path Q that avoids v_x. Notice that if Q contains a vertex of S, then it is either v_y or $v_{y'}$. Consider the $v_y, v_{y'}$-walk obtained by the concatenation of the v_y, u-subpath of P, Q, and the $u', v_{y'}$-subpath of P'. This walk avoids the vertices of $S \setminus \{v_y, v_{y'}\}$. It means that v_y and $v_{y'}$ are adjacent in the graph obtained from G by the elimination of X; a contradiction.

Now we prove the second claim. Let $y_1, \ldots, y_l$ be the children of an inner vertex x of H. To obtain a contradiction, suppose that there is a bicomp B that is a child of v_x in the bicomp-tree for which, contrary to our claim, there are two different children y_i, y_j of x such that $v_{y_i}, v_{y_j} \in G_{v_x B}$. By Lemma 5, $v_{y_s} \in B$ for at most one child y_s of x. If such a child exists, then we assume that $i = s$. Since x is adjacent to y_i, y_j in H, G has a v_x, v_{y_i}-path P_i and a v_x, v_{y_j}-path P_j that avoids the vertices of $S \setminus \{v_x, v_{y_i}\}$ and $S \setminus \{v_x, v_{y_j}\}$, respectively. If P_i and P_j have a common vertex except v_x, then v_{y_i} and v_{y_j} are adjacent in the graph obtained from G by the elimination of X; a contradiction. Hence, v_x is the only common vertex of P and P'. Let u_i and u_j be the vertices adjacent to v_x in P_i and P_j respectively. Clearly, $u_i, u_j \in V(B)$. The 2-connected graph B has u_i, u_j-path Q that avoids v_x. Notice that if Q contains a vertex of S, then it is the vertex v_{y_i}. Consider the v_{y_i}, v_{y_j}-walk obtained by the concatenation of the v_{y_i}, u_i-subpath of P_i, Q, and the u_j, v_{y_j}-subpath of P_j. This walk avoids the vertices of $S \setminus \{v_{y_i}, v_{y_j}\}$. It means that v_{y_i} and v_{y_j} are adjacent in the graph obtained from G by the elimination of X. However, y_i and y_j are children of x in H, and are therefore not adjacent. This contradiction completes the proof of Lemma 9. □

We need one more lemma in the correctness proof of our dynamic programming algorithm in the proof of Theorem 5. Recall that H is an elimination of G with witness $S = \{v_x \mid x \in V(H)\}$, and that we have rooted H at vertex z and the bicomp-tree $\mathcal{T}_G$ at vertex $r = v_z$. Let H_x be a subtree of H for some $x \in V(H)$. Informally speaking, the following lemma shows that H_x is an elimination not only of the subgraph G_{v_x}, i.e., of the subgraph of G corresponding to the subtree of $\mathcal{T}_G$ rooted at v_x, but also of any subgraph of G corresponding to a subtree of $\mathcal{T}_G$ rooted at a vertex $c \in C(G)$ appearing "higher" in $\mathcal{T}_G$ than v_x, i.e., a vertex $c \in C(G)$ such that v_x is a descendant of c in $\mathcal{T}_G$.

Lemma 10. *For any inner vertex $x \in V(H)$ and any cut-vertex $c \in V(G)$ such that v_x is a descendant of c, H_x is an elimination of G_c with witness $S_x = \{v_y \mid y \in V(H_x)\}$, and there is a c, v_x-path in G_c that avoids $S_x \setminus \{v_x\}$. Moreover, for any inner vertex $x \in V(H)$ with a child y, any cut-vertex $c \in V(G)$ such that v_x is a descendant of c, and any bicomp B such that B is a child of c in $\mathcal{T}_G$ and $v_y \in V(G_{cB})$, H_{xy} is an elimination of G_{cB} with witness $S_{xy} = \{v_y \mid y \in V(H_{xy})\}$, and there is a c, v_x-path in G_{cB} that avoids $S_{xy} \setminus \{v_x\}$.*

Proof. First observe that, by Lemma 9, for any $p \in V(H) \setminus \{x\}$, p is a descendant of x in H if and only if v_p is a descendant of v_x in G. Hence, to prove the lemma it is sufficient to observe that for any two vertices v_{p_1}, v_{p_2} in G_{v_x}, there is a v_{p_1}, v_{p_2}-path in G that avoids the vertices of $S \setminus \{v_{p_1}, v_{p_2}\}$ if and only if there is a v_{p_1}, v_{p_2}-path in G_{v_x} that avoids the vertices of $S \cap V(G_{v_x}) \setminus \{v_{p_1}, v_{p_2}\}$, because v_x is a cut-vertex in G. Moreover, since v_x is a cut-vertex and G_c is connected, G_c has a c, v_x-path that avoids $\{v_y \mid y \in V(H_x) \setminus \{x\}\}$. The second claim is proved by the same arguments. □

We now present a polynomial-time algorithm for deciding whether a given tree H is an elimination of a given graph G.

Theorem 5. Elimination *can be solved in time $O(n^{9/2})$ when H is a tree.*

Proof. Let G and H be an instance of Elimination where H is a tree. Since H is a connected graph, the graph G can be eliminated to H if and only if at least one connected component of G can be eliminated to this graph. Hence, we assume without loss of generality that G is connected. Since any graph with at least one vertex can be eliminated to K_1, and K_2 is an elimination of any graph that contains at least one edge, we may also assume that H has at least three vertices. Since we made exactly the same assumptions at the start of Subsection 4.1, we may now apply all the lemmas that were proved in the subsection.

Clearly, if $|V(H)| > n$, then we have a no-instance of the problem. Hence, we assume that $|V(H)| \leq n$. For the tree H, we choose an arbitrary inner vertex z and make it the root of H. For the graph G, we find the set of cut-vertices $C(G)$ and the set of bicomps $\mathcal{B}$, and construct the bicomp-tree $\mathcal{T}_G$. We then construct a set $U \subseteq V(G)$ as follows: for each bicomp B that is a leaf of $\mathcal{T}_G$, we choose an arbitrary vertex $u \in V(B) \setminus C(G)$ and include it in U. As we already pointed out just below Lemma 8, it follows immediately from Lemmas 6, 7 and 8 that H is an elimination of G if only if G can be eliminated to H with a witness $S \subseteq C(G) \cup U$.

Suppose H is an elimination of G with a witness $S = \{v_x \mid x \in V(H)\}$. Since we chose z to be an inner vertex of H, the vertex v_z is a cut-vertex of G due to Lemma 6. Hence, by Lemma 9, there is a cut-vertex r in G such that if y is a descendant of x in H rooted at z, then v_y is a descendant of v_x in G when we root $\mathcal{T}_G$ at r. We check all cut-vertices $r \in C(G)$, and for each r, we root $\mathcal{T}_G$ at r and try to find a witness $S \subseteq C(G) \cup U$ that satisfies this condition using the dynamic programming approach described below. Due to the above arguments, H is an elimination of G if and only if we find such a witness for some r, and we have a no-instance of Elimination otherwise.

For every possible choice of the root r of $\mathcal{T}_G$, i.e., for every cut-vertex $r \in C(G)$, we perform the following dynamic programming algorithm. For each vertex $u \in C(G) \cup U$, the algorithm will create a set R_u which, informally speaking, consists of those vertices x of H for which the subtree H_x is an elimination of the subgraph of G corresponding to the subtree of $\mathcal{T}_G$ rooted at u. Formally, for every $u \in C(G) \cup U$, the algorithm constructs a set $R_u \subseteq V(H)$ such that:

- for any $u \in U$, $R_u = L(H)$;
- for any $u \in C(G)$, R_u is the set of all vertices x of H such that H_x is an elimination of G_u with witness $\{v_y \mid y \in V(H)\}$ such that for any $y, y' \in V(H_x)$, if y' is a descendant of y in H_x, then $v_{y'}$ is a descendant of v_y in G_r.

The algorithm returns "yes" if the set R_r contains z, and proceeds to the next choice of r otherwise. If the algorithm has not returned "yes" after all the vertices $r \in C(G)$ have been considered, then it returns "no".

The sets R_u are constructed in a bottom-up manner, i.e., for every $u \in C(G) \cup U$, the set R_u is constructed only when the set R_w has been constructed for every $w \in C(G) \cup U$ that is a descendant of u. If $u \in U$, then there is no $w \in C(G) \cup U$ that is a descendant of u, and $R_u = L(H)$ by definition. Suppose $u \in C(G)$. Denote by $B_1, \ldots, B_k$ the bicomps of G that are children of the cut-vertex u in the bicomp-tree $\mathcal{T}_G$. Let $D_u = ((C(G) \cup U) \setminus \{u\}) \cap \bigcup_{i=1}^{k} V(B_i)$, i.e., D_u is the set of all vertices $w \in C(G) \cup U$ other than u that are descendants of u and are contained in some bicomp together with u. From the sets R_w for all $w \in D_u$, the set R_u is created in two steps as follows:

Step 1. For every $w \in D_u$, all the vertices of R_w are included in R_u.

Step 2. For every $i \in \{1, \ldots, k\}$, let $T_i = \cup_{w \in D_u \cap V(B_i)} R_w$. A vertex $x \in V(H)$ with children $y_1, \ldots, y_l$ is included in R_u if there is a set $\{i_1, \ldots, i_l\} \subseteq \{1, \ldots, k\}$ such that $y_j \in T_{i_j}$ for every $j \in \{1, \ldots, l\}$. Checking if such a set exists is equivalent to solving a matching problem on an auxiliary bipartite graph, for which we can use the well-known Hopcroft-Karp algorithm [12].

Let us prove the correctness of the algorithm. We observe that by Lemma 6, if H is an elimination of G with witness $\{v_y \mid y \in V(H)\}$, then $v_x \in U$ only if x is a leaf of H. Recall that any graph can be eliminated to an isolated vertex. Hence, each set R_u includes all the leaves of H, and $R_u = L(H)$ for any $u \in U$. For any cut-vertex u in G, u is either saved or not; recall that a vertex is saved if and only if it is not eliminated. Step 1 is applied to construct all elements of R_u that correspond to the partial solutions where u is not saved. The correctness of Step 1 follows from Lemma 10. We use Step 2 to find all elements of R_u that correspond to the partial solutions where u is a saved vertex. The correctness of this step follows from Lemmas 9 and 10.

Finally, we analyze the running time. It is well-known that for a connected graph G, the set of cut-vertices $C(G)$ and the set of bicomps $\mathcal{B}$ can be found and the bicomp-tree can be constructed in linear time. There are at most n cut-vertices that can be chosen as the root of G. For each choice, we run our dynamic programming algorithm. The initial assignment of R_u for each $u \in U$ can be done in $O(n)$ time, and we have at most n vertices in U. For each $u \in C(G)$, Step 1 can be done in $O(n \cdot |D_u|)$ time. For Step 2, for each $x \in V(H)$ we use the Hopcroft-Karp algorithm [12] to check existence of a system of distinct representatives $y_1, \ldots, y_l$ from the sets $T_1, \ldots, T_k$. Observe that we can omit running the algorithm if $l > k$; thus we can assume that $l \leq k \leq |D_u|$ and a single test runs in $O(|D_u|^{5/2})$ time. Hence, for vertex u, Step 2 can be done in $O(n \cdot |D_u|^{5/2})$ time. In total, performing Step 1 and Step 2 takes $O(n \cdot |D_u|^{5/2})$ time for each $u \in C(G)$. Observe that each vertex $u' \in C(G) \cup U$ appears in the set D_u for at most one u, so $\sum_{u \in C(G)} |D_u| \leq n$. As function $f(t) = t^{5/2}$ is increasing and convex, we

infer that $\sum_{u \in C(G)} |D_u|^{5/2} \leq n^{5/2}$. Therefore, the whole dynamic programming routine runs in $O(n^{7/2})$ time. Since we run the dynamic programming algorithm for $O(n)$ choices of the root r, it follows that the total running time is $O(n^{9/2})$. □

4.2 Eliminating from a Tree is NP-complete

In this subsection, we consider the case when G is a tree and H is an arbitrary graph. First, we make the following observation. A connected graph is called a *block graph* if each of its bicomps is a complete graph. Observe that if G is a block graph, then elimination of any vertex v results in another block graph, because this operation unites all maximal cliques that contain v into a single clique and then removes v. Since trees are block graphs, it gives us the following proposition.

Proposition 1. *If H is an elimination of a tree G, then H is a block graph.*

Despite the fact that graphs that are eliminations of trees have relatively simple structure, it turns out that Elimination remains NP-complete when G is assumed to be a tree. In order to prove this result (Theorem 6 below), we first prove three auxiliary results.

For a graph G, the *distance* $\mathrm{dist}_G(u, v)$ between a pair of vertices u and v of G is the number of edges on a shortest path between them. The *diameter* of G is defined as $\mathrm{diam}(G) = \max\{\mathrm{dist}_G(u, v) \mid u, v \in V(G)\}$. Our first auxiliary result follows directly from Observation 1.

Lemma 11. *Let H be a graph that is obtained by the elimination of a set of vertices X of a graph G, and let $S = V(G) \setminus X$. Then for any $u, v \in S$, $\mathrm{dist}_H(u, v) \leq \mathrm{dist}_G(u, v)$ and hence $\mathrm{diam}(H) \leq \mathrm{diam}(G)$.*

Let G be a tree, let $k \geq 2$ be an integer, and let $Q = \{v_1, \ldots, v_k\}$ be a subset of the vertices of G such that no v_i is an inner vertex of a path between any two vertices in Q. Then G contains a (unique) maximal subtree, denoted by T_Q, that contains all the vertices of Q as leaves. We point out that although the set of leaves of T_Q contains the set Q by definition, T_Q might have other leaves as well. To see this, note that if we take G to be a star with $s \geq 3$ leaves and Q to be any subset of exactly $s - 1$ leaves, then the tree $T_Q = G$ has $s > |Q|$ leaves.

Lemma 12. *Let H be a graph on at least two vertices that is obtained from a tree G by the elimination of a subset X of vertices of G. Let $Q = \{v_1, \ldots, v_k\}$ be a maximal clique of H. Then $k \geq 2$ and no vertex of Q is an inner vertex of a path in G between any two vertices of Q. Moreover, $V(T_Q) \setminus Q \subseteq X$. In particular, Q is obtained by the elimination of $V(T_Q) \setminus Q$.*

Proof. Because H is a graph on at least two vertices that is obtained from a connected graph, namely the tree G, we find that H is connected. Hence, $k \geq 2$. As Q forms a clique in H, for every two vertices $v_i, v_j \in Q$, the unique path between v_i and v_j in G does not contain other vertices from Q. This implies that the subtree T_Q of G, as defined just above Lemma 12, exists. We now prove that $V(T_Q) \setminus Q \subseteq X$.

For contradiction, suppose there is a vertex $u \in V(T_Q) \setminus Q$ such that $u \notin X$. Let T' be the unique subtree of T_Q whose leaves are exactly the vertices of Q. Note that u is not a vertex of T', as otherwise Q would not form a clique in H. Also note that u is not adjacent to any of the vertices in Q. Now let u' be the neighbor of u on the unique path in G from u to a vertex of T'. We may assume that $u' \in X$, as otherwise we can choose u' instead of u. We first eliminate all vertices of T' not equal to u'. Afterward, Q is a clique and u' is adjacent to all vertices of Q. Then eliminating u' makes u adjacent to all vertices of Q. Because $u \notin X$, this implies that H contains a clique $Q' = Q \cup \{u\}$. This contradicts the maximality of Q. Hence $V(T_Q) \setminus Q \subseteq X$, and consequently Q is obtained by the elimination of $V(T_Q) \setminus Q$. This completes the proof of Lemma 12. □

We now state the third auxiliary result that will be used in the proof of Theorem 6 below.

Lemma 13. *Let H be a graph on at least two vertices that is obtained from a tree G by the elimination of a subset X of vertices of G. For a vertex $v \notin X$, let $\{Q_1, \ldots, Q_r\}$ be the set maximal cliques of H that contain v. Then $d_G(v) \geq r$.*

Proof. Because H is a graph on at least two vertices that is obtained from a connected graph, namely the tree G, we find that H is connected. Hence, each Q_i contains at least two vertices. By Lemma 12, the trees T_{Q_i} exist, and the elimination of T_{Q_i} yields Q_i for $i = 1, \ldots, r$. This means that the sets $V(T_{Q_i}) \setminus Q_i$ are mutually disjoint. If $V(T_{Q_i}) \setminus Q_i$ is empty, then Q_i is an edge incident with v and some other vertex v_i, which is not contained in some Q_h with $h \neq i$, because the cliques $Q_1, \ldots, Q_r$ are maximal. We conclude that v must have at least r neighbors. □

We are now ready to present the main result of this subsection.

Theorem 6. Elimination *is* NP*-complete, even if G is restricted to be a tree.*

Proof. We reduce from Exact 3-Cover, which is an NP-complete problem (cf. [7]). It has as input a finite set X with $3n$ elements and a collection $\mathcal{C}$ of subsets of X, each of which contains exactly three elements, and the goal is to test whether $\mathcal{C}$ contains an *exact cover* of X, i.e., a subcollection $\mathcal{C}' \subseteq \mathcal{C}$ such that each element of X occurs in exactly one subset in $\mathcal{C}'$. Clearly, $|\mathcal{C}'| = n$ (if it exists).

Let $X = \{x_1, \ldots, x_k\}$, where $k = 3n$, and $\mathcal{C} = \{C_1, \ldots, C_m\}$ be an instance of Exact 3-Cover. We assume that $m > n \geq 2$.

First, we construct an auxiliary gadget $F_i(u, v)$ for $i \in \{1, \ldots, k\}$ as follows (see Figure 5 for an illustration):

- take two vertices u, v;
- join u and v by a path $v_0 \cdots v_{k+3}$ of length $k + 3$, where $u = v_0$ and $v = v_{k+3}$;
- introduce a pendant vertex x and make it adjacent to v_i.

We say that x is the *index* vertex of $F_i(u, v)$.

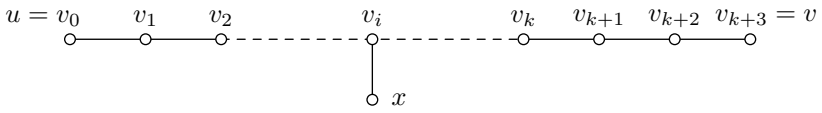

Fig. 5. The graph $F_i(u, v)$ with index vertex x.

Now, we construct a tree G as follows (see Figure 6):

- for all $j \in \{1, \ldots, m\}$, if $C_j = \{x_p, x_q, x_s\}$ then construct copies of $F_p(u_j^{(1)}, v_j^{(1)})$, $F_q(u_j^{(2)}, v_j^{(2)})$, $F_s(u_j^{(3)}, v_j^{(3)})$ that we denote by $T_j^{(p)}, T_j^{(q)}, T_j^{(s)}$ respectively, and introduce a vertex w_j and make w_j adjacent to $u_j^{(1)}, u_j^{(2)}, u_j^{(3)}$;
- introduce a vertex r and make it adjacent to $w_1, \ldots, w_m$;
- introduce a vertex r' and join it with r by a path P of length $k + 6$.

Finally, we construct a graph H as follows (see Figure 6):

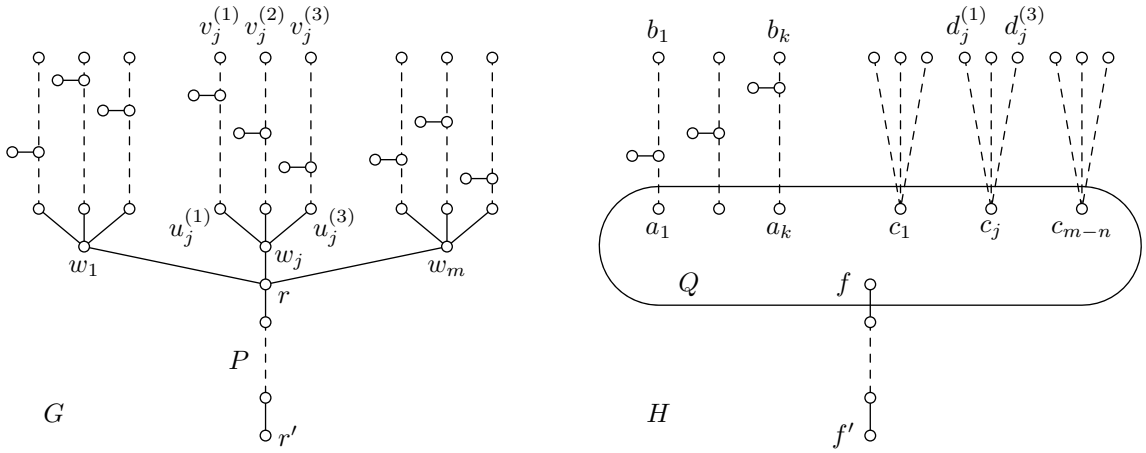

Fig. 6. The graphs G and H.

- for each $i \in \{1, \ldots, k\}$, construct a copy of $F_i(a_i, b_i)$;
- for each $j \in \{1, \ldots, m-n\}$, introduce a vertex c_j as well as three vertices $d_j^{(1)}, d_j^{(2)}, d_j^{(3)}$, and join them with c_j by paths of length $k+4$;
- introduce two vertices f, f' and join them by a path of length $k+5$;
- join the vertices $a_1, \ldots, a_k$, $c_1, \ldots, c_{m-n}$, f by edges to form a clique; denote this clique by Q.

We claim that $\mathcal{C}$ contains an exact cover $\mathcal{C}'$ of X if and only if H is an elimination of G. First suppose that $\mathcal{C}' = \{C_{j_1}, \ldots, C_{j_n}\}$ is exact cover of X. We eliminate the vertex r and the vertices $w_{j_1}, \ldots, w_{j_n}$. Then for $j \in \{1, \ldots, m\} \setminus \{j_1, \ldots, j_n\}$, we eliminate the index vertices of $T_j^{(p)}, T_j^{(q)}, T_j^{(s)}$. It is straightforward to check that the obtained graph is isomorphic to H. Hence, H is an elimination of G.

Now suppose that H is an elimination of G. Denote by X the set of eliminated vertices, and let $S = V(G) \setminus X$. Let also h be an isomorphism between H and the graph obtained from G by the elimination of X.

Any subtree of G that does not contain r as an inner vertex has at most 7 leaves. The size of the clique Q is $k + (m-n) + 1 > 7$, because $m > n \geq 2$. Hence, by Lemma 12, $r \in X$. Observe that the graph G' obtained from G by the elimination of r has diameter $2k + 10 = \mathrm{diam}(H)$. By Lemma 11, $V(P) \setminus \{r\} \subseteq S$, since the elimination of any vertex of $V(P) \setminus \{r\}$ results in a graph of diameter less that $2k + 10$. The vertex r' is the unique vertex of G' that has at least two vertices at distance $2k + 10$, and f' is the unique vertex of H with this property. By Lemma 11, we conclude that h maps f' to r'.

For $j \in \{1, \ldots, m\}$, consider the unique shortest $r', v_j^{(1)}$, $r', v_j^{(2)}$ and $r', v_j^{(3)}$-paths in G'. Observe that they have length $2k + 10$, and that the set of the last two vertices of such paths is the set of all vertices that are at distance at least $2k + 9$ from r' in G'. Also note that we have $3m$ such paths in total. Consider now the unique shortest f', b_i-paths and $f', d_j^{(1)}$, $f', d_j^{(2)}$, $f', d_j^{(3)}$-paths in H for $i \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, m-n\}$. They have length at least $2k + 9$, and the union of the sets of the last vertices of the f', b_i-paths and the set of the last two vertices of the $f', d_j^{(1)}$, $f', d_j^{(2)}$, $f', d_j^{(3)}$-paths is the set of vertices at distance at least $2k + 9$ from f' in H. The total number of the paths is $k + 3(m-n) = 3m$. Hence, for each

$j \in \{1,\ldots,m\}$, at most one vertex of each shortest $r', v_j^{(1)}$-path, $r', v_j^{(2)}$-path and $r', v_j^{(3)}$-path in G' is included in X, due to Lemma 11.

Only $w_1,\ldots,w_m$ and r have degrees at least four in G, and we already proved that $r \in X$. For each $j \in \{1,\ldots,n-m\}$, the vertex c_j is included in four maximal cliques of H. By Lemma 13, we then find that the isomorphism h maps the vertices $c_1,\ldots,c_{m-n}$ to the vertices from the set $\{w_1,\ldots,w_m\}$. Hence, at least $m-n$ vertices from $\{w_1,\ldots,w_m\}$ are in S.

Let K be the set of vertices in G that are mapped to the vertices of Q by h. By Lemma 12, the tree T_K exists in G, and K is obtained by the elimination of $V(T_K) \setminus K$. By the definition of T_K, we find that T_K has at least $|K| = |Q| = k+m-n+1 = 2n+m+1$ leaves. We will use this lower bound on the number of leaves of T_K in our reasoning below.

Because for each $j \in \{1,\ldots,m\}$, at most one vertex of each shortest $r', v_j^{(1)}$-path, $r', v_j^{(2)}$-path and $r', v_j^{(3)}$-path in G' is included in X and w_j belongs to each of these three paths, we deduce that if $w_j \in X$, then $u_j^{(1)}, u_j^{(2)}, u_j^{(3)} \in S$. Hence, when we denote the number of vertices of $\{w_1,\ldots,w_m\}$ in X by ℓ, we find that T_K has $3\ell+(m-\ell)+1$ leaves. We already deduced that T_K has at least $2n+m+1$ leaves. This means that we have found that $2\ell+m+1 \geq 2n+m+1$, which is equivalent to $\ell \geq n$. Recall that at least $m-n$ vertices from $\{w_1,\ldots,w_m\}$ are in S, which means that $\ell \leq n$. Hence, $\ell = n$. Then exactly n vertices of $\{w_1,\ldots,w_m\}$ are included in X, and consequently, exactly $m-n$ vertices of this set are in S. Let $w_{j_1},\ldots,w_{j_n}$ be the n vertices from $\{w_1,\ldots,w_m\}$ that are in X.

We let G'' be the graph obtained from G by the elimination of r and $w_{j_1},\ldots,w_{j_n}$. We note that G'' has $3(m-n)$ vertices at distance $2k+10$ from r', and these are the vertices $v_j^{(1)}, v_j^{(2)}, v_j^{(3)}$ for $j \in \{1,\ldots,m\} \setminus \{j_1,\ldots,j_n\}$. Then h maps $d_i^{(1)}, d_i^{(2)}, d_i^{(3)}$ for $i \in \{1,\ldots,m-n\}$ to $v_j^{(1)}, v_j^{(2)}, v_j^{(3)}$ for $j \in \{1,\ldots,m\} \setminus \{j_1,\ldots,j_n\}$ due to Lemma 11. Therefore, the index vertices of the gadgets $T_j^{(p)}, T_j^{(q)}, T_j^{(s)}$ for $j \in \{1,\ldots,m\} \setminus \{j_1,\ldots,j_n\}$ are in X.

We have now that X contains the vertex r, exactly n vertices $w_{j_1},\ldots,w_{j_n}$ from the set $\{w_1,\ldots,w_m\}$, as well as the index vertices of the gadgets $T_j^{(p)}, T_j^{(q)}, T_j^{(s)}$ for every $j \in \{1,\ldots,m\} \setminus \{j_1,\ldots,j_n\}$. Because the graph obtained after eliminating X contains the same number of vertices as H, we find that X contains no other vertices. We set $C' = \{C_{j_1},\ldots,C_{j_m}\}$. Because H and the graph obtained from G by the elimination of X are isomorphic, for each $i \in \{1,\ldots,k\}$, the isomorphism h maps T_i to exactly one gadget $T_j^{(p)}, T_j^{(q)}, T_j^{(s)}$ for some $j \in \{j_1,\ldots,j_n\}$. This means that C' is an exact cover for X, and we have completed the proof. □

Acknowledgements. We would like to thank Łukasz Kowalik for an inspiring discussion on the theorem of Kleitman and West. We also thank the two anonymous referees for their useful comments and suggestions that helped us to improve the presentation of our paper.

References

1. van Bevern, R., Komusiewicz, C., Moser, H., Niedermeier, R.: Measuring indifference: Unit interval vertex deletion. In: Thilikos, D.M. (ed.) WG 2010. LNCS, vol. 6410, pp. 232–243 (2010)
2. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. J. Comput. Syst. Sci. 75(8), 423–434 (2009)
3. Cai, L.: Fixed-parameter tractability of graph modification problems for hereditary properties. Inform. Process. Lett. 58(4), 171–176 (1996)
4. Courcelle, B.: The monadic second-order logic of graphs III: Tree-decompositions, minor and complexity issues. ITA 26, 257–286 (1992)

5. Cygan, M., Philip, G., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J.O.: Dominating set is fixed parameter tractable in claw-free graphs. Theor. Comput. Sci. 412, 6982–7000 (2011)
6. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer-Verlag (1999)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1990)
8. George, J.A., Liu, J.W.H.: Computer Solution of Large Sparse Positive Definite Systems. Prentice-Hall Inc. (1981)
9. Golovach, P.A., Heggernes, P., van 't Hof, P., Manne, F., Paulusma, D., Pilipczuk, M.: How to eliminate a graph. In: Golumbic, M.C., Stern, M. (eds.) WG 2012. LNCS, vol. 7551, pp. 320–331. Springer (2012)
10. Heggernes, P.: Minimal triangulations of graphs: A survey. Discrete Mathematics 306, 297–317 (2006)
11. Heggernes, P., van 't Hof, P., Jansen, B.M.P., Kratsch, S., Villanger, Y.: Parameterized complexity of vertex deletion into perfect graph classes. In: Owe, O., Steffen, M., Telle, J.A. (eds.) FCT 2011. LNCS, vol. 6914, pp. 240–251. Springer (2011)
12. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Comput. 2, 225–231 (1973)
13. Kawarabayashi, K., Reed, B.A.: An (almost) linear time algorithm for odd cycles transversal. In: Charikar, M. (ed.) SODA 2010. pp. 365–378. SIAM (2010)
14. Kleitman, D.J., West, D.B.: Spanning trees with many leaves. SIAM J. Discrete Math. 4, 99–106 (1991)
15. Marx, D.: Chordal deletion is fixed-parameter tractable. Algorithmica 57(4), 747–768 (2010)
16. Marx, D., Schlotter, I.: Obtaining a planar graph by vertex deletion. Algorithmica 62(3-4), 807–822 (2012)
17. Natanzon, A., Shamir, R., Sharan, R.: Complexity classification of some edge modification problems. Discrete Appl. Math. 113, 109–128 (2001)
18. Parter, S.: The use of linear graphs in Gauss elimination. SIAM Review 3, 119–130 (1961)
19. Philip, G., Raman, V., Villanger, Y.: A quartic kernel for pathwidth-one vertex deletion. In: Thilikos, D.M. (ed.) WG 2010. Lecture Notes in Computer Science, vol. 6410, pp. 196–207 (2010)
20. Robertson, N., Seymour, P.D., Thomas, R.: Quickly excluding a planar graph. J. Comb. Theory B 62, 323–348 (1994)
21. Rose, D.J., Tarjan, R.E., Lueker, G.S.: Algorithmic aspects of vertex elimination on graphs. SIAM J. Comput. 5, 266–283 (1976)
22. Samdal, E.: Minimum Fill-in Five Point Finite Element Graphs. Master's thesis, Department of Informatics, University of Bergen, Norway (2003)
23. Tarjan, R.E., Yannakakis, M.: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. SIAM Journal on Computing 13(3): 566–579 (1984)
24. Yannakakis, M.: Edge-deletion problems. SIAM J. Comput. 10, 297–309 (1981)