Domination and Cut Problems on Chordal Graphs with Bounded Leafage

Esther Galby
TU Hamburg, Germany

Dániel Marx
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Philipp Schepper
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Roohani Sharma
Max Planck Institute for Informatics, SIC, Saarbrücken, Germany

Prafullkumar Tale
Indian Institute of Science Education and Research, Pune, India

Abstract

The leafage of a chordal graph G is the minimum integer ℓ such that G can be realized as an intersection graph of subtrees of a tree with ℓ leaves. We consider structural parameterization by the leafage of classical domination and cut problems on chordal graphs. Fomin, Golovach, and Raymond [ESA 2018, Algorithmica 2020] proved, among other things, that Dominating Set on chordal graphs admits an algorithm running in time $2^{\mathcal{O}(\ell^2)} \cdot n^{\mathcal{O}(1)}$. We present a conceptually much simpler algorithm that runs in time $2^{\mathcal{O}(\ell)} \cdot n^{\mathcal{O}(1)}$. We extend our approach to obtain similar results for Connected Dominating Set and Steiner Tree. We then consider the two classical cut problems MultiCut with Undeletable Terminals and Multiway Cut with Undeletable Terminals. We prove that the former is W[1]-hard when parameterized by the leafage and complement this result by presenting a simple $n^{\mathcal{O}(\ell)}$-time algorithm. To our surprise, we find that Multiway Cut with Undeletable Terminals on chordal graphs can be solved, in contrast, in $n^{\mathcal{O}(1)}$-time.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases Chordal Graphs, Leafage, FPT Algorithms, Dominating Set, MultiCut with Undeletable Terminals, Multiway Cut with Undeletable Terminals

Digital Object Identifier 10.4230/LIPIcs.IPEC.2022.14

Related Version *Full Version*: `https://arxiv.org/abs/2208.02850` [23]

Funding Research supported by the European Research Council (ERC) consolidator grant No. 725978 SYSTEMATICGRAPH.
Esther Galby and Prafullkumar Tale: Part of the work was carried out when the authors were Post-Doctoral Researchers at CISPA Helmholtz Center for Information Security, Germany.
Philipp Schepper: Part of Saarbrücken Graduate School of Computer Science, Germany.

1 Introduction

The intersection graph of a family $\mathcal{F}$ of nonempty sets is the graph whose vertices are the elements of $\mathcal{F}$ with two vertices being adjacent if and only if their corresponding sets intersect. The most natural and famous example of such intersection graphs are *interval graphs* where $\mathcal{F}$ is a collection of subpaths of a path. Due to their applicability in scheduling, interval graphs have received a considerable attention in the realm of algorithmic graph theory. One useful characterization of an interval graph is that its maximal cliques can be linearly ordered such

that for every vertex, the maximal cliques containing that vertex occur consecutively [25]. This property proves very useful for the design of polynomial-time dynamic programming based or greedy algorithms on interval graphs.

Consider the generalization where $\mathcal{F}$ is a collection of subtrees of a tree instead of subpaths of a path. In this case, the corresponding class of intersection graphs is exactly that of *chordal graphs* [47, 24, 11]. Recall that a graph is chordal if every cycle of length at least 4 has a chord. Often, the algorithms of the types mentioned in the previous paragraph fail to generalize to this superclass as witnessed by the following problems that admit polynomial-time algorithms on interval graphs but are NP-complete on chordal graphs: Dominating Set [13, 8], Connected Dominating Set [3, 48], Steiner Tree [3, 48], Multicut with Undeletable Terminals [28, 44], Subset Feedback Vertex Set (Subset FVS) [45, 21], Longest Cycle [34, 27][1], Longest Path [32], Component Order Connectivity [19], s-Club Contraction [26], Independent Set Reconfiguration [5], Bandwidth [36], Cluster Vertex Deletion [35]. Also, Graph Isomorphism on chordal graphs is polynomial-time equivalent to the problem on general graphs whereas it admits a linear-time algorithm on interval graphs [40].

The problems above remain hard even on *split graphs*, another well-studied subclass of chordal graphs. A graph is a split graph if its vertex set can be partitioned into a clique and an independent set. The collection of split graphs is a (proper) subset of the class of intersection graphs where $\mathcal{F}$ is a collection of substars of a star. As interval graphs are intersection graphs of subpaths of a path (a tree with two leaves) and split graphs are intersection graphs of substars of a star (a tree with arbitrary number of leaves), a natural question to consider is what happens to these problems on subclasses of chordal graphs that are intersection graphs of subtrees of a tree with a bounded number of leaves. Motivated by such questions, we consider the notion of *leafage* introduced by Lin et al. [39]: the leafage of a chordal graph G is the minimum integer ℓ such that G can be realized as an intersection graph of a collection $\mathcal{F}$ of subtrees of a tree that has ℓ leaves. Note that the leafage of interval graphs is at most 2 while split graphs have unbounded leafage. Thus the leafage measures, in some sense, how close a chordal graph is to an interval graph. Alternately, an FPT or XP algorithm parameterized by the leafage can be seen as a generalization of the algorithm on interval graphs.

Related Work. Habib and Stacho [29] showed that we can compute the leafage of a connected chordal graph in polynomial time. Their algorithm also constructs a corresponding *representation tree*[2] T with the minimum number of leaves. In recent years, researchers have studied the structural parameterization of various graph problems on chordal graphs parameterized by the leafage. Fomin et al. [20] and Arvind et al. [2] proved, respectively, that the Dominating Set and Graph Isomorphism problems on chordal graphs are FPT parameterized by the leafage. Barnetson et al. [4] and Papadopoulos and Tzimas [46] presented XP-algorithms running in time $n^{\mathcal{O}(\ell)}$ for Fire Break and Subset FVS on chordal graphs, respectively. Papadopoulos and Tzimas [46] also proved that Subset FVS is W[1]-hard when parameterized by the leafage. Hochstättler et al [31] showed that we can compute the neighborhood polynomial of a chordal graph in $n^{\mathcal{O}(\ell)}$-time.

It is known that the size of *asteroidal set* in a chordal graph is upper bounded by its leafage [39]. See [30, 1] for the relationship between leafage and other structural properties of chordal graphs. Kratsch and Stewart [37] proved that we can effectively 2ℓ-approximate

[1] See Exercise 2 in Chapter 6 in [27].
[2] We present formal definitions of the terms used in this section in Section 2.

bandwidth of chordal graphs of leafage ℓ. Chaplick and Stacho [14] generalized the notion of leafage to *vertex leafage* and proved that, unlike leafage, it is hard to determine the optimal vertex leafage of a given chordal graph. Figueiredo et al. [18] proved that Dominating Set, Connected Dominating Set and Steiner Tree are FPT on chordal graphs when parameterized by the size of the solution plus the vertex leafage, provided that a tree representation with optimal vertex leafage is given as part of the input.

Our Results. We consider well-studied domination and cut problems on chordal graphs. As our first result, we prove that Dominating Set on chordal graphs of leafage at most ℓ admits an algorithm running in time $2^{\mathcal{O}(\ell)} \cdot n^{\mathcal{O}(1)}$. This improves upon the existing algorithm by Fomin et al. [20, Theorem 9] which runs in time $2^{\mathcal{O}(\ell^2)} \cdot n^{\mathcal{O}(1)}$. Despite being significantly simpler than the algorithm in [20], our algorithm in fact solves the Red-Blue Dominating Set problem, a well-known generalization of Dominating Set. In this generalized version, an input is a graph G with a partition (R, B) of its vertex set and an integer k, and the objective is to find a subset D of R that dominates every vertex in B, i.e., $B \subseteq N(D)$. We further use this algorithm to solve other related domination problems.

▶ **Theorem 1.** Dominating Set, Connected Dominating Set, *and* Steiner Tree *can be solved in* $2^{\mathcal{O}(\ell)} \cdot n^{\mathcal{O}(1)}$ *on chordal graphs of leafage at most* ℓ.

The reductions in [8] and [48] used to prove that these problems are NP-complete on chordal graphs imply that these problems do not admit $2^{o(n)}$, and hence $2^{o(\ell)} \cdot n^{\mathcal{O}(1)}$, algorithms unless the ETH fails.

Arguably, the two most studied cut problems are MultiCut and Multiway Cut. In the MultiCut problem, an input is graph G, a set of terminal pairs $P \subseteq V(G) \times V(G)$ and an integer k, and the objective is to find a subset $S \subseteq V(G)$ of size at most k such that no pair of vertices in P is connected in $G - S$. In the Multiway Cut problem, instead of terminal pairs, we are given a terminal set P and the objective is to find a subset $S \subseteq V(G)$ of size at most k such that no two vertices in P are connected in $G - S$. These problems and variations of them have received a considerable attention which lead to the development of new techniques [41, 42, 9, 16, 15]. Misra et al. [43] studied the parameterized complexity of these problems on chordal graphs. Guo et al. [28] proved that MultiCut with Deletable Terminals is NP-complete on interval graphs, thereby implying that this problem is paraNP-hard when parameterized by the leafage. We consider the MultiCut with Undeletable Terminals problem and prove the following result.

▶ **Theorem 2.** MultiCut with Undeletable Terminals *on chordal graphs is* W[1]-hard *when parameterized by the leafage* ℓ *and assuming the* ETH*, does not admit an algorithm running in time* $f(\ell) \cdot n^{o(\ell)}$ *for any computable function* f*. However, it admits an* XP*-algorithm running in time* $n^{\mathcal{O}(\ell)}$.

Next, we focus on the Multiway Cut with Undeletable Terminals problem. We find it somewhat surprising that the classical complexity of this problem on chordal graphs was not known. Bergougnoux et al. [7], using the result in [20], proved that the problem admits an XP-algorithm when parameterized by the leafage[3]. Our next result significantly improves upon this and [43, Theorem 2] which states that the problem admits a polynomial kernel when parameterized by the solution size.

[3] See the discussion after Corollary 2 on page 1388 in [7].

▶ **Theorem 3.** Multiway Cut with Undeletable Terminals *can be solved in* $n^{\mathcal{O}(1)}$*-time on chordal graphs.*

A well-known trick to convert an instance of Multiway Cut with Deletable Terminals into an instance of Multiway Cut with Undeletable Terminals is to add a pendant vertex to each terminal, remove that vertex from the set of terminals, and make the newly added vertex a terminal. As this reduction converts a chordal graph into another chordal graph, Theorem 3 implies that Multiway Cut with Deletable Terminals is also polynomial-time solvable on chordal graphs. Another closely related problem is Subset FVS which is NP-complete on split graphs [45]. To the best of our knowledge, this is the first graph class on which the classical complexity of these two problems differ.

Next, we revisit the problems on chordal graphs with bounded leafage and examine how far we can generalize this class. An *asteroidal triple* of a graph G is a set of three vertices such that each pair is connected by some path that avoids the closed neighborhood of the third vertex. Lekkerkerker and Boland [38] showed that a graph is an interval graph if and only if it is chordal and does not contain an asteroidal triple. They also listed all minimal chordal graphs that contain an asteroidal triple (see, for instance, [12, Figure 1]). Among this list, we found the *net graph* to be the most natural to generalize. For a positive integer $\ell \geq 3$, we define H_ℓ as a split graph on 2ℓ vertices with split partition (C, I) such that the only edges across C, I are a perfect matching. Note that H_3 is the net graph. As interval graphs are a proper subset of the collection of chordal graphs that do not contain a net graph as an induced subgraph, the collection of the chordal graph of leafage ℓ is a proper subset of the collection of chordal graphs that do not contain $H_{\ell+1}$ as an induced subgraph (see the full version [23]). We show that, although the considered domination problems are polynomial-time solvable for constant ℓ, the fixed-parameter tractability results are unlikely to extend to this larger class. Let us mention that the core reason these problems admit XP-algorithms parameterized by ℓ lies in the fact that H_ℓ-induced-subgraph-free chordal graphs have mim-width at most $\ell - 1$ [33] (all three problems are indeed known to be solvable in $n^{\mathcal{O}(m)}$ on graphs of mim-width at most m [6, 10]). Nonetheless, we present alternative algorithms which we believe to be simpler and more insightful. In fact, we give a $n^{\mathcal{O}(\ell)}$ algorithm for the more general Red-Blue Dominating Set problem and obtain the other results by simple reductions.

▶ **Theorem 4.** Dominating Set*,* Connected Dominating Set *and* Steiner Tree *on* H_ℓ*-induced-subgraph-free chordal graphs are* W[1]-hard *when parameterized by* ℓ *and assuming the* ETH*, do not admit an algorithm running in time* $f(\ell) \cdot n^{o(\ell)}$ *for any computable function* f*. However, they all admit* XP*-algorithms running in time* $n^{\mathcal{O}(\ell)}$*.*

We observe a similar trend with respect to MultiCut with Undeletable Terminals as its parameterized complexity jumps from W[1]-hard on chordal graph of leafage ℓ to paraNP-hard on H_ℓ-induced-subgraph-free chordal graphs when parameterized by ℓ.

▶ **Theorem 5.** MultiCut with Undeletable Terminals *is* NP-hard *even when restricted to* H_3*-induced-subgraph-free chordal graphs.*

Table 1 summarizes our results.

Our Methods. We briefly discuss the methods used in our two main algorithms, namely the algorithm for Dominating Set and the one for Multiway Cut.

Table 1 Overview of the known results and our contributions. Every graph class mentioned in the first column is a proper subset of the graph class mentioned below.

Input graph	Dom Set, ConnDom Set, Steiner Tree	MultiCut with UnDel Term	MultiwayCut
Interval Graphs	Poly-time [13, 3]	Poly-time [28]	Poly-time [7]
Chordal graphs of leafage ℓ	$2^{\mathcal{O}(\ell^2)} \cdot n^{\mathcal{O}(1)}$ algo [20] $2^{\mathcal{O}(\ell)} \cdot n^{\mathcal{O}(1)}$ algo (Thm 1)	W[1]-hard $n^{\mathcal{O}(\ell)}$ algo (Thm 2)	$n^{\mathcal{O}(\ell)}$ algo [7] Poly-time (Thm 3)
H_ℓ-induced subgraph-free chordal	W[1]-hard (Thm 4); $n^{\mathcal{O}(\ell)}$ algo (Thm 4, [33] + [6, 10])	NP-hard for $\ell \geq 3$ (Thm 5)	Poly-time (Thm 3)
Chordal graphs	NP-complete [8]	NP-complete [48]	Poly-time (Thm 3)

Red-Blue Dominating Set in Chordal Graphs. As mentioned earlier, the linear ordering of cliques in interval graphs is particularly useful for the design of polynomial-time algorithms. Such an ordering is not possible even if G is a chordal graph whose representation tree T is a star. Consider the case where the model of every red vertex in G includes the center of the star T (and possibly some leaves) and the model of every blue vertex is (only) a leaf. We can solve this instance by converting it to an instance of Set Cover and solving it using the FPT algorithm parameterized by the size of the universe. In this case, the size of the universe is at most the number of leaves which is upper bounded by the leafage. In the other case where the properties of red vertices and blue vertices are reversed, we obtain a similar result by creating an equivalent instance of Hitting Set.

These ideas can be used in a more general setting as long as the following two properties are satisfied: (1) the model of each vertex is *local*, that is, it contains at most one branching node, and (2) each branching node is contained only in models of either red vertices or blue vertices. Based on this observation, we introduce a restricted version of the problem in which the input graph is required to satisfy these two conditions. We then show that the general case reduces to this restricted version: indeed, we prove that there is a branching algorithm that constructs $2^{\mathcal{O}(\ell)}$ many instances (where ℓ is the leafage of the input graph) of the restricted version of the problem such that the input instance is a Yes-instance if and only if one of these newly created instances is a Yes-instance. These two properties ensure that the graph induced by the red and blue vertices whose model intersect the subtree rooted at a farthest branching node (from some fixed root) satisfies the premise of at least one of the cases mentioned in the previous paragraph. We then present a greedy procedure, based on solving the Set Cover and Hitting Set problems, that identifies some part of an optimum solution. Apart from this greedy selection procedure, all other steps of the algorithm run in polynomial time.

Multiway Cut in Chordal Graphs. We give a polynomial-time algorithm for Multiway Cut on chordal graphs by solving several instances of the (s,t)-Cut problem (not necessarily with unit capacities). Our strategy is based on a bottom-up dynamic programming (DP) on a tree representation of a chordal graph. An interesting aspect of our DP is that we need to look-up *all* DP table values that are already computed to compute a new entry. This is in contrast to typical DP-based algorithms that do computations only based on *local* entries.

We remark that we do not expect to design an algorithm for Multiway Cut on chordal graphs using much simpler arguments (like a simple dynamic programming procedure etc.) as the problem generalizes some well-studied cut-flow based problems. As an example, recall the Vertex Cover problem on bipartite graphs where given a bipartite graph G with bipartition (A, B), the goal is to find $A' \subseteq A$ and $B' \subseteq B$ such that $|A' \cup B'|$ is minimum and $N(A \setminus A') \subseteq B'$. The set $A' \cup B'$ is called a vertex cover of G. The Vertex Cover problem on bipartite graphs reduces to the Multiway Cut problem on chordal graphs: indeed, let G' be the graph obtained from G by making B a clique, adding new pendant vertex t_a to each vertex $a \in A$, and further adding another new vertex t that is adjacent to all vertices of B. Then G' is a chordal graph and letting $T = t \cup \{t_a \mid a \in A\}$, it is easy to see that $S \subseteq V(G)$ is a vertex cover of G if and only if S is a T-multiway-cut in G'. As mentioned earlier, our algorithm solves several instances of the (s,t)-Cut problem, which also sits at the heart of some algorithms for Vertex Cover on bipartite graphs. The above reduction suggests that an algorithm for Multiway Cut on chordal graphs using much simpler techniques, would imply an algorithm for Vertex Cover on bipartite graphs that uses much simpler techniques as well.

Note that a similar reduction would work from the weighted variant of the Vertex Cover problem on bipartite graphs. This can be achieved by further replacing each vertex of the graph G by a clique of size proportional to the weight of this vertex and making each vertex of the clique adjacent to all the neighbors of this vertex. This reduction still preserves the chordality of the resulting graph.

2 Preliminaries

For a directed graph H, we denote, for all $v \in V(H)$, by $N_H^+(v)$ the out-neighbors of v and by $N_H^-(v)$ the in-neighbors of v. If H is clear from the context, we omit the subscript H. Given a (directed) path P and two vertices $u, v \in V(P)$, we denote by $P[u, v]$ the subpath of P from u to v. For a tree T rooted at r, we define the function $\mathtt{parent}(t, T) : V(T) \setminus \{\mathsf{r}\} \mapsto V(T)$ to specify the unique parent of the nodes in T. For any node $t \in T$, we denote by T_t the subtree rooted at t.

It is well-known that a chordal graph G can be represented as intersection graphs of subtrees in a tree T. The pair $(T, \mathcal{M})$ is called a *tree representation* of G where for every $v \in V(G)$, we denote by $\mathcal{M}(v)$ the subtree corresponding to v and refer to $\mathcal{M}(v)$ as the *model* of v in T. The *leafage* of G, denoted by $\mathtt{lf}(G)$, is defined as the minimum number of leaves in the tree of a tree representation of G.

For every node $\alpha \in V(T)$, we let $\mathtt{ver}(\alpha) = \{v \in V(G) \mid \alpha \in \mathcal{M}(v)\}$ be the set of vertices in G that contain the node α is their model. A vertex $v \in V(G)$ whose model contains α may also be referred to as an *α-vertex*. Similarly, for every edge $e \in E(T)$, we define $\mathtt{ver}(e) = \{v \in V(G) \mid e \subseteq \mathcal{M}(v)\}$. Given a subtree T' of T, we denote by $G_{|T'}$ the subgraph of G induced by those vertices $x \in V(G)$ such that $V(\mathcal{M}(x)) \subseteq V(T')$.

3 Dominating Set

For a graph G, a set $X \subseteq V(G)$ is a *dominating set* if every vertex in $V(G) \setminus X$ has at least one neighbor in X, that is, $V(G) = N[X]$. In the Dominating Set problem (DomSet for short), the input is a graph G and an integer k, and the objective is to decide whether G has a dominating set of size at most k. We assume that the leafage of the input graph is given as part of the input. If not, recall that it can be computed in polynomial time [29]. We consider a generalized version of this problem as defined below.

Red-Blue Dominating Set (Red-Blue-DomSet)
Input: A graph G, a partition (R, B) of $V(G)$, and an integer k.
Question: Does there exist a set $X \subseteq R$ of size at most k such that $B \subseteq N(X)$?

To solve DomSet, it is sufficient to solve Red-Blue-DomSet even when the input is restricted to chordal graphs of leafage ℓ. A simple reduction, which is given in the full version [23], suffices to prove the following result.

▶ **Lemma 6.** *There is a polynomial-time algorithm that given an instance (G, k) of DomSet constructs an equivalent instance $(G', (R', B'), k)$ of Red-Blue-DomSet such that if G has leafage at most ℓ, then so does G'.*

In the remainder of this section, we present an FPT algorithm for Red-Blue-DomSet when parameterized by the leafage ℓ of the input graph. The algorithm consists of two parts. In the first part, the algorithm constructs $2^{\mathcal{O}(\ell)}$ many instances of a "restricted version" of the problem such that the input instance is a Yes-instance if and only if one of these newly created instances is a Yes-instance. Moreover, the graphs in the newly created instances satisfy certain properties that allow us to design a fast algorithm. See Lemma 7 for the formal statement. In the second part (cf. Lemma 8), the algorithm solves the restricted version of Red-Blue-DomSet which is defined as follows.

Restricted-Red-Blue Dominating Set (Rest-Red-Blue-DomSet)
Input: A chordal graph G, a partition (R, B) of $V(G)$, an integer k and tree representation $(T, \mathcal{M})$ of G such that
- for every vertex in G, its model contains at most one branching node of T, and
- for all branching nodes $\gamma \in V(T)$, there are either only red γ-vertices or only blue γ-vertices.

Question: Does there exist a set $D \subseteq R$ of size at most k such that $B \subseteq N(D)$?

The first step of the algorithm is summarized in the following lemma which is proven in Appendix A.

▶ **Lemma 7.** *Let $\mathcal{I} = (G, (R, B), k)$ be an instance of Red-Blue-DomSet where G is a chordal graph of leafage at most ℓ. We can construct, in time $2^{\mathcal{O}(\ell)} \cdot n^{\mathcal{O}(1)}$, a collection $\{\mathcal{I}_i = (G_i, (R_i, B_i), k) \mid i \in [2^{\mathcal{O}(\ell)}]\}$ of Rest-Red-Blue-DomSet instances such that*
- *for every $i \in [2^{\mathcal{O}(\ell)}]$, G_i is a chordal graph of leafage at most 2ℓ, and*
- *$\mathcal{I}$ is a Yes-instance of Red-Blue-DomSet if and only if at least one of the instances in the collection is a Yes-instance of Rest-Red-Blue-DomSet.*

The second step of the algorithm solves Rest-Red-Blue-DomSet. Formally, we prove the following lemma.

▶ **Lemma 8.** Rest-Red-Blue-DomSet *admits an algorithm running in time $2^{\mathcal{O}(\ell)} \cdot n^{\mathcal{O}(1)}$.*

We first state some easy reduction rules before we handle two cases based on whether the farthest branching node[4] is contained only in the models of red vertices or blue vertices. We present Greedy Select 14 and Greedy Select 16 to handle these cases. The proof of the lemma follows from the correctness of the Greedy Select 14 and 16 and the fact that each application of the greedy selection procedure deletes some vertices in the graph.

[4] We assume that the tree in the tree representation is rooted and thus, by farthest branching node, we mean farthest from the root.

We first introduce some notations. Recall that an instance of Rest-Red-Blue-DomSet contains a chordal graph G, a partition (R, B) of $V(G)$, an integer k and tree representation $(T, \mathcal{M})$ of G such that for every vertex in G, its model contains at most one branching node of T, and for all branching nodes $\gamma \in V(T)$, there are either only red γ-vertices or only blue γ-vertices. We assume, without loss of generality, that the tree T is rooted at node r. Unless mentioned otherwise, α denotes the farthest branching node in T from the root, that is, each proper subtree of T_α is a path. If there are more than one branching node that satisfy the property, we arbitrarily select one of them. Let β be the closest branching ancestor of α, that is, no internal node in the unique path from α to β is a branching node in T.[5] Recall that for a vertex $v \in V(G)$, we define $\mathtt{top}_{\mathcal{M}}(v)$ as the node $\eta \in \mathcal{M}(v)$ that is closest to the root. Likewise if a leaf λ is fixed, we define $\mathtt{bot}^{\lambda}_{\mathcal{M}}(v)$ as the node $\eta \in \mathcal{M}(v)$ that is closest to λ. For ease of notation, we omit λ as it is always clear from the context.

▶ **Definition 9.** *Let γ be a node of the tree T. We define the following sets of vertices in G.*

- *$B^{\cap}_{\gamma}, R^{\cap}_{\gamma}, V^{\cap}_{\gamma}$ are the sets of, respectively, blue, red, all vertices $v \in V(G)$ whose models intersect the tree rooted at γ, i.e., $\mathcal{M}(v) \cap V(T_\gamma) \neq \emptyset$.*
- *$B^{\subseteq}_{\gamma}, R^{\subseteq}_{\gamma}, V^{\subseteq}_{\gamma}$ are the sets of, respectively, blue, red, all vertices $v \in V(G)$ whose models are completely contained inside the tree rooted at γ, i.e., $\mathcal{M}(v) \subseteq V(T_\gamma)$.*
- *$B^{\subseteq\dagger}_{\gamma}, R^{\subseteq\dagger}_{\gamma}, V^{\subseteq\dagger}_{\gamma}$ are the sets of blue, red, all vertices $v \in V(G)$ where the model is completely contained inside the tree rooted at γ but does not contain γ, respectively, i.e. $\mathcal{M}(v) \subseteq V(T^{\dagger}_{\gamma}) = V(T_\gamma) \setminus \{\gamma\}$.*
- *$B^{\in}_{\gamma}, R^{\in}_{\gamma}, V^{\in}_{\gamma}$ are the sets of, respectively, blue, red, all vertices $v \in V(G)$ whose models contains γ, i.e., $\gamma \in \mathcal{M}(v)$.*

Simplifications. We first apply the following easy reduction rules whose correctness readily follows from the definition of the problem. It is also easy to see that the reduction rules can be applied in polynomial time and the reduced instance is also a valid instance of Rest-Red-Blue-DomSet.

▶ **Reduction Rule 10.** *If there is a blue vertex, which is not adjacent to a red vertex, or if $k < 0$, then return a trivial No-instance.*

▶ **Reduction Rule 11.**

- *If there are two blue vertices u, v such that $\mathcal{M}(u) \subseteq \mathcal{M}(v)$, then delete v.*
- *If there are two red vertices u, v such that $\mathcal{M}(u) \subseteq \mathcal{M}(v)$, then delete u.*

Consider a blue vertex v in G whose model is contained in the subtree rooted at α. Moreover, let v be such a vertex for which $\mathtt{top}_{\mathcal{M}}(v)$ is farthest from the root and v is not adjacent to a red vertex whose model contains α. Hence, there is a natural ordering amongst the red neighbors of v. Note that such an ordering is not possible if some of its neighbors contain α in their models. As any solution contains a red neighbor of v, it is safe to include its neighbor v_r for which $\mathtt{top}_{\mathcal{M}}(v_r)$ is closest to α.

▶ **Reduction Rule 12.** *Suppose that there is a blue vertex $v \in B^{\subseteq\dagger}_{\alpha}$ such that $\mathtt{top}_{\mathcal{M}}(v)$ is farthest from the root and v is not adjacent to any red α-vertices. Moreover, amongst all the red neighbors of v, let v_r be the node such that $\mathtt{top}_{\mathcal{M}}(v_r)$ is closest to α. Then, remove v_r and all of its blue neighbors and decrease k by 1.*

[5] If α is the root of the tree, then we can add an artificial new root β which is not contained in the model of any vertex.

We remark that the above reduction rule is applicable irrespective of the fact whether either all α-vertices are red or all α-vertices are blue.

Case-1: All the vertices that contain α in their models are red vertices. Let β be the closest branching ancestor of α. Consider the blue vertices whose model intersect the path from α to β. Note that there may not be any such blue vertex; however, we find it convenient to present an uniform argument. With a slight abuse of notation, let $b_1, \ldots, b_d$ be these blue vertices ordered according to their endpoint in the direction of α, that is, for $i < j$ we have either $\texttt{bot}_{\mathcal{M}}(b_i) = \texttt{bot}_{\mathcal{M}}(b_j)$ or $\texttt{bot}_{\mathcal{M}}(b_i)$ is closer to α than $\texttt{bot}_{\mathcal{M}}(b_j)$. For each $i \in [d]$, we compute an optimal solution for dominating the vertices whose model is in the tree rooted at α (i.e., the vertices of $B_\alpha^{\subseteq\dagger}$) and the vertex b_i while only using red α-vertices. Formally, we want to compute an optimal solution for the following instance: $\mathcal{I}_i \coloneqq G[R_\alpha^{\cap} \cup B_\alpha^{\subseteq} \cup \{b_i\}]$. We also define instance $\mathcal{I}_0 \coloneqq G[R_\alpha^{\cap} \cup B_\alpha^{\subseteq}]$ to handle the cases when there are no blue vertices whose model intersects the path from α to β or when b_1 (and hence, the other blue vertices mentioned above) are not dominated by red α-vertices in an optimum solution. To simplify notation we set $\mathrm{OPT}_i \coloneqq \mathrm{OPT}(\mathcal{I}_i)$ in the following. If $\mathcal{I}_i$ is not defined, then we set $\mathrm{OPT}_i = \infty$. Note that the solution OPT_i also dominates the blue vertices $b_1, \ldots, b_{i-1}$ due to the ordering of the b_is. Hence, for any $i, j \in [0, d]$ such that $i < j$, we have $|\mathrm{OPT}_i| \leq |\mathrm{OPT}_j|$. We use this monotonicity to prove the following structural lemma.

▶ **Lemma 13.** *Let $q \in [0, d]$ be the largest value such that $|OPT_q| = |OPT_0|$. If there is a solution, then there is an optimum solution containing OPT_q.*

Proof. Let OPT be an optimum solution of $(G, (R, B), k)$. Let S denote the collection of vertices in OPT whose model contains nodes in the subtree rooted at α, i.e., $S \coloneqq \mathrm{OPT} \cap R_\alpha^{\cap}$. We claim that we can replace S by a super-set S' of OPT_q of equal size to obtain another solution.

Let $j \in [0, d]$ be the largest integer such that b_j is dominated by some vertex in S. If $j \leq q$, then by our choice of q, $|S| = |\mathrm{OPT}_q|$. By the definition of the $\mathcal{I}_i$s, we get that OPT_q is also a solution for $\mathcal{I}_i$. Hence, we can replace S by OPT_q to get another optimal solution. Suppose therefore that $j > q$. By our choice of q, we have $|S| > |\mathrm{OPT}_q|$. Let r_j be the red α-vertex with $\texttt{top}_{\mathcal{M}}(r_j)$ closest to β such that b_j is a neighbor of r_j. Such a vertex exists, as by assumption, S contains one of these vertices which dominates b_j. Then we replace S by $S' = \mathrm{OPT}_q \cup \{r_j\}$. As $|S| > |\mathrm{OPT}_q|$, we have $|S'| \leq |S|$. Moreover, observe that $S' \cup \mathrm{OPT} \setminus S$ is still a solution as all vertices in $B_\alpha^{\subseteq\dagger}$ and the vertices $b_1, \ldots, b_q$ are dominated by some vertex in OPT_q, vertex r_j dominates the vertices $b_{q+1}, \ldots, b_j$ and, by the choice of j, the vertices $b_{j+1}, \ldots, b_d$ are dominated by some vertex not contained in S. ◀

We devise a greedy selection step based on the above lemma which can be completed in time $2^{\mathcal{O}(\ell)} \cdot n^{\mathcal{O}(1)}$ (cf. full version in [23]).

▶ **Greedy Select 14.** *Let $q \in [0, d]$ be the largest value such that $|OPT_q| = |OPT_0|$. Include the vertices of OPT_q in the solution, i.e., delete the red vertices in OPT_q, the blue vertices that are adjacent to vertices in OPT_q, and decrease k by $|OPT_q|$.*

Case-2: All the vertices that contain α in their models are blue vertices. Let β be the closest branching ancestor of α. We consider two cases depending on whether there is a red vertex whose model intersects the path from α to β. If there is no such red vertex, then we

consider the graph induced by all the red vertices whose model is (properly) contained in the subtree rooted at α and the blue vertices whose model intersects the subtree rooted at α. Formally, we define $\mathcal{I}_0 = G[R_\alpha^{\subseteq} \cup B_\alpha^{\cap}]$.

Consider the other case and suppose that there are $d \geq 1$ many red vertices whose model intersects the path from α to β. Let $r_1, \ldots, r_d$ be these vertices ordered according to their endpoints in the direction of α, that is, for $i < j$, we have either $\texttt{bot}_{\mathcal{M}}(r_i) = \texttt{bot}_{\mathcal{M}}(r_j)$ or $\texttt{bot}_{\mathcal{M}}(r_i)$ is closer to α than $\texttt{bot}_{\mathcal{M}}(r_j)$. For each such red vertex v_i, we compute the optimal solution to dominate the vertices in $B_\alpha^{\cap}$ by vertices in $R_\alpha^{\subseteq}$ assuming that v_i is already selected. Note that we only have to focus on the blue vertices in $B_\alpha^{\cap}$ which are not adjacent to v_i. Formally we define $\mathcal{I}_i = G[R_\alpha^{\subseteq} \cup (B_\alpha^{\cap} \setminus N[v_i])]$. It is possible that the optimum solution does not include any of the vertices in $\{r_1, r_2, \ldots, r_d\}$. To handle this case, we define $\mathcal{I}_{d+1} = G[R_\alpha^{\subseteq} \cup B_\alpha^{\cap}]$. To simplify notation, we set $\mathrm{OPT}_i \coloneqq \mathrm{OPT}(\mathcal{I}_i)$ in the following. Note that for the instance defined above, R_i is same for every instance whereas $B_i \subseteq B_{i+1}$ because of the ordering. Hence, for any $i, j \in [d+1]$ such that $i < j$, we have $|\mathrm{OPT}_i| \leq |\mathrm{OPT}_j|$. We use this monotonicity to prove the following structural lemma.

▶ **Lemma 15.** *If there is a red vertex whose model intersects the path from α to β, let $q \in [d+1]$ be the largest value such that $|OPT_q| = |OPT_1|$. Otherwise, define $OPT_q = OPT_0$. If there is a solution for the instance, then there is an optimum solution OPT such that $OPT \cap R_\alpha^{\subseteq} = OPT_q$.*

Proof. If there is no red vertices whose model intersects the path from α to β, then all the red vertices in G that are adjacent to blue vertices in $\mathcal{I}_0$ are the red vertices in $\mathcal{I}_0$. Hence, the statement of the lemma follows.

We now consider the case where there are red vertices whose model intersects the path from α to β. Let OPT be an optimum solution of $(G, (R, B), k)$. Let S denote the collection of vertices in OPT whose model is (properly) contained in the subtree rooted at α, i.e., $S \coloneqq \mathrm{OPT} \cap R_\alpha^{\subseteq\dagger}$. We claim that we can replace S by a super-set S' of OPT_q of equal size to obtain another optimum solution.

Let $j \in [d]$ be the smallest index such that v_j is contained in OPT. Note that, by definition, $j \neq d+1$ as there are only d red vertices with the said property. If $j \leq q$, then by our choice of q, $|S| \geq |\mathrm{OPT}_j|$. By the definition of $\mathcal{I}_j$ and the fact blue vertices in $\mathcal{I}_j$ are subset of blue vertices in $\mathcal{I}_q$, OPT_q is also a solution for $\mathcal{I}_j$. Hence, we can replace S by OPT_q to get another optimal solution. Suppose therefore that $j > q$. By our choice of q, we have $|\mathrm{OPT}_j| > |\mathrm{OPT}_q|$. As OPT is a solution, all vertices in $B_\alpha^{\cap}$ must be covered by OPT. Hence, we can replace S by $S' = \mathrm{OPT}_q \cup \{r_q\}$ and get a solution of not larger size which still dominates all vertices in $B_\alpha^{\cap}$. Indeed, the vertices which are not dominated by OPT_q are dominated by r_q. ◀

We devise a greedy selection step based on the above lemma which can be completed in time $2^{\mathcal{O}(\ell)} \cdot n^{\mathcal{O}(1)}$ (cf. full version in [23]).

▶ **Greedy Select 16.** *If there is a red vertex whose model intersects the path from α to β, let $q \in [d+1]$ be the largest value such that $|OPT_q| = |OPT_1|$. Otherwise, define $OPT_q = OPT_0$. Include OPT_q in the solution, i.e., delete the red vertices in OPT_q, the blue vertices that are adjacent to vertices in OPT_q, and decrease k by $|OPT_\ell|$.*

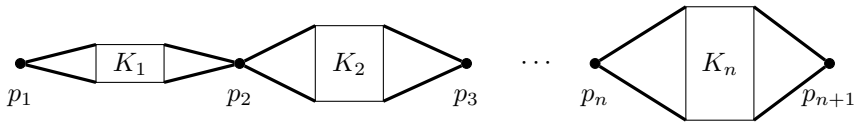


Figure 1 The auxiliary graph B. Rectangles represent cliques and thick edges indicate that the corresponding vertex is complete to the corresponding cliques.

4 Multicut with Undeletable Terminals

This section considers the MultiCut with Undeletable Terminals problem formally defined as follows.

MultiCut with Undeletable Terminals (MultiCut with UnDel Term)
Input: An undirected graph G, a set $P \subseteq V(G) \times V(G)$, and an integer k.
Question: Is there a set $S \subseteq V(G) \setminus V(P)$ such that $|S| \leq k$ and for all $(p, p') \in P$, there is no path between p and p' in $G - S$?

In the following, a set $S \subseteq V(G) \setminus V(P)$ such that for all $(p, p') \in P$, there is no path between p and p' in $G - S$ is called a *P-multicut* in G. We first prove that when the input is restricted to chordal graphs, the problem is unlikely to admit an FPT algorithm when parameterized by the leafage. We then complement this result with an XP-algorithm parameterized by the leafage. We restate the theorem with the precise statement for the reader's convenience.

▶ **Theorem 2.** MultiCut with Undeletable Terminals *on chordal graphs is* W[1]*-hard when parameterized by the leafage ℓ and assuming the* ETH*, does not admit an algorithm running in time $f(\ell) \cdot n^{o(\ell)}$ for any computable function f. However, it admits an* XP*-algorithm running in time $n^{\mathcal{O}(\ell)}$.*

To prove that the problem is W[1]-hard, we present a parameter preserving reduction from Multicolored Clique. An instance of this problem consists of a simple graph G, an integer q, and a partition $(V_1, V_2, \ldots, V_q)$ of $V(G)$. The objective is to determine whether there is a clique in G that contains exactly one vertex from each part V_i. Such a clique is called a *multicolored clique*. We assume, without loss of generality, that each V_i is an independent set and that $|V_1| = \ldots = |V_q| = n$.[6] This implies, in particular, that $|E(G)| < n^2 \cdot q^2$. For every $i \in [q]$, we denote by $v_1^i, \ldots, v_n^i$ the vertex set of V_i and for every $i \neq j \in [q]$, we denote by $E_{i,j} \subseteq E(G)$ the set of edges between V_i and V_j. We define $M := (n+1)^2 \cdot q^2$.

Reduction. The reduction takes as input an instance $(G, q, (V_1, \ldots, V_q))$ of Multicolored Clique and outputs an instance (H, P, k) of MultiCut with UnDel Term which is constructed as follows.

- The reduction starts by constructing an auxiliary graph B. The vertex set of B consists of $n+1$ vertices $p_1, \ldots, p_{n+1}$ and n vertex-disjoint cliques $K_1, \ldots, K_n$ such that $|K_a| = a \cdot M$ for every $a \in [n]$. Then, it adds edges so that p_1 is complete to K_1, p_{n+1} complete to K_n, and p_a complete to $K_{a-1} \cup K_a$ for every $a \in [n] \setminus \{1\}$. This completes the construction of B (see Figure 1).

[6] Unlike in the rest of the article, we *do not* use n to denote the total number of vertices in G to keep notation simple while presenting the reduction.

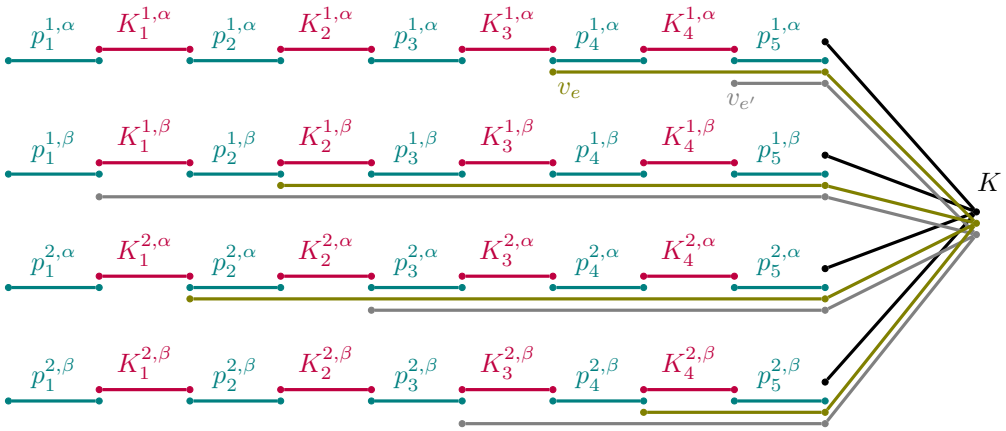


Figure 2 A tree representation of the graph H restricted to the gadgets representing V_1, V_2 and $E_{1,2}$ where $n = 4$ and $E_{1,2} = \{e = v_3^1v_1^2, e' = v_4^1v_2^2\}$.

- For each $i \in [q]$, the reduction introduces two vertex-disjoint copies $B^{i,\alpha}$ and $B^{i,\beta}$ of B. For every $i \in [q]$, let $p_1^{i,\alpha}, \ldots, p_{n+1}^{i,\alpha}$ denote the copies of $p_1, \ldots, p_{n+1}$ in $B^{i,\alpha}$ and $K_1^{i,\alpha}, \ldots, K_n^{i,\alpha}$ denote the copies of $K_1, \ldots, K_n$ in $B^{i,\alpha}$. Moreover, for every $1 \leq a_1 \leq a_2 \leq n+1$, we define, for notational convenience,

$$p^{i,\alpha}[a_1, a_2] \coloneqq \{p_a^{i,\alpha} \mid a_1 \leq a \leq a_2\} \qquad \text{and} \qquad K^{i,\alpha}[a_1, a_2] \coloneqq \bigcup_{a_1 \leq a \leq a_2} K_a^{i,\alpha}.$$

  We define $p_a^{i,\beta}$, $K_a^{i,\beta}$, $p^{i,\beta}[a_1, a_2]$, and $K^{i,\beta}[a_1, a_2]$ in a similar way.
- For $i \in [q]$ and $a \in [n]$, the reduction uses $p_a^{i,\alpha}$, $p_{n+1-a}^{i,\beta}$, $K_a^{i,\alpha}$, and $K_{n+1-a}^{i,\beta}$ to encode vertex v_a^i.
- For every edge $e = v_{a_i}^i v_{a_j}^j \in E(G)$, the reduction introduces an *edge-vertex* v_e and adds edges so that v_e is complete to the following sets.
  - $p^{i,\alpha}[a_i + 1, n + 1]$ and $K^{i,\alpha}[a_i, n + 1]$ in $V(B^{i,\alpha})$.
  - $p^{j,\alpha}[a_j + 1, n + 1]$ and $K^{i,\alpha}[a_j, n + 1]$ in $V(B^{j,\alpha})$.
  - $p^{i,\beta}[n + 1 - a_i + 1, n + 1]$ and $K^{i,\beta}[n + 1 - a_i + 1, n + 1]$ in $V(B^{i,\beta})$.
  - $p^{j,\beta}[n + 1 - a_j + 1, n + 1]$ and $K^{i,\beta}[n + 1 - a_j + 1, n + 1]$ in $V(B^{j,\beta})$.

  Note that v_e is adjacent to vertices in $K^{i,\alpha}[a_i] \cup K^{j,\alpha}[a_j]$ but not to any vertex in $K^{i,\beta}[n + 1 - a_i] \cup K^{j,\beta}[n + 1 - a_j]$.
- The reduction introduces a central clique K of size $2M^2$ and makes it complete to $\{p_{n+1}^{i,\alpha}, p_{n+1}^{i,\beta} \mid i \in [q]\}$ and V_E where $V_E = \{v_e \mid e \in E(G)\}$ is the set of edge-vertices. This completes the construction of H.
- The reduction further defines

$$P \coloneqq \{(p_a^{i,\alpha}, p_{n+2-a}^{i,\beta}) \mid a \in [n] \text{ and } i \in [q]\}, \qquad \text{and} \qquad k \coloneqq q(n+1)M + |E(G)| - q(q-1)/2.$$

The reduction returns (H, P, k) as the instance of MultiCut with UnDel Term. This completes the reduction. It is easy to see that H is chordal and has leafage at most $2q$. See Figure 2 for a tree representation of H.

Intuition. We first provide the intuition behind the reduction. Recall that the reduction uses $p_a^{i,\alpha}$, $p_{n+1-a}^{i,\beta}$, $K_a^{i,\alpha}$, and $K_{n+1-a}^{i,\beta}$ to encode vertex v_a^i where $i \in [q]$ and $a \in [n]$. Hence, for $a, b \in [n]$, if $a + b = n + 1$, then $p_a^{i,\alpha}$ and $p_b^{i,\beta}$ correspond to the same vertex. Note that the

pairs in P *do not* correspond to the vertices associated with v_a^i. Rather, $p_{a+1}^{i,\alpha}$ is paired with $p_{n+1-a}^{i,\beta}$. Conversely, for $a, b \in [n]$, if $a + b = n + 2$, then $(p_a^{i,\alpha}, p_b^{i,\beta}) \in P$. By the construction of H and P, for a P-multicut S of H, if there is a path from $p_a^{i,\alpha}$ to $p_b^{i,\beta}$ in $H - S$, then $a + b \geq n + 3$.

Now, consider the terminal pairs $(p_1^{i,\alpha}, p_{n+1}^{i,\beta})$ in P for some $i \in [q]$. Because of the size constraints, S cannot contain all the vertices of the central clique K. Since S cannot contain a terminal, it needs to include one clique from $B^{i,\alpha}$. Let $a_i \in [n]$ be the largest index such that $K_{a_i}^{i,\alpha} \subseteq S$. Using similar arguments, there must also exist $b_i \in [n]$ such that $K_{b_i}^{i,\beta} \subseteq S$ and b_i is largest such index. By definition of a_i, b_i and construction of H, there is a path from $p_{a_i+1}^{i,\alpha}$ to $p_{b_i+1}^{i,\beta}$ in $H - S$. The discussion in the previous paragraph implies that $a_i + 1 + b_i + 1 \geq n + 3$, i.e., $a_i + b_i \geq n + 1$. However, by definition of the solution size k and the size of the cliques, we have $a_i + b_i \leq n + 1$. Hence, the structure of the auxiliary graphs and the terminal pairs ensure that the selected cliques in $S \cap V(B^{i,\alpha})$ and $S \cap V(B^{i,\beta})$ encode selecting a vertex in V_i in G.

Suppose that $\{v_{a_1}^1, v_{a_2}^2, \ldots, v_{a_q}^q\}$ are the vertices in G that are selected by S. Recall that V_E is the collection of edge-vertices in H. Considering the remaining budget, a solution S can include at most $|E(G)| - q(q-1)/2$ many vertices in V_E. We argue that $q(q-1)/2$ edges in G corresponding to vertices in $V_E \setminus S$ should have their endpoints in $\{v_{a_1}^1, v_{a_2}^2, \ldots, v_{a_q}^q\}$ as otherwise some terminal pair is connected in $H - S$. Hence, a P-multicut S of H corresponds to a multicolored clique in G. We give the formal proof in the full version [23].

Finally, it is known that, assuming the ETH, there is no algorithm that can solve Multicolored Clique on instance $(G, q, (V_1, V_2, \ldots V_q))$ in time $f(q) \cdot |V(G)|^{o(q)}$ for any computable function f (see, e.g., [17, Corollary 14.23]). Thus, together with the fact that the reduction takes polynomial time in the size of the input, the proof of correctness, and arguments that are standard for parameter preserving reductions, we conclude that the following holds.

▶ **Lemma 17.** Multicut with Undeletable Terminals *on chordal graphs is* W[1]-hard *when parameterized by leafage ℓ and assuming the* ETH*, does not admit an algorithm running in time $f(\ell) \cdot n^{o(\ell)}$ for any computable function f.*

We defer the XP-algorithm for Multicut with UnDel Term on chordal graphs to the full version of this paper [23]. Together with Lemma 17 this proves Theorem 2.

5 Multiway Cut with Undeletable Terminals on Chordal Graphs

In this section, we consider the Multiway Cut with Undeletable Terminals problem formally defined below. Given a graph G and a set $P \subseteq V(G)$, a set $S \subseteq V(G) \setminus P$ is a called a *P-multiway-cut* in G if $G - S$ has no (p, p')-path for any two distinct $p, p' \in P$.

Multiway Cut with Undeletable Terminals (MWC)
Input: An undirected graph G and a set $P \subseteq V(G)$ of terminals.
Question: Find the size of a minimum P-multiway-cut in G.

The aim of this section is to prove Theorem 3 which states that Multiway Cut with Undeletable Terminals can be solved in $n^{\mathcal{O}(1)}$-time on chordal graphs. Before turning to the proof, we first start with a few definitions. Let $(T, \mathcal{M})$ a tree representation of a chordal graph G where T is rooted at an arbitrary node $\mathsf{r} \in V(T)$. Given a subtree T' of T and a set $Q \subseteq V(G)$, we let $Q_{|T'} \subseteq Q$ be the set of vertices $x \in Q$ such that $\mathcal{M}(x) \subseteq V(T')$. Now let

$Q \subseteq V(G)$ be an independent set of G such that for every leaf η of T, $\mathtt{ver}(\eta) \cap Q \neq \emptyset$. Then the *truncated tree w.r.t.* Q is the tree T_Q^{trunc} obtained from T as follows. Let $\{\eta_1, \ldots, \eta_q\}$ be the set of leaves of T. For each $i \in [q]$, let $Q_i \subseteq Q \setminus \mathtt{ver}(\mathsf{r})$ be the set of vertices $p \in Q \setminus \mathtt{ver}(\mathsf{r})$ such that $\mathtt{top}_{\mathcal{M}}(p)$ is on the (η_i, r)-path in T, and let $p_i \in Q_i$ be the vertex of Q_i such that $\mathtt{top}_{\mathcal{M}}(p_i)$ is closest to r. Then T_Q^{trunc} is obtained from T by deleting the subtrees rooted at the children of the nodes in $\{\mathtt{top}_{\mathcal{M}}(p_i) \mid i \in [q]\}$. Note that, by construction, the set of leaves of T_Q^{trunc} is $\{\mathtt{top}_{\mathcal{M}}(p_i) \mid i \in [q]\}$ and that, apart from the vertices in $\{p_i \mid i \in [q]\}$, there is at most one other vertex in Q whose model intersects $V(T_Q^{\mathsf{trunc}})$, namely the potential vertex in $Q \cap \mathtt{ver}(\mathsf{r})$ (note that if such a vertex exists, its model is in fact fully contained in T_Q^{trunc}). Finally, given a set $P \subseteq V(G)$, a P-multiway-cut X in G is said to *destroy* an edge $e \in E(T)$ if $\mathtt{ver}(e) \subseteq X$.

We now turn to the proof of Theorem 3. Throughout the remaining of this section, we let (G, P) be an instance of MWC, where G is a n-vertex chordal graph, and further let $(T, \mathcal{M})$ be a tree representation of G. First, we may assume that P is an independent set: indeed, if there exist $p, p' \in P$ such that $pp' \in E(G)$, then (G, P) is a No-instance. Furthermore, if a vertex $v \in V(G)$ does not belong to any (p, p')-path in G, where $p, p' \in P$, then it can be safely deleted as no minimal P-multiway-cut in G may contain v. Hence, we assume that every vertex in G participates in some (p, p')-path where $p, p' \in P$; in particular, we may assume that for every leaf η of T, $\mathtt{ver}(\eta) \cap P \neq \emptyset$. Note that, consequently, for every internal node $\alpha \in V(T)$, the truncation of T_α w.r.t. $P_{|T_\alpha}$ exists.

Now let T_0 be the tree obtained by adding a new node r_0 and connecting it to an arbitrary node $\mathsf{r} \in V(T)$. Observe that $(T_0, \mathcal{M})$ is also a tree representation of G. In the following, we root T_0 at r_0. To prove Theorem 3, we design a dynamic program that computes, in a bottom-up traversal of T_0, the entries of a table $\mathtt{A}$ whose content is defined as follows. The table $\mathtt{A}$ is indexed over the edges of $E(T_0)$. For each node $\alpha \in V(T)$, $\mathtt{A}[\alpha\, \mathtt{parent}_{T_0}(\alpha)]$ stores the size of a minimum $P_{|T_\alpha}$-multiway-cut in $G_{|T_\alpha}$. The size of a minimum P-multiway-cut in G may then be found in $\mathtt{A}[\mathsf{rr}_0]$. We describe below how to compute the entries of $\mathtt{A}$.

Update Procedure. For every leaf η of T, we set $\mathtt{A}[\eta\, \mathtt{parent}_{T_0}(\eta)] = 0$. Consider now an internal node α of T. We show how to compute $\mathtt{A}[\alpha\, \mathtt{parent}_{T_0}(\alpha)]$ assuming that for every edge $e \in E(T_\alpha)$, the entry $\mathtt{A}[e]$ is correctly filled.

Let $\widetilde{T}$ be the truncation of T_α w.r.t. $P_{|T_\alpha}$ and let $\widetilde{G} = G_{|\widetilde{T}}$. Denote by $\eta_1, \ldots, \eta_q$ the leaves of $\widetilde{T}$. Recall that, by construction, for every $i \in [q]$, there exists $p_i \in P_{|T_\alpha}$ such that $\eta_i = \mathtt{top}_{\mathcal{M}}(p_i)$: we let $\widetilde{P} = \{p_i \mid i \in [q]\}$. Furthermore, it may be that $P_{|T_\alpha} \cap \mathtt{ver}(\mathsf{r})$ is nonempty: we let $\widetilde{P}_{\mathsf{r}} = P_{|T_\alpha} \cap \mathtt{ver}(\mathsf{r})$. Note that $|\widetilde{P}_{\mathsf{r}}| \leq 1$: if $\widetilde{P}_{\mathsf{r}} \neq \emptyset$ then we refer to the terminal in $\widetilde{P}_{\mathsf{r}}$ as the *root terminal*. Observe that $V(\widetilde{G}) \cap P_{|T_\alpha} = V(\widetilde{G}) \cap P = \widetilde{P} \cup \widetilde{P}_{\mathsf{r}}$ by construction. To compute $\mathtt{A}[\alpha\, \mathtt{parent}_{T_0}(\alpha)]$, we distinguish two cases:

(1) if $\widetilde{P}_{\mathsf{r}} \neq \emptyset$ then we construct a unique instance $(H_0, \mathtt{s}, \mathtt{t}, \mathtt{wt}_0)$ of (s,t)-Cut;

(2) otherwise, for every $i \in [0, q]$, we construct an instance $(H_i, \mathtt{s}, \mathtt{t}, \mathtt{wt}_i)$ of (s,t)-Cut.

We describe below how such instances are constructed. First, recall that an instance of the (s,t)-Cut problem consists of a digraph D, vertices $s, t \in V(D)$, a weight function $\mathtt{wt} : E(D) \to \mathbb{N} \cup \{\infty\}$, and the goal is to find a set $X \subseteq E(D)$ such that $D - X$ has no (s,t)-path and $\mathtt{wt}(X)$ is minimum with this property, where $\mathtt{wt}(X) = \sum_{u \in X} \mathtt{wt}(u)$.

Construction of the (s,t)-Cut Instances. For every $i \in [q]$, let us denote by $\widetilde{P}_i = \widetilde{P} \setminus \{p_i\}$ and let $\widetilde{P}_0 = \widetilde{P}$. Consider $i \in [0, q]$. Before turning to the formal construction of the instance $(H_i, \mathtt{s}, \mathtt{t}, \mathtt{wt}_i)$, let us first give an intuitive idea of the construction. The digraph H_i

is obtained from $\widetilde{T}$ by orienting all edges of $\widetilde{T}$ towards its root $\tilde{\mathtt{r}} = \alpha$ and further adding vertices and weighted arcs to encode the graph $G_{|T_\alpha}$. The arcs in H_i corresponding to the edges of $\widetilde{T}$ are called the *tree arcs* and the nodes in H_i corresponding to the nodes of $\widetilde{T}$ are called the *tree nodes*. The idea is that we separate, for each terminal $p \in \widetilde{P}_i$, the node $\mathtt{top}_{\mathcal{M}}(p)$ from the root $\tilde{\mathtt{r}}$. To achieve this, we add a *source* node $\mathtt{s}$ and *source* arcs from $\mathtt{s}$ to $\mathtt{top}_{\mathcal{M}}(p)$ (of infinite weight) and look for an $(\mathtt{s}, \tilde{\mathtt{r}})$-cut in H_i. Since the edges of T can presumably not be independently destroyed in a P-multiway-cut, we need some additional vertices to encode these dependencies. For each vertex $v \in V(\widetilde{G}) \setminus \widetilde{P}_i$, we introduce a node $\gamma(v)$ in H_i which is reachable via *connection* arcs (with infinite weight) from all the tree nodes that are contained in the model of v. This node $\gamma(v)$ is further connected via a *sink* arc (of weight one) to $\mathtt{top}_{\mathcal{M}}(v)$ which ensures that if we want to cut a tree arc, we also have to cut all the sink arcs associated to vertices containing the corresponding edge in their model. The index i is then used to specify which root-to-leaf path of $\widetilde{T}$ is uncut: if $i = 0$ then every such path is cut, otherwise the $(\eta_i, \tilde{\mathtt{r}})$-path is uncut. To encode the rest of the solution, we associate with each tree arc (β, δ) a weight $\mathtt{wt}_i((\beta, \delta))$ corresponding to the size of a minimum $P_{|\beta}$-multiway-cut in $G_{|\beta}$.

We proceed with the formal construction of H_i. The vertex set of H_i is $V(H_i) = V(\widetilde{T}) \uplus \{\mathtt{s}\} \uplus \{\Gamma\}$ where $\Gamma = \{\gamma(v) \mid v \in V(\widetilde{G}) \setminus \widetilde{P}\}$, that is, Γ contains a node of every non-terminal vertex in $\widetilde{G}$. For every $z \in \Gamma$, we denote by $\gamma^{-1}(z)$ the corresponding vertex in $V(\widetilde{G}) \setminus \widetilde{P}$. The arc set of H_i is partitioned into four sets:

- the set $E_{\widetilde{T}}$ of *tree arcs* containing all the edges of $\widetilde{T}$ oriented towards the root $\tilde{\mathtt{r}}$,
- the set $E^i_{\mathtt{source}} = \{(\mathtt{s}, \mathtt{top}_{\mathcal{M}}(p)) \mid p \in \widetilde{P}_i\}$ of *source* arcs,
- the set $E_{\mathtt{conn}} = \{(\alpha, \gamma(v)) \mid \gamma(v) \in \Gamma, \alpha \in \mathcal{M}(v) \cap V(\widetilde{T})\}$ of *connection* arcs and
- the set $E_{\mathtt{sink}} = \{(\gamma(v), \mathtt{top}_{\mathcal{M}}(v)) \mid v \in V(\widetilde{G}) \setminus \widetilde{P}\}$ of *sink* arcs.

Furthermore, if $\widetilde{P}_{\mathtt{r}} \neq \emptyset$, then we let $E_{\mathtt{rterm}} \subseteq E_{\widetilde{T}}$ be the set of tree arcs $(\beta, \delta) \in E_{\widetilde{T}}$ such that the edge $\beta\delta$ is contained in the model of the root terminal; otherwise, we let $E_{\mathtt{rterm}} = \emptyset$. The weight function $\mathtt{wt}_i : E(H_i) \to \mathbb{N} \cup \{\infty\}$ is defined as follows. For every $j \in [q]$, let ρ_j be the path in $\widetilde{T}$ from η_j to $\tilde{\mathtt{r}}$ and let $\overrightarrow{\rho_j}$ be the corresponding directed path in H_i (that is, $\overrightarrow{\rho_j}$ is the path in H_i from η_j to $\tilde{\mathtt{r}}$ consisting only of tree arcs). Then for every arc e of H_i,

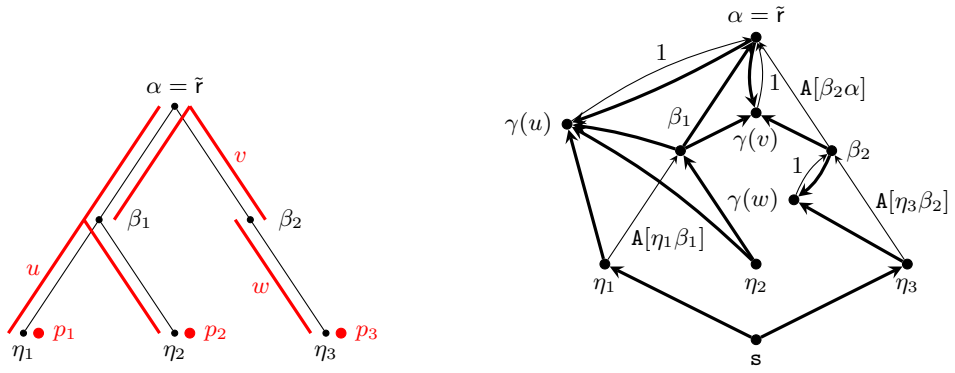
$$\mathtt{wt}_i(e) = \begin{cases} \mathtt{A}[e] & \text{if } i = 0 \text{ and } e \in E_{\widetilde{T}} \setminus E_{\mathtt{rterm}} \\ \mathtt{A}[e] & \text{if } i \neq 0,\ e \in E_{\widetilde{T}} \text{ and } e \text{ does not belong to the path } \overrightarrow{\rho_i} \\ 1 & \text{if } e \in E_{\mathtt{sink}} \\ \infty & \text{otherwise.} \end{cases}$$

Note, in particular, that every arc in $E_{\mathtt{rterm}}$ (if any) has infinite weight. Similarly, if $i \neq 0$, then every arc of the path $\overrightarrow{\rho_i}$ has infinite weight. This completes the construction of the instance $(H_i, \mathtt{s}, \mathtt{t} = \tilde{\mathtt{r}}, \mathtt{wt}_i)$ (see Figure 3). It is easy to see that such an instance can be constructed in $\mathcal{O}(n^2)$-time.

Now let X_0 be an $(\mathtt{s}, \tilde{\mathtt{r}})$-cut in H_0 such that $\mathtt{wt}_0(X_0)$ is minimum; and if $\widetilde{P}_{\mathtt{r}} = \emptyset$, then for every $i \in [q]$, further let X_i be an $(\mathtt{s}, \tilde{\mathtt{r}})$-cut in H_i such that $\mathtt{wt}_i(X_i)$ is minimum. For each $i \in [q]$, let us denote by $\mathtt{cost}_i = \mathtt{A}[\eta_i\, \mathtt{parent}_{T_0}(\eta_i)]$ and let $\mathtt{cost}_0 = 0$. Then we set

$$\mathtt{A}[\alpha\, \mathtt{parent}_{T_0}(\alpha)] = \begin{cases} |X_0| & \text{if } \widetilde{P}_{\mathtt{r}} \neq \emptyset \\ \min_{i \in [0,q]} \{|X_i| + \mathtt{cost}_i\} & \text{otherwise} \end{cases}$$

In the following, for convenience, we let $I = [0, q]$ if $\widetilde{P}_{\mathtt{r}} = \emptyset$, and $I = \{0\}$ otherwise. We prove in Appendix B that the entry $\mathtt{A}[\alpha\, \mathtt{parent}_{T_0}(\alpha)]$ is updated correctly. To this end, we show that $G_{|T_\alpha}$ has a $P_{|T_\alpha}$-multiway-cut of size at most k if and only if there exists $i \in I$ such that H_i has an $(\mathtt{s}, \tilde{\mathtt{r}})$-cut of weight at most $k - \mathtt{cost}_i$ w.r.t. $\mathtt{wt}_i$.



(a) The tree representation $(\widetilde{T}, \mathcal{M}_{|V(\widetilde{G})})$ of $\widetilde{G}$ where $V(\widetilde{G}) = \{p_1, p_2, p_3, u, v, w\}$ and $\widetilde{P}_{\mathtt{r}} = \emptyset$.

(b) The instance $(H_2, \mathtt{s}, \tilde{\mathtt{r}}, \mathtt{wt}_2)$ (thick arcs have infinite weight).

Figure 3 An illustration of the construction of the (s,t)-Cut instances.

From the correctness (cf. Lemmas 18 and 21) we conclude that $\mathtt{A}[\alpha\, \mathtt{parent}_{T_0}(\alpha)]$ indeed stores the size of a minimum $P_{|T_\alpha}$-multiway-cut in $G_{|T_\alpha}$. Since the construction of each H_i takes polynomial-time, an $(\mathtt{s}, \mathtt{t})$-cut in H_i can be computed in polynomial time (see, for instance, [22]) and the number of H_is is at most n, it takes plynomial-time to update $\mathtt{A}[\alpha\, \mathtt{parent}_{T_0}(\alpha)]$. Finally, since the number of edges of T is linear in n, the overall running time is polynomial in n, which proves Theorem 3. We remark that a more careful analysis of the running time of the algorithm leads to an upper bound of $\mathcal{O}(n^4)$.

6 Conclusion

In this article, we presented improved and new results regarding domination and cut problems on chordal graphs with bounded leafage. We presented an FPT algorithm running in time $2^{\mathcal{O}(\ell)} \cdot n^{\mathcal{O}(1)}$ for the Dominating Set problem on chordal graphs. Regarding cut problems, we proved that MultiCut with Undeletable Terminals on chordal graphs is W[1]-hard when parameterized by the leafage. We also presented a polynomial-time algorithm for Multiway Cut with Undeletable Terminals on chordal graphs. We find it surprising that the complexity of this problem was not known before.

In the case of chordal graphs, we believe the leafage to be a more natural parameter than other popular parameters such as vertex cover, feedback vertex set or treewidth. It would be interesting to examine the structural parameterized complexity of problems such as Longest Cycle, Longest Path, Component Order Connectivity, s-Club Contraction, Independent Set Reconfiguration, Bandwidth, or Cluster Vertex Deletion. These problems are known to be NP-complete on split graphs and admit polynomial-time algorithms on interval graphs. Hence it is plausible that they admit an FPT or XP algorithm on chordal graphs parameterized by the leafage. We believe it is a representative list, though not exhaustive, of problems that exhibit this behavior. In fact, it would be fascinating to find a natural problem that does not exhibit this behavior, i.e., a problem that is NP-complete on interval graphs but admits a polynomial-time algorithm on split graphs.

References

1 Liliana Alcón. On asteroidal sets in chordal graphs. *Discret. Appl. Math.*, 164:482–491, 2014. doi:10.1016/j.dam.2013.04.019.

2 Vikraman Arvind, Roman Nedela, Ilia Ponomarenko, and Peter Zeman. Testing isomorphism of chordal graphs of bounded leafage is fixed-parameter tractable. *CoRR*, abs/2107.10689, 2021. arXiv:2107.10689.

3 Hari Balakrishnan, Anand Rajaraman, and C. Pandu Rangan. Connected domination and steiner set on asteroidal triple-free graphs. In Frank K. H. A. Dehne, Jörg-Rüdiger Sack, Nicola Santoro, and Sue Whitesides, editors, *Algorithms and Data Structures, Third Workshop, WADS '93, Montréal, Canada, August 11-13, 1993, Proceedings*, volume 709 of *Lecture Notes in Computer Science*, pages 131–141. Springer, 1993. doi:10.1007/3-540-57155-8_242.

4 Kathleen D. Barnetson, Andrea C. Burgess, Jessica A. Enright, Jared Howell, David A. Pike, and Brady Ryan. The firebreak problem. *Networks*, 77(3):372–382, 2021. doi:10.1002/net.21975.

5 Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, Yota Otachi, and Florian Sikora. Token sliding on split graphs. *Theory Comput. Syst.*, 65(4):662–686, 2021. doi:10.1007/s00224-020-09967-8.

6 Benjamin Bergougnoux and Mamadou Moustapha Kanté. More applications of the d-neighbor equivalence: Acyclicity and connectivity constraints. *SIAM J. Discret. Math.*, 35(3):1881–1926, 2021. doi:10.1137/20M1350571.

7 Benjamin Bergougnoux, Charis Papadopoulos, and Jan Arne Telle. Node multiway cut and subset feedback vertex set on graphs of bounded mim-width. *Algorithmica*, 84(5):1385–1417, 2022. doi:10.1007/s00453-022-00936-w.

8 Alan A. Bertossi. Dominating sets for split and bipartite graphs. *Inf. Process. Lett.*, 19(1):37–40, 1984. doi:10.1016/0020-0190(84)90126-1.

9 Nicolas Bousquet, Jean Daligault, and Stéphan Thomassé. Multicut is FPT. *SIAM J. Comput.*, 47(1):166–207, 2018. doi:10.1137/140961808.

10 Binh-Minh Bui-Xuan, Jan Arne Telle, and Martin Vatshelle. Fast dynamic programming for locally checkable vertex subset and vertex partitioning problems. *Theor. Comput. Sci.*, 511:66–76, 2013. doi:10.1016/j.tcs.2013.01.009.

11 Peter Buneman. A characterisation of rigid circuit graphs. *Discret. Math.*, 9(3):205–212, 1974. doi:10.1016/0012-365X(74)90002-8.

12 Yixin Cao. Linear recognition of almost interval graphs. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1096–1115. SIAM, 2016. doi:10.1137/1.9781611974331.ch77.

13 Maw-Shang Chang. Efficient algorithms for the domination problems on interval and circular-arc graphs. *SIAM J. Comput.*, 27(6):1671–1694, 1998. doi:10.1137/S0097539792238431.

14 Steven Chaplick and Juraj Stacho. The vertex leafage of chordal graphs. *Discret. Appl. Math.*, 168:14–25, 2014. doi:10.1016/j.dam.2012.12.006.

15 Rajesh Hemant Chitnis, Marek Cygan, Mohammad Taghi Hajiaghayi, and Dániel Marx. Directed subset feedback vertex set is fixed-parameter tractable. *ACM Trans. Algorithms*, 11(4):28:1–28:28, 2015. doi:10.1145/2700209.

16 Rajesh Hemant Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. *SIAM J. Comput.*, 42(4):1674–1696, 2013. doi:10.1137/12086217X.

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

18 Celina M. H. de Figueiredo, Raul Lopes, Alexsander Andrade de Melo, and Ana Silva. Parameterized algorithms for steiner tree and dominating set: Bounding the leafage by the vertex leafage. In Petra Mutzel, Md. Saidur Rahman, and Slamin, editors, *WALCOM: Algorithms and Computation - 16th International Conference and Workshops, WALCOM 2022, Jember, Indonesia, March 24-26, 2022, Proceedings*, volume 13174 of *Lecture Notes in Computer Science*, pages 251–262. Springer, 2022. doi:10.1007/978-3-030-96731-4_21.

19 Pål Grønås Drange, Markus S. Dregi, and Pim van 't Hof. On the computational complexity of vertex integrity and component order connectivity. *Algorithmica*, 76(4):1181–1202, 2016. doi:10.1007/s00453-016-0127-x.

20 Fedor V. Fomin, Petr A. Golovach, and Jean-Florent Raymond. On the Tractability of Optimization Problems on H-graphs. *Algorithmica*, 82(9):2432–2473, 2020. doi:10.1007/s00453-020-00692-9.

21 Fedor V. Fomin, Pinar Heggernes, Dieter Kratsch, Charis Papadopoulos, and Yngve Villanger. Enumerating minimal subset feedback vertex sets. *Algorithmica*, 69(1):216–231, 2014. doi:10.1007/s00453-012-9731-6.

22 L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.

23 Esther Galby, Dániel Marx, Philipp Schepper, Roohani Sharma, and Prafullkumar Tale. Domination and cut problems on chordal graphs with bounded leafage. *CoRR*, abs/2208.02850, 2022. doi:10.48550/arXiv.2208.02850.

24 Fanica Gavril. The intersection graphs of subtrees in tree are exactly the chordal graphs. *Combinatorica*, 1974.

25 P. C. Gilmore and A. J. Hoffman. A characterization of comparability graphs and of interval graphs. *Canadian Journal of Mathematics*, 16:539–548, 1964. doi:10.4153/CJM-1964-055-5.

26 Petr A. Golovach, Pinar Heggernes, Pim van 't Hof, and Christophe Paul. Hadwiger number of graphs with small chordality. *SIAM J. Discret. Math.*, 29(3):1427–1451, 2015. doi:10.1137/140975279.

27 Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*. Elsevier, 2004.

28 Jiong Guo, Falk Hüffner, Erhan Kenar, Rolf Niedermeier, and Johannes Uhlmann. Complexity and exact algorithms for vertex multicut in interval and bounded treewidth graphs. *Eur. J. Oper. Res.*, 186(2):542–553, 2008. doi:10.1016/j.ejor.2007.02.014.

29 Michel Habib and Juraj Stacho. Polynomial-time algorithm for the leafage of chordal graphs. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 290–300. Springer, 2009. doi:10.1007/978-3-642-04128-0_27.

30 Michel Habib and Juraj Stacho. Reduced clique graphs of chordal graphs. *Eur. J. Comb.*, 33(5):712–735, 2012. doi:10.1016/j.ejc.2011.09.031.

31 Winfried Hochstättler, Johann L. Hurink, Bodo Manthey, Daniël Paulusma, Britta Peis, and Georg Still. In memoriam walter kern. *Discret. Appl. Math.*, 303:2–3, 2021. doi:10.1016/j.dam.2021.08.034.

32 Kyriaki Ioannidou, George B. Mertzios, and Stavros D. Nikolopoulos. The longest path problem has a polynomial solution on interval graphs. *Algorithmica*, 61(2):320–341, 2011. doi:10.1007/s00453-010-9411-3.

33 Dong Yeap Kang, O-joung Kwon, Torstein J. F. Strømme, and Jan Arne Telle. A width parameter useful for chordal and co-comparability graphs. *Theor. Comput. Sci.*, 704:1–17, 2017. doi:10.1016/j.tcs.2017.09.006.

34 J. Mark Keil. Finding hamiltonian circuits in interval graphs. *Inf. Process. Lett.*, 20(4):201–206, 1985. doi:10.1016/0020-0190(85)90050-X.

35 Athanasios L. Konstantinidis and Charis Papadopoulos. Cluster deletion on interval graphs and split related graphs. *Algorithmica*, 83(7):2018–2046, 2021. doi:10.1007/s00453-021-00817-8.

36 Dieter Kratsch. Finding the minimum bandwidth of an interval graphs. *Inf. Comput.*, 74(2):140–158, 1987. doi:10.1016/0890-5401(87)90028-9.

37 Dieter Kratsch and Lorna Stewart. Approximating bandwidth by mixing layouts of interval graphs. *SIAM J. Discret. Math.*, 15(4):435–449, 2002. doi:10.1137/S0895480199359624.

38 C. Lekkeikerker and J. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962. URL: http://eudml.org/doc/213681.

39 In-Jen Lin, Terry A. McKee, and Douglas B. West. The leafage of a chordal graph. *Discuss. Math. Graph Theory*, 18(1):23–48, 1998. doi:10.7151/dmgt.1061.

40 George S. Lueker and Kellogg S. Booth. A linear time algorithm for deciding interval graph isomorphism. *J. ACM*, 26(2):183–195, 1979. doi:10.1145/322123.322125.

41 Dániel Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006. doi:10.1016/j.tcs.2005.10.007.

42 Dániel Marx and Igor Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. *SIAM J. Comput.*, 43(2):355–388, 2014. doi:10.1137/110855247.

43 Pranabendu Misra, Fahad Panolan, Ashutosh Rai, Saket Saurabh, and Roohani Sharma. Quick separation in chordal and split graphs. In Javier Esparza and Daniel Král', editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 70:1–70:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.70.

44 Charis Papadopoulos. Restricted vertex multicut on permutation graphs. *Discret. Appl. Math.*, 160(12):1791–1797, 2012. doi:10.1016/j.dam.2012.03.021.

45 Charis Papadopoulos and Spyridon Tzimas. Polynomial-time algorithms for the subset feedback vertex set problem on interval graphs and permutation graphs. *Discret. Appl. Math.*, 258:204–221, 2019. doi:10.1016/j.dam.2018.11.017.

46 Charis Papadopoulos and Spyridon Tzimas. Computing a minimum subset feedback vertex set on chordal graphs parameterized by leafage. In Cristina Bazgan and Henning Fernau, editors, *Combinatorial Algorithms - 33rd International Workshop, IWOCA 2022, Trier, Germany, June 7-9, 2022, Proceedings*, volume 13270 of *Lecture Notes in Computer Science*, pages 466–479. Springer, 2022. doi:10.1007/978-3-031-06678-8_34.

47 James Richard Walter. *Representations of rigid cycle graphs*. Wayne State University, 1972.

48 Kevin White, Martin Farber, and William R. Pulleyblank. Steiner trees, connected domination and strongly chordal graphs. *Networks*, 15(1):109–124, 1985. doi:10.1002/net.3230150109.

A Proof of Lemma 7: Constructing Rest-Red-Blue-DomSet Instances

Let G be a chordal graph and let $(T, \mathcal{M})$ be a tree representation of G. We define the following functions.

- Let $f_T(G)$ denote the number of branching nodes $\gamma \in V(T)$ such that there exist both a red vertex and a blue vertex whose models contain γ.
- Let $f_r(G)$ denote the number of pairs of consecutive branching nodes α, β in T (that is, no node on the unique path in T from α to β is a branching node) such that there is red vertex whose model contains both α and β.
- Similarly, let $f_b(G)$ denote the number of pairs of consecutive branching nodes α, β in T such that there is blue vertex whose model contains both α and β.

We further define $\mu(G) := \mathtt{lf}(G) + 2 \cdot (f_T(G) + f_r(G) + f_b(G))$. Note that, by definition, $\mu(G) \geq \mathtt{lf}(G)$. We design a polynomial-time branching algorithm whose measure μ decreases in each branch. We first show that if $\mu(G) = \mathtt{lf}(G)$ then $(G, (R, B), k)$ is in fact an instance of Rest-Red-Blue-DomSet and then show how the branching algorithm proceeds.

Assume therefore that $\mu(G) = \mathtt{lf}(G)$. Then $f_T(G) = f_r(G) = f_b(G) = 0$ by definition. However, when $f_T(G) = 0$, then, by definition, for every branching node $\gamma \in V(T)$, all the vertices containing γ in their model are either red or blue; and when $f_r(G) = f_b(G) = 0$

then, considering the fact that every model is a subtree in T, for every vertex in G, its model contains at most one branching node in T. Therefore if $\mu(G) = \texttt{lf}(G)$, then $(G, (R, B), k)$ is also an instance of Rest-Red-Blue-DomSet.

Now assume that $\mu(G) > \texttt{lf}(G)$. Then $f_T(G) + f_r(G) + f_b(G) > 0$. We consider the following three exhaustive cases.

Case-I. $f_T(G) > 0$. Let γ be a branching node in T such that there is both a red-vertex and a blue-vertex whose models contain γ. Suppose that $\mathcal{I}$ is a Yes-instance of Red-Blue-DomSet and let D be a solution. Consider first the case where D includes a red vertex whose model contains γ. In this case, we return the instance $\mathcal{I}_1 = (G_1, (R_1, B_1), k)$ which is obtained as follows.

- Initialize $V(G_1) = V(G)$, $R_1 = R$, $B_1 = B$.
- Let T_1 be the tree obtained from T by adding a node δ and making it adjacent to γ only. Note that $V(T_1) \setminus \{\delta\} \subseteq V(T)$.
- For every red vertex $v \in V(G_1)$ such that $\gamma \in \mathcal{M}(v)$, add δ to its model, i.e., $\mathcal{M}_1(v) = \mathcal{M}(v) \cup \{\delta\}$.
- For every blue vertex $v \in V(G_1)$ such that $\gamma \in \mathcal{M}(v)$, delete v from $V(G_1)$.
- Add a new blue vertex x to $V(G_1)$ and to B_1 with $\mathcal{M}_1(x) = \{\delta\}$.
- For every (red or blue) vertex $v \in V(G)$ such that $\gamma \notin \mathcal{M}(v)$, define $\mathcal{M}_1(v_1) = \mathcal{M}(v)$.

It is easy to verify that $(T_1, \mathcal{M}_1)$ is a tree representation of G_1 and that T_1 has exactly one more leaf than T, i.e., $\texttt{lf}(G_1) \leq \texttt{lf}(G) + 1$. However, since we have deleted all the blue vertices whose models contained γ, $f_T(G_1) = f_T(G) - 1$. As the other parts of the measure do not change, $\mu(G_1) < \mu(G)$.

In the second case where no vertex in D contains γ in its model, we return the instance $\mathcal{I}_2 = (G_2, (R_2, B), k)$ where G_2, R_2 are obtained from G, R, respectively, by deleting red vertices whose model contains γ. It is easy to verify that $\mu(G_2) < \mu(G)$.

If $\mathcal{I}$ is a Yes-instance, then at least one of $\mathcal{I}_1$ or $\mathcal{I}_2$ is a Yes-instance as these two branches are exhaustive. If $\mathcal{I}_1$ is a Yes-instance, then any optimum solution must include a red γ-vertex because of the newly added vertex x. As $R_2 \subseteq R$, if $\mathcal{I}_2$ is a Yes-instance, then $\mathcal{I}$ is a Yes-instance. Hence, this branching step is correct.

Case-II. $f_T(G) = 0$ *and* $f_r(G) > 0$. Let α, β be two consecutive branching nodes in T such that there is a red vertex whose model contains both α and β. Suppose that $\mathcal{I}$ is a Yes-instance of Red-Blue-DomSet and let D be a solution. Consider the case where D includes a red vertex whose model contains both α and β. In this case, we return the instance $\mathcal{I}_1 = (G_1, (R_1, B_1), k)$ which is obtained as follows.

- Initialize $V(G_1) = R_1 = B_1 = \emptyset$.
- Let T_1 be the tree obtained from T by contracting the unique path $P_{\alpha\beta}$ from α to β in T and let $\gamma_{\alpha\beta}$ be the node resulting from this contraction. Add a node δ to T_1 and make it adjacent to $\gamma_{\alpha\beta}$ only. Note that $V(T_1) \setminus \{\gamma_{\alpha\beta}, \delta\} \subseteq V(T)$.
- For every red vertex $v \in V(G)$ such that $\mathcal{M}(v) \cap V(P_{\alpha\beta}) \neq \emptyset$, add a red vertex v_1 to $V(G_1)$ (and to R_1) with $\mathcal{M}_1(v_1) = (\mathcal{M}(v) \setminus V(P_{\alpha\beta})) \cup \{\gamma_{\alpha\beta}, \delta\}$.
- Add a new blue vertex x to $V(G_1)$ with $\mathcal{M}_1(x) = \{\delta\}$.
- For every (red or blue) vertex $v \in V(G)$ such that $\mathcal{M}(v) \cap V(P_{\alpha\beta}) = \emptyset$, add v_1 to G_1 (and to, respectively, either R_1 or B_1) with $\mathcal{M}_1(v_1) = \mathcal{M}(v)$.

Note that for every blue vertex $v \in V(G)$ such that $\mathcal{M}(G) \cap V(P_{\alpha\beta}) \neq \emptyset$, there is no corresponding blue vertex in G_1. It is easy to verify that $(T_1, \mathcal{M}_1)$ is a tree representation of G_1 and that T_1 has one more leaf than T which implies $\texttt{lf}(G_1) \leq \texttt{lf}(G) + 1$. Since we have contracted the path $P_{\alpha\beta}$ to obtain the node $\gamma_{\alpha\beta}$, $f_r(G_1) < f_r(G)$. As the other parts of the measure do not change, $\mu(G_1) < \mu(G)$.

In the second case where no vertex in D contains both α and β in its model, we return an instance $\mathcal{I}_2 = (G_2, (R_2, B), k)$ where G_2, R_2 are obtained from G, R, respectively, by deleting red vertices whose model contains both α and β. It is easy to verify that $\mu(G_2) < \mu(G)$. We argue as in the previous case for the correctness of this branching steps.

Case-III. *$f_T(G) = 0$ and $f_b(G) > 0$.* Let α, β be two consecutive branching nodes in T such that there is a blue vertex whose model contains both α and β. Note that since $f_T(G) = 0$, for every red vertex $v \in V(G)$ such that $\mathcal{M}(v) \cap V(P_{\alpha\beta}) \neq \emptyset$, in fact $\mathcal{M}(v) \subseteq V(P_{\alpha\beta}) \setminus \{\alpha, \beta\}$. Suppose that $\mathcal{I}$ is a Yes-instance of Red-Blue-DomSet and let D be a solution. Consider first the case where D includes a red vertex whose model is in $V(P_{\alpha\beta}) \setminus \{\alpha, \beta\}$. In this case, we return the instance $\mathcal{I}_1 = (G_1, (R, B_1), k)$ where G_1, B_1 are obtained from G and B as follows.

- Delete all the blue vertices whose model contains both α and β.
- Add a blue vertex x to $V(G_1)$ (and to B_1) with $\mathcal{M}(x) = V(P_{\alpha\beta}) \setminus \{\alpha, \beta\}$.

It is easy to verify that $(T, \mathcal{M})$ is a tree representation of G_1 and $f_b(G_1) < f_b(G)$. As the other parts of the measure do not change, $\mu(G_1) < \mu(G)$.

In the second case where there is no vertex in D whose model is in $V(P_{\alpha\beta}) \setminus \{\alpha, \beta\}$, we consider the following two subcases. If there is a blue vertex v such that $\mathcal{M}(v) \subseteq V(P_{\alpha\beta})$, then we return a trivial No-instance. Otherwise, we return the instance $\mathcal{I}_2 = (G_2, (R_2, B_2), k)$ which is constructed as follows.

- Initialize $V(G_2) = R_2 = B_2 = \emptyset$.
- Let T_2 be the tree obtained from T by contracting the path $P_{\alpha\beta}$ from α to β in T and let $\gamma_{\alpha\beta}$ be the node resulting from this contraction. Note that $V(T_2) \setminus \{\gamma_{\alpha\beta}\} \subseteq V(T)$.
- For every (red or blue) vertex $v \in G$ such that $\mathcal{M}(v) \cap V(P_{\alpha\beta}) = \emptyset$, add a vertex v_2 to G_2 (and to, respectively, either R_2 or B_2) with $\mathcal{M}_2(v_2) = \mathcal{M}(v)$.
- For every blue vertex $v \in V(G)$ such that $\mathcal{M}(v) \cap V(P_{\alpha\beta}) \neq \emptyset$, add a blue vertex v_2 to $V(G_2)$ (and to B_2) with $\mathcal{M}_2(v_2) = (\mathcal{M}(v_2) \setminus V(P_{\alpha\beta})) \cup \{\gamma_{\alpha\beta}\}$.

Note that for any red vertex $v \in V(G)$ such that $\mathcal{M}(v) \subseteq V(P_{\alpha\beta}) \setminus \{\alpha, \beta\}$, there is no corresponding red vertex in G_2. It is easy to verify that $(T_2, \mathcal{M}_2)$ is a tree representation of G_2. Furthermore, the number of leaves of T_2 is the same as T and $f_b(G_2) < f_b(G)$. As the other parts in the measure do not change, $\mu(G_2) < \mu(G)$.

The correctness of this branching step follows from the same arguments as in the previous cases and the fact that in the second case, since there is no red vertex whose model intersects $V(P_{\alpha\beta})$, it is safe to contract that path.

Finishing the Proof. The correctness of the overall algorithm follows from the correctness of branching steps in the above three cases. To bound its running time and the number of instances it outputs, note that $f_T(G) + f_r(G) + f_b(G) \leq 3 \cdot \texttt{lf}(G)$ as these functions either count the number of branching nodes or the unique paths containing exactly two (consecutive) branching nodes. ◀

B Correctness of the Algorithm for Multiway-Cut

▶ **Lemma 18.** *For any $i \in I$, if H_i has an $(\mathtt{s}, \tilde{\mathtt{r}})$-cut Y such that $\mathtt{wt}_i(Y) \leq k - \mathtt{cost}_i$, then $G_{|T_\alpha}$ has a $P_{|T_\alpha}$-multiway-cut of size at most k.*

Proof. Assume that there exists $i \in I$ such that H_i has an $(\mathtt{s}, \tilde{\mathtt{r}})$-cut Y where $\mathtt{wt}_i(Y) \leq k - \mathtt{cost}_i$. For every $j \in [q] \setminus \{i\}$, let A_j be the set of tree arcs on the path $\overrightarrow{\rho_j}$ belonging to Y (recall that $\overrightarrow{\rho_j}$ is the path in H_i from η_j to $\tilde{\mathtt{r}}$ consisting only of tree arcs). Note that since Y is an $(\mathtt{s}, \tilde{\mathtt{r}})$-cut, $A_j \neq \emptyset$ for every $j \in [q] \setminus \{i\}$.

▷ Claim 19. For every terminal $j \in [q] \setminus \{i\}$, there exists an arc $(x, y) \in A_j$ such that for every $z \in N^+_{H_i}(x) \setminus (N^-_{H_i}(x) \cup \{y\})$, the sink arc with tail z belongs to Y.

Proof. Suppose for a contradiction that this does not hold for some index $j \in [q] \setminus \{i\}$, that is, for every arc $(x, y) \in A_j$, there exists $z \in N^+_{H_i}(x) \setminus (N^-_{H_i}(x) \cup \{y\})$ such that the sink arc with tail z does not belong to Y. Let $(x_1, y_1), \ldots, (x_a, y_a)$ be the arcs of A_j ordered according to their order of appearance when traversing the path $\overrightarrow{\rho_j}$. We show that, in this case, there is a path from $\mathtt{s}$ to $\tilde{\mathtt{r}}$ in $H - Y$. For every $b \in [a]$, denote by $Z_b \subseteq N^+_{H_i}(x_b) \setminus (N^-_{H_i}(x_b) \cup \{y_b\})$ the set of vertices z such that the sink arc with tail z does not belong to Y. Let $b_1, \ldots, b_w \in [a]$ be the longest sequence defined as follows:

- $b_1 \in [a]$ is the largest index such that $Z_1 \cap Z_{b_1} \neq \emptyset$ and
- for every $l > 1$, $b_l \in [a]$ is the largest index such that $Z_{b_{l-1}+1} \cap Z_{b_l} \neq \emptyset$.

For every $l \in [w]$, consider a vertex $z_{b_l} \in Z_{j_l}$ and let $h_{b_l} \in N^+_{H_i}(z_{b_l})$ be the head of the sink arc with tail z_{b_l}. Then for every $l \in [w-1]$, h_{b_l} lies on the path $\overrightarrow{\rho_j}[y_{b_l}, x_{b_l+1}]$: indeed, since $z_{b_l} \notin Z_{b_l+1}$ by the choice of b_l, either $z_{b_l} \notin N^+_{H_i}(x_{b_l+1})$ or $z_{b_l} \in N^+_{H_i}(x_{b_l+1}) \cap N^-_H(x_{b_l+1})$; but $z_{b_l} \in N^+_{H_i}(x_{b_l}) \setminus N^-_{H_i}(x_{b_l})$ by construction, and so, h_{b_l} necessarily lies on $\overrightarrow{\rho_j}[y_{b_l}, x_{b_l+1}]$.

Now observe that, by maximality of the sequence, $b_w = a$: indeed, if $b_w < a$ then the sequence could be extended as $Z_{b_w+1} \neq \emptyset$ by assumption. Since $z_{b_w} \notin N^-_{H_i}(x_{b_w})$, this implies, in particular, that h_{b_w} lies on the path $\overrightarrow{\rho_j}[y_{b_w}, \tilde{\mathtt{r}}]$. It follows that

$$\mathtt{s}\overrightarrow{\rho_j}[\eta_j, x_1] z_{b_1} \overrightarrow{\rho_j}[h_{b_1}, x_{b_1+1}] z_{b_2} \ldots z_{b_l} \overrightarrow{\rho_j}[h_{b_l}, x_{b_l+1}] z_{b_{l+1}} \ldots \overrightarrow{\rho_j}[h_{b_{w-1}}, x_{b_{w-1}+1}] z_{b_w} L[h_{b_w}, \tilde{\mathtt{r}}]$$

is a path from $\mathtt{s}$ to $\tilde{\mathtt{r}}$ in $H - Y$, a contradiction which proves our claim. ◁

For every $j \in [q] \setminus \{i\}$, let $e_j = (x_j, y_j) \in A_j$ be the arc closest to $\tilde{\mathtt{r}}$ such that for every $z \in N^+_{H_i}(x_j) \setminus (N^-_{H_i}(x_j) \cup \{y_j\})$, the sink arc with tail z belongs to Y (note that we may have $e_j = e_{j'}$ for two distinct $j, j' \in [q] \setminus \{i\}$). Denote by $E = \{e_j \mid j \in [q] \setminus \{i\}\} \cup \{e^*\}$ where $e^* = (\eta_i, \mathtt{parent}(\eta_i))$. For every $e = (x, y) \in E$, let $\widetilde{P}_e \subseteq \widetilde{P}_i$ be the set of terminals in $\widetilde{P}_i$ which are also terminals in the instance restricted to T_x. Note that $\{P_e \mid e \in E \setminus \{e^*\}\}$ is a partition of $\widetilde{P}_i$: indeed, by construction, every $p \in \widetilde{P}_i$ belongs to at least one such set and if there exist $e, e' \in E \setminus \{e^*\}$ such that $\widetilde{P}_e \cap \widetilde{P}_{e'} \neq \emptyset$, then for any $j \in [q] \setminus \{i\}$ such that $p_j \in P_e \cap P_{e'}$, $e, e' \in A_j$; in particular, both e and e' lie on the path $\overrightarrow{\rho_j}$, a contradiction to the choice of the arc in A_j.

Now for every $e = (x, y) \in E$, let S_e be a minimum $P_{|T_x}$-multiway-cut in $G_{|T_x}$ and denote by $N_e = N^+_{H_i}(x) \setminus (N^-_{H_i}(x) \cup \{y\})$. We define

$$S = S_{e^*} \cup \bigcup_{e \in E \setminus \{e^*\}} S_e \cup \{\gamma^{-1}(z) \mid z \in N_e\}.$$

▷ Claim 20. S is a $P_{|T_\alpha}$-multiway-cut in $G_{|T_\alpha}$.

Proof. Since for every $e = (x, y) \in E$, S_e is a $P_{|T_x}$-multiway-cut in $G_{|T_x}$, it is in fact enough to show that for every $e, e' \in E$, $p \in \widetilde{P}_e$ and $p' \in \widetilde{P}_{e'}$, there is no path from p to p' in $G_{|T_\alpha} - S$.

Consider therefore $j, j' \in [q] \setminus \{i\}$ such that $p_j \in \widetilde{P}_e$ and $p_{j'} \in \widetilde{P}_{e'}$ for two distinct $e, e' \in E$. Since, as shown above, $\{\widetilde{P}_f \mid f \in E \setminus \{e^*\}\}$ is a partition of $\widetilde{P}_i$, $p_{j'} \notin \widetilde{P}_e$ and $p_j \notin \widetilde{P}_{e'}$; in particular, e' does not lie on the path $\overrightarrow{\rho_j}$ and e does not lie on the path $\overrightarrow{\rho_{j'}}$. It follows that any path in $G_{|T_\alpha}$ from p_j to $p_{j'}$ contains at least one vertex x whose model contains the edge corresponding to e; but then, $\gamma(x) \in N_e$ and so, $x \in S$ by construction. Thus, there is no path from p_j to $p_{j'}$ in $G_{|T_\alpha} - S$. ◁

Finally, note that, by construction,

$$\begin{aligned}
|S| &= |S_{e^*}| + \sum_{e \in E \setminus \{e^*\}} |S_e| + \left| \bigcup_{e \in E \setminus \{e^*\}} \{\gamma^{-1}(z) \mid z \in N_e\} \right| \\
&= |S_{e^*}| + \sum_{e \in E \setminus \{e^*\}} \mathtt{wt}_i(e) + \sum_{z \in \bigcup_{e \in E \setminus \{e^*\}} N_e} \mathtt{wt}_i((z, \mathtt{top}_{\mathcal{M}}(\gamma^{-1}(z)))) \\
&\leq \mathtt{cost}_i + \mathtt{wt}_i(Y) \leq k
\end{aligned}$$

which concludes the proof. ◀

▶ **Lemma 21.** *If $G_{|T_\alpha}$ has a $P_{|T_\alpha}$-multiway-cut X of size at most k, then there exists $i \in I$ such that H_i has an $(\mathtt{s}, \tilde{\mathtt{r}})$-cut Y where $\mathtt{wt}_i(Y) \leq k - \mathtt{cost}_i$.*

Proof. Recall that for every $j \in [q]$, ρ_j is the unique $(\eta_j, \tilde{\mathtt{r}})$-path in $\widetilde{T}$. To prove the lemma, we first show the following.

▷ Claim 22. If there exists $i \in [q]$ such that $G_{|T_\alpha}$ has a $P_{|T_\alpha}$-multiway-cut X of size at most k where

(1) X does not destroy any edge of ρ_i and

(2) for every $j \in [q] \setminus \{i\}$, X destroys an edge of ρ_j,

then H_i has an $(\mathtt{s}, \tilde{\mathtt{r}})$-cut Y such that $\mathtt{wt}_i(Y) \leq k - \mathtt{cost}_i$.

Proof. Assume that such an index $i \in [q]$ exists and let X be a $P_{|T_\alpha}$-multiway-cut X of size at most k satisfying item (1) and (2). Note that since X does not destroy any edge of ρ_i, $\widetilde{P}_{\mathtt{r}} = \emptyset$ for, otherwise, p_i and the root terminal would be in the same connected component of $G_{|T_\alpha} - X$ thereby contradicting the fact that X is a $P_{|T_\alpha}$-multiway-cut. For every $j \in [q] \setminus \{i\}$, let $e_j \in E(\widetilde{T})$ be the closest edge to η_j on ρ_j such that $\mathtt{ver}(e_j) \subseteq X$ (note that the edges $e_1, \ldots, e_q$ are not necessarily pairwise distinct). Denote by $E = \{e_j \mid j \in [q] \setminus \{i\}\}$. We construct an $(\mathtt{s}, \tilde{\mathtt{r}})$-cut Y in H_i as follows: Y contains the tree arcs of H_i corresponding to the edges in E and for each $v \in X$ such that $\mathcal{M}(v)$ contains at least one edge of E (that is, $v \in \mathtt{ver}(e)$ for some edge $e \in E$), we include in Y the sink arc $(\gamma(v), \mathtt{top}_{\mathcal{M}}(v))$ of $E(H_i)$. Let us show that Y is indeed an $(\mathtt{s}, \tilde{\mathtt{r}})$-cut in H_i.

For every $j \in [q] \setminus \{i\}$, let $V^j_- \subseteq V(\widetilde{T})$ ($V^j_+ \subseteq V(\widetilde{T})$, respectively) be the set of nodes of the subpath of ρ_j from η_j to the tail of e_j (the head of e_j to $\tilde{\mathtt{r}}$, respectively). We contend that for every $j \in [q] \setminus \{i\}$, there is no (V^j_-, V^j_+)-path in $H_i - Y$. Note that if true, this would prove that Y is indeed an $(\mathtt{s}, \tilde{\mathtt{r}})$-cut in H_i. For the sake of contradiction, suppose that, for some $j \in [q] \setminus \{i\}$, there is a path L in $H_i - Y$ from a vertex $x \in V^j_-$ to a vertex $y \in V^j_+$. Since the tree arc in H_i corresponding e_j belongs to Y, there must exist a vertex $z \in V(L)$ such that $N^-_{H_i}(z) \cap V^j_- \cap V(L) \neq \emptyset$ and $N^+_{H_i}(z) \cap V^j_+ \cap V(L) \neq \emptyset$; in particular, the sink arc e with

tail z must belong to L. By construction of H_i, it must then be that $\mathcal{M}(\gamma^{-1}(z))$ contains the edge e_j, that is, $\gamma^{-1}(z) \in \mathtt{ver}(e_j)$; but then, $\gamma^{-1}(z) \in X$ and so, $e \in Y$ by construction, a contradiction which proves our claim.

Let us finally show that $\mathtt{wt}_i(Y) \leq k - \mathtt{cost}_i$. To this end, for every $e \in E$, let $X_e \subseteq X$ be the restriction of X to T_{t_e} where t_e is the endpoint of e the furthest from $\tilde{\mathsf{r}}$ (note that for any two distinct $e, e' \in E$, $X_e \cap X_{e'} = \emptyset$). Then, for every $e \in E$, X_e is a $P_{|T_{t_e}}$-multiway-cut in $G_{|T_{t_e}}$ and so, $\mathtt{wt}_i(e) \leq |X_e|$. Similarly, the restriction X_i of X to T_{η_i} is a $P_{|T_{\eta_i}}$-multiway-cut in $G_{|T_{\eta_i}}$ and so, $|X_i| \geq \mathtt{cost}_i$ (note that, by construction, $X_i \cap X_e = \emptyset$ for every $e \in E$). Letting $X' = \bigcup_{e \in E} \mathtt{ver}(e)$, it then follows from the definition of Y that

$$\mathtt{wt}_i(Y) = |X'| + \sum_{e \in E} \mathtt{wt}_i(e) \leq |X'| + \sum_{e \in E} |X_e| \leq |X| - |X_i| \leq k - \mathtt{cost}_i$$

as $X' \cap X_i = \emptyset$ and for every $e \in E$, $X' \cap X_e = \emptyset$. ◁

Using similar arguments, we can also prove the following.

▷ Claim 23. If $G_{|T_\alpha}$ has a $P_{|T_\alpha}$-multiway-cut X of size at most k such that for every $i \in [q]$, X destroys an edge of ρ_i, then H_0 has an $(\mathtt{s}, \tilde{\mathsf{r}})$-cut Y such that $\mathtt{wt}_i(Y) \leq k$.

To conclude the proof of Lemma 21, let us show that for any $P_{|T_\alpha}$-multiway-cut S in $G_{|T_\alpha}$, S destroys an edge of every root-to-leaf path of $\widetilde{T}$, except for at most one when $\widetilde{P}_{\mathsf{r}} = \emptyset$. Note that if the claim is true, the lemma would then follow from Claims 22 and 23.

Let S be a $P_{|T_\alpha}$-multiway-cut in $G_{|T_\alpha}$. Observe first that if $\widetilde{P}_{\mathsf{r}} \neq \emptyset$ then for every $i \in [q]$, S must destroy an edge of ρ_i for, otherwise, p_i and the root terminal are in the same connected component of $G_{|T_\alpha} - S$, thereby contradicting the fact that S is a $P_{|T_\alpha}$-multiway-cut. Assume therefore that $\widetilde{P}_{\mathsf{r}} = \emptyset$ and suppose, for the sake of contradiction, that there exist two distinct indices $i, j \in [q]$ such that S destroys no edge of ρ_i and no edge of ρ_j. Then for every edge e of $\rho_i \cup \rho_j$, $\mathtt{ver}(e) \setminus S \neq \emptyset$: for each such edge e, let $\alpha_e \in \mathtt{ver}(e) \setminus S$. It is now not difficult to see that there is a path in $G_{|T_\alpha} - S$ from p_i to p_j using only vertices from $\{\alpha_e \mid e \text{ is an edge of } \rho_i \cup \rho_j\}$, a contradiction to the fact that S be a $P_{|T_\alpha}$-multiway-cut in $G_{|T_\alpha}$. ◀