Computing a Minimum Subset Feedback Vertex Set on Chordal Graphs Parameterized by Leafage†

Charis Papadopoulos*[1] and Spyridon Tzimas[2]

[1]Department of Mathematics, University of Ioannina, Greece, charis@uoi.gr
[2]Department of Mathematics, University of Ioannina, Greece, roytzimas@hotmail.com

Abstract

Chordal graphs are characterized as the intersection graphs of subtrees in a tree and such a representation is known as the tree model. Restricting the characterization results in well-known subclasses of chordal graphs such as interval graphs or split graphs. A typical example of a problem that does not behave computationally the same in all subclasses of chordal graphs is the Subset Feedback Vertex Set (SFVS) problem: given a vertex-weighted graph $G = (V, E)$ and a set $S \subseteq V$, we seek for a vertex set of minimum weight that intersects all cycles containing a vertex of S. SFVS is known to be polynomial-time solvable on interval graphs, whereas SFVS remains NP-complete on split graphs and, consequently, on chordal graphs. Towards a better understanding of the complexity of SFVS on subclasses of chordal graphs, we exploit structural properties of a tree model in order to cope with the hardness of SFVS. Here we consider the *leafage*, which measures the minimum number of leaves in a tree model. We show that SFVS can be solved in polynomial time for every chordal graph with bounded leafage. In particular, given a chordal graph on n vertices with leafage ℓ, we provide an algorithm for solving SFVS with running time $n^{O(\ell)}$, thus improving upon $n^{O(\ell^2)}$, which is the running time of an approach that utilizes the previously known algorithm for graphs with bounded mim-width. We complement our result by showing that SFVS is W[1]-hard parameterized by ℓ. Pushing further our positive result, it is natural to also consider the *vertex leafage*, which measures the minimum upper bound on the number of leaves of every subtree in a tree model. However, we show that it is unlikely to obtain a similar result, as we prove that SFVS remains NP-complete on undirected path graphs, i.e., chordal graphs having vertex leafage at most two. Lastly, we provide a polynomial-time algorithm for solving SFVS on rooted path graphs, a proper subclass of undirected path graphs and graphs with mim-width one, which is faster than the approach of constructing a graph decomposition of mim-width one and applying the previously known algorithm for graphs with bounded mim-width.

1 Introduction

Several fundamental optimization problems are known to be intractable on chordal graphs, however they admit polynomial-time algorithms when restricted to a proper subclass of chordal graphs such as interval graphs. Typical examples of problems that exhibit this behavior are

†A preliminary version of this article appeared in the Proceedings of the 33rd International Workshop on Combinatorial Algorithms (IWOCA 2022), volume 13270 of LNCS, pages 466–479. This full version contains all missing proofs and detailed analysis of the claimed algorithms and the hardness results.
Research supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "First Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research grant", Project FANTA (eFficient Algorithms for NeTwork Analysis), number HFRI-FM17-431.

*Corresponding author

domination and induced path problems [2, 5, 13, 25, 27, 34]. Towards a better understanding of why many intractable problems on chordal graphs admit polynomial-time algorithms on interval graphs, we consider the algorithmic usage of a structural parameter named leafage. Leafage, introduced by Lin et al. [32], is a graph parameter that captures how close a chordal graph is to being an interval graph. As it concerns chordal graphs, leafage essentially measures the smallest number of leaves in a clique tree, an intersection representation of the given graph [21]. Here we are concerned with the Subset Feedback Vertex Set problem, SFVS for short: given a vertex-weighted graph and a set S of its vertices, compute a vertex set of minimum weight that intersects all cycles containing a vertex of S. Although Subset Feedback Vertex Set does not fall under the themes of domination or induced path problems, it is known to be NP-complete on chordal graphs [18], whereas it becomes polynomial-time solvable on interval graphs [36]. Thus our research study investigates to what extent the structure of the underlying tree representation influences the computational complexity of Subset Feedback Vertex Set.

An interesting remark concerning Subset Feedback Vertex Set is the fact that the computational complexities of its unweighted and weighted variants do not align on hereditary graph classes. For example, Subset Feedback Vertex Set is NP-complete on H-free graphs for some fixed graphs H, while its unweighted variant admits a polynomial-time algorithm on the same class of graphs [8, 37]. Subset Feedback Vertex Set remains NP-complete on bipartite graphs [41] and planar graphs [20], as a generalization of Feedback Vertex Set. Notable differences between the two latter problems regarding their complexity status concern the classes of split graphs and $4P_1$-free graphs on which Subset Feedback Vertex Set is NP-complete [18, 37], whereas the Feedback Vertex Set problem is polynomial-time solvable [8, 12, 40]. Inspired by the NP-completeness on chordal graphs, Subset Feedback Vertex Set restricted on (subclasses of) chordal graphs has attracted several researchers to obtain fast, still exponential-time, algorithms [23, 38].

On the positive side, Subset Feedback Vertex Set can be solved in polynomial time when restricted on a number of other graph classes [7, 8, 36, 37]. Cygan et al. [15] and Kawarabayashi and Kobayashi [31] independently showed that Subset Feedback Vertex Set is fixed-parameter tractable (FPT) parameterized by the solution size, while Hols and Kratsch [26] provided a randomized polynomial kernel for the problem. Related to the structural parameter mim-width, Bergougnoux et al. [1] recently proposed an $n^{O(w^2)}$-time algorithm that solves Subset Feedback Vertex Set given a decomposition of the input graph of mim-width w. As leaf power graphs are a subclass of chordal graphs that admit a decomposition of mim-width one [28], via the latter algorithm Subset Feedback Vertex Set can be solved in polynomial time on leaf power graphs if an intersection model is given as input. However, to the best of our knowledge, it is not known whether an intersection model of a leaf power graph can be constructed in polynomial time. Moreover, even for graphs of mim-width one that do admit an efficient construction of the corresponding decomposition, the exponent of the running time of the algorithm proposed by Bergougnoux et al. [1] is relatively high.

Habib and Stacho [24] showed that the leafage of a connected chordal graph can be computed in polynomial time. Their described algorithm also constructs a corresponding clique tree with the minimum number of leaves. Regarding other problems that behave well with the leafage, we mention the Minimum Dominating Set problem for which Fomin et al. [19] showed that the problem is FPT parameterized by the leafage of the given graph. Here we show that Subset Feedback Vertex Set is polynomial-time solvable for every chordal graph with bounded leafage. In particular, we provide an algorithm that given a chordal graph and a tree model of it with ℓ leaves solves the problem in $O(n^{2\ell+1})$ time. Thus, by combining the algorithm of Habib and Stacho [24] with our algorithm, we deduce that Subset Feedback Vertex Set is in XP parameterized by the leafage.

One advantage of leafage over mim-width is that we can compute the leafage of a chordal

graph in polynomial time, whereas we do not know how to compute the mim-width of a chordal graph in polynomial time. However we note that a graph of bounded leafage implies a graph of bounded mim-width and, further, a decomposition of bounded mim-width can be computed in polynomial time [19]. This can be seen through the notion of H-graphs. For some fixed graph H, a graph is an H-graph if it is the intersection graph of connected subgraphs of some subdivision of H. The intersection model of subtrees of a tree T having ℓ leaves is a T'-graph where T' is obtained from T by contracting nodes of degree two. Thus the size of T' is at most 2ℓ, since T has ℓ leaves. Moreover, given an H-graph and its intersection model, a (linear) decomposition of mim-width at most $2|E(H)|$ can be computed in polynomial time [19]. Therefore, given a graph of leafage ℓ, there is a polynomial-time algorithm that computes a decomposition of mim-width $O(\ell)$. Combined with the algorithm via mim-width [1], one can solve Subset Feedback Vertex Set in time $n^{O(\ell^2)}$ on graphs having leafage ℓ. Notably, our $n^{O(\ell)}$-time algorithm is a non-trivial improvement on the running time obtained from the mim-width approach.

We complement our algorithmic result by showing that Subset Feedback Vertex Set is W[1]-hard parameterized by the leafage of a chordal graph. Thus we can hardly avoid the dependence of the exponent in the stated running time. Our reduction is inspired by the W[1]-hardness of Feedback Vertex Set parameterized by the mim-width given by Jaffke et al. [29]. However we note that our result holds on graphs with arbitrary vertex weights and we are not aware if the unweighted variant of Subset Feedback Vertex Set admits the same complexity behavior.

Our algorithm works on an expanded tree model that is obtained from the given tree model and maintains all intersecting information without increasing the number of leaves. Then in a bottom-up dynamic programming fashion, we visit every node of the expanded tree model in order to compute partial solutions. At each intermediate step, we store all necessary information of subsets of vertices that are of size $O(\ell)$. As a byproduct of our dynamic programming scheme and the expanded tree model, we show how our approach can be extended in order to handle rooted path graphs. Rooted path graphs are the intersection graphs of directed paths in a rooted tree. They form a subclass of leaf powers and we observe that they form a class of unbounded leafage. Although rooted path graphs admit a decomposition of mim-width one [28] and such a decomposition can be constructed in polynomial time [16, 22], the running time obtained through the bounded mim-width approach is rather impractical as it requires to store a table of size $O(n^{13})$ even in this particular case [1]. By analyzing further subsets of vertices at each intermediate step, we manage to derive an algorithm for Subset Feedback Vertex Set on rooted path graphs that runs in $O(n^2m)$ time. Observe that the stated running time is comparable to the $O(nm)$ time of the previously known algorithm on interval graphs [36]. Interval graphs form a proper subclass of rooted path graphs.

Inspired by the algorithm on bounded leafage graphs, we also consider the natural relaxation of the leafage that is the *vertex leafage* of a graph. Chaplick and Stacho [11] introduced the vertex leafage of a graph G as the smallest number k such that there exists a tree model for G in which every subtree corresponding to a vertex of G has at most k leaves. As leafage measures the closeness to interval graphs (graphs with leafage at most two), vertex leafage measures the closeness to undirected path graphs which are the intersection graphs of paths in a tree (graphs with vertex leafage at most two). We prove that the unweighted variant of Subset Feedback Vertex Set is NP-complete on undirected path graphs and, thus, the problem is para-NP-complete parameterized by the vertex leafage. An interesting trait of our NP-completeness proof is that our reduction comes from the Max Cut problem as opposed to known reductions for Subset Feedback Vertex Set which are usually based on, more natural, covering problems [18, 37]. From our results for rooted path graphs and undirected path graphs, we obtain a complexity dichotomy of the problem with respect to the vertex leafage: if the vertex leafage is at most one (i.e., rooted path graphs) then Subset Feedback Vertex Set is polynomial-

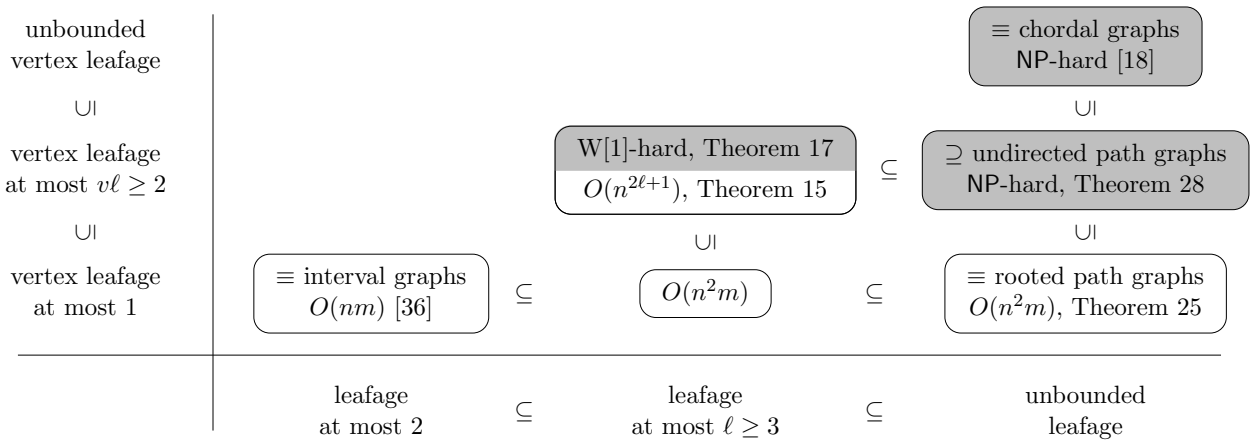

Figure 1: Computational complexity of the SFVS problem parameterized by leafage and vertex leafage.

time solvable; otherwise, if the vertex leafage is at least two, Subset Feedback Vertex Set is NP-complete. Our findings are summarized in Figure 1.

2 Preliminaries

All graphs considered here are finite undirected graphs without loops and multiple edges. We refer to the textbook by Bondy and Murty [4] for any undefined graph terminology and to the book of Cygan et al. [14] for an introduction to Parameterized Complexity. For a positive integer p, we use $[p]$ and $-[p]$ to denote the sets $\{1, 2, \ldots, p\}$ and $\{-1, -2, \ldots, -p\}$ respectively. For a graph $G = (V_G, E_G)$, we use V_G and E_G to denote the sets of vertices and edges respectively. We use n to denote the number of vertices of the graph and use m for its number of edges. Given $x \in V_G$, we denote by $N_G(x)$ the neighborhood of x. The *degree* of x is the number of edges incident to x. Given $X \subseteq V_G$, we let $N(X) = \bigcup_{v \in X} N(v) \setminus X$ and $N[X] = N(X) \cup X$. We denote by $G - X$ the graph obtained from G by the removal of the vertices of X. If $X = \{u\}$, we also write $G - u$. The *subgraph induced by* X is denoted by $G[X]$, and has X as its vertex set and $\{uv \in E_G \mid u, v \in X\}$ as its edge set.

A *clique* is a set $K \subseteq V_G$ such that $G[K]$ is a complete graph. An *ℓ-star* is an undirected tree that has exactly $\ell + 1$ nodes, ℓ of which are leaves. Notice that this implies that all ℓ leaves of an ℓ-star are adjacent to its remaining node.

Given a collection $\mathcal{C}$ of sets, the graph $G = (\mathcal{C}, \{\{X, Y\} \mid X, Y \in \mathcal{C} \text{ and } X \cap Y \neq \emptyset\})$ is called the *intersection graph of* $\mathcal{C}$. Structural properties and recognition algorithms are known for intersection graphs of (directed) paths in (rooted) trees [10, 33, 35]. Depending on the collection $\mathcal{C}$, we say that a graph is

- *chordal* if $\mathcal{C}$ is a collection of subtrees of a tree,
- *undirected path* if $\mathcal{C}$ is a collection of paths of a tree,
- *rooted path* if $\mathcal{C}$ is a collection of directed paths of a rooted tree, and
- *interval* if $\mathcal{C}$ is a collection of subpaths of a path.

Let T be a rooted tree. We use $r(T)$ to denote its root. We assume that the edges of T are directed away from $r(T)$. We denote the unique directed path from a node s to a node t by

$s \to t$. If $s \to t$ exists in T, we say that t is a *descendant* of s and that s is an *ancestor* of t. The leaves of an undirected tree T are exactly the nodes of T having degree at most one. The leaves of a rooted tree T are exactly the nodes of T having in-degree at most one and out-degree zero. For any tree T, we use $L(T)$ to denote the set of its leaves. Observe that for an undirected tree T we have $|L(T)| = 1$ if and only if T has no edges, whereas for a rooted tree T we have $|L(T)| = 1$ if and only if T is a directed path.

A binary relation defined on a set is called a *partial order* if it is transitive and anti-symmetric. Let X be a set and $\leq$ be a partial order on X. We say that two elements u and v of X are *comparable* with respect to $\leq$ if $u \leq v$ or $v \leq u$; otherwise, u and v are called *incomparable* with respect to $\leq$. If $u \leq v$ and $u \neq v$, then we simply write $u < v$. For all $X' \subseteq X$, we write $\min_{\leq} X'$ and $\max_{\leq} X'$ to denote the sets of all minimal and maximal elements of X' with respect to $\leq$ respectively. Given a rooted tree $T = (V_T, E_T)$, we define a partial order on the nodes of T as follows: for every $x, y \in V_T$, $x \leq_T y \Leftrightarrow y \to x$ exists in T. Regarding $\leq_T$ we make the following observations.

Observation 1. *Let $T = (V_T, E_T)$ be a rooted tree. For every $x, y, y' \in V_T$, if $x \leq_T y$ and $x \leq_T y'$, then y and y' are comparable with respect to $\leq_T$.*

Proof. Let $x \leq_T y$ and $x \leq_T y'$. Then by definition $y \to x$ and $y' \to x$ exist in T. Since T is a rooted tree, every node has at most one parent in T. By induction, for every $k \in \mathbb{N}$, every node has at most one ancestor at distance k in T. We set y_c and y_f to be the nodes among y, y' which are the closest and the farthest away from x respectively. Then observe that $y_f \to x$ contains y_c and in particular $y_f \to y_c$ exists in T, which implies that $y_c \leq_T y_f$ by definition, so y and y' are comparable with respect to $\leq_T$. □

Observation 2. *Let T be a rooted tree and let T' be a subtree of T. For every $l, r \in V(T')$ such that $l < r$, every node $b \in V(T)$ such that $l < b < r$ is also in $V(T')$.*

Proof. By definition, $l < b < r$ implies that $r \to b$ and $b \to l$ exist in T. In other words, $r \to l$ exists in T and contains b. Since T' is connected and $l, r \in V(T')$, we conclude that all the nodes in $r \to l$ are in $V(T')$. In particular $b \in V(T')$. □

Observation 3. *Let T be a rooted tree and let V be a set of pairwise incomparable nodes of T with respect to $\leq_T$. Then $|V| \leq |L(T)|$.*

Proof. Notice that for every node x of T, there exists a leaf l of T such that $l \leq_T x$. Assume that $|L(T)| < |V|$. Then there exists a leaf l of T and two distinct nodes $x, y \in V$ such that $l \leq_T x$, and $l \leq_T y$. By Observation 1, the nodes x and y are comparable with respect to $\leq_T$, a contradiction. We conclude that $|V| \leq |L(T)|$. □

Leafage and vertex leafage A *tree model* of a graph $G = (V_G, E_G)$ is a pair $(T, \{T_v\}_{v \in V_G})$ such that (1) T is a tree, called a *host tree*[1], (2) for each $v \in V_G$, T_v is a subtree of T, and (3) for each $u, v \in V_G$ such that $u \neq v$, $uv \in E_G$ if and only if $V(T_u) \cap V(T_v) \neq \emptyset$. It is known that a graph is chordal if and only if it admits a tree model [9, 21]. The tree model of a chordal graph is not necessarily unique. The *leafage* of a chordal graph G, denoted by $\ell(G)$, is the minimum number of leaves of the host tree among all tree models of G, that is, $\ell(G)$ is the smallest integer ℓ such that there exists a tree model $(T, \{T_v\}_{v \in V_G})$ of G with $|L(T)| = \ell$ [32]. Moreover, every chordal graph G admits a tree model for which its host tree T has the minimum $|L(T)|$ and $|V(T)| \leq n$ [11, 24]; such a tree model can be constructed in $O(n^3)$ time [24]. Thus the leafage $\ell(G)$ of a chordal graph G is computable in polynomial time.

[1]The host tree is also known as a *clique tree*, usually when we are concerned with the maximal cliques of a chordal graph [21].

A relaxation of the leafage is the *vertex leafage* introduced by Chaplick and Stacho [11]. The vertex leafage of a chordal graph G, denoted by $v\ell(G)$, is the smallest integer $v\ell$ such that there exists a tree model $(T, \{T_v\}_{v \in V_G})$ of G where $|L(T_v)| \leq v\ell$ for all $v \in V_G$. Clearly, we have $v\ell(G) \leq \ell(G)$. Unlike the leafage, deciding whether the vertex leafage of a chordal graph is at most $v\ell$ is NP-complete for every fixed integer $v\ell \geq 3$ [11].

Notice that for any tree model $(T, \{T_v\}_{v \in V_G})$ and for any $\ell, v\ell \in \mathbb{N}^*$ such that $|L(T)| \leq \ell$ and $|L(T_v)| \leq v\ell$ for all $v \in V_G$, after rooting T in an arbitrary node, the same conditions still hold. Henceforth, we will only consider tree models with host trees that are rooted trees unless otherwise stated. Under these terms, observe that (1) $\ell(G) \leq 1 \Leftrightarrow G$ is an interval graph[2], (2) $v\ell(G) \leq 1 \Leftrightarrow G$ is a rooted path graph, and (3) $v\ell(G) \leq 2$ if G is an undirected path graph.

S-forests and S-triangles By an induced cycle of G we mean a chordless cycle. A *triangle* is a cycle on 3 vertices. Chordal graphs are exactly the graphs that do not contain induced cycles on more than 3 vertices. In this work, we consider graph classes that are subclasses of chordal graphs.

Given a graph G and $S \subseteq V(G)$, we say that a cycle of G is an *S-cycle* if it contains a vertex in S. Moreover, we say that an induced subgraph F of G is an *S-forest* if F does not contain an S-cycle. Thus an induced subgraph F of a chordal graph is an S-forest if and only if F does not contain any S-triangle. A set of vertices such that its removal results in an S-forest is referred to as a *subset feedback vertex set*. In these terms, the (unweighted) Subset Feedback Vertex Set problem asks for a subset feedback vertex set of minimum weight (size). In our dynamic programming algorithms, we focus on the equivalent formulation of computing a maximum-weighted S-forest.

For a collection $\mathcal{C}$ of sets, we write $\max_{\text{weight}}\{C \in \mathcal{C}\}$ to denote $\arg\max\{weight(C) \mid C \in \mathcal{C}\}$, where $weight(C)$ is the sum of weights of all vertices in C. Let $G = (V_G, E_G)$ be a graph and let $S \subseteq V_G$. The collection of all S-forests of G is denoted by $\mathcal{F}_S$. For any $X, Y \subseteq V_G$ such that $X \cap Y = \emptyset$ and $G[Y] \in \mathcal{F}_S$, we write A_X^Y to denote an arbitrary element of the collection $\max_{\text{weight}}\{U \subseteq X \mid G[U \cup Y] \in \mathcal{F}_S\}$. As the sets A_X^Y are not necessarily unique, we use the $\leftrightarrow$ operator between any two expressions involving such sets to denote that for any particular evaluation of one there exists an evaluation of the other such that both yield the same result. Our desired optimal solution is any element $A_{V_G}^{\emptyset}$ of $\max_{\text{weight}}\{U \subseteq V_G \mid G[U] \in \mathcal{F}_S\}$. We will subsequently show that in order to compute $A_{V_G}^{\emptyset}$, if G is a chordal graph with leafage ℓ for any fixed $\ell \in \mathbb{N}^*$ or a rooted path graph, then it is sufficient to compute A_X^Y for a polynomial number of sets X and Y.

Let $G = (V_G, E_G)$ be a chordal graph, let $S \subseteq V_G$ and let $X, Y \subseteq V_G$ such that $X \cap Y = \emptyset$ and $G[Y] \in \mathcal{F}_S$. A partition $\mathcal{P}$ of X is called *nice* if for any S-triangle S_t of $G[X \cup Y]$, there is a part $P \in \mathcal{P}$ such that $V(S_t) \cap X \subseteq P$. In other words, any S-triangle of $G[X \cup Y]$ is involved with at most one part of any nice partition of X. With respect to the defined optimal solutions A_X^Y, we observe the following:

Observation 4. *Let $G = (V_G, E_G)$ be a chordal graph, let $S \subseteq V_G$ and let $X, Y \subseteq V_G$ such that $X \cap Y = \emptyset$ and $G[Y] \in \mathcal{F}_S$. Then the following hold:*

(1) $A_X^Y \leftrightarrow \bigcup_{P \in \mathcal{P}} A_P^Y$ *for any nice partition $\mathcal{P}$ of X.*

(2) $A_X^Y \leftrightarrow A_X^{Y'}$ *where $Y' = Y \cap N(X')$ for any $X \supseteq X' \supseteq X \setminus \{u \in X \setminus S \mid Y \cap N(u) \subseteq Y \setminus S\}$.*

[2]If the host tree T is an undirected tree, then $\ell(G) \leq 2 \Leftrightarrow G$ is an interval graph [11].

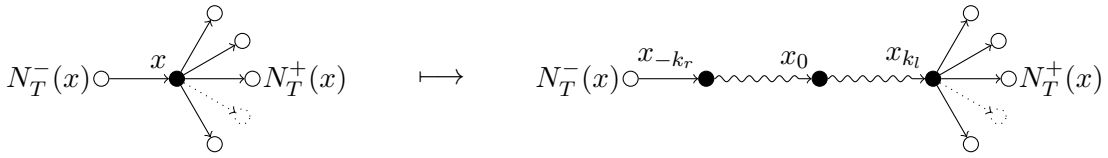

Figure 2: We replace node x of T by the directed path $\langle x_{-k_r}, \ldots, x_0, \ldots, x_{k_l} \rangle$ such that in T' the node of $N_T^-(x)$ points to x_{-k_r} and all nodes of $N_T^+(x)$ are pointed by x_{k_l} instead.

Proof. For the first statement, assume that there is an S-triangle S_t in $G[X \cup Y]$. Then it must contain a vertex of some part P of $\mathcal{P}$, as $G[Y]$ is an S-forest. By the definition of a nice partition, we have $V(S_t) \cap X \subseteq P$. Therefore, we deduce $A_X^Y \cap P \leftrightarrow A_P^Y$, which shows the claim.

For the second statement, observe that $G[Y'] \in \mathcal{F}_S$, as $Y' \subseteq Y$ and $G[Y] \in \mathcal{F}_S$. Also, notice that any S-triangle in $G[X \cup Y']$ remains an S-triangle in $G[X \cup Y]$. Consider an S-triangle in $G[X \cup Y]$ induced by $\{x, y, z\}$ where $x \in X$ and $y \in Y$. We show that $y \in Y'$ and $z \in X \cup Y'$. If $x \in X'$, then $y \in Y'$ and $z \in X \cup Y'$ by the fact that $Y' = Y \cap N(X')$. Suppose that $x \in X \setminus S$ such that $Y \cap N(x) \subseteq Y \setminus S$. Then $y \in Y \setminus S$ and $z \in X \cup (Y \setminus S)$. This means that z must be in S and in particular $z \in X \cap S \subseteq X'$. By the fact that $Y' = Y \cap N(X')$, we conclude that $y \in Y'$. Thus, any S-triangle in $G[X \cup Y]$ remains an S-triangle in $G[X \cup Y']$, which concludes the proof. □

Observation 4 suggests how to reduce the computation of A_X^Y to the computation of optimal solutions to smaller instances. More precisely, by Observation 4 (1), if we obtain a nice partition $\mathcal{P}$ of the vertex set X, then we can reduce the computation of A_X^Y to the computation of A_P^Y for every $P \in \mathcal{P}$, which are optimal solutions to smaller and pairwise-independent instances, and Observation 4 (2) states that for computing A_X^Y, it is sufficient to consider only the vertices y of Y which have neighbours x in X such that at least one of x and y is in S.

3 Expanded tree model

Given a tree model of a chordal graph, we are interested in defining a partial order on the vertices of the graph that takes advantage of the underlying tree structure. For this purpose, it is necessary that each of the subtrees of the tree model corresponds to at most one vertex of the graph. Here we show how a tree model can be altered in order to obtain this property in a formal way. Assume that G is a chordal graph.

Definition 1. A tree model $(T, \{T_v\}_{v \in V_G})$ of G is called *expanded tree model* if the sets of the collection $\{\{r(T_v)\}\}_{v \in V_G} \cup \{L(T_v)\}_{v \in V_G}$ are paiwise-disjoint.

We show that for any tree model M of a chordal graph G, there exists an expanded tree model M' of G that is structurally close to M. In fact, we provide an algorithm that, given a tree model of G, constructs such an expanded tree model of G.

Lemma 5. *For any tree model $(T, \{T_v\}_{v \in V_G})$ of G and for any $\ell, v\ell \in \mathbb{N}^*$ such that $|L(T)| = \ell$ and $|L(T_v)| \leq v\ell$ for all $v \in V_G$, there is an expanded tree model $(T', \{T'_v\}_{v \in V_G})$ of G such that:*

- *$|L(T')| = |L(T)| = \ell$ and $|L(T'_v)| = |L(T_v)| \leq v\ell$ for all $v \in V_G$, and*
- *$|V(T')| \leq |V(T)| + (1 + v\ell)n$.*

Moreover, given $(T, \{T_v\}_{v \in V_G})$, the expanded tree model can be constructed in time $O(n^2)$.

Proof. Consider a node x of T. Assume that x is the root of k_r subtrees $T_{v_{-1}}, \ldots, T_{v_{-k_r}}$ and a leaf of k_l subtrees $T_{v_1}, \ldots, T_{v_{k_l}}$ of $\{T_v\}_{v \in V_G}$ where $k_r + k_l \geq 2$. We replace the node x in T by the gadget shown in Figure 2. We also modify every subtree T_v of $\{T_v\}_{v \in V_G}$ as follows:

- If there exists an $i \in -[k_r]$ such that $T_v = T_{v_i}$ and $T_v \neq T_{v_j}$ for all $j \in [k_l]$, then we replace x in T_v by the part of the gadget involving the vertices $x_i, \ldots, x_0, \ldots, x_{k_l}$.
- If $T_v \neq T_{v_i}$ for all $i \in -[k_r]$ and there exists a $j \in [k_l]$ such that $T_v = T_{v_j}$, then we replace x in T_v by the part of the gadget involving the vertices $x_{-k_r}, \ldots, x_0, \ldots, x_j$.
- If there exists an $i \in -[k_r]$ such that $T_v = T_{v_i}$ and a $j \in [k_l]$ such that $T_v = T_{v_j}$, then we replace x in T_v by the part of the gadget involving the vertices $x_i, \ldots, x_0, \ldots, x_j$.
- If $T_v \neq T_{v_i}$ for all $i \in -[k_r]$ and $T_v \neq T_{v_j}$ for all $j \in [k_l]$, then
  - if x is a (necessarilly internal) node of T_v, we replace x in T_v by the whole gadget, otherwise $T'_v = T_v$.

To see that $(T', \{T'_v\}_{v \in V_G})$ is indeed a tree model of G, observe that for every $T_u, T_w \in \{T_v\}_{v \in V_G}$:

- if $x \in V(T_u) \cap V(T_w)$, then $x_0 \in V(T'_u) \cap V(T'_w)$, and
- if $x \notin V(T_u) \cap V(T_w)$, then $x_{-k_l}, \ldots, x_{k_r} \notin V(T'_u) \cap V(T'_w)$.

Thus the intersection graph of $(T', \{T'_v\}_{v \in V_G})$ is isomorphic to G. Observe that among the sets $\{r(T_{v_{-k_r}})\}$, $\ldots$, $\{r(T_{v_{-1}})\}$, $L(T_{v_1})$, $\ldots$, $L(T_{v_{k_l}})$, the node x_i is only in $\{r(T_{v_i})\}$ for all $i \in -[k_r]$ and only in $L(T_{v_i})$ for all $i \in [k_r]$. Iteratively applying the above modifications to $(T, \{T_v\}_{v \in V_G})$ results in an tree model of G that satisfies Definition 1 for being an expanded tree model.

Observe that the iterative procedure described above preserves the number of leaves of the tree and of all subtrees in the collection of the tree model. Let us now bound $|V(T')|$. Recall that every subtree in $\{T_v\}_{v \in V_G}$ has at most $v\ell$ leaves. In the worst case, every subtree in $\{T_v\}_{v \in V_G}$ has exactly $v\ell$ leaves and no node of T is the root of one subtree and a leaf of no subtree of $\{T_v\}_{v \in V_G}$ or vice versa. In this case, the iterative procedure described above will add $\sum_{v \in V_G}(|\{r(T_v)\}| + |L(T_v)|) = (1 + v\ell)n$ nodes to T. We conclude that the procedure adds at most $(1+v\ell)n$ nodes to T as well as to the n subtrees in $\{T_v\}_{v \in V_G}$, resulting in the total running time of $O(n^2)$. □

An example of an expanded tree model produced by the iterative procedure described in the proof of Lemma 5 is shown in Figure 3. Hereafter we assume that $(T, \{T_v\}_{v \in V_G})$ is an expanded tree model of G. For any vertex u of G, we denote the node $r(T_u)$ by $r(u)$ for simplicity. We define the following partial order on the vertices of G: for all $u, v \in V_G$, $u \leq_G v \Leftrightarrow r(u) \leq_T r(v)$. In other words, two vertices of G are comparable with respect to $\leq_G$ if and only if there is a directed path between their corresponding roots in T. Since we defined $\leq_G$ and $\leq_T$ on disjoint sets, we will subsequently omit mentioning the relevant partial order explicitly.

Observation 6. *Let $u, v, w, z \in V_G$. Then, the following hold:*

(1) If $uv \in E_G$, then u and v are comparable.

(2) If $u \leq v$, $z \leq w$, and u and z are comparable, then v and w are comparable.

(3) If $u < v < w$ and $uw \in E_G$, then $vw \in E_G$.

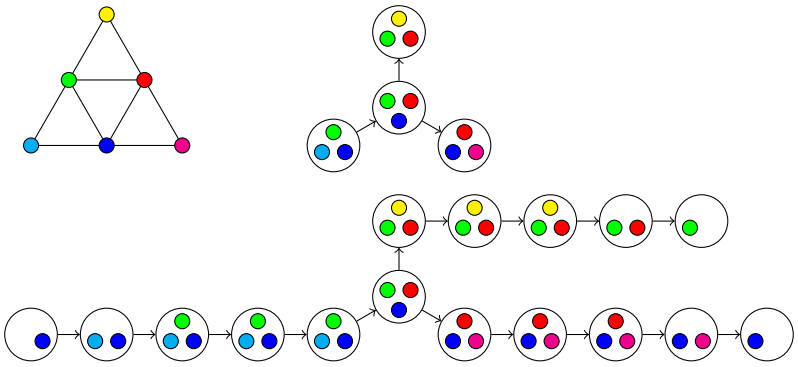

Figure 3: Illustration of a chordal graph G (top left), a tree model of G (top center) and an expanded tree model of G obtained via the iterative procedure described in the proof of Lemma 5 (bottom). The paths of the two models corresponding to each vertex of G are the ones formed from the nodes containing the color of that vertex.

Proof. For the first statement, assume that $x \in V(T_u) \cap V(T_v)$, which exists as $uv \in E_G$. Then the paths $r(u) \to x$ and $r(v) \to x$ exist in T. Equivalently, $x \leq r(u)$ and $x \leq r(v)$ hold. By Observation 1, we get that $r(u)$ and $r(v)$ are comparable, which implies that u and v are also comparable.

For the second statement, assume first that $u \leq z$. Then $r(u) \leq r(z) \leq r(w)$ because also $z \leq w$. Additionally $r(u) \leq r(v)$ because $u \leq v$. Just as before, by Observation 1, we get that $r(v)$ and $r(w)$ are comparable, which implies that v and w are also comparable. The case for $z \leq u$ is completely symmetrical.

For the third statement, observe that $u < v < w$ implies that $r(u) < r(v) < r(w)$. We show that $r(v) \in V(T_w)$. Since u and w are adjacent, there exists a node $x \in V(T_u) \cap V(T_w)$. Then the paths $r(u) \to x$ and $r(w) \to x$ exist in T, implying that $x \leq r(u)$ and $x \leq r(w)$. Since $x, r(w) \in V(T_w)$ and $x < r(v) < r(w)$, by Observation 2, we get that $r(v) \in V(T_w)$, so v and w are adjacent. □

For all $u \in V_G$, we define V_u to be the set $\{u' \in V_G \mid u' \leq u\}$. We also define $\lhd u$ to be the set $\max\{u' \in V_G \mid u' < u\} = \max(V_u \setminus \{u\})$. Moreover, for all $uv \in E_G$, we define $\lhd uv$ to be the set $\max\{u' \in V_G \mid u' < u, v \text{ and } (u'u \notin E_G \text{ or } u'v \notin E_G)\} = \max((V_u \cap V_v) \setminus (N[u] \cap N[v]))$. Recall that for any edge $uv \in E_G$, either $u < v$ or $v < u$ by Observation 6 (1). If $u < v$ holds, then $\lhd uv = \max(V_u \setminus (N[u] \cap N(v)))$. For all $U \subseteq V_G$, we define $\mathcal{V}_U$ to be the collection $\{V_u\}_{u \in U}$. For the example of Figure 3, denoting the red, green, blue, cyan, magenta and yellow vertices by r, g, b, c, m and y respectively, the following hold:

$$
\begin{array}{llllll}
V_r = \{r, m, y\} & \lhd r = \{m, y\} & \lhd rg = \{m\} & \lhd gb = \{m, y\} & \lhd bc = \{r\} \\
V_g = \{r, g, m, y\} & \lhd g = \{r\} & \lhd rb = \{y\} & \lhd gc = \{r\} & \lhd bm = \emptyset \\
V_b = \{r, g, b, c, m, y\} & \lhd b = \{c\} & \lhd rm = \emptyset & \lhd gy = \emptyset & \\
V_c = \{r, g, c, m, y\} & \lhd c = \{g\} & \lhd ry = \emptyset & & \\
V_m = \{m\} & \lhd m = \emptyset & & & \\
V_y = \{y\} & \lhd y = \emptyset & & &
\end{array}
$$

Having defined all the primary components, we can now provide a brief outline of our dynamic programming algorithms.

Step 1: Construction of expanded tree model. From a tree model of the chordal graph G that we are given as input, we produce an expanded tree model $(T, \mathcal{T})$ as described in the proof of Lemma 5.

Step 2: Computation of auxiliary vertex sets. Traversing T from its leaves to its root, upon reaching each node $v \in V_T$, if there exists a vertex $u \in V_G$ such that $r(u) = v$, then we compute the set $\lhd u$. Similarly for all other auxiliary vertex sets that are necessary for each algorithm.

Step 3: Computation of optimal solutions to subproblems. Traversing T again from its leaves to its root, upon reaching each node $v \in V_T$, if there exists a vertex $u \in V_G$ such that $r(u) = v$, then we compute the optimal solution $A^{\emptyset}_{V_u}$ from previously computed optimal solutions to smaller subproblems. Similarly for optimal solutions to all other subproblems that are necessary for each algorithm.

The following two lemmas provide nice partitions of X, to be used in the application of Observation 4 (1), in certain cases of X that are considered by both our algorithms.

Lemma 7. *For every $u \in V_G$, the collection $\mathcal{V}_{\lhd u}$ is a partition of $V_u \setminus \{u\}$ into pairwise disconnected sets. For every $u, v \in V_G$ such that $u < v$ and $uv \in E_G$, the collection $\mathcal{V}_{\lhd uv}$ is a partition of $V_u \setminus (N[u] \cap N(v))$ into pairwise disconnected sets.*

Proof. We prove the first statement. The proof of the second statement is completely analogous. Firstly notice that, by definition, the vertices of $\lhd u$ are pairwise incomparable. Consider two vertices u'_1 and u'_2 such that $u'_1 \leq u_1$ and $u'_2 \leq u_2$ where u_1 and u_2 are two vertices of $\lhd u$. Clearly, $u'_1 \in V_{u_1}$ and $u'_2 \in V_{u_2}$. By Observation 1 and Observation 6 (1–2), it follows that the vertices u'_1 and u'_2 are distinct and non-adjacent. □

Lemma 8. *For every $u \in V_G$, the collection $\mathcal{V}_{\lhd u}$ is a nice partition of $V_u \setminus \{u\}$. For every $u, v \in V_G$ such that $u < v$ and $uv \in E_G$, the collection $\mathcal{V}_{\lhd uv}$ is a nice partition of $V_u \setminus (N[u] \cap N(v))$.*

Proof. We prove the first statement. The proof of the second statement is completely analogous. Let $X = V_u \setminus \{u\}$ and $Y \subseteq V_G$ such that $X \cap Y = \emptyset$. Suppose that S_t is an S-triangle of $G[X \cup Y]$ for which the intersection of $V(S_t)$ and a part of $\mathcal{V}_{\lhd u}$ is non-empty for at least two such parts. Assume that P_1 and P_2 are two of those parts and let $u_1 \in V(S_t) \cap P_1$ and $u_2 \in V(S_t) \cap P_2$. Then u_1 and u_2 must be adjacent, which is in contradiction to Lemma 7. □

The following Lemma simplifies the calculation of Y' in Observation 4 (2) in the case of $X = V_u$ for some $u \in V_G$.

Lemma 9. *Let $u \in V_G$ and $Y \subseteq V_G \setminus V_u$. Then $Y \cap N(V_u) = Y \cap N(u)$.*

Proof. From the facts that $u \in V_u$ and $Y \subseteq V_G \setminus V_u$, it directly follows that $Y \cap N(u) \subseteq Y \cap N(V_u)$. We will now show that also $Y \cap N(V_u) \subseteq Y \cap N(u)$. It suffices to show that $N(V_u) \subseteq N(u)$. Let $w \in N(V_u)$. Then there exists a vertex $v \in V_u$ such that $vw \in E_G$. It suffices to show that $w \in N(u)$. Assume that $u \neq v$, as otherwise the claim trivially holds. Then $v < u$, because $v \in V_u$. Moreover, Observation 6 (1) implies that either $w < v$ or $v < w$. Since $w < v < u$ contradicts the fact that $w \notin V_u$, we conclude that $v < w$. Then by applying Observation 6 (2) we obtain that u and w are comparable. Since $w \notin V_u$, it must be that $v < u < w$ and by Observation 6 (3) we conclude that $uw \in E_G$. □

We are now ready to show the first recursive expressions of optimal solutions to subproblems, to be used for the computation of A^Y_X in certain cases of X and Y that are considered by both our algorithms. Both statements involve the application of Observation 4 in combination with Lemma 8 and Lemma 9.

Lemma 10. *Let $u \in V_G$ and $Y \subseteq V_G \setminus V_u$. If $u \notin A^Y_{V_u}$, then $A^Y_{V_u} \leftrightarrow \bigcup_{u' \in \lhd u} A^{Y \cap N(u')}_{V_{u'}}$.*

Proof. Since $u \notin A^Y_{V_u}$, we have $A^Y_{V_u} \leftrightarrow A^Y_{V_u \setminus \{u\}}$. According to Lemma 8, the collection $\mathcal{V}_{\lhd u}$ is a nice partition of $V_u \setminus \{u\}$. By Observation 4 and Lemma 9, we get $A^Y_{V_u \setminus \{u\}} \leftrightarrow \bigcup_{u' \in \lhd u} A^Y_{V_{u'}} \leftrightarrow \bigcup_{u' \in \lhd u} A^{Y \cap N(V_{u'})}_{V_{u'}} \leftrightarrow \bigcup_{u' \in \lhd u} A^{Y \cap N(u')}_{V_{u'}}$. □

Lemma 11. *Let $u \in V_G$. If $u \in A^{\emptyset}_{V_u}$, then $A^{\emptyset}_{V_u} \leftrightarrow \{u\} \cup \bigcup_{u' \in \lhd u} A^{\{u\} \cap N(u')}_{V_{u'}}$.*

Proof. Assume that $u \in A^{\emptyset}_{V_u}$. Then $A^{\emptyset}_{V_u} \leftrightarrow \{u\} \cup A^{\{u\}}_{V_u \setminus \{u\}}$. Recall that the collection $\mathcal{V}_{\lhd u}$ is a nice partition of $V_u \setminus \{u\}$. By Observation 4 and Lemma 9, we get $A^{\{u\}}_{V_u \setminus \{u\}} \leftrightarrow \bigcup_{u' \in \lhd u} A^{\{u\}}_{V_{u'}} \leftrightarrow \bigcup_{u' \in \lhd u} A^{\{u\} \cap N(V_{u'})}_{V_{u'}} \leftrightarrow \bigcup_{u' \in \lhd u} A^{\{u\} \cap N(u')}_{V_{u'}}$. □

4 SFVS on graphs with bounded leafage

In this section our goal is to show that SFVS can be solved in polynomial time on chordal graphs with bounded leafage. In particular, we consider chordal graphs that have an intersection model tree with at most ℓ leaves and we show that SFVS can be solved in $n^{O(\ell)}$ time. We subsequently assume that we are given a chordal graph G that admits an expanded tree model $(T, \{T_v\}_{v \in V_G})$ with $|L(T)| = \ell$ due to Lemma 5.

Given a set of vertices of G, we collect the nodes and the leaves and of their corresponding subtrees: for every $U \subseteq V_G$, we define $V(U) = \bigcup_{u \in U} V(T_u)$ and $L(U) = \bigcup_{u \in U} L(T_u)$. Notice that for any non-empty $U \subseteq V_G$, the sets $V(U)$ and $L(U)$ are also non-empty, and the nodes of $L(U)$ admit a partial order $\leq_T$. Moreover, given a set of nodes of T, we collect the vertices corresponding to the subtrees of which they are leaves: for every $V \subseteq V_T$, we define $L^{-1}(V)$ to be the set $\{u \in V_G \mid L(T_u) \cap V \neq \emptyset\}$.

Observation 12. *Let $U \subseteq V_G$ and $V \subseteq L(U)$. Then $L^{-1}(V) \subseteq U$.*

Proof. The fact that $V \subseteq L(U)$ yields $L^{-1}(V) \subseteq L^{-1}(L(U))$. We will show that $L^{-1}(L(U)) \subseteq U$. Let u be a vertex of G such that $u \notin U$. Then, since $(T, \{T_v\}_{v \in V_G})$ is an expanded tree model, Definition 1 implies that $L(T_u) \cap L(U) = \emptyset$. Thus $u \notin L^{-1}(L(U))$. □

For every $U \subseteq V_G$, we define the *representation* of U to be the set $R_{\leq 2}(U) = R_1(U) \cup R_2(U)$ where $R_1(U) = L^{-1}(\min L(U))$ and $R_2(U) = L^{-1}(\min L(U \setminus R_1(U)))$. Observation 12 implies that $R_{\leq 2}(U) \subseteq U$ for every $U \subseteq V_G$. Observe that for any $V \subseteq V_T$, the set $\min V$ of minimal nodes of V is a set of pairwise-incomparable nodes, so $|\min V| \leq |L(T)| = \ell$ by Observation 3. This implies that $|R_{\leq 2}(U)| \leq 2\ell$ for all $U \subseteq V_G$. Representations have the following property.

Observation 13. *Let $u \in V_G$ and let $Y \subseteq V_G \setminus V_u$. Then $V(Y) \cap V(V_u) = V(R_1(Y)) \cap V(V_u)$ and $V(Y \setminus R_1(Y)) \cap V(V_u) = V(R_2(Y)) \cap V(V_u)$.*

Proof. We show that the first equation holds. Showing that the second equation holds is completely analogous. By Observation 12, we get $R_1(Y) \subseteq Y \Rightarrow V(R_1(Y)) \subseteq V(Y) \Rightarrow V(R_1(Y)) \cap V(V_u) \subseteq V(Y) \cap V(V_u)$. We will show that also $V(Y) \cap V(V_u) \subseteq V(R_1(Y)) \cap V(V_u)$. Let $b \in V(Y) \cap V(V_u)$. Then there exist $u' \in V_u$ and $v \in Y$ such that $b \in V(T_{u'}) \cap V(T_v)$. Since $b \in V(T_{u'})$, we get $b \leq r(u')$. Since $b \in V(T_v)$, there exists an $l \in L(T_v) \subseteq L(Y)$ such that $l \leq b$. Then there exists an $l' \in \min L(Y)$ such that $l' \leq l$. By definition of $R_1(Y)$, there exists a vertex $v' \in R_1(Y)$ such that $l' \in L(T_{v'})$. Putting it all together yields $l' \leq l \leq b \leq r(u') \leq r(u) < r(v')$. By Observation 2, we conclude that $b \in V(T_{v'}) \subseteq V(R_1(Y))$. □

We subsequently show that for computing A^Y_X, the vertex set Y can be substituted by its representation $R_{\leq 2}(Y)$ in certain cases of X and Y that are considered by the algorithm of this section.

Lemma 14. *Let $u \in V_G$ and $W \subseteq V_G \setminus V_u$ such that $W \neq \emptyset$, $G[W] \in \mathcal{F}_S$ and $\{u\} \cup W$ is a clique, and let $u \in A^W_{V_u}$.*

- *If $(\{u\} \cup W) \cap S \neq \emptyset$, then $W = \{w\}$ and no vertex of $V_u \cap N(u) \cap N(w)$ belongs to $A^{\{w\}}_{V_u}$.*
- *If $(\{u\} \cup W) \cap S = \emptyset$, then $A^{(\{u\} \cup W) \cap N(u')}_{V_{u'}} \leftrightarrow A^{R_{\leq 2}((\{u\} \cup W) \cap N(u'))}_{V_{u'}}$ for every vertex $u' \in \lhd u$.*

Proof. Assume that some vertex of $\{u\} \cup W$ is in S. Further assume that $|W| \geq 2$. Then there are $w_1, w_2 \in W$ such that $\{u, w_1, w_2\} \cap S \neq \emptyset$. Since $\{u\} \cup W$ is a clique, we have that $\{u, w_1, w_2\}$ induces an S-triangle, contradicting the fact that u belongs to $A^W_{V_u}$. We deduce that $W = \{w\}$ because $W \neq \emptyset$. Observe that for any $u' \in V_u \cap N(u) \cap N(w)$, the vertex set $\{u', u, w\}$ induces an S-triangle, since u and w are adjacent. Thus, no vertex of $V_u \cap N(u) \cap N(w)$ is in $A^{\{w\}}_{V_u}$.

Assume that no vertex of $\{u\} \cup W$ is in S. Consider a vertex $u' \in \lhd u$. Observe that for any two vertices $a \in V_{u'}$ and $b \in V_G \setminus V_{u'}$ to be adjacent, since $r(a) \leq r(u') < r(b)$ already holds, there must exist an $l \in L(T_b)$ such that $l < r(a)$ also holds. Let $W' = (\{u\} \cup W) \cap N(u')$ and $R = R_{\leq 2}(W')$. We will show that $A^{W'}_{V_{u'}} \leftrightarrow A^R_{V_{u'}}$.

- Assume there are two vertices $u''_1, u''_2 \in V_{u'}$ and a vertex $w' \in W'$ such that $\{u''_1, u''_2, w'\}$ induces an S-triangle. Then u''_1, u''_2 are adjacent and consequently, by Observation 6 (1), comparable, so without loss of generality we may assume that $r(u''_1) < r(u''_2)$. Let $l' \in L(T_{w'})$ such that $l' < r(u''_1)$. By definition of R, there is a vertex $w'' \in R$ for which there is a node $l'' \in L(T_{w''})$ such that $l'' \leq l'$. This implies that the set $\{u''_1, u''_2, w''\}$ also induces an S-triangle.

- Assume there is a vertex $u'' \in V_{u'}$ and two vertices $w'_1, w'_2 \in W'$ such that $\{u'', w'_1, w'_2\}$ induces an S-triangle. Let $l'_1 \in L(T_{w'_1})$ and $l'_2 \in L(T_{w'_2})$ such that $l'_1, l'_2 < r(u'')$. By definition of R, there are two distinct vertices $w''_1, w''_2 \in R$ for which there are nodes $l''_1 \in L(T_{w''_1})$ and $l''_2 \in L(T_{w''_2})$ such that $l''_1 \leq l'_1$ and $l''_2 \leq l'_2$. This implies that the set $\{u'', w''_1, w''_2\}$ also induces an S-triangle. □

We next show that Lemma 10, Lemma 11 and Lemma 14 suffice for the development of a dynamic programming scheme. As the size of the representation of any subset of V_G is bounded by 2ℓ, we need to store only a bounded number of optimal solutions to subproblems. In particular, we show that we need to compute A^Y_X only for $O(n)$ cases of X and only for cases of Y such that $|Y| \leq 2\ell$ holds.

Theorem 15. *There is an algorithm that, given a connected chordal graph G and an expanded tree model $(T, \mathcal{T})$ of G with $|L(T)| = \ell$, solves the weighted* Subset Feedback Vertex Set *problem in $O(n^{2\ell+1})$ time.*

Proof. Let $u_{\max}$ denote the (unique) vertex of $\max V_G$. Our task is to solve SFVS on G by computing $A^{\emptyset}_{V_{u_{\max}}}$. For doing so, we device a dynamic programming algorithm that visits the nodes of T in a bottom-up fashion starting from its leaves and moving towards its root. At each node $v \in V_T$, if there exists a vertex $u \in V_G$ such that $r(u) = v$, we store the values of $A^{\emptyset}_{V_u}$ and $A^W_{V_u}$ for every $W \subseteq V_G \setminus V_u$ such that $W \neq \emptyset$, $G[W] \in \mathcal{F}_S$ and $\{u\} \cup W$ is a clique. In order to compute $A^{\emptyset}_{V_u}$, we apply Lemma 10 and Lemma 11. In particular, after retrieving all necessary values being stored on the corresponding descendants of v, we apply the formula

$$A^{\emptyset}_{V_u} = \max_{\text{weight}} \left\{ \bigcup_{u' \in \lhd u} A^{\emptyset}_{V_{u'}}, \quad \{u\} \cup \bigcup_{u' \in \lhd u} A^{\{u\} \cap N(u')}_{V_{u'}} \right\}.$$

Proof: The first case in the above formula is the case of $u \notin A^{\emptyset}_{V_u}$ and is due to Lemma 10, whereas the second case is the case of $u \in A^{\emptyset}_{V_u}$ and is due to Lemma 11. ⌟

For computing $A^W_{V_u}$, we apply Lemma 10 and Lemma 14. In particular, depending on W, after retrieving all necessary values being stored on the corresponding descendants of v, we apply the appropriate formula, as follows:

- If $(\{u\} \cup W) \cap S \neq \emptyset$ and $|W| \geq 2$, then we apply $A^W_{V_u} = \bigcup_{u' \in \lhd u} A^{W \cap N(u')}_{V_{u'}}$.

  Proof: Lemma 14 implies that $u \notin A^W_{V_u}$. By Lemma 10 we get the above formula. ⌟

- If $(\{u\} \cup W) \cap S \neq \emptyset$ and $W = \{w\}$, then we apply $A^W_{V_u} = \max_{\text{weight}} \left\{ \bigcup_{u' \in \lhd u} A^{\{w\} \cap N(u')}_{V_{u'}}, \quad \{u\} \cup \bigcup_{u' \in \lhd uw} A^{\{u,w\} \cap N(u')}_{V_{u'}} \right\}$.

  Proof: The first case in the above formula is the case of $u \notin A^{\{w\}}_{V_u}$ and is due to Lemma 10. For the second case, assume that $u \in A^{\{w\}}_{V_u}$. Lemma 14 implies that $A^{\{w\}}_{V_u} \leftrightarrow \{u\} \cup A^{\{u,w\}}_{V_u \setminus (N[u] \cap N(w))}$. According to Lemma 8, the collection $\mathcal{V}_{\lhd uw}$ is a nice partition of $V_u \setminus (N[u] \cap N(w))$. By Observation 4 and Lemma 9 we get $A^{\{u,w\}}_{V_u \setminus (N[u] \cap N(w))} \leftrightarrow \bigcup_{u' \in \lhd uw} A^{\{u,w\}}_{V_{u'}} \leftrightarrow \bigcup_{u' \in \lhd uw} A^{\{u,w\} \cap N(V_{u'})}_{V_{u'}} \leftrightarrow \bigcup_{u' \in \lhd uw} A^{\{u,w\} \cap N(u')}_{V_{u'}}$. ⌟

- If $(\{u\} \cup W) \cap S = \emptyset$, then we apply $A^W_{V_u} = \max_{\text{weight}} \left\{ \bigcup_{u' \in \lhd u} A^{W \cap N(u')}_{V_{u'}}, \quad \{u\} \cup \bigcup_{u' \in \lhd u} A^{R_{\leq 2}((\{u\} \cup W) \cap N(u'))}_{V_{u'}} \right\}$.

  Proof: The first case in the above formula is the case of $u \notin A^{\{w\}}_{V_u}$ and is due to Lemma 10. For the second case, assume that $u \in A^W_{V_u}$. Then $A^W_{V_u} \leftrightarrow \{u\} \cup A^{\{u\} \cup W}_{V_u \setminus \{u\}}$. According to Lemma 8, the collection $\mathcal{V}_{\lhd u}$ is a nice partition of $V_u \setminus \{u\}$. By Observation 4, Lemma 9 and Lemma 14 we get $A^{\{u\} \cup W}_{V_u \setminus \{u\}} \leftrightarrow \bigcup_{u' \in \lhd u} A^{\{u\} \cup W}_{V_{u'}} \leftrightarrow \bigcup_{u' \in \lhd u} A^{(\{u\} \cup W) \cap N(V_{u'})}_{V_{u'}} \leftrightarrow \bigcup_{u' \in \lhd u} A^{(\{u\} \cup W) \cap N(u')}_{V_{u'}} \leftrightarrow \bigcup_{u' \in \lhd u} A^{R_{\leq 2}((\{u\} \cup W) \cap N(u'))}_{V_{u'}}$. ⌟

Regarding the correctness of the algorithm, we show that applying any of the above recursive formulas requires only sets that can also be computed via these formulas. Notice that an induced subgraph of a graph in $\mathcal{F}_S$ is also a graph in $\mathcal{F}_S$ and that a subset of a clique is also a clique. Now observe that applying any of the above recursive formulas requires only sets $A^{\emptyset}_{V_{u'}}$ and $A^{W'}_{V_{u'}}$ where $u' \in V_G$ and $W' \subseteq V_G \setminus V_{u'}$ such that $W' \neq \emptyset$, $G[W'] \in \mathcal{F}_S$ and $\{u'\} \cup W'$ is a clique. We conclude that all sets A^Y_X that are required for the application of any of these formulas can also be computed via these formulas.

We now analyze the running time of our algorithm. We begin by determining for every pair (x, y) of distinct nodes of the host tree T whether $x < y$ or not. As the act of discovering all nodes x that precede a node y in $\leq_T$ takes $O(n)$ time by traversing T once, we complete this task in $O(n^2)$ time. We then compute the sets $\lhd u$ and $\lhd uv$ for all $u \in V_G$ and for all $v \in V_G$ such that $uv \in E_G$. Since any such set can be computed in $O(n)$ time by traversing T once, we compute all such sets in $O(n(n+m))$ time, which is simply $O(nm)$ time as G is connected. Let us also bound the size of such sets. Observe that by definition any such set U is a set of pairwise-incomparable vertices. By definition of $\leq_G$, this implies that the set $V = \{r(u) \mid u \in U\}$ is a set of pairwise-incomparable nodes, yielding $|V| \leq |L(T)| = \ell$ by Observation 3. We conclude that any such set U contains at most ℓ vertices. We proceed

with the computation of the sets A_X^Y where $X, Y \subseteq V_G$ such that $X \cap Y = \emptyset$ and $G[Y] \in \mathcal{F}_S$. According to the recursive formulas shown above and due to the fact that $|R_{\leq 2}(U)| \leq 2\ell$ for all $U \subseteq V_G$, it is sufficient to compute for every $u \in V_G$ the sets A_X^Y where $X = V_u$ and either $Y = \emptyset$ or $Y \subseteq N(u) \setminus V_u$ such that $G[Y] \in \mathcal{F}_S$, $\{u\} \cup Y$ is a clique and $|Y| \leq 2\ell$. Therefore, it suffices to compute $O(n^{2\ell+1})$ sets A_X^Y. Now consider such a set A_X^Y. Its computation requires the retrieval of a number of stored values. Due to the bound on the size of the auxiliary sets shown above, that number is at most 2ℓ. If the set A_X^Y is computed via the last of the formulas shown above, we must also compute at most ℓ representations $R_{\leq 2}(U) = R_1(U) \cup R_2(U)$ of sets $U \subseteq V_G$ such that $|U| \leq 2\ell + 1$. Consider one such set U. Computing $R_1(U)$ (resp. $R_2(U)$) requires the computation of $\min L(U)$ (resp. $\min L(U \setminus R_1(U))$), which in turn requires determining for every pair (x, y) of distinct nodes in $L(U)$ (resp. $L(U \setminus R_1(U))$) whether $x < y$ or not. Since all these are predetermined, it suffices to retrieve a number of stored values. Recall that $|L(T_v)| \leq \ell$ for all $v \in V_G$. We get that $|L(U \setminus R_1(U))| \leq |L(U)| = \sum_{v \in U} |L(T_v)| \leq (2\ell + 1)\ell$, which implies that the number of stored values to be retrieved is $O(\ell^4)$. We conclude that the running time for computing the set A_X^Y is $O(\ell^5)$, which is constant time. Thus the total running time of our algorithm is $O(n^{2\ell+1})$. □

Notice that in the special case of $\ell = 1$, the number of sets A_X^Y that the algorithm of Theorem 15 computes is actually $O(nm)$ and consequently its total running time is $O(nm)$. If we let the leafage of a chordal graph be the maximum leafage over all of its connected components, then we obtain the following result.

Corollary 16. *The weighted* Subset Feedback Vertex Set *problem can be solved on chordal graphs with leafage at most ℓ in $n^{O(\ell)}$ time.*

Proof. Let G be a chordal graph. For every connected component C of G, we first recongize if C is an interval graph and if so construct a tree model $M(C)$ of C in linear time [6], otherwise we determine the leafage of C and construct a tree model $M(C)$ of C via the $O(n^3)$-time algorithm of Habib and Stacho [24]. We then construct an expanded tree model $M'(C)$ from $M(C)$ in $O(n^2)$ time by Lemma 5. Applying Theorem 15 on $M'(C)$, we compute $A_{V(C)}^{\emptyset}$ in $n^{O(\ell)}$ time. It is not difficult to see that the collection $\{V(C) \mid C \text{ is a connected component of } G\}$ is a nice partition of $V(G)$. Thus by Observation 4 the set $A_{V(G)}^{\emptyset}$ is the union of $A_{V(C)}^{\emptyset}$ over all of the connected components C of G. Therefore, all above steps result in solving SFVS on G and can be carried out in $n^{O(\ell)}$ time. □

Notice that in the special case of the input graph being an interval graph, the algorithm of Corollary 16 actually runs in $O(nm)$ time, which is also the running time of the previously known algorithm for solving SFVS on interval graphs [36]. We next prove that we can hardly avoid the dependence of the exponent in the stated running time, since we show that weighted Subset Feedback Vertex Set is W[1]-hard parameterized by the leafage of a chordal graph. Our reduction is inspired by the W[1]-hardness of Feedback Vertex Set parameterized by the mim-width given by Jaffke et al. [29].

Theorem 17. *The weighted* Subset Feedback Vertex Set *decision problem on chordal graphs is W[1]-hard when parameterized by its leafage.*

Proof. We provide a reduction from the Multicolored Clique problem. Given a graph $G = (V, E)$ and a partition $\{V_i\}_{i \in [k]}$ of V into k parts, the Multicolored Clique (MCC) problem asks whether G has a clique that contains exactly one vertex of V_i for every $i \in [k]$. It is known that MCC is W[1]-hard when parameterized by k [17, 39].

Let $(G = (V, E), \{V_i\}_{i \in [k]})$ be an instance of MCC. We assume that $k \geq 10$ and without loss of generality that there exists $p \in \mathbb{N}$ such that $V_i = \{v_i^j\}_{j \in [p]}$ for every $i \in [k]$. We consider

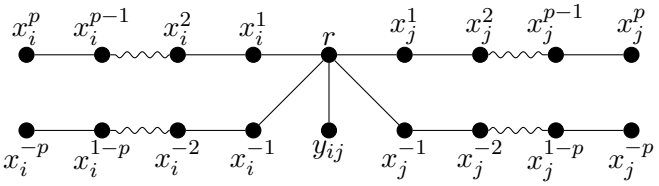

Figure 4: The subtree $T(\{x_i^+, x_i^-, y_{ij}, x_j^+, x_j^-\})$ of T for some $i, j \in [k]$ such that $i < j$.

the $\frac{k}{2}(k+3)$-star T with internal node r and leaves x_i^+, x_i^- for every $i \in [k]$ and y_{ij} for every $i, j \in [k]$ such that $i < j$. We modify the star T as follows: for every $i \in [k]$, through a series of edge subdivisions, we replace the edge $\langle r, x_i^+ \rangle$ by the path $\langle r = x_i^0, x_i^1, \ldots, x_i^p = x_i^+ \rangle$ and the edge $\langle r, x_i^- \rangle$ by the path $\langle r = x_i^0, x_i^{-1}, \ldots, x_i^{-p} = x_i^- \rangle$. Given a set X of nodes of T, we write $T(X)$ to denote the minimal subtree of T containing all nodes of X. The subtree $T(X)$ for a particular choice of X is depicted in Figure 4. We define the following subtrees of T:

- For every $i, j \in [k]$ such that $i < j$ and for every $a, b \in [p]$ such that $v_i^a v_j^b \in E$, we define $e_{ij}^{ab} = T(\{x_i^a, x_i^{a-p}, y_{ij}, x_j^b, x_j^{b-p}\})$. We denote by R the collection of all these subtrees.
  - For all $i \in [k]$, we denote by R_i the collection $\{e_{ij}^{ab} \in R \mid j \in [k] \text{ and } a, b \in [p]\} \cup \{e_{ji}^{ba} \in R \mid j \in [k] \text{ and } a, b \in [p]\}$.
  - For all $i \in [k]$ and for all $a \in [p]$, we denote by R_i^a the collection $\{e_{ij}^{ab} \in R \mid j \in [k] \text{ and } b \in [p]\} \cup \{e_{ji}^{ba} \in R \mid j \in [k] \text{ and } b \in [p]\}$.
  - For all $i, j \in [k]$ such that $i < j$, we denote by R_{ij} the collection $\{e_{ij}^{ab} \in R \mid a, b \in [p]\}$.
- For every $i \in [k]$ and $a \in [p]$, we define $s_i^{a,1} = s_i^{a,2} = T[\{x_i^a\}]$ and $s_i^{-a,1} = s_i^{-a,2} = T[\{x_i^{-a}\}]$. We denote by S_V the collection of all these subtrees.
  - For all $i \in [k]$, we denote by S_i the collection $\{s_i^{a,c} \in S_V \mid a \in -[p] \cup [p] \text{ and } c \in \{1, 2\}\}$.
  - For all $i \in [k]$ and for all $a \in [p]$, we denote by S_i^a the collection $\{s_i^{a',c} \in S_V \mid a' \in -[p-a] \cup [a] \text{ and } c \in \{1, 2\}\}$.
- For every $i, j \in [k]$ such that $i < j$, we define $s_{ij} = T[\{y_{ij}\}]$. We denote by S_E the collection of all these subtrees.

We further denote by S the collection $S_V \cup S_E$ and by $\mathcal{T}$ the collection $R \cup S$. We construct a graph G' that is the intersection graph of the undirected tree model $(T, \mathcal{T})$. Notice that G' is a chordal graph of leafage at most $\frac{k}{2}(k+3)$. We identify the vertices of G' with their corresponding subtrees in $\mathcal{T}$. By the construction of $(T, \mathcal{T})$, regarding the adjacencies between vertices of G' we observe the following:

- R is a clique, because all its elements contain the node r.
- For every $i \in [k]$ and $a \in -[p] \cup [p]$, we have $N(s_i^{a,1}) \cap S = \{s_i^{a,2}\}$ and $N(s_i^{a,2}) \cap S = \{s_i^{a,1}\}$.
- For every $i \in [k]$ and $a \in [p]$, we have $N(e) \cap S_i = S_i^a$ for all $e \in R_i^a$.
- For every $i, j \in [k]$ such that $i < j$, we have $N(s_{ij}) = R_{ij}$.

We set the weight of all vertices of R, S_V and S_E to be $\frac{p}{2}$, 1 and $\frac{p}{2}m$, respectively. We will show that $(G, \{V_i\}_{i \in [k]})$ is a YES-instance of MCC if and only if there exists a solution to SFVS on (G', S) having weight $\frac{p}{2}(m - \frac{k}{2}(k-9))$.

For the forward direction, let $\{v_1^{a_1}, \ldots, v_k^{a_k}\}$ be a solution of MCC on $(G, \{V_i\}_{i\in[k]})$. We set R_C to be the collection $\{e_{ij}^{a_i a_j} \in R \mid i, j \in [k]\}$. Observe that R_C contains exactly one element of R_{ij} for each $i, j \in [k]$ such that $i < j$. We further set $U = (R \setminus R_C) \cup \bigcup_{i\in[k]} S_i^{a_i}$. Now observe that in $G - U$ each of the remaining vertices of S has exactly one neighbour. Thus U is a solution to SFVS on (G', S) having weight $\frac{p}{2}(m - \frac{k}{2}(k-1)) + 2pk = \frac{p}{2}(m - \frac{k}{2}(k-9))$.

For the reverse direction, let U be a solution to SFVS on (G', S) having weight $\frac{p}{2}(m - \frac{k}{2}(k-9))$. Notice that no element of S_E can be in U. Consequently, for every $i, j \in [k]$ such that $i < j$, we have $|R_{ij} \setminus U| \leq 1$, since any two elements of R_{ij} along with s_{ij} form an S-triangle of G'. Any remaining S-triangle of G' is formed by either

- an element of R_i^a and two adjacent elements of S_i^a or
- an element of R_i^a, an element of $R_i^{a'}$ and an element of $S_i^a \cap S_i^{a'}$

for a particular choice of $i \in [k]$ and $a, a' \in [p]$. Let $i \in [k]$.

Claim 17.1. *If $|R_i \setminus U| \geq 1$, then $|S_i \cap U| \geq p$.*

Proof: Assume that $e \in R_i \setminus U$. Then there exists an $a \in [p]$ such that $e \in R_i^a$. We conclude that for every $a' \in -[p-a] \cup [a]$, at least one of $s_i^{a',1}, s_i^{a',2}$ must be in U, yielding $|S_i \cap U| \geq p$. ⌟

Claim 17.2. *If $|R_i \setminus U| \geq 2$, then $|S_i \cap U| \geq 2p$.*

Proof: Assume that e, e' are two distinct elements of $R_i \setminus U$. Then there exist $a, a' \in [p]$ such that $e \in R_i^a$ and $e' \in R_i^{a'}$. Without loss of generality, assume that $a \leq a'$. We conclude that

- for every $a'' \in -[p-a'] \cup [a]$ both $s_i^{a'',1}$ and $s_i^{a'',2}$ must be in U and
- for every $a'' \in (-[p-a] \cup [a']) \setminus (-[p-a'] \cup [a])$ at least one of $s_i^{a'',1}, s_i^{a'',2}$ must be in U,

yielding $|S_i \cap U| \geq 2((p-a')+a) + 1(((p-a)+a') - ((p-a')+a)) = 2p$. ⌟

Claim 17.3. *If $|R_i \setminus U| \geq 3$, then $|S_i \cap U| = 2p$ only if there exists $a \in [p]$ such that $R_i \setminus U \subseteq R_i^a$.*

Proof: Assume that e, e', e'' are three distinct elements of $R_i \setminus U$. Then there exist $a, a', a'' \in [p]$ such that and $e \in R_i^a$, $e' \in R_i^{a'}$ and $e'' \in R_i^{a''}$. Without loss of generality, assume that $a \leq a' \leq a''$. We conclude that

- for every $a''' \in -[p-a'] \cup [a']$ both $s_i^{a''',1}$ and $s_i^{a''',2}$ must be in U and
- for every $a''' \in (-[p-a] \cup [a'']) \setminus (-[p-a'] \cup [a'])$ at least one of $s_i^{a''',1}, s_i^{a''',2}$ must be in U,

yielding $|S_i \cap U| \geq 2p + 1(((p-a)+a'') - ((p-a')+a')) = 2p + (a''-a)$. Therefore, for $|S_i \cap U|$ to be $2p$, it must hold that $a = a' = a''$, so it must hold that $e, e', e'' \in R_i^a$. ⌟

Assume that $|\{i \in [k] \mid |R_i \setminus U| = 1\}| = k'$ and $|\{i \in [k] \mid |R_i \setminus U| \geq 2\}| = k''$. Then notice that $|R \setminus U| \leq k' + \frac{k''}{2}(k''-1)$, so $|R \cap U| \geq m - k' - \frac{k''}{2}(k''-1)$. Also, according to Claims 17.1 and 17.2, we have $|S_V \cap U| = \sum_{i\in[k]} |S_i \cap U| \geq p(k' + 2k'')$. Lastly, recall that $|S_E \cap U| = \emptyset$. Consequently, the weight of U must be at least

$$\frac{p}{2}\left(m - k' - \frac{k''}{2}(k''-1)\right) + p(k' + 2k'') = \frac{p}{2}\left(m + k' - \frac{k''}{2}(k''-9)\right) = B(k', k'').$$

Clearly, $k', k'' \in \{0, 1, \ldots, k\}$. Regarding the values of B, we observe the following:

- $B(k', k'') < B(k'+1, k'')$ for all $k' \in \{0, 1, \ldots, k-1\}$ and for all $k'' \in \{0, 1, \ldots, k\}$,

- $B(k', k'') \geq B(k', 9)$ for all $k' \in \{0, 1, \ldots, k\}$ and for all $k'' \in \{0, 1, \ldots, 8\}$, and
- $B(k', k'') > B(k', k'' + 1)$ for all $k' \in \{0, 1, \ldots, k\}$ and for all $k'' \in \{9, 10, \ldots, k-1\}$.

These imply that $B(k', k'')$ is minimum if and only if $k' = 0$ and $k'' = k$. Therefore, a weight of $\frac{p}{2}(m - \frac{k}{2}(k-9)) = B(0, k)$ is within bounds for U only if $k' = 0$ and $k'' = k$. Furthermore, the weight of U is $B(0, k)$ only if $|R \setminus U| = \frac{k}{2}(k-1)$ and $|S_i \cap U| = 2p$ for all $i \in [k]$. Now recall that $|R_{ij} \setminus U| \leq 1$ for all $i, j \in [k]$ such that $i < j$. We deduce that $|R_{ij} \setminus U| = 1$ for all $i, j \in [k]$ such that $i < j$, which implies that $|R_i \setminus U| = k - 1$ for all $i \in [k]$. Then, by Claim 17.3, for every $i \in [k]$, there exists an $a_i \in [p]$ such that $R_i \setminus U \subseteq R_i^{a_i}$. We conclude that the set $\{v_1^{a_1}, \ldots, v_k^{a_k}\}$ is a solution to MCC on $(G, \{V_i\}_{i \in [k]})$. □

5 SFVS on rooted path graphs

Here we show how to extend our previous approach to solving SFVS to rooted path graphs. Recall that rooted path graphs are exactly the intersection graphs of directed paths on a rooted tree. We observe that rooted path graphs are a graph class of unbounded leafage.

Proposition 18. *There are rooted path graphs on n vertices having leafage $\Theta(n)$.*

Proof. Let G be the graph obtained from the ℓ-star with $\ell \geq 2$ by subdividing all of its edges. Notice that $n = 2\ell + 1$. We show that G is a rooted path graph having leafage $\ell - 1$.

Let $v_1, \ldots, v_\ell$ be the leaf vertices of G, let $u_1, \ldots, u_\ell$ be the vertices of G such that $u_i v_i \in E(G)$ for every $i \in [\ell]$ and let t be the remaining vertex of G. To show that G is a rooted path graph, we construct a tree model $(T, \{T_v\}_{v \in V(G)})$ of G as follows. We obtain the host tree T as the union of paths $P_0 = (x_\ell, x_{\ell-1}, \ldots, x_1)$, $P_\ell = (y_\ell, x_\ell)$ and $P_i = (x_i, y_i)$, $i \in [\ell - 1]$. We choose the subtrees to be $T_t = P_0$, $T_{u_i} = P_i$ and $T_{v_i} = (y_i)$, $i \in [\ell]$. Notice that T is a tree rooted on y_ℓ such that $L(T) = \ell - 1$ and all T_v, $v \in V(G)$ are directed subpaths of T. It is not difficult to see that $(T, \{T_v\}_{v \in V(G)})$ is a tree model of G, which shows that G is indeed a rooted path graph.

Let us now show that for every tree model $(T', \{T'_v\}_{v \in V(G)})$ of G, it holds that $|L(T')| \geq \ell - 1$. The maximal cliques of G are $C_i = \{t, u_i\}$ and $D_i = \{u_i, v_i\}$, $i \in [\ell]$. For every maximal clique C of G, there exists a node $c \in V(T')$ such that for every $v \in V(G)$, it holds that $c \in V(T'_v) \Leftrightarrow v \in C$. Moreover, all these nodes are pairwise distinct. For every C_i and for every D_i, we select one such node and denote it by c_i and d_i respectively. For all $V \subseteq V(T')$, we define $\triangleright V$ to be the (unique) node of the set $\min\{y \in V(T') \mid \forall x \in V : x \leq y\}$. Observe that for every $v \in V(G)$, for every $V \subseteq V(T'_v)$, the node $\triangleright V$ is also in $V(T'_v)$ because T'_v is connected. For all $i \in [\ell]$, we denote by b_i the node $\triangleright\{c_i, d_i\}$ where $\{c_i, d_i\} \subseteq V(T'_{u_i})$. For all $i, j \in [\ell]$, we denote by a_{ij} the node $\triangleright\{c_i, c_j\}$ where $\{c_i, c_j\} \subseteq V(T'_t)$.

Claim 18.1. *For every $i, j \in [\ell]$, if $d_i < d_j$ holds, then $d_i < a_{ij} < d_j$ holds.*

Proof: For every $i, j \in [\ell]$, we have that the following relations hold:

$$c_i \leq b_i \qquad d_i \leq b_i \qquad c_j \leq b_j \qquad d_j \leq b_j \qquad c_i \leq a_{ij} \qquad c_j \leq a_{ij}$$

By Observation 1, we obtain that a_{ij} is comparable to both b_i and b_j. Assume that $d_i < d_j$ holds. By Observation 1, we obtain that b_i is comparable to both b_j and d_j. If $d_i < d_j \leq b_i$ holds, then by Observation 2, we obtain that $d_j \in V(T_{u_i})$, which is a contradiction, so $d_i \leq b_i < d_j \leq b_j$ must hold. Now by Observation 1, we obtain that a_{ij} is comparable to d_j. If $c_i \leq b_i < d_j \leq a_{ij}$ holds, then by Observation 2, we obtain that $d_j \in V(T'_t)$, which is a contradiction, so $a_{ij} < d_j$ must hold. Assume $a_{ij} \leq b_i$ holds. Then both $c_i \leq a_{ij} \leq b_i$ and $c_j \leq a_{ij} < b_j$ hold. By Observation 2, we obtain that $a_{ij} \in V(T'_{u_i}) \cap V(T'_{u_j})$, which is a contradiction to $(T', \{T'_v\}_{v \in V(G)})$ being a tree model of G. We conclude that $d_i < a_{ij} < d_j$ holds. ⌟

We will show that there are at least $\ell - 1$ pairwise incomparable nodes in $D = \{d_i\}_{i \in [\ell]}$. Assume that there exist $i, j, k \in [\ell]$ such that $d_i < d_j < d_k$ holds. Then by Claim 18.1, we obtain that $d_i < a_{ij} < d_j < a_{jk} < d_k$ holds, and by Observation 2, we obtain that $d_j \in V(T'_t)$, a contradiction. Now assume that there exist $i, j, k, l \in [\ell]$ such that both $d_i < d_j$ and $d_k < d_l$ hold and d_j and d_l are incomparable. Then by Claim 18.1, we obtain that both $d_i < a_{ij} < d_j$ and $d_k < a_{kl} < d_l$ hold. Let a be the node $\triangleright\{a_{ij}, a_{kl}\}$ where $\{a_{ij}, a_{kl}\} \subseteq V(T'_t)$. By Observation 1, we obtain that a is comparable to both d_j and d_l. If both $a \leq d_j$ and $a \leq d_l$ hold, then by Observation 1 we obtain that d_j and d_l are comparable, a contradiction, so at least one of $d_j < a$ and $d_l < a$ holds. Without loss of generality, assume that $d_j < a$ holds. Then by Observation 2, we obtain that $d_j \in V(T'_t)$, a contradiction. We conclude that there exists at most one node in D that succeeds another node in D, which implies that at least $\ell - 1$ nodes in D are pairwise incomparable. By Observation 3, we obtain that $|L(T')| \geq \ell - 1$, which concludes the proof. □

Our goal in this section is to device an algorithm for solving SFVS on rooted path graphs in polynomial time. Just as for the algorithm of Section 4, we will derive recursive formulas for optimal solutions A_X^Y and subsequently bound the number of optimal solutions to subproblems that the algorithm requires for solving the complete problem.

Assume that $G = (V_G, E_G)$ is a rooted path graph, $S \subseteq V_G$ and $(T, \{T_v\}_{v \in V_G})$ is an expanded tree model of G. For every $u \in V_G$, we denote by $l(u)$ the (unique) leaf of its corresponding directed path T_u. For devicing the algorithm of this section, further special vertices and subsets are required. For every $u, v \in V_G$ such that $u < v$, we define $u \triangleleft v$ to be the (unique) vertex of $\max(\{u\} \cup \{u' \in V_G \setminus S \mid u < u' < v \text{ and } u' \in N(u)\}) = \max(\{u\} \cup \{u' \in V_G \setminus S \mid l(u') < r(u) < r(u') < r(v)\}$. Moreover, for every $V_1, V_2, V_3 \subseteq V_G$, we define the following subsets of $V_G \setminus S$:

$$V[V_1;;] = \{u \in V_G \setminus S \mid \nexists v_1 \in V_1 : l(u) < r(v_1)\} \qquad V[V_1; V_2;] = V[V_1;;] \cap V[; V_2;]$$
$$V[; V_2;] = \{u \in V_G \setminus S \mid \exists v_2 \in V_2 : l(u) < r(v_2) < r(u)\} \qquad V[V_1;; V_3] = V[V_1;;] \cap V[;; V_3]$$
$$V[;; V_3] = \{u \in V_G \setminus S \mid \exists v_3 \in V_3 : r(u) \leq r(v_3)\} \qquad V[; V_2; V_3] = V[; V_2;] \cap V[;; V_3]$$

$$V[V_1; V_2; V_3] = V[V_1;;] \cap V[; V_2;] \cap V[;; V_3]$$

For any $u, v \in V_G$, we denote the set $V_u \cup V[; \{u\}; \{v\}]$ by $V_{u,v}$ for simplicity. Observe that the set $V_{u,u}$ is simply V_u.

Lemma 19. *Let $u, w \in V_G$ such that $u < w$ and $uw \in E_G$. Then the collection $\mathcal{V} = \{V[\triangleleft uw;; \triangleleft u]\} \cup \{V_{u', u' \triangleleft u}\}_{u' \in \triangleleft uw}$ is a nice partition of $X = (V_u \setminus \{u\}) \setminus (N(u) \cap N(w) \cap S)$ for every $Y \subseteq V_G \setminus X$ such that $Y \cap S = \emptyset$.*

Proof. We first show that $\mathcal{V}$ is a partition of X. Recall that $\triangleleft uw$ is a set of pairwise incomparable vertices by definition. Consider a vertex $u'' \in X$. Then exactly one of the following statements holds:

$$\begin{cases} \exists u' \in \triangleleft uw : r(u'') \leq r(u') \\ \exists u' \in \triangleleft uw : l(u'') < r(u') < r(u'') \\ \nexists u' \in \triangleleft uw : l(u'') < r(u') \end{cases} \Leftrightarrow \begin{cases} \exists u' \in \triangleleft uw : u'' \in V_{u'} \\ \exists u' \in \triangleleft uw : u'' \in V[; \{u'\}; \{u' \triangleleft u\}] \\ u'' \in V[\triangleleft uw;; \triangleleft u] \end{cases}$$

By Observation 1 and the definition of $\leq_G$, the vertex $u' \in \triangleleft uw$ in the first two cases above is unique, otherwise we obtain a contradiction to the vertices of $\triangleleft uw$ being pairwise incomparable. This fact implies our claim.

Now let $Y \subseteq V_G \setminus X$ such that $Y \cap S = \emptyset$ and consider an S-triangle S_t of $G[X \cup Y]$. Then there exists a vertex $u' \in \triangleleft uw$ such that $V(S_t) \cap V_{u'} \cap S \neq \emptyset$. Since S_t is a triangle, every vertex in $V(S_t) \setminus V_{u'}$ is adjacent to every vertex in $V(S_t) \cap V_{u'}$. Then by Lemma 9, for every vertex $u'' \in V(S_t) \setminus V_{u'}$, the vertex u'' is adjacent to u', which implies that $l(u'') < r(u') < r(u'')$ holds. We conclude that $V(S_t) \subseteq V_{u', u' \triangleleft u}$. □

Observation 20. *Let $X, Y \subseteq V_G$ such that $X \cap Y = \emptyset$ and $G[Y] \in \mathcal{F}_S$ and let $X' \subseteq X$. If $\mathcal{P}$ is a nice partition of X for Y, then $\mathcal{P}' = \{P \cap X' \mid P \in \mathcal{P} : P \cap X' \neq \emptyset\}$ is a nice partition of X' for Y.*

Proof. The fact that $\mathcal{P}'$ is a partition of X' follows trivially from the definition. For showing that $\mathcal{P}'$ is a nice partition of X' for Y, it suffices to notice that any S-triangle in $G[X' \cup Y]$ remains an S-triangle in $G[X \cup Y]$. □

We are now ready to show the recursive expressions that hold exclusively for G being a rooted path graph and are required by the algorithm of this section. First we obtain an expression to be used for the computation of sets A_X^Y in case of $X = V_u$ and $Y = \{w\}$ where $u, w \in V_G$ such that $u < w$ and $uw \in E_G$.

Lemma 21. *Let $u, w \in V_G$ such that $u < w$ and $uw \in E_G$, and let $u \in A_{V_u}^{\{w\}}$.*

- *If $u \in S$ or $w \in S$, then $A_{V_u}^{\{w\}} \leftrightarrow \{u\} \cup \bigcup_{u' \in \lhd uw} A_{V_{u'}}^{\{u,w\} \cap N(u')}$.*
- *If $u, w \notin S$, then $A_{V_u}^{\{w\}} \leftrightarrow \{u\} \cup V[\lhd uw; ; \lhd u] \cup \bigcup_{u' \in \lhd uw} A_{V_{u'}, u' \lhd u}^{\{u,w\} \cap N(u')}$.*

Proof. Observe that $A_{V_u}^{\{w\}} \leftrightarrow \{u\} \cup A_{V_u \setminus \{u\}}^{\{u,w\}}$ by definition. Regarding triangles of $G[V_u \cup \{w\}]$, we observe the following property:

(P1) By the hypothesis, the vertices u and w are adjacent. Thus, for any $u' \in V_u \cap N(u) \cap N(w)$, the vertex set $\{u', u, w\}$ induces a triangle.

If $u \in S$ or $w \in S$, then no vertex of $V_u \cap N(u) \cap N(w)$ is in $A_{V_u}^{\{w\}}$ because of (P1). By definition, we get $A_{V_u \setminus \{u\}}^{\{u,w\}} \leftrightarrow A_{V_u \setminus (N[u] \cap N(w))}^{\{u,w\}}$. According to Lemma 8, the collection $\mathcal{V}_{\lhd uw}$ is a nice partition of $V_u \setminus (N[u] \cap N(w))$. By Observation 4 and Lemma 9, we get $A_{V_u \setminus (N[u] \cap N(w))}^{\{u,w\}} \leftrightarrow \bigcup_{u' \in \lhd uw} A_{V_{u'}}^{\{u,w\}} \leftrightarrow \bigcup_{u' \in \lhd uw} A_{V_{u'}}^{\{u,w\} \cap N(V_{u'})} \leftrightarrow \bigcup_{u' \in \lhd uw} A_{V_{u'}}^{\{u,w\} \cap N(u')}$.

If $u, w \notin S$, then no vertex of $V_u \cap N(u) \cap N(w) \cap S$ is in $A_{V_u}^{\{w\}}$ because of (P1). Let $X = (V_u \setminus \{u\}) \setminus (N(u) \cap N(w) \cap S)$. By definition, we get $A_{V_u \setminus \{u\}}^{\{u,w\}} \leftrightarrow A_X^{\{u,w\}}$. According to Lemma 19, the collection $\{V[\lhd uw; ; \lhd u]\} \cup \{V_{u', u' \lhd u}\}_{u' \in \lhd uw}$ is a nice partition of X for $Y = \{u, w\}$. By Observation 4 and Lemma 9, we get $A_X^{\{u,w\}} \leftrightarrow A_{V[\lhd uw; ; \lhd u]}^{\{u,w\}} \cup \bigcup_{u' \in \lhd uw} A_{V_{u', u' \lhd u}}^{\{u,w\}} \leftrightarrow V[\lhd uw; ; \lhd u] \cup \bigcup_{u' \in \lhd uw} A_{V_{u', u' \lhd u}}^{\{u,w\} \cap N(V_{u'})} \leftrightarrow V[\lhd uw; ; \lhd u] \cup \bigcup_{u' \in \lhd uw} A_{V_{u', u' \lhd u}}^{\{u,w\} \cap N(u')}$. □

We next obtain recursive expressions to be used for the computation of sets A_X^Y in case of $X = V_{u,v}$ where $u \in V_G$ and $v \in V_G \setminus S$ such that $u < v$ and $uv \in E_G$. The first two Lemmas follow directly from the definition of sets A_X^Y and the fact that $V_{u,v} \setminus \{v\} = V_{u, u \lhd v}$ for every $u, v \in V_G$ such that $u < v$.

Lemma 22. *Let $u \in V_G$ and $v \in V_G \setminus S$ such that $u < v$ and $uv \in E_G$ and let $Y \subseteq V_G \setminus V_{u,v}$. If $v \notin A_{V_{u,v}}^Y$, then $A_{V_{u,v}}^Y \leftrightarrow A_{V_{u, u \lhd v}}^Y$.*

Lemma 23. *Let $u \in V_G$ and $v \in V_G \setminus S$ such that $u < v$ and $uv \in E_G$. If $v \in A_{V_{u,v}}^{\emptyset}$, then $A_{V_{u,v}}^{\emptyset} \leftrightarrow \{v\} \cup A_{V_{u, u \lhd v}}^{\{v\}}$.*

Lemma 24. *Let $u \in V_G$ and $v, w \in V_G \setminus S$ such that $u < v < w$ and $\{u, v, w\}$ is a clique and let $v \in A_{V_{u,v}}^{\{w\}}$. Then $A_{V_{u,v}}^{\{w\}} \leftrightarrow \{v\} \cup V[\lhd vw; \{u\}; \{u \lhd v\}] \cup \bigcup_{u' \in V_u \cap \lhd vw} A_{V_{u', u' \lhd v}}^{\{v,w\} \cap N(u')}$.*

Proof. Observe that $A^{\{w\}}_{V_{u,v}} \leftrightarrow \{v\} \cup A^{\{v,w\}}_{V_{u,v}\setminus\{v\}} \leftrightarrow \{v\} \cup A^{\{v,w\}}_{V_{u,u\lhd v}}$ by definition. Regarding triangles of $G[V_{u,v} \cup \{w\}]$, we observe the following property:

(P2) By the hypothesis, the vertices v and w are adjacent. Thus, for any $u' \in V_{u,v} \cap N(v) \cap N(w)$, the vertex set $\{u', v, w\}$ induces a triangle.

Since $v, w \notin S$, no vertex of $V_{u,v} \cap N(v) \cap N(w) \cap S$ is in $A^{\{w\}}_{V_{u,v}}$ because of (P2). Let $X = (V_v \setminus \{v\}) \setminus (N(v) \cap N(w) \cap S)$ and $X' = V_{u,u\lhd v} \setminus (N(v) \cap N(w) \cap S)$. By definition, we get $A^{\{v,w\}}_{V_{u,u\lhd v}} \leftrightarrow A^{\{v,w\}}_{X'}$. Now notice that $X' \subseteq X$. According to Lemma 19 and Observation 20, the collection $\{V[\lhd vw; \{u\}; \{u\lhd v\}]\} \cup \{V_{u',u'\lhd v}\}_{u' \in V_u \cap \lhd vw}$ is a nice partition of X' for $Y = \{v, w\}$. By Observation 4 and Lemma 9, we get $A^{\{v,w\}}_{X'} \leftrightarrow A^{\{v,w\}}_{V[\lhd vw;\{u\};\{u\lhd v\}]} \cup \bigcup_{u' \in V_u \cap \lhd vw} A^{\{v,w\}}_{V_{u',u'\lhd v}} \leftrightarrow V[\lhd vw; \{u\}; \{u\lhd v\}] \cup \bigcup_{u' \in V_u \cap \lhd vw} A^{\{v,w\} \cap N(V_{u'})}_{V_{u',u'\lhd v}} \leftrightarrow V[\lhd vw; \{u\}; \{u\lhd v\}] \cup \bigcup_{u' \in V_u \cap \lhd vw} A^{\{v,w\} \cap N(u')}_{V_{u',u'\lhd v}}$. □

Now we are in position to state our claimed result, which is obtained via an algorithm similar to the one in the proof of Theorem 15.

Theorem 25. *The weighted* Subset Feedback Vertex Set *problem can be solved on rooted path graphs in $O(n^2m)$ time.*

Proof. We first describe the algorithm. Given a rooted path graph $G = (V_G, E_G)$, we construct a tree model $(T, \{T_v\}_{v \in V_G})$ of G such that all subtrees T_v, $v \in V_G$ are directed paths in $O(n+m)$ time [16, 22]. We apply the iterative procedure described in the proof of Lemma 5 and obtain an expanded tree model $(T', \{T'_v\}_{v \in V_G})$ of G such that all subtrees T'_v, $v \in V_G$ are directed paths in $O(n^2)$ time. As the host tree T of G has at most n nodes [11, 24], the expanded host tree T' has $O(n)$ nodes. If G is an interval graph, then SFVS can be solved via the algorithm described in the proof of Corollary 16 in $O(nm)$ time. Otherwise, we solve SFVS by computing $A^{\emptyset}_{V_{u_{\max}}}$ where $u_{\max}$ is the (unique) vertex of $\max V_G$.

For this purpose, we device a dynamic programming algorithm for computing $A^{\emptyset}_{V_{u_{\max}}}$. The algorithm works on T' traversing it in a bottom-up fashion starting from its leaves and moving towards its root. It maintains tables for storing the values of computed sets A^Y_X in the following four cases of X and Y:

- $X = V_u$ and $Y = \emptyset$ for every $u \in V_G$.
- $X = V_u$ and $Y = \{w\}$ for every $u, w \in V_G$ such that $u < w$ and $uw \in E_G$.
- $X = V_{u,v}$ and $Y = \emptyset$ for every $u \in V_G$ and $v \in V_G \setminus S$ such that $u < v$ and $uv \in E_G$.
- $X = V_{u,v}$ and $Y = \{w\}$ for every $u \in V_G$ and $v, w \in V_G \setminus S$ such that $u < v < w$ and $\{u, v, w\}$ is a clique.

For computing these sets, we derive the following recursive formulas:

- Let $u \in V_G$. Lemma 10 and Lemma 11 imply that

$$A^{\emptyset}_{V_u} = \max_{\text{weight}} \left\{ \bigcup_{u' \in \lhd u} A^{\emptyset}_{V_{u'}}, \quad \{u\} \cup \bigcup_{u' \in \lhd u} A^{\{u\} \cap N(u')}_{V_{u'}} \right\}.$$

- Let $u, w \in V_G$ such that $u < w$ and $uw \in E_G$. Lemma 10 and Lemma 21 imply the following:

– If $u \in S$ or $w \in S$, then

$$A_{V_u}^{\{w\}} = \max_{\text{weight}} \left\{ \bigcup_{u' \in \lhd u} A_{V_{u'}}^{\{w\} \cap N(u')}, \quad \{u\} \cup \bigcup_{u' \in \lhd uw} A_{V_{u'}}^{\{u,w\} \cap N(u')} \right\}.$$

– If $u, w \notin S$, then

$$A_{V_u}^{\{w\}} = \max_{\text{weight}} \left\{ \bigcup_{u' \in \lhd u} A_{V_{u'}}^{\{w\} \cap N(u')}, \quad \{u\} \cup V[\lhd uw; ; \lhd u] \cup \bigcup_{u' \in \lhd uw} A_{V_{u', u' \lhd u}}^{\{u,w\} \cap N(u')} \right\}.$$

- Let $u \in V_G$ and $v \in V_G \setminus S$ such that $u < v$ and $uv \in E_G$. Lemma 22 and Lemma 23 imply that

$$A_{V_{u,v}}^{\emptyset} = \max_{\text{weight}} \left\{ A_{V_{u,u \lhd v}}^{\emptyset}, \quad \{v\} \cup A_{V_{u,u \lhd v}}^{\{v\}} \right\}.$$

- Let $u \in V_G$ and $v, w \in V_G \setminus S$ such that $u < v < w$ and $\{u, v, w\}$ is a clique. Lemma 22 and Lemma 24 imply that

$$A_{V_{u,v}}^{\{w\}} = \max_{\text{weight}} \left\{ A_{V_{u,u \lhd v}}^{\{w\}}, \quad \{v\} \cup V[\lhd vw; \{u\}; \{u \lhd v\}] \cup \bigcup_{u' \in V_u \cap \lhd vw} A_{V_{u', u' \lhd v}}^{\{v,w\} \cap N(u')} \right\}.$$

Regarding the correctness of the algorithm, observe that applying any of the above recursive formulas requires only sets A_X^Y that can also be computed via these formulas.

To evaluate the running time of the algorithm, we assume that the input graph is a connected rooted path graph. If not, observe that we can simply run our algorithm on each connected component of the input graph and subsequently combine all output solutions into a solution for the input graph. Notice that the number of sets A_X^Y described above that the algorithm computes and subsequently stores their values in corresponding table entries is $O(nm)$. Computing a single such set A_X^Y via any of the recursive formulas shown above requires the retrieval of stored values from $O(n)$ entries. These entries are determined via precomputed auxiliary objects. For some of these formulas, their application additionally requires the computation of a set $V[V_1; V_2; V_3]$. It is not difficult to see that any such set can be computed via a single transversal of the host tree T'. As there are $O(n)$ nodes in T', traversing it once takes $O(n)$ time. Thus the total processing time is $O(n^2 m)$. Now consider the aforementioned auxiliary objects: they are the vertex sets $\lhd v$, $\lhd vw$ and $V_u \cap \lhd vw$ and the vertices $u \lhd v$ for appropriate $u, v, w \in V_G$. For computing the vertex sets $\lhd v$, it is sufficient the traverse T' once for every $v \in V_G$. Similarly, for computing the vertex sets $\lhd vw$, it is sufficient to traverse T' once for every $v, w \in V_G$ such that $v < w$ and $vw \in E_G$, and for computing the vertices $u \lhd v$, it is sufficient to traverse T' once for every $u, v \in V_G$ such that $u < v$ and $uv \in E_G$. We also determine for every pair (x, y) of distinct nodes of T' whether $x < y$ or not. As mentioned in the proof of Theorem 15, this can be accomplished in $O(n^2)$ time. Then for every $u, v, w \in V_G$ such that $u < v < w$ and $\{u, v, w\}$ is a clique, we compute the vertex set $V_u \cap \lhd vw$ in $O(n)$ time by checking for every $u' \in \lhd vw$ whether $u' \leq u$. Thus the total preprocessing time is $O(n^2 m)$. Therefore, the total running time of our algorithm is $O(n^2 m)$. □

6 SFVS on undirected path graphs

The results of Theorem 15 and Corollary 16 motivate us to investigate whether our approach can be further extended to provide similar results on larger classes of chordal graphs. The class of graphs with bounded vertex leafage is a natural candidate to consider for such an investigation. However we show that Subset Feedback Vertex Set is NP-complete on undirected path graphs which are a subclass of graphs with vertex leafage at most two. In particular, we provide

a polynomial reduction from the NP-complete Max Cut problem. Given a graph G, the Max Cut problem concerns the finding of a partition of $V(G)$ into two sets A and $\overline{A}$ such that the number of edges with one endpoint in A and the other one in $\overline{A}$ is maximum among all such partitions. For two disjoint sets of vertices X and Y, we denote by $E(X,Y)$ the set $\{\{x,y\} \mid x \in X \text{ and } y \in Y\}$. The *cut-set* of a set $A \subseteq V(G)$ in G is the set of edges of G with exactly one endpoint in A, which is $E(A, V(G) \setminus A) \cap E(G)$. In such terminology, Max Cut concerns the finding of a set $A \subseteq V(G)$ such that its cut-set in G is of maximum size. The Max Cut problem is known to be NP-hard on general graphs [30] and to remain NP-hard even when the input graph is restricted to be a split or 3-colorable or undirected path graph [3]. We mention that our reduction is based on Max Cut on general graphs.

Towards the claimed reduction, to any graph G on n vertices and m edges, we will associate a graph H_G on $12n^2+4n+2m$ vertices. First we describe the vertex set of H_G. For every vertex $v \in V(G)$, we consider the following sets:

- $X(v) = \{x_v^1, x_v^2, \ldots, x_v^{2n}\}$ and $\overline{X}(v) = \{\overline{x}_v^1, \overline{x}_v^2, \ldots, \overline{x}_v^{2n}\}$,
- $Y(v) = \{y_v^1, y_v^2, \ldots, y_v^{2n+1}\}$ and $\overline{Y}(v) = \{\overline{y}_v^1, \overline{y}_v^2, \ldots, \overline{y}_v^{2n+1}\}$,
- $Z(v) = \{z_v^1, z_v^2, \ldots, z_v^{2n+1}, \overline{z}_v^1, \overline{z}_v^2, \ldots, \overline{z}_v^{2n+1}\}$, and
- $W(v) = \{(v,v') \mid \{v,v'\} \in E(G)\}$.

We consider all these sets to be pairwise-disjoint. The vertex set of H_G is precisely the union of all these sets. Notice that for every edge $\{u,v\} \in E(G)$, the ordered pairs (u,v) and (v,u) are both vertices of H_G. We denote by $\overline{W}(v)$ the set $\{(v',v) \mid \{v',v\} \in E(G)\}$. The edge set of H_G contains precisely the following:

- all edges required for the set $\bigcup_{v \in V(G)}(Y(v) \cup \overline{Y}(v) \cup W(v))$ to be a clique and
- for every vertex $v \in V(G)$:
  - all elements of the sets $E(X(v), Y(v))$, $E(\overline{X}(v), \overline{Y}(v))$, $E(X(v), W(v))$, $E(\overline{X}(v), \overline{W}(v))$,
  - for every $i \in [n]$, the edges $\{x_v^i, x_v^{n+i}\}, \{\overline{x}_v^i, \overline{x}_v^{n+i}\}$, and
  - for every $j \in [2n+1]$, the edges $\{y_v^j, z_v^j\}, \{y_v^j, \overline{z}_v^j\}, \{\overline{y}_v^j, z_v^j\}, \{\overline{y}_v^j, \overline{z}_v^j\}$.

Observe that x_v^i, x_v^{n+i} are true twins and $\overline{x}_v^i, \overline{x}_v^{n+i}$ are true twins, whereas $z_v^j, \overline{z}_v^j$ are false twins. This completes the construction of H_G. An example of a graph G and its associated graph H_G is given in Figure 5.

Lemma 26. *For any graph G, the graph H_G is an undirected path graph.*

Proof. In order to show that H_G is an undirected path graph, we construct an undirected tree $T(H_G)$ such that the vertices of H_G correspond to particular paths of $T(H_G)$. To distinguish the vertex sets between G and $T(H_G)$, we refer to the vertices of $T(H_G)$ as nodes. In order to construct $T(H_G)$, starting from a particular node r, for every vertex $v \in V(G)$, we consider the following paths:

- $P_X(v) = \langle r, x_1^{(v)}, \ldots, x_n^{(v)} \rangle$ and $P_{\overline{X}}(v) = \langle r, \overline{x}_1^{(v)}, \ldots, \overline{x}_n^{(v)} \rangle$ and
- for every $j \in [2n+1]$, $P_Z(v,j) = \langle r, z_1^{(v,j)}, z_2^{(v,j)} \rangle$.

Notice that the initial node r is contained in all these paths. We consider them to be otherwise pairwise-disjoint. The tree $T(H_G)$ is precisely the union of all these paths. Next, we describe the paths of $T(H_G)$ that correspond to the vertices of H_G.

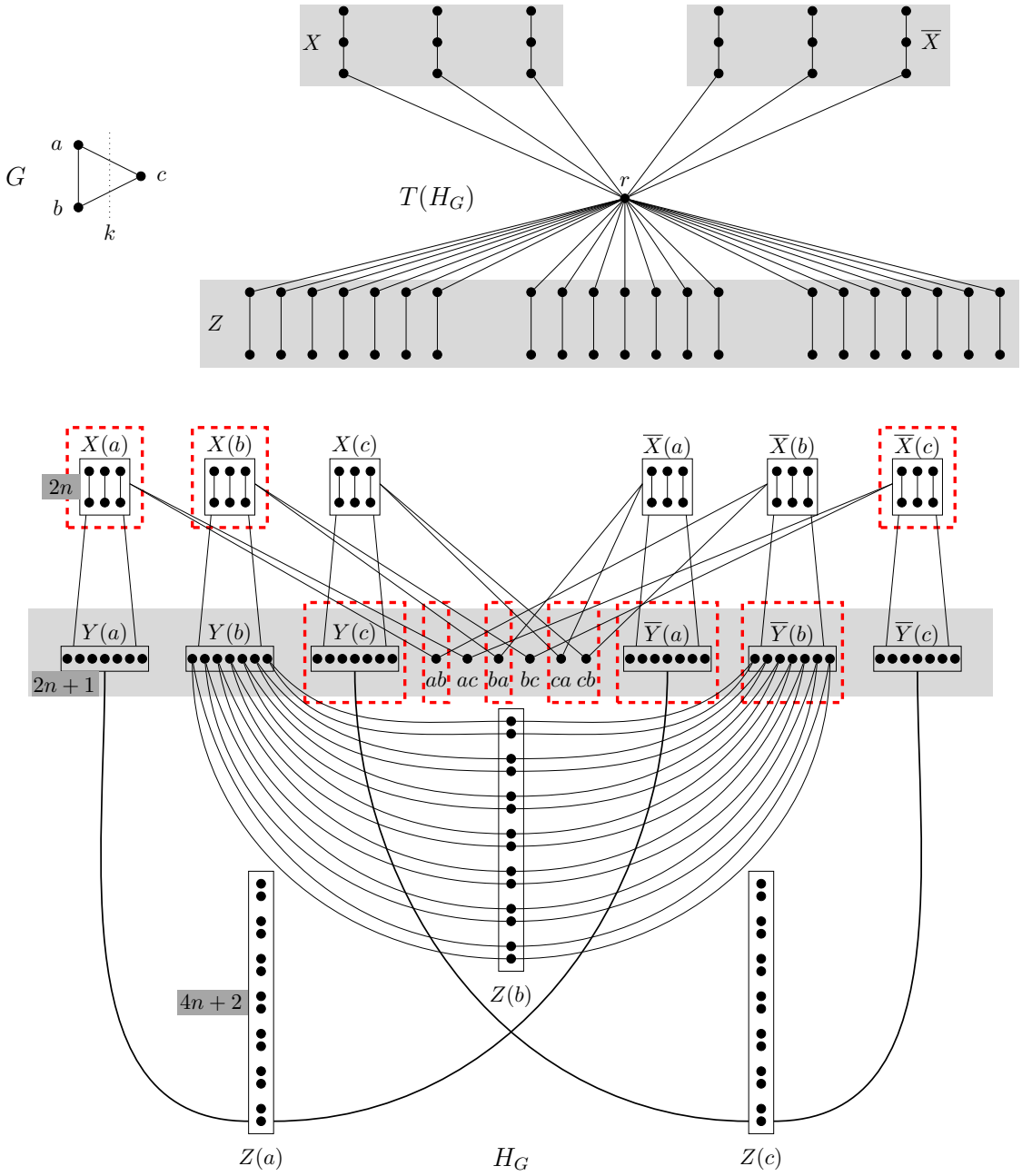

Figure 5: Illustration of a graph G on three vertices (top left) and its associated undirected path graph H_G (bottom). We also show the tree $T(H_G)$ described in the proof of Lemma 26 (top right). The vertices of H_G that lie in the gray area form a clique.

- For every $v \in V(G)$ and $i \in [n]$, to each of the vertices x_v^i, x_v^{n+i} (resp. $\overline{x}_v^i, \overline{x}_v^{n+i}$) we correspond the path $\langle x_i^{(v)} \rangle$ (resp. $\langle \overline{x}_i^{(v)} \rangle$).
- For every $v \in V(G)$ and $j \in [2n+1]$,
  - to the vertices y_v^j and $\overline{y}_v^j$ we correspond the paths $\langle x_n^{(v)}, \ldots, x_1^{(v)}, r, z_1^{(v,j)}, z_2^{(v,j)} \rangle$ and $\langle \overline{x}_n^{(v)}, \ldots, \overline{x}_1^{(v)}, r, z_1^{(v,j)}, z_2^{(v,j)} \rangle$) respectively, and
  - to the vertices z_v^j and $\overline{z}_v^j$ we correspond the paths $\langle z_1^{(v,j)} \rangle$ and $\langle z_2^{(v,j)} \rangle$ respectively.

- For every $\{u, v\} \in E(G)$, to the vertices (u, v) and (v, u) we correspond the paths $\langle x_n^{(u)}, \ldots, x_1^{(u)}, r, \overline{x}_1^{(v)}, \ldots, \overline{x}_n^{(v)} \rangle$ and $\langle x_n^{(v)}, \ldots, x_1^{(v)}, r, \overline{x}_1^{(u)}, \ldots, \overline{x}_n^{(u)} \rangle$ respectively.

Now it is not difficult to see that the intersection graph of the collection that contains precisely all these paths is isomorphic to H_G. Observe that all paths containing node r correspond to the vertices of the clique $\bigcup_{v \in V(G)} Y(v) \cup \overline{Y}(v) \cup W(v)$ and for every $v \in V(G)$, all paths that are subpaths of $P_X(v)$ and $P_{\overline{X}}(v)$ correspond to the vertices of $X(v)$ and $\overline{X}(v)$ respectively, and all paths that are subpaths of $P_Z(v, j)$, $j \in [2n + 1]$ correspond to the vertices of $Z(v)$. Therefore, H_G is an undirected path graph. □

Let us now show that to any cut-set in G, there is an associated subset feedback vertex set in H_G. We first introduce some additional notation: $X = \bigcup_{v \in V(G)} X(v)$, $\overline{X} = \bigcup_{v \in V(G)} \overline{X}(v)$, $Y = \bigcup_{v \in V(G)} Y(v)$, $\overline{Y} = \bigcup_{v \in V(G)} \overline{Y}(v)$, $Z = \bigcup_{v \in V(G)} Z(v)$, $W = \bigcup_{v \in V(G)} W(v) = \bigcup_{v \in V(G)} \overline{W}(v)$ and for every $A, B \subseteq V(G)$, $W(A, B) = (\bigcup_{a \in A} W(a)) \cap (\bigcup_{b \in B} \overline{W}(b))$. For every $A \subseteq V(G)$, we denote by $\overline{A}$ the set $V(G) \setminus A$ for simplicity. We also define the vertex set

$$U(A) = \left(\bigcup_{v \in A} (X(v) \cup \overline{Y}(v)) \right) \cup \left(\bigcup_{v \in \overline{A}} (\overline{X}(v) \cup Y(v)) \right) \cup (W \setminus W(A, \overline{A})).$$

Observe that $|U(A)| = n(2n + (2n + 1)) + (2m - |W(A, \overline{A})|) = 4n^2 + n + 2m - |W(A, \overline{A})|$ and $|W(A, \overline{A})|$ is the size of the cut-set of A in G.

Lemma 27. *Let G be a graph and let $A \subseteq V(G)$. Then $U(A)$ is a subset feedback vertex set of (H_G, S) where $S = X \cup \overline{X} \cup Z$.*

Proof. We show that the undirected path graph $H_G - U(A)$ is an S-forest. Assume for contradiction that there is an S-triangle S_t in $H_G - U(A)$. Then S_t contains at least one vertex of S. We consider the following three cases:

- Let $x \in V(S_t) \cap X$. Then there exists $v \in V(G)$ such that $x \in X(v)$. Since $X(v') \subseteq U(A)$ for every $v' \in A$, the vertex v must be in $\overline{A}$. By the construction of H_G, any vertex of $X(v)$ has exactly one neighbor in $X(v)$. Thus there exists $y \in V(S_t)$ such that $y \notin X(v)$. Again by the construction of H_G, the neighborhood of any vertex of $X(v)$ in $H_G - X(v)$ is $Y(v) \cup W(v)$, which implies that $y \in Y(v) \cup W(v)$. Hence we reach a contradiction to the definition of $U(A)$, since $Y(v) \subseteq U(A)$ and $W(v) \subseteq W \setminus W(A, \overline{A}) \subseteq U(A)$.

- Let $\overline{x} \in V(S_t) \cap \overline{X}$. Arguments that are completely symmetrical to the ones employed in the previous case yield a contradiction to the definition of $U(A)$.

- Let $z \in V(S_t) \cap Z$. Then there exists $v \in V(G)$ such that $z \in Z(v)$ and by the construction of H_G, there exist $y \in Y(v)$ and $\overline{y} \in \overline{Y}(v)$ such that $N(z) = \{y, \overline{y}\}$. Thus $V(S_t)$ must be $\{z, y, \overline{y}\}$, which implies that $y, \overline{y} \notin U(A)$. Hence we reach a contradiction to the definition of $U(A)$, since either $Y(v) \subseteq U(A)$ and $\overline{Y}(v) \cap U(A) = \emptyset$ or vice versa.

Since we obtained a contradiction in all three cases, we conclude that there is no S-triangle in $H_G - U(A)$. □

Now we are ready to show the main result of this section. Its forward direction follows from the previous lemma. Its reverse direction is obtained through a series of claims. These claims imply a procedure which progressively reconfigures any arbitrary initial subset feedback vertex set of H_G until it becomes one that is associated to a cut-set of G.

Theorem 28. *The unweighted* Subset Feedback Vertex Set *decision problem is* NP*-complete on undirected path graphs.*

Proof. We provide a polynomial reduction from the NP-complete Max Cut problem. Given a graph G on n vertices and m edges for the Max Cut problem, we construct the graph H_G. Observe that the size of H_G is polynomial and the construction of H_G can be done in polynomial time. By Lemma 26, H_G is an undirected path graph. We set $S = X \cup \overline{X} \cup Z$. We claim that G admits a cut-set of size at least k if and only if (H_G, S) admits a subset feedback vertex set of size at most $4n^2 + n + 2m - k$.

Lemma 27 provides the forward direction. Here we show the reverse direction. For every $U \subseteq V(H_G)$, we denote by $\overline{U}$ the set $V(H_G) \setminus U$ for simplicity. We also define the vertex set $A_U = \{v \in V(G) \mid X(v) \subseteq U\}$. Let U be a subset feedback vertex set of (H_G, S). Then it is not difficult to see that $U(A_U) = U$ holds if and only if U satisfies the following four properties:

(1) $Z \subseteq \overline{U}$.

(2) For all $v \in V(G)$, either $X(v) \subseteq U$ or $X(v) \subseteq \overline{U}$, and either $\overline{X}(v) \subseteq U$ or $\overline{X}(v) \subseteq \overline{U}$.

(3) For all $v \in V(G)$, either $X(v) \cup \overline{Y}(v) \subseteq U$ and $\overline{X}(v) \cup Y(v) \subseteq \overline{U}$, or vice versa.

(4) $W \setminus W(A_U, \overline{A_U}) \subseteq U$ and $W(A_U, \overline{A_U}) \subseteq \overline{U}$.

For every $i \in \{0, 1, \ldots, 4\}$, we say that a subset of $V(H_G)$ is a *tier-i sfvs* if it is a subset feedback vertex set of (H_G, S) that satisfies the first i properties listed above. Notice that if a subset of $V(H_G)$ is a tier-i sfvs, then it is a tier-j sfvs for all $j \in \{0, 1, \ldots, i\}$. Also notice that any subset feedback vertex set of (H_G, S) is a tier-0 sfvs. We will now show through a series of claims that for every tier-0 sfvs, there exists a tier-4 sfvs of at most equal size.

Claim 28.1. *For every tier-0 sfvs U, there exists a tier-1 sfvs U' such that $|U'| \leq |U|$.*

Proof: Let U be a tier-0 sfvs. If $Z \cap U = \emptyset$, then U is already a tier-1 sfvs. Assume otherwise. Then there exist $v \in V(G)$ and $j \in [2n+1]$ such that $\{z_v^j, \overline{z}_v^j\} \cap U \neq \emptyset$. We construct the set $U' = (U \setminus \{z_v^j, \overline{z}_v^j\}) \cup \{\overline{y}_v^j\}$. Notice that $|\{z_v^j, \overline{z}_v^j\} \cap U| \geq 1$ and $|\{\overline{y}_v^j\} \setminus U| \leq 1$, yielding $|U'| \leq |U|$. Observe that by the construction of H_G, the neighborhoods of z_v^j and $\overline{z}_v^j$ in $H_G - U'$ are the same subset of $\{y_v^j\}$, thus neither z_v^j nor $\overline{z}_v^j$ is a vertex of any triangle of $H_G - U'$. As z_v^j and $\overline{z}_v^j$ are the only vertices being removed from a tier-0 sfvs, this implies that U' is also a tier-0 sfvs. Iteratively following this argumentation for every $v \in V(G)$ and $j \in [2n+1]$ such that $\{z_v^j, \overline{z}_v^j\} \cap U \neq \emptyset$, we obtain a tier-1 sfvs U' such that $|U'| \leq |U|$. ⌟

Claim 28.2. *For every tier-1 sfvs U and $v \in V(G)$, it holds that $|(Y(v) \cup \overline{Y}(v)) \cap U| \geq 2n + 1$.*

Proof: Let U be a tier-1 sfvs. Consider a vertex $v \in V(G)$. Then $Z(v) \subseteq \overline{U}$. Since $Z(v) \subset S$ also holds, the triangles induced by the sets $\{y_v^j, \overline{y}_v^j, \overline{z}_v^j\}$, $j \in [2n+1]$ are $2n+1$ vertex-disjoint S-triangles of H_G. Therefore, at least $2n+1$ vertices of $Y(v) \cup \overline{Y}(v)$ must be in U, because $H_G - U$ is an S-forest. ⌟

Claim 28.3. *For every tier-1 sfvs U, there exists a tier-2 sfvs U' such that $|U'| \leq |U|$.*

Proof: Let U be a tier-1 sfvs. Consider a vertex $v \in V(G)$ such that both $X(v) \cap U \neq \emptyset$ and $X(v) \cap \overline{U} \neq \emptyset$. Assume that $x \in X(v) \cap \overline{U}$. By the construction of H_G, we have that $N(x) \setminus X(v) = Y(v) \cup W(v)$ is a clique. Since $X(v) \subset S$ and $H_G - U$ is an S-forest, at most one vertex of $Y(v) \cup W(v)$ is in $\overline{U}$. We construct the set $U' = (U \setminus X(v)) \cup (Y(v) \cup W(v))$. Notice that $|X(v) \cap U| \geq 1$ and $|(Y(v) \cup W(v)) \setminus U| \leq 1$, yielding $|U'| \leq |U|$. The set U' is a tier-1 sfvs. This follows from the fact that by the construction of H_G, there are no triangles in $H_G[X(v)]$ and $N(X(v)) = Y(v) \cup W(v) \subseteq U'$. Now consider a vertex $v \in V(G)$ such that both $\overline{X}(v) \cap U \neq \emptyset$ and

$\overline{X}(v) \cap \overline{U} \neq \emptyset$. Via completely symmetrical arguments, the set $U'' = (U' \setminus \overline{X}(v)) \cup (\overline{Y}(v) \cup \overline{W}(v))$ is a tier-1 sfvs such that $|U''| \leq |U'|$. Iteratively following the argumentations regarding all applicable cases for every $v \in V(G)$, we obtain a tier-2 sfvs U' such that $|U'| \leq |U|$. ⌟

Before we continue with our claims, we observe that for every tier-0 sfvs U and $v \in V(G)$, if $X(v) \subseteq \overline{U}$, then $Y(v) \cup W(v) \subseteq U$, and if $\overline{X}(v) \subseteq \overline{U}$, then $\overline{Y}(v) \cup \overline{W}(v) \subseteq U$. This follows from the facts that $H_G - U$ is an S-forest, $X(v) \cup \overline{X}(v) \subset S$ and by the construction of H_G, for every $y \in Y(v) \cup W(v)$ (resp. $\overline{y} \in \overline{Y}(v) \cup \overline{W}(v)$), the set $\{x_v^1, x_v^{n+1}, y\}$ (resp. $\{\overline{x}_v^1, \overline{x}_v^{n+1}, \overline{y}\}$) induces a triangle of H_G. In proving our remaining claims, we implicitly apply this observation.

Claim 28.4. *For every tier-2 sfvs U, there exists a tier-3 sfvs U' such that $|U'| \leq |U|$.*

Proof: Let U be a tier-2 sfvs. Consider a vertex $v \in V(G)$. Then exactly one of the following holds:

(1) $X(v) \cup \overline{X}(v) \subseteq \overline{U}$

(2) $X(v) \subseteq U$ and $\overline{X}(v) \subseteq \overline{U}$

(3) $X(v) \subseteq \overline{U}$ and $\overline{X}(v) \subseteq U$

(4) $X(v) \cup \overline{X}(v) \subseteq U$

Assume that (1) holds. Then $Y(v) \cup \overline{Y}(v) \subseteq U$ holds. We construct the set $U' = (U \setminus Y(v)) \cup X(v)$. Notice that $|Y(v) \cap U| = 2n+1$ and $|X(v) \setminus U| = 2n$, yielding $|U'| < |U|$. It is not difficult to show that the set U' is a tier-2 sfvs.

Now assume that (2) holds. Then $\overline{Y}(v) \subseteq U$ holds. We construct the set $U' = U \setminus Y(v)$. Clearly, $|U'| \leq |U|$. It is not difficult to show that the set U' is a tier-2 sfvs. Assuming that (3) holds, completely symmetrical arguments yield that the set $U' = U \setminus \overline{Y}(v)$ is a tier-2 sfvs such that $|U'| \leq |U|$.

Lastly assume that (4) holds. By Claim 28.2, we have $|(Y(v) \cup \overline{Y}(v)) \setminus U| \leq 2n+1$. Without loss of generality, assume that $|\overline{Y}(v) \setminus U| \leq n$. We construct the set $U' = (U \setminus (\overline{X}(v) \cup Y(v))) \cup (\overline{Y}(v) \cup \overline{W}(v))$. Notice that $|(\overline{X}(v) \cup Y(v)) \cap U| \geq 2n + 0 = 2n$ and $|(\overline{Y}(v) \cup \overline{W}(v)) \setminus U| \leq n + (n-1) < 2n$, yielding $|U'| < |U|$. It is not difficult to show that the set U' is a tier-2 sfvs.

Iteratively following the argumentation regarding the appropriate case for every $v \in V(G)$, we obtain a tier-3 sfvs $|U'|$ such that $|U'| \leq |U|$. ⌟

Claim 28.5. *For every tier-3 sfvs U, there exists a tier-4 sfvs U' such that $|U'| \leq |U|$.*

Proof: Let U be a tier-3 sfvs. Consider a vertex $w \in W$. Then there exist $u, v \in V(G)$ such that $w = (u, v)$ is the (unique) vertex of $W(u) \cap \overline{W}(v)$. Assume that $w \in W \setminus W(A_U, \overline{A_U})$. Then $u \notin A_U$ or $v \notin \overline{A_U}$, which implies that $X(u) \subseteq \overline{U}$ or $\overline{X}(v) \subseteq \overline{U}$. In both cases, it follows that $w \in U$. Now assume that $w \in W(A_U, \overline{A_U})$. Then $u \in A_U$ and $v \in \overline{A_U}$, which implies that $X(u) \subseteq U$ and $\overline{X}(v) \subseteq U$. By the construction of H_G, we have that $N(w) \cap S = X(u) \cup \overline{X}(v)$. It follows that the set $U' = U \setminus W(A_U, \overline{A_U})$ is a tier-4 sfvs such that $|U'| \leq |U|$. ⌟

To conclude our proof, we assume that U is a tier-0 sfvs such that $|U| \leq 4n^2+n+2m-k$. Due to Claims 28.1, 28.3–28.5, there exists a tier-4 sfvs U' such that $|U'| \leq |U|$. Then $U(A_{U'}) = U'$ holds. Recall that $|U(A_{U'})| = 4n^2 + n + 2m - |W(A_{U'}, \overline{A_{U'}})|$ and $|W(A_{U'}, \overline{A_{U'}})|$ is the size of the cut-set of $A_{U'}$ in G. All of the above imply that the size of the cut-set of $A_{U'}$ in G is at least k. □

7 Concluding Remarks

We provided a systematic and algorithmic study towards the classification of the complexity of Subset Feedback Vertex Set on subclasses of chordal graphs. We considered the structural parameters of leafage and vertex leafage as natural tools to exploit insights of the corresponding tree representation. Our proof techniques revealed a fast algorithm for the class of rooted path graphs. Naturally, it is interesting to settle whether the unweighted Subset Feedback Vertex Set problem is FPT when parameterized by the leafage of a chordal graph. We note that it is not unlikely for the unweighted and weighted variants of the problem to behave computationally differently in this case, since they do so in other cases [8, 37]. We also believe that our NP-hardness proof on undirected path graphs carries along the class of directed path graphs which are the intersection graphs of directed paths taken from a directed tree that has no constraint on the directions of its edges. Towards a more complete picture on the behavior of the problem on subclasses of chordal graphs, strongly chordal graphs are a candidate subclass for further investigation as they are incomparable to the subclasses of bounded leafage.

Furthermore, it would be interesting to consider the closely related Subset Odd Cycle Transversal problem in which the task is to hit all odd S-cycles. Notice that all of our results for Subset Feedback Vertex Set are still valid for Subset Odd Cycle Transversal, as in chordal graphs any induced cycle is a triangle which is an odd induced cycle. Preliminary results indicate that the two problems align on particular hereditary classes of graphs [7, 8]. Finally, an interesting direction for further research involving the parameters of leafage and vertex leafage is to consider induced path problems that are polynomially solvable and NP-hard on interval graphs and split graphs respectively [2, 25, 27, 34].

Acknowledgement

We thank Steven Chaplick for helpful remarks on a preliminary version. We also thank the reviewers for the careful reading of our manuscript and for useful comments.

Declarations

Conflict of Interest All authors declare no conflicts of interests.

References

[1] 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.

[2] Alan A. Bertossi and Maurizio A. Bonuccelli. Hamiltonian circuits in interval graph generalizations. *Inf. Process. Lett.*, 23(4):195–200, 1986.

[3] H. L. Bodlaender and K. Jansen. On the complexity of the maximum cut problem. *Nord. J. Comput.*, 7(1):14–31, 2000.

[4] J. Adrian Bondy and Uppaluri S. R. Murty. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2008.

[5] Kellogg S. Booth and J. Howard Johnson. Dominating sets in chordal graphs. *SIAM J. Comput.*, 11:191–199, 1982.

[6] Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *J. Comput. Syst. Sci.*, 13(3):335–379, 1976.

[7] Nick Brettell, Matthew Johnson, Giacomo Paesani, and Daniël Paulusma. Computing subset transversals in H-free graphs. *Theor. Comput. Sci.*, 902:76–92, 2022.

[8] Nick Brettell, Matthew Johnson, and Daniël Paulusma. Computing weighted subset transversals in H-free graphs. In *Algorithms and Data Structures - 17th International Symposium, WADS 2021, Proceedings*, volume 12808 of *Lecture Notes in Computer Science*, pages 229–242. Springer, 2021.

[9] P. Buneman. A characterization of rigid circuit graphs. *Discret. Math.*, 9:205–212, 1974.

[10] Steven Chaplick. Intersection graphs of non-crossing paths. In *Graph-Theoretic Concepts in Computer Science - 45th International Workshop, WG 2019, Revised Papers*, volume 11789 of *Lecture Notes in Computer Science*, pages 311–324. Springer, 2019.

[11] Steven Chaplick and Juraj Stacho. The vertex leafage of chordal graphs. *Discret. Appl. Math.*, 168:14–25, 2014.

[12] D. G. Corneil and J. Fonlupt. The complexity of generalized clique covering. *Discret. Appl. Math.*, 22(2):109–118, 1988.

[13] D. G. Corneil and Y. Perl. Clustering and domination in perfect graphs. *Discret. Appl. Math.*, 9(1):27–39, 1984.

[14] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

[15] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. Subset feedback vertex set is fixed-parameter tractable. *SIAM J. Discrete Math.*, 27(1):290–309, 2013.

[16] P. F. Dietz. *Intersection graph algorithms*. PhD thesis, Cornell University, 1984.

[17] M. R. Fellows, D. Hermelin, F. A. Rosamond, and S. Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410(1):53–61, 2009.

[18] F. V. Fomin, P. Heggernes, D. Kratsch, C. Papadopoulos, and Y. Villanger. Enumerating minimal subset feedback vertex sets. *Algorithmica*, 69(1):216–231, 2014.

[19] 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.

[20] M. R. Garey and D. S. Johnson. *Computers and Intractability*. W. H. Freeman and Co., 1978.

[21] Fănică Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *J. Comb. Theory, Ser. B*, 16(1):47–56, 1974.

[22] Fănică Gavril. A recognition algorithm for the intersection graphs of directed paths in directed trees. *Discret. Math.*, 13(3):237–249, 1975.

[23] P. A. Golovach, P. Heggernes, D. Kratsch, and R. Saei. Subset feedback vertex sets in chordal graphs. *J. Discrete Algorithms*, 26:7–15, 2014.

[24] Michel Habib and Juraj Stacho. Polynomial-time algorithm for the leafage of chordal graphs. In *Algorithms - ESA 2009, 17th Annual European Symposium, Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 290–300. Springer, 2009.

[25] Pinar Heggernes, Pim van 't Hof, Erik Jan van Leeuwen, and Reza Saei. Finding disjoint paths in split graphs. *Theory Comput. Syst.*, 57(1):140–159, 2015.

[26] E. C. Hols and S. Kratsch. A randomized polynomial kernel for subset feedback vertex set. *Theory Comput. Syst.*, 62:54–65, 2018.

[27] 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.

[28] Lars Jaffke, O-joung Kwon, Torstein J. F. Strømme, and Jan Arne Telle. Mim-width III. graph powers and generalized distance domination problems. *Theor. Comput. Sci.*, 796:216–236, 2019.

[29] Lars Jaffke, O-joung Kwon, and Jan Arne Telle. Mim-width II. the feedback vertex set problem. *Algorithmica*, 82(1):118–145, 2020.

[30] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[31] K. Kawarabayashi and Y. Kobayashi. Fixed-parameter tractability for the subset feedback set problem and the s-cycle packing problem. *J. Comb. Theory, Ser. B*, 102(4):1020–1034, 2012.

[32] 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.

[33] Clyde L. Monma and Victor K. Wei. Intersection graphs of paths in a tree. *J. Comb. Theory, Ser. B*, 41(2):141–181, 1986.

[34] Sridhar Natarajan and Alan P. Sprague. Disjoint paths in circular arc graphs. *Nord. J. Comput.*, 3(3):256–270, 1996.

[35] B. S. Panda. The separator theorem for rooted directed vertex graphs. *J. Comb. Theory, Ser. B*, 81(1):156–162, 2001.

[36] 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.

[37] Charis Papadopoulos and Spyridon Tzimas. Subset feedback vertex set on graphs of bounded independent set size. *Theor. Comput. Sci.*, 814:177–188, 2020.

[38] Geevarghese Philip, Varun Rajan, Saket Saurabh, and Prafullkumar Tale. Subset feedback vertex set in chordal and split graphs. *Algorithmica*, 81(9):3586–3629, 2019.

[39] K. Pietrzak. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *J. Comput. Syst. Sci.*, 67(4):757–771, 2003.

[40] Jeremy P. Spinrad. *Efficient graph representations*, volume 19 of *Fields Institute monographs*. American Mathematical Society, 2003.

[41] M. Yannakakis. Node-deletion problems on bipartite graphs. *SIAM J. Comput.*, 10(2):310–327, 1981.