How to Eliminate a Graph*

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

[1] School of Engineering and Computing Sciences, Durham University, UK, {petr.golovach,daniel.paulusma}@durham.ac.uk

[2] Department of Informatics, University of Bergen, Norway, {pinar.heggernes, pim.vanthof, fredrik.manne, michal.pilipczuk}@ii.uib.no

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 study the parameterized complexity of the Elimination problem. 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 surprisingly 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 u and v. Thus we can view the described problem as a relaxation of the well-known Clique problem.

The above problem can be stated in terms of a well-known graph operation related to Gaussian elimination: vertex elimination [17]. 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

* This work is supported by EPSRC (EP/G043434/1) and Royal Society (JP100692), and by the Research Council of Norway (197548/F20).

clique. In fact, we state a more general problem: the ELIMINATION problem takes as input two graphs G 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 [17]. The resulting *Elimination Game* [17] repeatedly chooses a vertex and eliminates it from the graph until the graph becomes empty. The amount of edges added during the 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; see e.g., [8, 9, 17, 20]. 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 Fixed-Parameter Tractability; see e.g., [1, 3, 6, 10, 12, 14–16, 18, 22]. Given the wide use of the vertex elimination operation, we find it surprising that the ELIMINATION problem does not seem to have been studied before. The only related study we are aware of is by Samdal [21], 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 has a kernel with at most $5|V(H)|$ vertices on connected graphs, which contrasts the hardness of the CLIQUE problem. We also study the cases when one of the input graphs is a tree. Surprisingly, 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 on the maximum number of leaves in a spanning tree of a graph, similar to a proof by Kleitman and West [13]. We find this a contribution of independent interest.

Notation. All graphs in this paper are undirected, finite, and simple. Our notation mostly follows the notation used by Rose et al. [20] and Heggernes [9]. Let $G = (V, E)$ be a graph. We sometimes use $V(G)$ and $E(G)$ to denote V and E, 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 vertex $v \in V(G)$, $G - v$ is the graph $G[V \setminus \{v\}]$.

A *clique* is a set of vertices that are all 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 the partition of V into two independent sets, then we denote it as $G = (A, B, E)$ and we call (A, B) the *bipartition* of G. A graph is a *split graph* it its vertex set can be partitioned into a clique and an independent set. A vertex is a *cut-vertex* if the removal of the vertex leaves the graph with more connected components than before.

In this extended abstract, proofs of some theorems and lemmas, which are marked with the symbol ♠, have been placed in an appendix.

2 Preliminaries and Hardness of Elimination

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 ([20]) *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 FPT 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.*

Since Elimination is unlikely to be FPT in general, 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 tractibility 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.*

We point out that the reductions presented 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 an integer k, and asks whether the complete graph K_k is an elimination of G. Since Clique Elimination is $W[2]$-hard when parameterized by $|V(G)|-k$ due to Theorem 2, we choose k as the parameter throughout this section.

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 [19], Courcelle's Theorem [4] implies that Clique Elimination is FPT 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 *admits a kernel with at most* $5k$ *vertices for connected graphs.*

We would like to remark that the assumption that the input graph is connected is probably necessary, as Clique Elimination in general graphs admits a simple composition algorithm that takes disjoint union of instances, so 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, an algorithm with single-exponential dependence on k can be obtained by kernelizing every connected component of the input graph separately, and then running a brute-force search on each kernel. This gives us a better running time than the aforementioned combination of meta-theorems.

Corollary 2. Clique Elimination *can be solved in* $\binom{5k}{k} n^{O(1)} \leq 12.21^k\, n^{O(1)}$ *time and polynomial space.*

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 [13] 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. Similar to the proof of Kleitman and West [13], we then show 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 all internal vertices of which 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 equivalent instance (G',k'). Each time, we apply the rule with the smallest number among the applicable ones. We argue that if none of the rules is applicable, then the modified graph has no vertices of degree 1 and no edge between any two vertices of degree 2. Recognizing whether a rule can be applied, as well as the application of the rule itself, will be trivially polynomial-time operations. The total number of applications will be bounded by a polynomial in the input size.

Reduction Rule 1 *If $k \leq 3$ or $n \leq 3$, then resolve the instance in polynomial time via a brute-force algorithm, and output a trivial* `yes-` *or* `no`*-instance, depending on the result.*

The safeness of this rule is obvious. The lemmas that prove that the following three rules are safe are given in Appendix B.

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

Reduction Rule 3 *If 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).*

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

If, after applying our four reduction rules exhaustively, we have not yet solved the problem, then we have obtained a graph G^* with 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 instance 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 the next result, which is our modified version of the aforementioned result by Kleitman and West [13]. This concludes the proof of Theorem 3.

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.*

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 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 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, let us 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 have grown T into a spanning tree using the rules; 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 this step is given in Appendix B.

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. We would like to note that the first three rules were already used in the original proof of Kleitman and West.

Growing Rule 1 If some leaf of T has at least two neighbors from $G \setminus T$, expand it. 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 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 of T, expand this leaf (which results in adding only v to T, as Rule 1 was not applicable) and then 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 of T, expand this leaf (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, all the neighbors of v' were not 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 at least $d-2$. Hence, the increase of the potential is at least $4(d-2)-(d+1) = 3d-9 \geq 0$, as $d \geq 3$.

It remains to argue that 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. In both cases, a new leaf of T is created that was not counted when we determined a lower bound on the increase of the potential, even though this leaf might be dead after the application of the rule. If this newly added leaf is in fact dead, then the potential increases by at least 1. Since we are sure that this leaf will be dead after the last rule application, the potential will increase by at least 1 during the whole process. 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. □

The bound $|V(G)|/5+2$ is best possible. A family of examples with tight inequality can be obtained by connecting a number of diamonds in the way as shown in Figure 1.

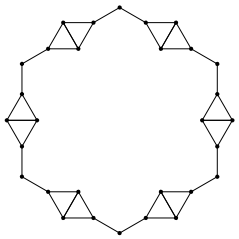

Fig. 1: A graph on 30 vertices for which the maximum possible number of leaves in a spanning tree is exactly 8. This example shows that the bound in Theorem 4 is tight.

4 Elimination on Trees

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

For a tree H with at least two vertices, we denote by $L(H)$ the set of leaves of H. The remaining set of vertices is denoted by $I(H) = V(H) \setminus L(H)$ and called the *inner vertices*. 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.

Let G and H be an instance of ELIMINATION where H is a tree. Since a graph G can be eliminated to a connected graph H if and only if at least one connected component of G can be eliminated to H, we assume without loss of generality that G is connected. Also it is easy to see that any graph G with at least one vertex can be eliminated to K_1, and K_2 is an elimination of a graph G whenever G has at least one edge. Hence, we can assume that H has at least three vertices. Therefore, $L(H) \neq \emptyset$ and $I(H) \neq \emptyset$.

Suppose that H is an elimination of G. Let $S = \{v_x \mid x \in V(H)\}$ be the 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 2 (♠). *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 $xy \in E(H)$.*

Lemma 3 (♠). *For any $x \in I(H)$, $v_x \in C(G)$.*

Now 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 3, 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 x and y in H, such that neither is a descendant of the other, and let p be their lowest common ancestor. A crucial observation in our algorithm is that v_x and v_y are descendants of v_p in G, but they do not appear in the same subgraph G_{v_pB} for some bicomp B that is a child of v_p. The following lemma formalizes this idea.

Lemma 4 (♠). *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 $v_y \in G_{v_xB_i}$.*

We are now ready to describe our algorithm in the proof of the following theorem.

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. Clearly, if $|V(H)| > n$, then we have a `no`-instance of the problem. Hence, we assume that $|V(H)| \leq n$. Recall that it is sufficient to solve the problem for connected graphs G and trees H with at least three vertices. 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 $\mathcal{T}_G$. Then we 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. It can be shown that H is an elimination of G if only if G can be eliminated to H with a witness $S \subseteq C(G) \cup U$. A formal proof of this statement requires some additional lemmas, and can be found in Appendix C.

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 3. Hence, by Lemma 4 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 rooted at r. We check all cut-vertices $r \in C(G)$, and for each r, we root G at r and try to find a witness that satisfies this condition. Clearly, 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.

From now on, we assume that the root vertex r of G is fixed, and we construct a dynamic programming algorithm. For each vertex $u \in C(G) \cup U$, the algorithm will create 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 the property 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, where $v_y, v_{y'}$ are the saved vertices in G_u corresponding to y, y'.

The algorithm returns `yes` if R_r contains z, and `no` otherwise.

Notice that the sets for $u \in U$ are already defined. The sets R_u for cut-vertices u are constructed as follows. Denote by $B_1, \ldots, B_k$ the bicomps of G that are children of the cut-vertex u in the bicomp-tree $\mathcal{T}_\mathcal{G}$. Let D_u be 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. In other words, $D_u = (C(G) \cup U \setminus \{u\}) \cap \bigcup_{i=1}^{k} V(B_i)$. Suppose that the sets R_w have already been constructed for all $w \in D_u$. We then create R_u in two steps.

Step 1. All the vertices that are in R_w for some $w \in D_u$ are included in R_u.

Step 2. Let $T_i = \cup_{w \in D_u \cap V(B_i)} R_w$ for $i \in \{1, \ldots, k\}$. 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 $i \in \{1, \ldots, l\}$.

In order to perform Step 2, whose correctness is guaranteed by Lemma 4, we need to solve a matching problem on an auxiliary graph. The full proof of correctness and the running time analysis of our algorithm can be found in Appendix C. □

Finally, 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 intractable when G is assumed to be a tree.

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

Acknowledgements. We would like to thank Łukasz Kowalik for an inspiring discussion on the theorem of Kleitman and West.

References

1. van Bevern, R., Komusiewicz, C., Moser, H., Niedermeier, R.: Measuring indifference: Unit interval vertex deletion. In: WG. pp. 232–243 (2010), LNCS 6410
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. Inf. Process. Lett. 58, 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. Heggernes, P.: Minimal triangulations of graphs: A survey. Discrete Mathematics 306, 297–317 (2006)
10. Heggernes, P., van 't Hof, P., Jansen, B.M.P., Kratsch, S., Villanger, Y.: Parameterized complexity of vertex deletion into perfect graph classes. In: FCT. pp. 240–251 (2011), LNCS 6914
11. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Comput. 2, 225–231 (1973)
12. Kawarabayashi, K., Reed, B.A.: An (almost) linear time algorithm for odd cyles transversal. In: SODA. pp. 365–378. ACM-SIAM (2010)
13. Kleitman, D.J., West, D.B.: Spanning trees with many leaves. SIAM J. Discrete Math. 4, 99–106 (1991)
14. Marx, D.: Chordal deletion is fixed-parameter tractable. In: WG. pp. 37–48 (2006), LNCS 4271
15. Marx, D., Schlotter, I.: Obtaining a planar graph by vertex deletion. In: WG. pp. 292–303 (2007), LNCS 4769
16. Natanzon, A., Shamir, R., Sharan, R.: Complexity classification of some edge modification problems. Discrete Applied Mathematics 113, 109–128 (2001)
17. Parter, S.: The use of linear graphs in Gauss elimination. SIAM Review 3, 119–130 (1961)
18. Philip, G., Raman, V., Villanger, Y.: A quartic kernel for pathwidth-one vertex deletion. In: WG. pp. 196–207 (2010), LNCS 6410
19. Robertson, N., Seymour, P.D., Thomas, R.: Quickly excluding a planar graph. J. Comb. Theory, Ser. B 62, 323–348 (1994)
20. Rose, D.J., Tarjan, R.E., Lueker, G.S.: Algorithmic aspects of vertex elimination on graphs. SIAM J. Comput. 5, 266–283 (1976)
21. Samdal, E.: Minimum Fill-in Five Point Finite Element Graphs. Master's thesis, Department of Informatics, University of Bergen, Norway (2003)
22. Yannakakis, M.: Edge-deletion problems. SIAM J. Comput. 10, 297–309 (1981)

Appendix A: Proofs Omitted from Section 2

This appendix contains the proofs of the two hardness results stated in Section 2.

Theorem 1 (restated). 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. □

Theorem 2 (restated). 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. From an instance (G, k, c) of Colorful Red-Blue Dominating Set, where $G = (R, B, E)$, we create an instance $(G^*, |V(G)| - k - 1)$ of Clique 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 2. We claim that (G, k, c) is a `yes`-instance of Colorful Red-Blue Dominating Set if and only if $(G^*, |V(G)|-k-1)$ is a `yes`-instance of Clique Elimination, i.e., if we can transform G^* into a complete graph by eliminating $k + 1$ vertices.

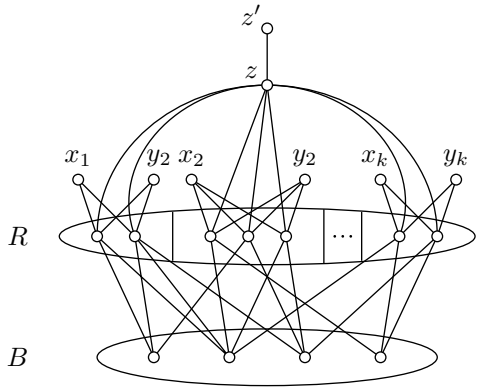

Fig. 2: The graph G^* constructed in the proof of Theorem 2.

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. □

Appendix B: Lemmas and Proofs Omitted from Section 3

In this appendix, we first prove that Reduction Rules 2, 3 and 4, used in our kernelization algorithm, are safe. We then present the part of the proof of Theorem 7 that was omitted from the main body of the paper.

Lemma *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 $v \notin S$. Otherwise, since v is adjacent only to v', $k \leq 2$ and we could have applied Reduction Rule 1. We now set $S' = (S \setminus \{v'\}) \cup \{v\}$ to obtain a witness set of the same cardinality that does not contain v'. □

Lemma *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 $v' \in S$. As Reduction Rule 1 was not applicable, we find that $k > 3$. Then 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'. □

Lemma *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 $v_0 \in S$. As Reduction Rule 1 was not applicable, we find that $k > 3$. Hence, 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. □

Proof of Theorem 4: how to choose a starting tree. Observe that the maximum degree of G is at least 3. The cases are as follows (see Figure 3 for guidance along the proof).

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.

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 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 the theorem holds in this case.

3. Now we can safely assume that G contains two adjacent vertices of degree 3; denote them 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. □

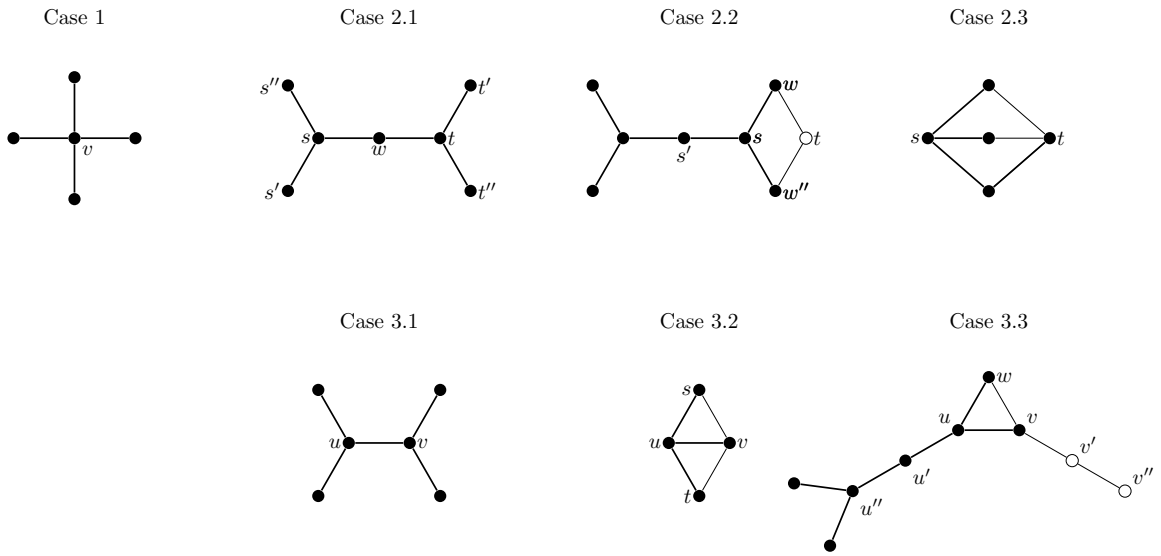

Fig. 3: The trees T chosen to start the growing process, for each of the cases.

Appendix C: Lemmas and Proofs Omitted from Section 4

This appendix contains the proofs of Theorems 5 and 6, as well as some lemmas that are used in the proofs of these theorems. We start by restating Theorem 5.

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

Before we present the proof of correctness and running time analysis of the algorithm given in the main body of the paper, we state and prove several lemmas that are used in the correctness proof.

Let G and H be an instance of Elimination where H is a tree. Recall that we may assume that G is connected, and that H has at least three vertices. Suppose that H is an elimination of G. Let $S = \{v_x \mid x \in V(H)\}$ be witness, where v_x is the vertex of G that corresponds to the vertex x of H. Let $X = V(G) \setminus S$ be the corresponding solution yielding H. Our algorithm is based on several structural results. We first prove Lemmas 2, 3 and 4 that were already stated in the main body of the paper.

Lemma 2 (restated). *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. If at least one internal vertex of one of these paths is not eliminated, we have a cycle in H, but it is impossible. Hence, all internal vertices of these paths are eliminated, and then v_x, 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 3 (restated). *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 2, 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. □

Lemma 4 (restated). *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 2. 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 2, $v_{y_s} \in B$ 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 4. □

We now state and prove some additional structural results.

Lemma 5. *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 3. Then by Lemma 2, 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 3, $v_y \in C(G)$. 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 2, 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 6. *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 2, 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 3, $v_y \in V(B) \cap C(G)$ 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 $i = 1, 2$, there are $v_{z_i} \in S$ such that $v_{z_i} \notin V(G^{c_i})$. Then there are c_i, v_{z_i}-paths P'_i for $i = 1, 2$. 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 $i = 1, 2$. 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 5, we can replace v_x by c in S. □

Lemma 7. *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 the set of eliminated vertices $V(G_c) \setminus$*

$\{v_y \mid y \in V(H_x)\}$, and there is a c, v_x-path in G_c that avoids $\{v_y \mid y \in V(H_x) \setminus \{x\}\}$. 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 the bicomp-tree and $v_y \in V(G_{cB})$, H_{xy} is an elimination of G_{cB} with the set of eliminated vertices $V(G_{cB}) \setminus \{v_s \mid s \in V(H_{xy})\}$, and there is a c, v_x-path in G_{cB} that avoids $\{v_s \mid s \in V(H_{xy}) \setminus \{x\}\}$.

Proof. First observe that, by Lemma 4, for any $z \in V(H) \setminus \{x\}$, z is a descendant of x in H if and only if v_z is a descendant of v_x in G. Hence, to prove the lemma it is sufficient to observe that for any two vertices v_{z_1}, v_{z_2} in G_{v_x}, there is a v_{z_1}, v_{z_2}-path in G that avoids the vertices of $S \setminus \{v_{z_1}, v_{z_2}\}$ if and only if there is a v_{z_1}, v_{z_2}-path in G_{v_x} that avoids the vertices of $S \cap V(G_{v_x}) \setminus \{v_{z_1}, v_{z_2}\}$, 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 are now ready to present the correctness proof and running time analysis of our algorithm, thereby completing the proof of Theorem 5.

Proof of Theorem 5 (continued). In the first part of the proof of Theorem 5 in the main body of the paper, we claimed that H is an elimination of G if only if G can be eliminated to H with a witness $S \subseteq C(G) \cup U$. This follows immediately from Lemmas 3, 5 and 6.

Let us prove the correctness of the algorithm. We observe that by Lemma 3, if H is an elimination of G and a saved vertex v_x corresponding to a vertex $x \in V(H)$ is a vertex of U, then 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. 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 7. 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 4 and 7.

Finally, we estimate 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 [11] 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 $O(n^{9/2})$. □

The remainder of this appendix is devoted to the proof of Theorem 6.

Theorem 6 (restated). ELIMINATION *is* NP*-complete, even if* G *is restricted to be a tree.*

Before we present the proof of this theorem, 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 8. *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* $\mathrm{diam}(H) \leq \mathrm{diam}(G)$*.*

For some $k \geq 2$, let $Q = \{v_1, \ldots, v_k\}$ be a subset of vertices in a tree G, such that no v_i is an inner vertex of a path between two vertices v_h and v_j. Then G contains a maximal subtree that contains all the vertices of Q as leaves (and that may have some other leaves as well). We denote this tree by T_Q. We use this definition in the statement of the second auxiliary result.

Lemma 9. *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 the tree* T_Q *exists and* $V(T_Q) \setminus Q \subseteq X$*. 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. Denote this path $P_{i,j}$.

Let T_Q be a subtree of T that

(1) does not contain vertices from Q as inner vertices;
(2) contains maximum number of vertices from Q as leaves among trees satisfying (1);
(3) contains maximum number of edges among trees satisfying (1) and (2).

We shall prove that T_Q contains all the vertices of Q. This suffices to prove the lemma.

Let ℓ be the number of vertices from Q contained in T_Q. Observe that $\ell \geq 2$, as every path $P_{i,j}$ satisfies property (1). For the sake of contradiction, assume that $\ell < k$, hence $k \geq 3$. Without loss of generality, v_1 is not contained in T_Q, but v_2 and v_3 are contained in T_Q. Let F be the forest that is obtained from G by removing edges of T_Q.

We claim that the path $P_{1,2}$ is entirely contained in F. Assume otherwise. Let w be the vertex from $P_{1,2}$ that is closest to v_1 among those that are contained in T_Q. By assumption, $w \neq v_2$. Note that also $w \neq v_1$ as $v_1 \notin V(T_Q)$. Hence, T_Q could be extended by adding the edge that is placed immediately before w on $P_{1,2}$. This contradicts properties (2) and (3). A symmetrical reasoning proves that $P_{1,3}$ is also entirely contained in F.

We infer that v_2 and v_3 are contained in the same connected component of F. Hence, there exist two edge-disjoint paths between v_2 and v_3: one entirely contained in T_Q and one entirely contained in F. This is a contradiction with G being a tree. □

Here is our third auxiliary result.

Lemma 10. *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 9, 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 proof of Theorem 6.

Proof of Theorem 6. 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 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 $i \in \{1, \ldots, k\}$ (see Fig. 4):

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

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

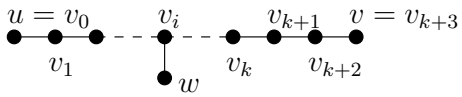

Fig. 4: The graph $F_i(u, v)$.

Now, we construct a tree G (see Fig. 5).

- 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 (see Fig. 5).

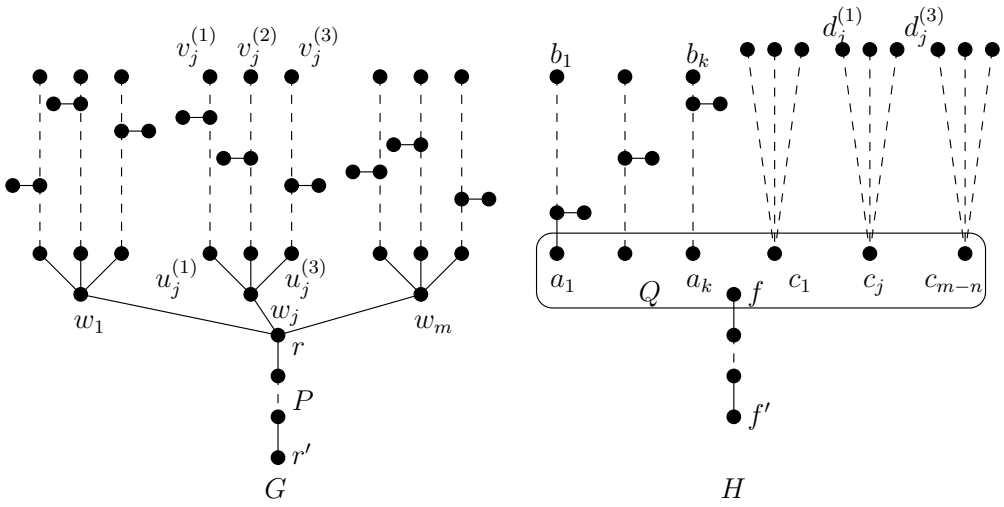

Fig. 5: The graphs G and H.

- 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 and also introduce 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 C contain an exact cover C' of X if and only if H is an elimination of G.

First suppose that $C' = \{C_{j_1}, \ldots, C_{j_n}\}$ 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 9, $r \in X$. Observe that the graph G' obtained from G by the elimination of r has diameter $2k + 10 = \mathrm{diam}(H)$. By Lemma 8, $V(P) \setminus \{r\} \subseteq S$, 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 8, 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 8.

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 10, 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 9, 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 8. 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 $\{w_1,\ldots,w_m\}$, and index vertices of the gadgets $T_j^{(p)}, T_j^{(q)}, T_j^{(s)}$ for $j \in \{1,\ldots,m\}\setminus\{j_1,\ldots,j_n\}$. 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. □