Parameterized Max Min Feedback Vertex Set

Michael Lampis

Université Paris-Dauphine, PSL University, CNRS, LAMSADE, 75016, Paris, France

Nikolaos Melissinos

Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, Czech Republic

Manolis Vasilakis

Université Paris-Dauphine, PSL University, CNRS, LAMSADE, 75016, Paris, France

Abstract

Given a graph G and an integer k, Max Min FVS asks whether there exists a minimal set of vertices of size at least k whose deletion destroys all cycles. We present several results that improve upon the state of the art of the parameterized complexity of this problem with respect to both structural and natural parameters.

Using standard DP techniques, we first present an algorithm of time $\mathrm{tw}^{O(\mathrm{tw})}n^{O(1)}$, significantly generalizing a recent algorithm of Gaikwad et al. of time $\mathrm{vc}^{O(\mathrm{vc})}n^{O(1)}$, where tw, vc denote the input graph's treewidth and vertex cover respectively. Subsequently, we show that both of these algorithms are essentially optimal, since a $\mathrm{vc}^{o(\mathrm{vc})}n^{O(1)}$ algorithm would refute the ETH.

With respect to the natural parameter k, the aforementioned recent work by Gaikwad et al. claimed an FPT branching algorithm with complexity $10^k n^{O(1)}$. We point out that this algorithm is incorrect and present a branching algorithm of complexity $9.34^k n^{O(1)}$.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases ETH, Feedback vertex set, Parameterized algorithms, Treewidth

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Related Version *Full Version*: `https://arxiv.org/abs/2302.09604`

Funding *Michael Lampis*: Partially supported by the ANR project ANR-21-CE48-0022 (S-EX-AP-PE-AL).
Nikolaos Melissinos: Supported by the CTU Global postdoc fellowship program.

Acknowledgements Work primarily conducted while Nikolaos Melissinos was affiliated with Université Paris-Dauphine.

1 Introduction

We consider a MaxMin version of the well-studied feedback vertex set problem where, given a graph $G = (V, E)$ and a target size k, we are asked to find a set of vertices S with the following properties: (i) every cycle of G contains a vertex of S, that is, S is a feedback vertex set (ii) no proper subset of S is a feedback vertex set, that is, S is minimal (iii) $|S| \geq k$. Although much less studied than its minimization cousin, Max Min FVS has recently attracted attention in the literature as part of a broader study of MaxMin versions of standard problems, such as Max Min Vertex Cover and Upper Dominating Set. The main motivation of this line of research is the search for a deeper understanding of the performance of simple greedy algorithms: given an input, we would like to compute what is the worst possible solution that would still not be improvable by a simple heuristic, such as removing redundant vertices. Nevertheless, over recent years MaxMin problems have been found to possess an interesting combinatorial structure of their own and have now become an object of more widespread study (we survey some such results below).

It is not surprising that Max Min FVS is known to be NP-complete and is in fact significantly harder than Minimum FVS in most respects, such as its approximability or its amenability to algorithms solving special cases. Given the problem's hardness, in this paper we focus on the parameterized complexity of Max Min FVS, since parameterized complexity is one of the main tools for dealing with computational intractability[1]. We consider two types of parameterizations: the natural parameter k; and the parameterization by structural width measures, such as treewidth. In order to place our results into perspective, we first recall the current state of the art.

Previous work. Max Min FVS was first shown to be NP-complete even on graphs of maximum degree 9 by Mishra and Sikdar [32]. This was subsequently improved to NP-completeness for graphs of maximum degree 6 by Dublois et al. [20], who also present an approximation algorithm with ratio $n^{2/3}$ and proved that this is optimal unless P=NP. A consequence of the polynomial time approximation algorithm of [20] was the existence of a kernel of order $O(k^3)$, which implied that the problem is fixed-parameter tractable with respect to the natural parameter k. Some evidence that this kernel size may be optimal was later given by [2]. We note also that the problem can easily be seen to be FPT parameterized by treewidth (indeed even by clique-width) as the property that a set is a minimal feedback vertex set is MSO_1-expressible, so standard algorithmic meta-theorems apply.

Given the above, the state of the art until recently was that this problem was known to be FPT for the two most well-studied parameterizations (by k and by treewidth), but concrete FPT algorithms were missing. An attempt to advance this state of the art and systematically study the parameterized complexity of the problem was recently undertaken by Gaikwad et al. [23], who presented exact algorithms for this problem running in time $10^k n^{O(1)}$ and $\mathrm{vc}^{O(\mathrm{vc})} n^{O(1)}$, where vc is the input graph's vertex cover, which is known to be a (much) more restrictive parameter than treewidth. Leveraging the latter algorithm, [23] also present an FPT approximation scheme which can $(1-\varepsilon)$-approximate the problem in time $2^{O(\mathrm{vc}/\varepsilon)} n^{O(1)}$, that is, single-exponential time with respect to vc.

Our contribution. We begin our work by considering Max Min FVS parameterized by the most standard structural parameter, treewidth. We observe that, using standard DP techniques, we can obtain an algorithm running in time $\mathrm{tw}^{O(\mathrm{tw})} n^{O(1)}$, that is, slightly super-exponential with respect to treewidth. Note that this slightly super-exponential running time is already present in the $\mathrm{vc}^{O(\mathrm{vc})} n^{O(1)}$ algorithm of [23], despite the fact that vertex cover is a much more severely restricted parameter. Hence, our algorithm generalizes the algorithm of [23] without a significant sacrifice in the running time.

Despite the above, our main contribution with respect to structural parameters is not our algorithm for parameter treewidth, but an answer to a question that is naturally posed given the above: can the super-exponential dependence present in both our algorithm and the algorithm of [23] be avoided, that is, can we obtain a $2^{O(\mathrm{tw})} n^{O(1)}$ algorithm? We show that this is likely impossible, as the existence of an algorithm running in time $\mathrm{vc}^{o(\mathrm{vc})} n^{O(1)}$ is ruled out by the ETH (and hence also the existence of a $\mathrm{tw}^{o(\mathrm{tw})} n^{O(1)}$ algorithm). This result is likely to be of wider interest to the parameterized complexity community, where one of

[1] Throughout the paper we assume that the reader is familiar with the basics of parameterized complexity, as given in standard textbooks [16].

the most exciting developments of the last fifteen years has arguably been the development of the Cut&Count technique (and its variations). One of the crowning achievements of this technique is the design of single-exponential algorithms for connectivity problems – indeed an algorithm running in time $3^{\mathrm{tw}}n$ for Minimum FVS is given in [17]. It has therefore been of much interest to understand which connectivity problems admit single-exponential algorithms using such techniques (see e.g. [7] and the references within). Curiously, even though several cousins of Minimum Feedback Vertex Set have been considered in this context (such as Subset Feedback Vertex Set and Restricted Edge-Subset Feedback Edge Set), for Max Min FVS, which is arguably a very natural variant, it was not known whether a single-exponential algorithm for parameter treewidth is possible. Our work thus adds to the literature a natural connectivity problem where Cut&Count can provably not be applied (under standard assumptions). Interestingly, our lower bound even applies to the case of vertex cover, which is rare, as most problems tend to become rather easy under this very restrictive parameter.

We then move on to consider the parameterization of the problem by k, the size of the sought solution. Observe that a $k^{O(k)}n^{O(1)}$ algorithm can easily be obtained by the results sketched above and a simple win/win argument: start with any minimal feedback vertex set S of the given graph G: if $|S| \geq k$ we are done; if not, then $\mathrm{tw}(G) \leq k$ and we can solve the problem using the algorithm for treewidth. It is therefore only interesting to consider algorithms with a single-exponential dependence on k. Such an algorithm, with complexity $10^k n^{O(1)}$, was claimed by [23]. Unfortunately, as we explain in detail in Section 5, this algorithm contains a significant flaw[2].

Our contribution is to present a corrected version of the algorithm of [23], which also achieves a slightly better running time of $9.34^k n^{O(1)}$, compared to the $10^k n^{O(1)}$ of the (flawed) algorithm of [23]. Our algorithm follows the same general strategy of [23], branching and placing vertices in the forest or the feedback vertex set. However, we have to rely on a more sophisticated measure of progress, because simply counting the size of the selected set is not sufficient. We therefore measure our progress towards a restricted special case we identify, namely the case where the undecided part of the graph induces a linear forest. Though this special case sounds tantalizingly simple, we show that the problem is still NP-complete under this restriction, but obtaining an FPT algorithm is much easier. We then plug in our algorithm to a more involved branching procedure which aims to either reduce instances into this special case, or output a certifiable minimal feedback vertex set of the desired size.

Finally, motivated by the above we note that a blocking point in the design of algorithms for Max Min FVS seems to be the difficulty of the extension problem: given a set S_0, decide if a minimal fvs S that extends S_0 exists. As mentioned, Casel et al. [13] showed that this problem is W[1]-hard parameterized by $|S_0|$. Intriguingly, however, it is not even known if this problem is in XP, that is, whether it is solvable in polynomial time for fixed k. We show that this is perhaps not surprising, as obtaining a polynomial time algorithm in this case would imply the existence of a polynomial time algorithm for the notorious k-in-a-Tree problem: given k terminals in a graph, find an induced tree that contains them. Since this problem was solved for $k = 3$ in a breakthrough by Chudnovsky and Seymour [15], the complexity for fixed $k \geq 4$ has remained a big open problem (for example [29] states that "Solving it in polynomial time for constant k would be a huge result"). It is therefore perhaps not surprising that obtaining an XP algorithm for the extension problem for minimal feedback vertex sets of fixed size is challenging, since such an algorithm would settle another

[2] Saket Saurabh, one of the authors of [23], confirmed so via private communication with Michael Lampis.

long-standing problem.

Other relevant work. As mentioned, Max Min FVS is an example of a wider class of MaxMin problems which have recently attracted much attention in the literature, among the most well-studied of which are Maximum Minimal Vertex Cover [2, 11, 12, 34] and Upper Dominating Set (which is the standard name for Maximum Minimal Dominating Set) [1, 3, 5, 21]. Besides these problems, MaxMin or MinMax versions of cut and separations problems [19, 26, 30], knapsack problems [22, 24], matching problems [14], and coloring problems [6] have also been studied.

The question of which connectivity problems admit single-exponential algorithms parameterized by treewidth has been well-studied over the last decade. As mentioned, the main breakthrough was the discovery of the Cut&Count technique [16], which gave randomized $2^{O(\mathrm{tw})}n^{O(1)}$ algorithms for many such problems, such as Steiner Tree, Hamiltonicity, Connected Dominating Set and others. Follow-up work also provided deterministic algorithms with complexity $2^{O(\mathrm{tw})}n^{O(1)}$ [8]. It is important to note that the discovery of these techniques was considered a surprise at the time, as the conventional wisdom was that connectivity problems probably require $\mathrm{tw}^{O(\mathrm{tw})}$ time to be solved [31]. Naturally, the topic was taken up with much excitement, in an attempt to discover the limits of such techniques, including problems for which they cannot work. In this vein, [33] gave a meta-theorem capturing many tractable problems, and also an example problem that cannot be solved in time $2^{o(\mathrm{tw}^2)}n^{O(1)}$ under the ETH. Several other examples of connectivity problems which require slightly super-exponential time parameterized by treewidth are now known [4, 27], with the most relevant to our work being the feedback vertex set variants studied in [7, 10], as well as the digraph version of the minimum feedback vertex set problem (parameterized by the treewidth of the underlying graph) [9]. The results of our paper seem to confirm the intuition that the Cut&Count technique is rather fragile when applied to feedback vertex set problems, since in many variations or generalizations of this problem, a super-exponential dependence on treewidth is inevitable (assuming the ETH).

2 Preliminaries

Throughout the paper, we use standard graph notation [18]. Moreover, for vertex $u \in V(G)$, let $\deg_X(u)$ denote its degree in $G[X \cup \{u\}]$, where $X \subseteq V(G)$. A *multigraph* G is a graph which is permitted to have multiple edges with the same end nodes, thus, two vertices may be connected by more than one edge. Given a (multi)graph G, where $e = \{u, v\} \in E(G)$ is a not necessarily unique edge connecting distinct vertices u and v, the *contraction* of e results in a new graph G' such that $V(G') = (V(G) \setminus \{u, v\}) \cup \{w\}$, while for each edge $\{u, x\}$ or $\{v, x\}$ in $E(G)$, there exists an edge $\{w, x\}$ in $E(G')$. Any edge $e \in E(G)$ not incident to u, v also belongs to $E(G')$. If u and v were additionally connected by an edge apart from e, then w has a self loop.

For $i \in \mathbb{N}$, $[i]$ denotes the set $\{1, \dots, i\}$. A feedback vertex set S of G is minimal if and only if $\forall s \in S$, $G[(V(G) \setminus S) \cup \{s\}]$ contains a cycle, namely a *private cycle* of s [21]. Lastly, we make use of a weaker version of ETH, which states that 3-SAT cannot be determined in time $2^{o(n)}$, where n denotes the number of the variables [28].

Finally, note that the proofs of all lemmas and theorems marked with ($\star$) are in the appendix.

3 Treewidth Algorithm

Here we will present an algorithm for Max Min FVS parameterized by the treewidth of the input graph, arguably the most well studied structural parameter. As a corollary of the lower bound established in Section 4, it follows that the running time of the algorithm is essentially optimal under the ETH.

▶ **Theorem 1.** ($\star$) *Given an instance $\mathcal{I} = (G, k)$ of Max Min FVS, as well as a nice tree decomposition of G of width* tw*, there exists an algorithm that decides $\mathcal{I}$ in time $\mathrm{tw}^{O(\mathrm{tw})} n^{O(1)}$.*

Proof sketch. The main idea lies on performing standard dynamic programming on the nodes of the nice tree decomposition. To this end, for each node, we will consider all the partial solutions, corresponding to (not necessarily minimal) feedback vertex sets of the subgraph induced by the vertices of the nodes of the corresponding subtree of the tree decomposition. We will try to extend such a feedback vertex set to a minimal feedback vertex set of G, that respects the partial solution. For each partial solution, it is imperative to identify, apart from the vertices of the bag that belong to the feedback vertex set, the connectivity of the rest of the vertices in the potential final forest. In order to do so, we consider a coloring indicating that, same colored vertices of the forest of the partial solution, should be in the same connected component of the potential final forest. Moreover, we keep track of which vertices of the forest of the partial solution are connected via paths containing forgotten vertices. Finally, for each vertex of the feedback vertex set of the partial solution, we need to identify one of its private cycles. To do so, we first guess the connected component of the potential final forest that "includes" such a private cycle, while additionally keeping track of the number of edges between the vertex and said component. ◀

4 ETH Lower Bound

In this section we present a lower bound on the complexity of solving Max Min FVS parameterized by vertex cover. Starting from a 3-SAT instance on n variables, we produce an equivalent Max Min FVS instance on a graph of vertex cover $O(n/\log n)$, hence any algorithm solving the latter problem in time $\mathrm{vc}^{o(\mathrm{vc})} n^{O(1)}$ would refute the ETH. As already mentioned, vertex cover is a very restrictive structural parameter, and due to known relationships of vertex cover with more general parameters, such as treedepth and treewidth, analogous lower bounds follow for these parameters. We first state the main theorem.

▶ **Theorem 2.** *There is no $\mathrm{vc}^{o(\mathrm{vc})} n^{O(1)}$ time algorithm for Max Min FVS, where* vc *denotes the size of the minimum vertex cover of the input graph, unless the ETH fails.*

Before we present the details of our construction, let us give some high-level intuition. Our goal is to "compress" an n-variable instance of 3-SAT, into an Max Min FVS instance with vertex cover roughly $n/\log n$. To this end, we will construct $\log n$ choice gadgets, each of which is supposed to represent $n/\log n$ variables, while contributing only $n/\log^2 n$ to the vertex cover. Hence, each vertex of each such gadget must be capable of representing roughly $\log n$ variables.

Our choice gadget may be thought of as a variation of a bipartite graph with sets L, R, of size roughly $n/\log^2 n$ and $\sqrt{n}$ respectively. If one naively tries to encode information in such a gadget by selecting which vertices of $L \cup R$ belong in an optimal solution, this would only give 2 choices per vertex, which is not efficient enough. Instead, we engineer things in a way that all vertices of $L \cup R$ must belong in the forest in an optimal solution, and the interesting choice for a vertex ℓ of L is with which vertex r of R we will place ℓ in the same component. In this sense, a vertex ℓ of L has $|R|$ choices, which is sufficient to encode the assignment for $\Omega(\log n)$ variables. What remains, then, is to add machinery that enforces this basic setup, and then clause checking vertices which for each clause verify that the clause is satisfied by testing if an ℓ vertex that represents one of its literals is in the same component as an r vertex that represents a satisfying assignment for the clause.

4.1 Preliminary Tools

Before we present the construction that proves Theorem 2, we give a variant of 3-SAT from which it will be more convenient to start our reduction, as well as a basic force gadget that we will use in our construction to ensure that some vertices must be placed in the forest in order to achieve an optimal solution.

3P3SAT. We first define a constrained version of 3-SAT, called 3-PARTITIONED-3-SAT (3P3SAT for short), and establish its hardness under the ETH.

3-PARTITIONED-3-SAT **Input:** A formula ϕ in 3-CNF form, together with a partition of the set of its variables V into three disjoint sets V_1, V_2, V_3, with $\|V_i\| = n$, such that no clause contains more than one variable from each V_i. **Task:** Determine whether ϕ is satisfiable.

▶ **Theorem 3.** ($\star$) *3-PARTITIONED-3-SAT cannot be decided in time $2^{o(n)}$, unless the ETH fails.*

Force gadgets. We now present a gadget that will ensure that a vertex u must be placed in the forest in any solution that finds a large minimal feedback vertex set. In the remainder, suppose that A is a sufficiently large value (we give a concrete value to A in the next section). When we say that we attach a *force gadget* to a vertex u, we introduce $A+1$ new vertices $\bar{u}, u_1, \dots, u_A$ to the graph such that the vertices u_i form an independent set, while there exist edges $\{u, u_i\}, \{\bar{u}, u_i\}$ for all $i \in [A]$, as well as the edge $\{u, \bar{u}\}$. We refer to vertex $\bar{u}$ as the *gadget twin* of u, while the rest of the vertices will be referred to as the *gadget leaves* of u. Intuitively, the idea here is that if u (or $\bar{u}$) is contained in a minimal feedback vertex set, then none of the A leaves of the gadget can be taken, because these vertices cannot have private cycles. Hence, setting A to be sufficiently large will allow us to force u to be in the forest.

4.2 Construction

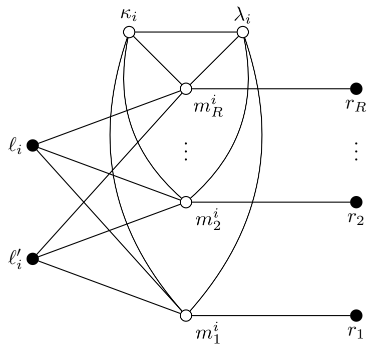
Let ϕ be a 3P3SAT instance of m clauses, where $|V_p| = n$ for $p \in [3]$ and, without loss of generality, assume that n is a power of 4 (this can be achieved by adding dummy variables to the instance if needed). Partition each variable set V_p to $\log n$ subsets V_p^q of size at most $\lceil \frac{n}{\log n} \rceil$, where $p \in [3]$ and $q \in [\log n]$. Let $L = \lceil \frac{n}{\log^2 n} \rceil$. Moreover, partition each variable subset V_p^q into $2L$ subsets $\mathcal{V}_\alpha^{p,q}$ of size as equal as possible, where $\alpha \in [2L]$. In the following we will omit p and q and instead use the notation $\mathcal{V}_\alpha$, whenever p, q are clear from the context. Define $R = \sqrt{n}$, $A = n^2 + m$ and $k = (4AL + AR + 2LR) \cdot 3 \log n + m$. We will proceed with the construction of a graph G such that G has a minimal feedback vertex set of size at least k if and only if ϕ is satisfiable.

For each variable subset V_p^q, we define the choice gadget graph G_p^q as follows:

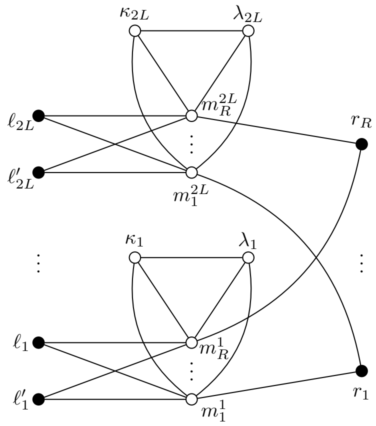
- $V(G_p^q) = \{\ell_i, \ell_i', \kappa_i, \lambda_i \mid i \in [2L]\} \cup \{r_j \mid j \in [R]\} \cup \{m_j^i \mid i \in [2L], j \in [R]\}$,
- all the vertices ℓ_i, ℓ_i' and r_j have an attached force gadget,
- for $i \in [2L]$, $N(\kappa_i) = M_i \cup \{\lambda_i\}$ and $N(\lambda_i) = M_i \cup \{\kappa_i\}$, where $M_i = \{m_j^i \mid j \in [R]\}$,
- for $i \in [2L]$ and $j \in [R]$, m_j^i has an edge with ℓ_i, ℓ_i' and r_j.

We will refer to the set $X_i = M_i \cup \{\kappa_i, \lambda_i\}$ as the *choice set* i.

Intuitively, one can think of this gadget as having been constructed as follows: we start with a complete bipartite graph that has on one side the vertices ℓ_i and on the other the vertices r_j; we subdivide each edge of this graph, giving the vertices m_j^i; for each $i \in [2L]$ we add $\ell_i', \kappa_i, \lambda_i$, connect them to the same m_j^i vertices that ℓ_i is connected to and connect κ_i to λ_i; we attach force gadgets to all ℓ_i, ℓ_i', r_j. Hence, as sketched before, the idea of this gadget is that the choice of a vertex ℓ_i is to pick an r_j with which it will be in the same component in the forest, and this will be expressed by picking one m_j^i that will be placed in the forest.



(a) Part of the construction concerning X_i.



(b) The whole choice gadget graph G_p^q.

Figure 1 Black vertices have a force gadget attached.

Each vertex ℓ_α of G_p^q is used to represent a variable subset $\mathcal{V}_\alpha^{p,q} \subseteq V_p^q$ containing at most

$$|\mathcal{V}_\alpha^{p,q}| \leq \left\lceil \frac{|V^{p,q}|}{2L} \right\rceil \leq \left\lceil \frac{\lceil \frac{n}{\log n} \rceil}{2L} \right\rceil = \left\lceil \frac{n}{2L \log n} \right\rceil \leq \left\lceil \frac{n}{2\frac{n}{\log^2 n} \log n} \right\rceil = \left\lceil \frac{\log n}{2} \right\rceil = \frac{\log n}{2}$$

variables of ϕ, where we used Theorem 3.10 of [25], for $f(x) = x/2L$. We fix an arbitrary one-to-one mapping so that every vertex m_β^α, where $\beta \in [R]$, corresponds to a different assignment for this subset, which is dictated by which element of M_α was not included in the final feedback vertex set. Since $R = 2^{\log n/2} = \sqrt{n}$, the size of M_α is sufficient to uniquely encode all the different assignments of $\mathcal{V}_\alpha$.

Finally, introduce vertices c_i, where $i \in [m]$, each of which corresponds to a clause of ϕ, and define graph G as the union of these vertices as well as all graphs G_p^q, where $p \in [3]$ and $q \in [\log n]$. For a clause vertex c, add an edge to ℓ_α when $\mathcal{V}_\alpha$ contains a variable appearing in c, as well as to the vertices r_β for each such ℓ_α, such that $m_\beta^\alpha \notin S$ corresponds to an assignment of $\mathcal{V}_\alpha$ satisfying c, where S denotes a minimal feedback vertex set. Notice that since no clause contains multiple variables from the same variable set V_i, due to the refinement of the partition of the variables, it holds that all the variables of a clause will be represented by vertices appearing in distinct G_p^q.

4.3 Correctness

Having constructed the previously described instance (G, k) of Max Min FVS, it remains to prove its equivalence with the initial 3-Partitioned-3-SAT instance.

▶ **Lemma 1.** ($\star$) *Any minimal feedback vertex set S of G of size at least k has the following properties:*
(i) *S does not contain any vertex attached with a force gadget or its gadget twin,*
(ii) *$|M_i \setminus S| \leq 1$, for every G_p^q and $i \in [2L]$,*
(iii) *$|S \cap V(G_p^q)| = 4AL + AR + 2LR$,*
where $p \in [3]$ and $q \in [\log n]$.

▶ **Lemma 2.** ($\star$) *If ϕ has a satisfying assignment, then G has a minimal feedback vertex set of size at least k.*

▶ **Lemma 3.** ($\star$) *If G has a minimal feedback vertex set of size at least k, then ϕ has a satisfying assignment.*

▶ **Lemma 4.** ($\star$) $\mathrm{vc}(G) = O(n/\log n)$.

Using the previous lemmas, we can prove Theorem 2.

Proof of Theorem 2. Let ϕ be a 3-Partitioned-3-SAT formula. In polynomial time, we can construct a graph G such that, due to Lemmas 2 and 3, deciding if G has a minimal feedback vertex set of size at least k is equivalent to deciding if ϕ has a satisfying assignment. In that case, assuming there exists a $\mathrm{vc}^{o(\mathrm{vc})}$ algorithm for Max Min FVS, one could decide 3-Partitioned-3-SAT in time

$$\mathrm{vc}^{o(\mathrm{vc})} = \left(\frac{n}{\log n}\right)^{o(n/\log n)} = 2^{(\log n - \log\log n)o(n/\log n)} = 2^{o(n)},$$

which contradicts the ETH due to Theorem 3. ◀

Since for any graph G it holds that $\mathrm{tw}(G) \leq \mathrm{vc}(G)$, the following corollary holds.

▶ **Corollary 4.** *There is no $\mathrm{tw}^{o(\mathrm{tw})} n^{O(1)}$ time algorithm for Max Min FVS, where tw denotes the treewidth of the input graph, unless the ETH fails.*

5 Natural Parameter Algorithm

In this section we will present an FPT algorithm for Max Min FVS parameterized by the natural parameter, i.e. the size of the maximum minimal feedback vertex set k. The main theorem of this section is the following.

▶ **Theorem 5.** *Max Min FVS can be solved in time $9.34^k n^{O(1)}$.*

Structure of the Section. In Section 5.1 we define the closely related Annotated MMFVS problem, and prove that it remains NP-hard, even on some instances of specific form, called *path restricted instances.* Subsequently, we present an algorithm dealing with this kind of instances, which either returns a minimal feedback vertex set of size at least k or concludes that this is a No instance of Annotated MMFVS. Afterwards, in Section 5.2, we solve Max Min FVS by producing a number of instances of Annotated MMFVS and utilizing the previous algorithm, therefore proving Theorem 5.

Oversight of [23]. The algorithm of [23] performs a branching procedure which marks vertices as either belonging in the feedback vertex set or the remaining forest. The flaw is that the algorithm ceases the branching once k vertices have been identified as vertices of the feedback vertex set. However, this is not correct, since deciding if a given set S_0 can be extended into a minimal feedback vertex set $S \supseteq S_0$ is NP-complete and even W[1]-hard parameterized by $|S_0|$ [13]. Hence, identifying k vertices of the solution is not, in general, sufficient to produce a feasible solution and the algorithm of [23] is incomplete, because it does not explain how the guessed part of the feedback vertex set can be extended into a feasible minimal solution.

5.1 Annotated MMFVS and Path Restricted Instances

First, we define the following closely related problem, denoted by Annotated MMFVS for short.

> Annotated Maximum Minimal Feedback Vertex Set
> **Input:** A graph $G = (V, E)$, disjoint sets $S, F \subseteq V$ where $S \cup F$ is a feedback vertex set of G, as well as an integer k.
> **Task:** Determine whether there exists a minimal feedback vertex set S' of G of size $|S'| \geq k$ such that $S \subseteq S'$ and $S' \cap F = \emptyset$.

Remarks. Notice that if F is not a forest, then the corresponding instance always has a negative answer. For the rest of this section, let $U = V(G) \setminus (S \cup F)$. Moreover, let $H = \{s \in S \mid \deg_F(s) \geq 2 \text{ and } \deg_U(s) \leq 1\}$ denote the set of *good vertices* of S. An *interesting path* of $G[U]$ is a connected component of $G[U]$ such that for every vertex u belonging to said component, it holds that $\deg_{F \cup U}(u) = 2$. If every connected component of $G[U]$ is an interesting path, then this is a *path restricted instance*. Furthermore, given instance $\mathcal{I}$, let ammfvs$(\mathcal{I})$ be equal to 1 if it is a Yes instance and 0 otherwise.

Let $\mathcal{I} = (G, S, F, k)$ be a path restricted instance of Annotated MMFVS. We will present an algorithm that either returns a minimal feedback vertex set $S' \subseteq S \cup U$ of G of size at least k or concludes that this is a No instance of Annotated MMFVS. Notice that Annotated MMFVS remains NP-hard even on such instances, as dictated by Theorem 6. Therefore, we should not expect to solve path restricted instances of Annotated MMFVS in polynomial time.

▶ **Theorem 6.** ($\star$) *Annotated MMFVS is NP-hard on path restricted instances, even if all the paths are of length* 2.

We proceed by presenting the main algorithm of this subsection, which will be essential in proving Theorem 5.

▶ **Theorem 7.** ($\star$) *Let $\mathcal{I} = (G, S, F, k)$ be a path restricted instance of Annotated MMFVS, and let g denote the number of its good vertices. There is an algorithm running in time $O(3^{k-g} n^{O(1)})$ which either returns a minimal feedback vertex set $S' \subseteq S \cup U$ of G of size at least k or concludes that $\mathcal{I}$ is a No instance of Annotated MMFVS.*

Proof sketch. The main idea of the algorithm lies on the fact that we can efficiently handle instances where either $k = 0$ or $S = \emptyset$. Towards this, we will employ a branching strategy that, as long as S remains non empty, new instances with reduced k are produced. Prior to performing branching, we first observe that we can efficiently deal with the good vertices.

Afterwards, by employing said branching strategy, in every step we decide which vertex will be counted towards the k required, thereby reducing parameter k on each iteration. If at some point $k = 0$ or $S = \emptyset$, it remains to decide whether this comprises a viable solution S'. Notice that S' may not be a solution for the annotated instance, since even if $|S'| \geq k$, it does not necessarily hold that $S' \supseteq S$. ◀

5.2 Algorithm for Max Min FVS

We start by presenting a high level sketch of the algorithm for Max Min FVS. The starting point is a minimal feedback vertex set S_0 of G. Note that such a set can be obtained in polynomial time, while if it is of size at least k, we are done. Therefore, assume that $|S_0| < k$. Then, assuming there exists a minimal feedback vertex set S^*, where $|S^*| \geq k$ and $F^* = V(G) \setminus S^*$, we will guess $S_0 \cap S^*$, thereby producing instances $\mathcal{I}_0 = (G, S_0 \cap S^*, S_0 \cap F^*, k)$ of Annotated MMFVS. Subsequently, we will establish a number of *safe* reduction rules, which do not affect the answer of the instances. We will present a measure of progress μ, which guarantees that if an instance $\mathcal{I} = (G, S, F, k)$ of Annotated MMFVS has $\mu(\mathcal{I}) \leq 1$, then G has a minimal feedback vertex set $S' \subseteq S \cup U$ of size at least k, and employ a branching strategy which, given $\mathcal{I}_i$, will produce instances $\mathcal{I}^1_{i+1}, \mathcal{I}^2_{i+1}$ of lesser measure of progress, such that $\mathcal{I}_i$ is a Yes instance if and only if at least one of $\mathcal{I}^1_{i+1}, \mathcal{I}^2_{i+1}$ is also a Yes instance. If we can no further apply our branching strategy, and the measure of progress remains greater than 1, then it holds that $\mathcal{I}$ is a path restricted instance and Theorem 7 applies.

Measure of progress. Let $\mathcal{I} = (G, S, F, k)$ be an instance of Annotated MMFVS. We define as $\mu(\mathcal{I}) = k + cc(F) - g - p$ its *measure of progress*, where

- $cc(F)$ denotes the number of connected components of F,
- g denotes the number of good vertices of S,
- p denotes the number of interesting paths of $G[U]$.

It holds that if $\mu(\mathcal{I}) \leq 1$, then the underlying Max Min FVS instance has a positive answer, which does not necessarily respect the constraints dictated by the annotated version.

▶ **Lemma 5.** ($\star$) *Let $\mathcal{I} = (G, S, F, k)$ be an instance of Annotated MMFVS, where $\mu(\mathcal{I}) \leq 1$. Then, G has a minimal feedback vertex set $S' \subseteq S \cup U$ of size at least k.*

Reduction rules. In the following, we will describe some reduction rules which do not affect the answer of an instance of Annotated MMFVS, while not increasing its measure of progress.

▶ **Lemma 6.** ($\star$) *Let $G = (V, E)$ be a (multi)graph and $uv \in E(G)$. Then, G is acyclic if and only if G/uv is acyclic.*

Rule 1. Let $\mathcal{I} = (G, S, F, k)$ be an instance of Annotated MMFVS, $u, v \in F$ and $uv \in E$. Then, replace $\mathcal{I}$ with $\mathcal{I}' = (G', S, F', k)$, where $G' = G/uv$ occurs from the contraction of u and v into w, while $F' = (F \cup \{w\}) \setminus \{u, v\}$.

Rule 2. Let $\mathcal{I} = (G, S, F, k)$ be an instance of Annotated MMFVS, $u \in U$ and $\deg_{F \cup U}(u) = 0$. Then, replace $\mathcal{I}$ with $\mathcal{I}' = (G - u, S, F, k)$.

Rule 3. Let $\mathcal{I} = (G, S, F, k)$ be an instance of ANNOTATED MMFVS, $u \in U$ and $\deg_{F\cup U}(u) = 1$, while $v \in N(u) \cap (F \cup U)$. Then, replace $\mathcal{I}$ with $\mathcal{I}' = (G', S, F', k)$, where $G' = G/uv$ occurs from the contraction of u and v into w, while $F' = (F \cup \{w\}) \setminus \{v\}$ if $v \in F$, and $F' = F$ otherwise.

▶ **Lemma 7.** ($\star$)*Applying rules 1, 2 and 3 does not change the outcome of the algorithm and does not increase the measure of progress.*

After exhaustively applying the aforementioned rules, it holds that $\forall u \in U$, $\deg_{F\cup U}(u) \geq 2$, i.e. $G[U]$ is a forest containing trees, all the leaves of which have at least one edge to F. Moreover, $G[F]$ comprises an independent set. We proceed with a branching strategy that produces instances of ANNOTATED MMFVS of reduced measure of progress. If at some point $\mu \leq 1$, then Lemma 5 can be applied.

Branching strategy. Let $\mathcal{I} = (G, S, F, k)$ be an instance of ANNOTATED MMFVS, on which all of the reduction rules have been applied exhaustively, thus, it holds that a) $\forall u \in U$, $\deg_{F\cup U}(u) \geq 2$ and b) F is an independent set.

Define $u \in U$ to be an *interesting* vertex if $\deg_{F\cup U}(u) \geq 3$. As already noted, $G[U]$ is a forest, the leaves of which all have an edge towards F, otherwise Rule 3 could still be applied. Consider a root for each tree of $G[U]$. For some tree T, let v be an interesting vertex at maximum distance from the corresponding root, i.e. v is an interesting vertex of maximum height. Notice that such a tree cannot be an interesting path. We branch depending on whether u is in the feedback vertex set or not. Towards this end, let $S' = S \cup \{v\}$ and $F' = F \cup \{v\}$, while $\mathcal{I}_1 = (G, S', F, k)$ and $\mathcal{I}_2 = (G, S, F', k)$. It holds that $\mathcal{I}$ is a Yes instance if and only if at least one of $\mathcal{I}_1, \mathcal{I}_2$ is a Yes instance, while if $G[F']$ contains a cycle, $\mathcal{I}_2$ is a No instance and we discard it. We replace $\mathcal{I}$ with the instances $\mathcal{I}_1$ and $\mathcal{I}_2$.

▶ **Lemma 8.** ($\star$)*The branching strategy produces instances of reduced measure of progress, without reducing the number of good vertices.*

Complexity. Starting from an instance (G, k) of MAX MIN FVS, we produce a minimal feedback vertex set S_0 of G in polynomial time. If $|S_0| \geq k$, we are done. Alternatively, we produce instances of ANNOTATED MMFVS by guessing the intersection of S_0 with some minimal feedback vertex set of G of size at least k. Let $\mathcal{I} = (G, S, F, k)$ be one such instance. It holds that $\mu(\mathcal{I}) \leq k + c$, where $c = cc(F)$, therefore the branching will perform at most $k + c$ steps. Notice that, at any step of the branching procedure, the number of good vertices never decreases. Now, consider a path restricted instance $\mathcal{I}' = (G', S', F', k)$ resulting from branching starting on $\mathcal{I}$, on which branching, exactly ℓ times a vertex was placed in the feedback vertex set, therefore $|S'| - |S| = \ell$. There are at most $\binom{k+c}{\ell}$ different such instances, each of which has at least ℓ good vertices, thus Theorem 7 requires time at most $3^{k-\ell} n^{O(1)}$. Since $0 \leq \ell \leq k + c$, and there are at most $\binom{k}{c}$ different instances $\mathcal{I}$, the algorithm runs in time $9.34^k n^{O(1)}$, since

$$\sum_{c=0}^{k} \binom{k}{c} \sum_{\ell=0}^{k+c} \binom{k+c}{\ell} 3^{k-\ell} = 3^k \sum_{c=0}^{k} \binom{k}{c} \sum_{\ell=0}^{k+c} \binom{k+c}{\ell} 3^{-\ell} = 3^k \sum_{c=0}^{k} \binom{k}{c} \left(\frac{4}{3}\right)^{k+c}$$

$$= 4^k \sum_{c=0}^{k} \binom{k}{c} \left(\frac{4}{3}\right)^{c} = 4^k \left(\frac{7}{3}\right)^{k} \leq 9.34^k.$$

6 The Extension Problem

In this section we consider the following extension problem:

MINIMAL FVS EXTENSION **Input:** A graph $G = (V, E)$ and a set $S \subseteq V$. **Task:** Determine whether there exists $S^* \supseteq S$ such that S^* is a minimal feedback vertex set of G.

Observe that this is a special case of ANNOTATED MMFVS, since we essentially set $F = \emptyset$ and do not care about the size of the produced solution, albeit with the difference that now we will not focus on the case where $V \setminus S$ is already acyclic. This extension problem was already shown to be W[1]-hard parameterized by $|S|$ by Casel et al. [13]. One question that was left open, however, was whether it is solvable in polynomial time for fixed $|S|$, that is, whether it belongs in the class XP.

Though we do not settle the complexity of the extension problem for fixed k, we provide evidence that obtaining a polynomial time algorithm would be a challenging task, because it would imply a similar algorithm for the k-IN-A-TREE problem. In the latter, we are given a graph G and a set T of k terminals and are asked to find a set T^* such that $T \subseteq T^*$ and $G[T^*]$ is a tree [15, 29].

▶ **Theorem 8.** *k-IN-A-TREE parameterized by k is fpt-reducible to MINIMAL FVS EXTENSION parameterized by the size of the given set.*

Proof. Consider an instance $G = (V, E)$ of k-IN-A-TREE, with terminal set T. Let $T = \{t_1, \dots, t_k\}$. We add to the graph $k - 1$ new vertices, $s_1, \dots, s_{k-1}$ and connect each s_i to t_i and to t_{i+1}, for $i \in [k-1]$. We set $S = \{s_1, \dots, s_{k-1}\}$. This completes the construction. Clearly, this reduction preserves the value of the parameter.

To see correctness, suppose first that a tree $T^* \supseteq T$ exists in G. We set $S_1 = S \cup (V \setminus T^*)$ in the new graph. S_1 is a feedback vertex set, because removing it from the graph leaves T^*, which is a tree. S_1 contains S. Furthermore, if S_1 is not minimal, we greedily remove from it arbitrary vertices until we obtain a minimal feedback vertex set S_2. We claim that S_2 must still contain S. Indeed, each vertex s_i, for $i \in [k-1]$ has a private cycle, since its neighbors $t_i, t_{i+1} \in T^*$. For the converse direction, if there exists in the new graph a minimal feedback vertex set S^* that contains S, then the remaining forest $F^* = V \setminus S^*$ must contain T, since each vertex of S must have a private cycle in the forest, and vertices of S have degree 2. Furthermore, all vertices of T must be in the same component of F^*, because to obtain a private cycle for s_i, we must have a path from t_i to t_{i+1} in F^*, for all $i \in [k-1]$. Therefore, in this case we have found an induced tree in G that contains all terminals. ◀

7 Conclusions and Open Problems

We have precisely determined the complexity of MAX MIN FVS with respect to structural parameters from vertex cover to treewidth as being slightly super-exponential. One natural question to consider would then be to examine if the same complexity can be achieved when the problem is parameterized by clique-width. Regarding the complexity of the extension problem for sets of fixed size k, we have shown that this is at least as hard as the well-known (and wide open) k-IN-A-TREE problem. Barring a full resolution of this question, it would also be interesting to ask if the converse reduction also holds, which would prove that the two problems are actually equivalent.

References

1. Hassan AbouEisha, Shahid Hussain, Vadim V. Lozin, Jérôme Monnot, Bernard Ries, and Viktor Zamaraev. Upper domination: Towards a dichotomy through boundary properties. *Algorithmica*, 80(10):2799–2817, 2018. doi:10.1007/s00453-017-0346-9.
2. Júlio Araújo, Marin Bougeret, Victor A. Campos, and Ignasi Sau. Introducing lop-kernels: A framework for kernelization lower bounds. *Algorithmica*, 84(11):3365–3406, 2022. doi:10.1007/s00453-022-00979-z.
3. Júlio Araújo, Marin Bougeret, Victor A. Campos, and Ignasi Sau. Parameterized complexity of computing maximum minimal blocking and hitting sets. *Algorithmica*, 85(2):444–491, 2023. doi:10.1007/s00453-022-01036-5.
4. Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. A complexity dichotomy for hitting connected minors on bounded treewidth graphs: the chair and the banner draw the boundary. In *SODA*, pages 951–970. SIAM, 2020. doi:10.1137/1.9781611975994.57.
5. Cristina Bazgan, Ljiljana Brankovic, Katrin Casel, Henning Fernau, Klaus Jansen, Kim-Manuel Klein, Michael Lampis, Mathieu Liedloff, Jérôme Monnot, and Vangelis Th. Paschos. The many facets of upper domination. *Theor. Comput. Sci.*, 717:2–25, 2018. doi:10.1016/j.tcs.2017.05.042.
6. Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, and Yota Otachi. Grundy distinguishes treewidth from pathwidth. *SIAM Journal on Discrete Mathematics*, 36(3):1761–1787, 2022. doi:10.1137/20M1385779.
7. Benjamin Bergougnoux, Édouard Bonnet, Nick Brettell, and O-joung Kwon. Close relatives of feedback vertex set without single-exponential algorithms parameterized by treewidth. In *IPEC*, volume 180 of *LIPIcs*, pages 3:1–3:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.IPEC.2020.3.
8. Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.
9. Marthe Bonamy, Lukasz Kowalik, Jesper Nederlof, Michal Pilipczuk, Arkadiusz Socala, and Marcin Wrochna. On directed feedback vertex set parameterized by treewidth. In *WG*, volume 11159 of *Lecture Notes in Computer Science*, pages 65–78. Springer, 2018. doi:10.1007/978-3-030-00256-5_6.
10. Édouard Bonnet, Nick Brettell, O-joung Kwon, and Dániel Marx. Generalized feedback vertex set problems on bounded-treewidth graphs: Chordality is the key to single-exponential parameterized algorithms. *Algorithmica*, 81(10):3890–3935, 2019. doi:10.1007/s00453-019-00579-4.
11. Édouard Bonnet, Michael Lampis, and Vangelis Th. Paschos. Time-approximation trade-offs for inapproximable problems. *J. Comput. Syst. Sci.*, 92:171–180, 2018. doi:10.1016/j.jcss.2017.09.009.
12. Nicolas Boria, Federico Della Croce, and Vangelis Th. Paschos. On the max min vertex cover problem. *Discret. Appl. Math.*, 196:62–71, 2015. doi:10.1016/j.dam.2014.06.001.
13. Katrin Casel, Henning Fernau, Mehdi Khosravian Ghadikolaei, Jérôme Monnot, and Florian Sikora. On the complexity of solution extension of optimization problems. *Theor. Comput. Sci.*, 904:48–65, 2022. doi:10.1016/j.tcs.2021.10.017.
14. Juhi Chaudhary, Sounaka Mishra, and B. S. Panda. Minimum maximal acyclic matching in proper interval graphs. In *CALDAM*, volume 13947 of *Lecture Notes in Computer Science*, pages 377–388. Springer, 2023. doi:10.1007/978-3-031-25211-2_29.
15. Maria Chudnovsky and Paul D. Seymour. The three-in-a-tree problem. *Comb.*, 30(4):387–417, 2010. doi:10.1007/s00493-010-2334-4.
16. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
17. Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. *ACM Trans. Algorithms*, 18(2):17:1–17:31, 2022. doi:10.1145/3506707.

18 Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate texts in mathematics*. Springer, 2017. doi:10.1007/978-3-662-53622-3.

19 Gabriel L. Duarte, Hiroshi Eto, Tesshu Hanaka, Yasuaki Kobayashi, Yusuke Kobayashi, Daniel Lokshtanov, Lehilton L. C. Pedrosa, Rafael C. S. Schouery, and Uéverton S. Souza. Computing the largest bond and the maximum connected cut of a graph. *Algorithmica*, 83(5):1421–1458, 2021. doi:10.1007/s00453-020-00789-1.

20 Louis Dublois, Tesshu Hanaka, Mehdi Khosravian Ghadikolaei, Michael Lampis, and Nikolaos Melissinos. (in)approximability of maximum minimal FVS. *J. Comput. Syst. Sci.*, 124:26–40, 2022. doi:10.1016/j.jcss.2021.09.001.

21 Louis Dublois, Michael Lampis, and Vangelis Th. Paschos. Upper dominating set: Tight algorithms for pathwidth and sub-exponential approximation. *Theor. Comput. Sci.*, 923:271–291, 2022. doi:10.1016/j.tcs.2022.05.013.

22 Fabio Furini, Ivana Ljubic, and Markus Sinnl. An effective dynamic programming algorithm for the minimum-cost maximal knapsack packing problem. *Eur. J. Oper. Res.*, 262(2):438–448, 2017. doi:10.1016/j.ejor.2017.03.061.

23 Ajinkya Gaikwad, Hitendra Kumar, Soumen Maity, Saket Saurabh, and Shuvam Kant Tripathi. Maximum minimal feedback vertex set: A parameterized perspective. *CoRR*, abs/2208.01953, 2022. arXiv:2208.01953, doi:10.48550/arXiv.2208.01953.

24 Laurent Gourvès, Jérôme Monnot, and Aris Pagourtzis. The lazy bureaucrat problem with common arrivals and deadlines: Approximation and mechanism design. In *FCT*, volume 8070 of *Lecture Notes in Computer Science*, pages 171–182. Springer, 2013. doi:10.1007/978-3-642-40164-0_18.

25 Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science, 2nd Ed.* Addison-Wesley, 1994.

26 Tesshu Hanaka, Yasuaki Kobayashi, Yusuke Kobayashi, and Tsuyoshi Yagita. Finding a maximum minimal separator: Graph classes and fixed-parameter tractability. *Theor. Comput. Sci.*, 865:131–140, 2021. doi:10.1016/j.tcs.2021.03.006.

27 Ararat Harutyunyan, Michael Lampis, and Nikolaos Melissinos. Digraph coloring and distance to acyclicity. In *STACS*, volume 187 of *LIPIcs*, pages 41:1–41:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.STACS.2021.41.

28 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

29 Kai-Yuan Lai, Hsueh-I Lu, and Mikkel Thorup. Three-in-a-tree in near linear time. In *STOC*, pages 1279–1292. ACM, 2020. doi:10.1145/3357713.3384235.

30 Michael Lampis. Minimum stable cut and treewidth. In *ICALP*, volume 198 of *LIPIcs*, pages 92:1–92:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.92.

31 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. *SIAM J. Comput.*, 47(3):675–702, 2018. doi:10.1137/16M1104834.

32 Sounaka Mishra and Kripasindhu Sikdar. On the hardness of approximating some np-optimization problems related to minimum linear ordering problem. *RAIRO Theor. Informatics Appl.*, 35(3):287–309, 2001. doi:10.1051/ita:2001121.

33 Michal Pilipczuk. Problems parameterized by treewidth tractable in single exponential time: A logical approach. In *MFCS*, volume 6907 of *Lecture Notes in Computer Science*, pages 520–531. Springer, 2011. doi:10.1007/978-3-642-22993-0_47.

34 Meirav Zehavi. Maximum minimal vertex cover parameterized by vertex cover. *SIAM J. Discret. Math.*, 31(4):2440–2456, 2017. doi:10.1137/16M109017X.

A Proofs for Section 3 (Treewidth Algorithm)

▶ **Theorem 1.** *Given an instance $\mathcal{I} = (G, k)$ of* Max Min FVS*, as well as a nice tree decomposition of G of width* tw*, there exists an algorithm that decides $\mathcal{I}$ in time $\mathrm{tw}^{O(\mathrm{tw})} n^{O(1)}$.*

Proof. The main idea lies on performing standard dynamic programming on the nodes of the nice tree decomposition. For a node t, let B_t denote its bag, and $B_t^{\downarrow} \supseteq B_t$ denote the union of the bags in the subtree rooted at t.

Let $S^* \subseteq V$ be a minimal feedback vertex set of G, where $F^* = V \setminus S^*$ and $G[F^*]$ is a forest. For each $u \in S^*$, it holds that there exists a set of vertices $T_u \subseteq F^*$ such that $G[T_u]$ is a tree and $G[\{u\} \cup T_u]$ is not acyclic, as u has a private cycle containing at least two of its neighbors. Our goal is, for each node t, to build all partial solutions S, where $S \subseteq B_t^{\downarrow}$ is a feedback vertex set of $G[B_t^{\downarrow}]$ and for each $u \in S \setminus B_t$, its neighboring vertices in its private cycle belong to $B_t^{\downarrow}$. By considering all the partial solutions of the root node, and extending them appropriately, we can eventually determine the maximum minimal feedback vertex set of the input graph G.

More precisely, for each partial solution S of a node t, let S^* be a minimal feedback vertex set of G respecting S, in the sense of $S^* \supseteq S$ and $(V \setminus S^*) \supseteq (B_t^{\downarrow} \setminus S)$ (note that such an S^* does not necessarily exist). We keep the following information:

- the set $S \cap B_t$ as well as the size of S,
- which vertices of S have private cycle in $G[B_t^{\downarrow}]$,
- information regarding the connectivity of the forest $G[B_t^{\downarrow}]$: a coloring of $B_t \setminus S$, such that if 2 vertices share the same color, then they belong to the same connected component of $G[V \setminus S^*]$,
- information regarding the private cycle of the vertices of $S \cap B_t$: a coloring of all $u \in S \cap B_t$ which matches the color of the connected component T_u of $V \setminus S^*$, where $G[T_u \cup \{u\}]$ is not acyclic.

Note that we need at most $\mathrm{tw}+1$ different colors, as we cannot have more that $\mathrm{tw}+1$ connected components appearing in a bag and we can reuse the colors. We will keep these colors in a table C.

For the vertices of the partial solution, we also need to consider whether they have found both their neighbors in the private cycle or not. To do so, for a vertex $u \in S$ colored c, we will distinguish between two cases. First, consider the case where there exist two vertices $v_1, v_2 \in N(u)$ such that, $v_1, v_2 \in B_t^{\downarrow} \setminus B_t$ belong to the same connected component of $G[V \setminus S^*]$ and $C[v_1] = C[v_2] = C[u]$ when we considered the bags of nodes t_1, t_2 where $B_{t_1} \supseteq \{u, v_1\}$ and $B_{t_2} \supseteq \{u, v_2\}$ respectively. For the second case, no such two vertices have been found yet. Consequently, we need to remember the number $i \leq 1$ of same colored neighbors of u in $B_t^{\downarrow} \setminus B_t$ that belong to a connected component T of the forest $G[V \setminus S^*]$ that has vertices in B_t (i.e. $T \cap B_t \neq \emptyset$). We will store this information in a table D by setting $D[u] = 2$ in the first case and $D[u] = i$ in the second case, for each $u \in S \cap B_t$ that belongs to the partial solution.

Moreover, it is imperative that we keep information regarding the connectivity of the forest vertices that appear in B_t, since otherwise cycles might be formed when we consider Introduce or Join Nodes. In particular, when considering a node t, we want to remember the subsets of vertices $T \subseteq B_t \setminus S$ such that, all $u \in T$ have the same color, all $u \in T$ are in the same connected component in $G[B_t^{\downarrow} \setminus S]$, and $G[T]$ is disconnected. We will call such subsets as *interesting*.

In order to store this information, we employ a second coloring on the vertices, kept in a table F. In particular, let all vertices belonging to the same interesting subset share the same color, while vertices of different interesting subsets are distinguished by different colors. We are going to use colors f_i, for $i \in [\text{tw}+1]$, for these components. Lastly, we will also use an extra color f_0 to distinguish all the vertices of B_t that are not included in any interesting component (including the vertices of $S \cap B_t$).

All of the previously described information will be kept in a tuple, each one of which represents a (partial) solution, affiliated with a node t of the tree decomposition. In particular, each tuple is of the form $s = \{S \cap B_t, |S|, C, D, F\}$, where C, F are tables defined on the vertices of B_t and D table defined on vertices of $S \cap B_t$. Note that, if there exist two tuples s_1 and s_2, differing only on the cardinality of the partial solutions, it suffices to only keep the one of the largest size. Now we will explain how we deal with the different kind of nodes in the tree decomposition.

Leaf Nodes. Since the bags of Leaf Nodes are empty, it follows that we keep an empty partial solution and all the tables C, D and F are also empty.

Introduce Nodes. Let t be an Introduce Node, where t' is its child node and u is the newly introduced vertex. We will build all the partial solutions of t by considering the partial solutions of t' and all possibilities for vertex u. In particular, notice that, for any partial solution S of t, $S' = S \setminus \{u\}$ corresponds to a partial solution for t'. Assume that for the partial solution S' we have stored the tuple $s' = \{X', |S'|, C', D', F'\}$, where $X' = S' \cap B_{t'}$, for t'. We build the tuple for S by considering all cases for the vertex u.

First we consider the case where u belongs to the partial solution, i.e. $u \in S$, and has a private cycle using vertices of color c. Note that the values of C', D' and F' must remain the same for all vertices $v \in B_{t'}$ because u is included in the partial solution. So, for the tables C, D and F it suffices to extend them to C', D' and F' by setting $C[u] = c$, $D[u] = 0$ and $F[u] = f_0$ respectively. In particular, we create the tuple $s = \{X' \cup \{u\}, |S'| + 1, C', D', F'\}$ for t. Notice that $D[u] = 0$, since u has no neighbors in $B_t^{\downarrow} \setminus B_t$.

Now we consider the case where $u \notin S$ and it is colored c. Note that, if there exists vertex $v \in N(u) \cap (B_t \setminus S)$ such that $C[v] \neq c$, then we discard this solution, since it should hold that $C[u] = C[v]$, otherwise we will use two colors for the same connected component of the final forest. Also, if u has at least two neighbors $v_1, v_2 \in N(u) \cap B_t$ such that v_1, v_2 are in the same component of $G[B_{t'} \setminus S]$ or $F'[v_1] = F'[v_2] \neq f_0$, then $G[B_t^{\downarrow} \setminus S]$ contains a cycle and we discard this solution. If none of the previous hold, then this is a valid partial solution. We build the tuple $s = \{X, |S|, C, D, F\}$ for this solution as follows. For the tables C and D note that, for any vertex $v \in B_{t'}$, since $u \notin S$, $D[v] = D'[v]$ and $C[v] = C'[v]$. For u, we just set $C[u] = c$. We also need to modify the table F accordingly. Notice that, for the vertices v of $B_{t'}$ which do not belong to $N(u)$, it suffices to set $F[v] = F'[v]$, since the introduction of u does not create any extra interesting components involving those vertices. Also, the vertices v of S always have $F[v] = f_0$. It remains to determine the value of F for the vertices belonging to $N(u) \setminus S$, for which there are two cases.

Case 1. For all $v \in N(u) \setminus S$, $C'[v] = c$ and $F'[v] = f_0$. In this case, there is no interesting component $T \subseteq B_{t'} \setminus S$ that includes any $v \in N(u) \setminus S$. Also the addition of u does not create such a component. Therefore we set $F[u] = f_0$ and $F[v] = F'[v]$ for all $v \in N(u) \setminus S$.

Case 2. There is at least one vertex $v \in N(u) \setminus S$ such that $F[v] \neq f_0$. Note that, if there are more than one such vertices, then they must belong to different components of $G[B_t^{\downarrow} \setminus S]$ as otherwise we have discarded this solution. In this case, the modification we need to make is to change the color of table F of all vertices that can be reached by u. In particular, let $L_u = \{f_i \mid f_i = F'[v] \neq f_0,$ for a vertex $v \in N(u)\}$ be the list of colors different than f_0

that appear in the neighborhood of u when we consider the table F'. Here, we select a color $f \in L_u$ and we set $F[v] = f$ for all vertices $v \in B_{t'}$ such that $F'[v] \in L_u$ and all vertices colored f_0 that belong in the same component as u in $G[B_t \setminus S]$.

Join Nodes. Let t be a Join Node and t_1 and t_2 be its two children. Note that, for any partial solution S for t, $S_1 = S \cap B^{\downarrow}_{t_1}$ and $S_2 = S \cap B^{\downarrow}_{t_2}$ are partial solutions for t_1 and t_2 respectively. Also, in the tuples $s_1 = \{S_1 \cap B_{t_1}, |S_1|, C_1, D_1, F_1\}$ and $s_2 = \{S_2 \cap B_{t_2}, |S_2|, C_2, D_2, F_2\}$ that we have been stored for S_1 and S_2 in t_1 and t_2 respectively, we must have $S_1 \cap B_{t_1} = S_2 \cap B_{t_2} = S \cap B_t = X$ and $C_1[v] = C_2[v]$, for all $v \in B_t$. Therefore, we can create all partial solutions of t by considering the partial solutions of t_1 and t_2 that respect those constraints. Now, assume that we have two tuples $s_1 = \{X, |S_1|, C, D_1, F_1\}$ and $s_2 = \{X, |S_2|, C, D_2, F_2\}$, for partial solutions S_1 and S_2 for t_1 and t_2 respectively. We want to create a partial solution S for t only if S_1 and S_2 do not result in a cycle in $G[B^{\downarrow}_t \setminus (S_1 \cup S_2)]$. Since S_1 and S_2 are partial solutions of t_1 and t_2 respectively, such a cycle must use vertices in both $B^{\downarrow}_{t_1} \setminus (S_1 \cup B_{t_1})$ and $B^{\downarrow}_{t_2} \setminus (S_2 \cup B_{t_2})$. Additionally, note that such a cycle may appear only if at least two vertices $v_1, v_2 \in B_t$, where $C[v_1] = C[v_2]$, belong in different connected components in $G[B_t \setminus X]$ and in interesting components in both t_1 and t_2 (i.e. $F_1[v_1] = F_1[v_2]$ and $F_2[v_1] = F_2[v_2]$). In that case, we discard such a solution. Alternatively, it is a valid one.

If the solution is valid, we need to create the tables D and F. For any vertex $v \in X$, we set $D[v] = \min\{2, D_1[v] + D_2[v]\}$. To see that this is a correct value for $D[v]$ first recall that the maximum value of $D[u]$ is 2. Also assume that the color we have set for v is c. If $D_1[v] = 2$ or $D_2[v] = 2$ then we have already found the two needed neighbors so obviously $D[v] = \min\{2, D_1[v] + D_2[v]\}$. Otherwise, $D_1[v] \neq 1$ and $D_2[v] \neq 1$. Here, the correct value is $D_1[v] + D_2[v]$ since these vertices must belong in the same connected component as colored c vertices that remain in B_t.

Now, we need to create the table F. Since we have the tables F_1 and F_2, and also the colors for the vertices of $G[B_t \setminus X]$ we can build the table F in $\mathrm{tw}^{O(1)}$.

Finally, regarding the size of the partial solution, that is $|S| = |S_1| + |S_2| - |X|$, since the vertices of X are present in both S_1 and S_2.

Forget Nodes. Let t be a Forget Node, where t' denotes its child node and u the forgotten vertex. Note that any partial solution for t can be constructed by a partial solution of t'. Therefore, we construct the partial solutions for t as follows. Let $s' = \{X', |S'|, C', D', F'\}$ be a tuple representing a partial solution S' for node t'. If u is included in X', then we need to verify whether it has found at least 2 of its neighbors which are included in its private cycle in the potential final solution. To do so, we first define the set U as follows. If $C[u] = c$, we set $U = \{v \in (N(u) \cap B_{t'}) \setminus X' \mid C'[v] = c\}$. Now, if $D[u] + |U| \geq 2$, then we have a valid partial solution for t and we construct a tuple $s = \{X' \setminus \{u\}, |S'|, C, D, F\}$ where $C[v] = C'[v]$ for all $v \in B_t$, $D[v] = D'[v]$ for all $v \in X' \setminus \{u\}$ and $F[v] = F'[v]$ for all $v \in B_t$. Otherwise, $D[u] + |U| < 2$ and we discard this tuple.

In the case that $u \notin X'$, we need to consider the interesting components before and after its removal. As we have mentioned, we want all the vertices in $B_t \setminus X'$ that share the same color in table C to belong in the same component in the potential final forest. Because of that we need to consider several cases. Let $C'[u] = c$ and U_c be the connected component of $G[B'_t \setminus X']$ that u belongs in.

Case 1. For all vertices $v \in B_{t'} \setminus (X \cup \{u\})$, it holds that $C'[v] \neq C'[u]$, i.e. u is the only c colored vertex of $B_{t'} \setminus X'$. In this case, there is no connected component of $B_t \setminus X'$ colored c, and no interesting component is affected, thus $F[v] = F'[v]$, for all $v \in B_t$. However, we need to modify the table D'. For the vertices $v \in X$ such that $C'[v] \neq c$, it holds that $D[v] = D'[v]$ as their same colored forgotten neighbors remain the same. The same holds for the vertices $v \in X$ such that $C[v] = c$ and $D'[v] = 2$ as they already their neighbors in their private cycle. However, for any vertex $v \in X$ colored c that has $D'[v] < 2$, we need to find a new component colored c for its private cycle. Therefore, for these vertices, we set $D[v] = 0$ as we have no color c connected component at the moment.

Case 2. There is at least one vertex $v \in B_{t'} \setminus (X \cup \{u\})$, such that $C'[v] = C'[u]$, while $U_c = \{u\}$. Note that, if $F'[u] = f_0$ then u does not belong in any interesting component of $B_{t'} \setminus X$. Therefore, we discard this tuple as u should be connected to the other c colored vertices of $B_{t'} \setminus (X \cup \{u\})$ in the final forest. Consequently, we can assume that $F'[u] \neq f_0$. Now, let $U'_c \subseteq B_{t'}$ be the set $\{v \in B_{t'} \mid F'[v] = F'[u]\}$. We need to consider two cases, a) when all the vertices of $U'_c \setminus \{u\}$ are in the same connected component in $G[B_t \setminus X']$ and b) when they are not. In case a), we set $F[v] = f_0$ for all vertices $v \in U'_c \setminus \{u\}$ as they do not need any forgotten vertices in order to maintain connectivity between them. If $U'_c \setminus \{u\}$ is not a connected component, then we set $F[v] = F'[v] \neq f_0$ for all $G[B_{t'} \setminus X]$ as removing u does not change the fact that $U'_c \setminus \{u\}$ is still an interesting component of $B_t \setminus X$.

Case 3. There is at least one vertex $v \in B_{t'} \setminus (X \cup \{u\})$, such that $C'[v] = C'[u]$, while $U_c \supset \{u\}$. Now we consider two cases, $F'[u] = f_0$ and $F'[u] \neq f_0$.

Case 3.a. $F'[u] \neq f_0$. In this case, we know that all the vertices of $v \in U_c$ have $F'[v] = F'[u]$. Also, there are vertices $v \notin U_c$ such that $F'[v] = F'[u]$. Therefore, even if we remove u, the other vertices of U_c still belongs in the interesting component colored $F'[u]$. Finally the removal of u does not change the other interesting components. Thus, $F[v] = F'[v]$ for all $v \in B_t$.

Case 3.b. $F'[u] = f_0$. Here we need to consider the connectivity of $G[U_c \setminus \{u\}]$ in order to decide the values in F. If $G[U_c \setminus \{u\}]$ is connected then we do not need to change the colors of F' for any vertex in B_t. On the other hand, if $G[U_c \setminus \{u\}]$ is not connected then $U_c \setminus \{u\}$ comprises a new interesting component in B_t. Therefore, for every vertex v of $U_c \setminus \{u\}$ we set $F[v] = f$, where f is a color that does not appear in F'. Also we keep the same values for all other vertices in B_t.

Finally, for both cases 2 and 3, we need to create a new table D. For the same reasons as in case 1, for the vertices $v \in S$ such that $C[v] \neq c$ or $D[v] = 2$, we set $D[v] = D'[v]$. Also, for vertices $v \in S$, such that $C[v] = C[u]$ and $D'[v] < 2$, we need to check whether $u \in N(v)$ or not. If $u \notin N(v)$ we set $D[v] = D'[v]$ otherwise $D[v] = D'[v] + 1$.

Now we consider the running time. First we calculate the number of different partial solutions for each node. Observe that for each vertex of a bag we have two cases, since it is either included in the (partial) solution or not. Also, we have $\mathrm{tw} + 1$ different choices per vertex, for the tables C and F. Finally, for each vertex in the solution we have three choices for the table D. In total, we have $O(\mathrm{tw})$ choices per vertex. Therefore, we keep at most $\mathrm{tw}^{O(\mathrm{tw})}$ tuples for each node of the tree decomposition. Now, notice that in the dynamic programming part of the algorithm, we can create all the tuples for Introduce and Forget Nodes in time $T \cdot |V|^{O(1)}$ where T is the number of tuples we have stored for the child of the node we consider. Therefore, we can compute all tuples for these nodes in $\mathrm{tw}^{O(\mathrm{tw})}|V|^{O(1)}$ time. For the Join Nodes, in the worst case, we many need to consider all pairs s_1, s_2 of tuples where s_1 and s_2 are tuples corresponding to the first and second child of the Join Node respectively. However, as all the other calculations remain polynomial to the number of vertices, the time required to compute the tuples for this node is again $\mathrm{tw}^{O(\mathrm{tw})}|V|^{O(1)}$. Therefore, the total running time is $\mathrm{tw}^{O(\mathrm{tw})}$. ◀

B Proofs for Section 4 (ETH Lower Bound)

▶ **Theorem 3.** *3-Partitioned-3-SAT cannot be decided in time $2^{o(n)}$, unless the ETH fails.*

Proof. Let ϕ be a 3-SAT formula of m clauses, where V denotes the set of its variables and $|V| = n$. We will construct an equivalent instance ϕ' of 3-Partitioned-3-SAT as follows:

- For every variable $x \in V$, introduce variables $x_i \in V_i$, for $i \in [3]$.
- For every clause $x \vee y \vee z$ of ϕ, introduce a clause $x_1 \vee y_2 \vee z_3$ in ϕ'. In an analogous way, for every clause $x \vee y$ of ϕ, introduce a clause $x_1 \vee y_2$ in ϕ'.
- Introduce clauses $\neg x_1 \vee x_2$, $\neg x_2 \vee x_3$ and $\neg x_3 \vee x_1$ in ϕ'. Note that these clauses are all satisfied if and only if variables x_1, x_2 and x_3 share the same assignment, i.e. either all are true or false.

Let $V' = V_1 \cup V_2 \cup V_3$. Notice that this is a valid 3-Partitioned-3-SAT instance, since $|V_i| = n$ and in none of the $m + 3n$ clauses of ϕ' variables belonging to the same V_i appear.

It holds that ϕ is satisfiable if and only if ϕ' is satisfiable:

$\Longrightarrow$ If ϕ is satisfied by some assignment $f : V \to \{T, F\}$, then consider the assignment $f' : V' \to \{T, F\}$, where $f'(x_i) = f(x)$, for $i \in [3]$ and $x \in V$. This is a satisfying assignment for ϕ'.

$\Longleftarrow$ If ϕ' is satisfied by some assignment $f' : V' \to \{T, F\}$, then it holds that $f'(x_1) = f'(x_2) = f'(x_3)$. Then, consider the assignment $f : V \to \{T, F\}$ where $f(x) = f(x_i)$, for $x \in V$. This is a satisfying assignment for ϕ.

Lastly, assume there exists a $2^{o(|V_i|)}$ algorithm deciding whether ϕ' is satisfiable. Then, since $|V_i|$ is equal to the number of variables of ϕ, 3-SAT could be decided in $2^{o(n)}$, thus the ETH fails. Consequently, unless the ETH is false, there is no $2^{o(n)}$ algorithm deciding if ϕ' is satisfiable, where $n = |V_i|$. ◀

▶ **Lemma 1.** *Any minimal feedback vertex set S of G of size at least k has the following properties:*
(i) *S does not contain any vertex attached with a force gadget or its gadget twin,*
(ii) *$|M_i \setminus S| \leq 1$, for every G_p^q and $i \in [2L]$,*
(iii) *$|S \cap V(G_p^q)| = 4AL + AR + 2LR$,*
where $p \in [3]$ and $q \in [\log n]$.

Proof. Let S be a minimal feedback vertex set of size $|S| \geq k > (4L + R) \cdot 3A \log n$. Let u be a vertex attached with a force gadget, and $\bar{u}$ its gadget twin.

For the first statement, suppose that $u, \bar{u} \in S$. In that case, $S \setminus \{\bar{u}\}$ remains a feedback vertex set, thus S cannot be minimal. On the other hand, if one of $u, \bar{u}$ belongs to S, then $|S| \leq |G| - (A + 1)$, since S cannot include the rest of the vertices of the corresponding force gadget, due to minimality. However, for the defined A and sufficiently large n, this leads to a contradiction, since

$$
\begin{aligned}
(4L + R) \cdot 3A \log n \leq |G| - A - 1 \iff \\
(4L + R) \cdot 3A \log n \leq m + (8L + 4AL + 2R + AR + 2L(2 + R))3 \log n - A - 1 \iff \\
n^2 \leq (12L + 2R + 2LR)3 \log n - 1 = O\left(\frac{n\sqrt{n}}{\log n}\right).
\end{aligned}
$$

Consequently, $u, \bar{u} \notin S$, for any vertex u attached with a force gadget.

For the second statement, let G_p^q for some $p \in [3]$ and $q \in [\log n]$, and $Y_i = S \cap X_i$, for choice set X_i, where $i \in [2L]$. Since S does not contain any vertices attached with a force gadget, it must contain at least $R-1$ vertices of M_i. If not, there exists a $\ell_i, m^i_j, \ell'_i, m^i_{j'}$ cycle. Therefore, $|M_i \setminus S| \leq 1$.

Lastly, S should contain an additional vertex per choice set, since a $\kappa_i, \lambda_i, m^i_j$ cycle remains otherwise. Hence, $|Y_i| \geq R$. Suppose that $|Y_i| > R$. In that case, if $M_i \subseteq Y_i$, then Y_i contains at least one of κ_i and λ_i. However, $S' = S \setminus \{\kappa_i, \lambda_i\}$ remains a feedback vertex set, thus S is not minimal. Alternatively, Y_i contains both κ_i, λ_i and all but one element of M_i. However, $S' = S \setminus \{\lambda_i\}$ remains a feedback vertex set, thus S is not minimal.

Since S includes A vertices per force gadget and exactly R vertices per choice set, property (iii) follows. ◀

▶ **Lemma 2.** *If ϕ has a satisfying assignment, then G has a minimal feedback vertex set of size at least k.*

Proof. Assume that ϕ has a satisfying assignment $f : V \to \{T, F\}$. For each set of variables V^q_p, consider the corresponding G^q_p. For each vertex ℓ_α in G^q_p, which represents a subset $\mathcal{V}_\alpha \subseteq V^q_p$, there exists a β such that m^α_β corresponds to the restriction of f to $\mathcal{V}_\alpha$. Moreover, each variable $x \in V^q_p$ is uniquely represented by some vertex ℓ_α in G^q_p. Let S be a set of size k containing:

- all the A gadget leaves per force gadget,
- all the $2L \cdot 3 \log n$ vertices κ_i,
- $m^\alpha_{\beta'}$, with $\beta' \neq \beta$, for each G^q_p and each subset $\mathcal{V}_\alpha \subseteq V^q_p$, where m^α_β corresponds to the restriction of f to $\mathcal{V}_\alpha$,
- all clause vertices $c_1, \dots, c_m$.

Claim. S is a feedback vertex set: Since S contains all the clause vertices c_i, the only possible remaining cycles concern vertices in the same G^q_p. Since S contains all the gadget leaves per force gadget, all the vertices attached with a force gadget do not belong to S. All λ vertices have a single neighbor, hence cannot be part of any cycle. Moreover, vertices ℓ and ℓ' cannot be part of a cycle, since they are of degree 2 and one of their neighbors (their gadget twin) is a leaf. Therefore, any possible cycle contains vertices r and m. However, vertices m form an independent set, and each of them has a single vertex r as neighbor. Finally, vertices r also form an independent set. Consequently, $G - S$ cannot have any cycles.

Claim. S is a minimal feedback vertex set: Assume there exists $u \in S$ such that $S \setminus \{u\}$ is a feedback vertex set. In that case, u cannot be a vertex leaf introduced by a force gadget, since both the vertex it is attached to as well as the latter's gadget twin do not belong to S. On the other hand, if u were a vertex $m^\alpha_{\beta'}$, then a $\ell_\alpha - m^\alpha_\beta - \ell'_\alpha - m^\alpha_{\beta'}$ cycle would remain. Furthermore, if it were a κ_α vertex, then a $\kappa_\alpha - \lambda_\alpha - m^\alpha_\beta$ cycle would remain. Lastly, u cannot be any clause vertex c. Indeed, for any c, there exists a variable x due to which c is satisfied. Consequently, there exists ℓ_α representing $\mathcal{V}_\alpha \ni x$, as well as $m^\alpha_\beta \notin S$ encoding said satisfying assignment. Therefore, $\ell_\alpha - m^\alpha_\beta - r_\beta - c$ comprises a cycle, because we connect c to all vertices r_j that encode a satisfying assignment for c. ◀

▶ **Lemma 3.** *If G has a minimal feedback vertex set of size at least k, then ϕ has a satisfying assignment.*

Proof. Let S denote said minimal feedback vertex set. Due to Lemma 1, it follows that $c_i \in S$, for all $i \in [m]$, otherwise S cannot reach the stated size.

Since S is minimal, it holds that, for all clause vertices c, $S \setminus \{c\}$ is not a feedback vertex set. Consequently, $G - (S \setminus \{c\})$ contains at least one cycle involving vertex c. Notice that each such cycle can only involve vertices belonging to a specific G^q_p, since vertices not belonging to the same G_p^q can only be connected via paths containing vertices c_i, but only a single such vertex remains in $G-(S\setminus\{c\})$. Let $G_c = G[(V(G_p^q)\setminus S)\cup\{c\}]$ be a subgraph of G containing one such cycle.

We will show that the aforementioned cycle must be of the form $\ell_i - m_j^i - r_j - c$, for some i and j. In order to do so, first notice that there is no path in $G_c - \{c\}$ between any two r vertices. Suppose there exists such a path, connecting r_α and r_β, for $\alpha,\beta \in [R]$ and $\alpha \neq \beta$. This path cannot involve only r vertices, since they constitute an independent set. Additionally, it cannot involve only r and m vertices, since each m vertex has a single r vertex in its neighborhood, while m vertices also induce an independent set. Therefore, any path from r_α to r_β must include a vertex ℓ_γ or ℓ'_γ for some γ, denoted by w_γ. In that case, the shortest such path must be of the form $r_\alpha - m_\alpha^\gamma - w_\gamma - m_\beta^\gamma - r_\beta$. However, this cannot be the case, since G_c contains at most one vertex belonging to M_γ, due to Lemma 1.

Consequently, any cycle that contains c in G_c must include the unique vertex ℓ_i that is a neighbor of c. Moreover, as the only other vertices that are adjacent to c are r vertices, and there are no paths between any two r vertices, the cycle must be of the form $\ell_i - m_j^i - r_j - c$ for some j.

Now, consider the following assignment for the variables of ϕ: for a set of variables $\mathcal{V}_\alpha \subseteq V_p^q$ represented by ℓ_α in G_p^q, if there exists a vertex $m_\beta^\alpha \notin S$ for some β, then let these variables have the assignment encoded by this choice. Alternatively, if there is no such vertex m, let all of these variables have a truthful assignment. This is valid assignment, since every variable of ϕ appears in a single variable set $\mathcal{V}_\alpha \subseteq V_p^q$, for some $p \in [3]$ and $q \in [\log n]$, which is uniquely represented by a single vertex ℓ_α in G_p^q, while $|M_\alpha \setminus S| \leq 1$. Lastly, this is a satisfying assignment, since for every clause vertex c, there exist neighboring vertices ℓ_α and r_β, such that $m_\beta^\alpha \notin S$, i.e. for every clause, there exists at least one variable in $\mathcal{V}_\alpha$ encoded by ℓ_α such that its assignment satisfies the clause. ◀

▶ **Lemma 4.** $\mathrm{vc}(G) = O(n/\log n)$.

Proof. Notice that the deletion of all vertices $\ell_i, \ell'_i, r_i, \kappa_i$ and λ_i, as well as their gadget twins, induces an independent set. Therefore,

$$\mathrm{vc}(G) \leq (8L + 2R + 4L)\cdot 3\log n = O(n/\log n).$$

◀

C Proofs for Section 5 (Natural Parameter Algorithm)

▶ **Theorem 6.** *Annotated MMFVS is NP-hard on path restricted instances, even if all the paths are of length* 2.

Proof. Let graph $G = (V, E)$, where $|V| = n$ and $|E| = m$, be an instance of 3-Coloring. We will construct an equivalent (G', S, F, k) instance of Annotated MMFVS. Construct graph $G' = (V', E')$, such that

- introduce $w \in V'$,
- for every vertex $u_i \in V$, introduce $u_j^i \in V'$, where $j \in [3]$,
- for every edge $e_i \in E$, introduce $e_j^i \in V'$ and $\{e_j^i, w\} \in E'$, where $j \in [3]$,
- introduce edges $\{w, u_1^i\}, \{u_1^i, u_2^i\}, \{u_2^i, u_3^i\}$ and $\{u_3^i, w\}$ in E', for all $i \in [n]$,
- for every edge $e_i = \{u_k, u_\ell\} \in E$, introduce edges $\{e_j^i, u_j^k\}, \{e_j^i, u_j^\ell\} \in E'$, where $j \in [3]$.

Set $F = \{w\}$, $S = \{e_j^i \in V' \mid i \in [m], j \in [3]\}$ and $k = n+3m$. Moreover, let $U_i = \{u_1^i, u_2^i, u_3^i\}$, for all $i \in [n]$. Notice that this is a valid instance of Annotated MMFVS. In Figure 2 part

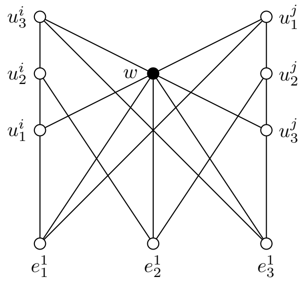


Figure 2 Part of the graph depicting vertices associated with $e_1 = \{u_i, u_j\} \in E$. Black vertex w belongs to F.

of the construction is shown, assuming there exists an edge $e_1 = \{u_i, u_j\} \in E$. It remains to show that the two problems are equivalent.

Assume that G has a valid 3-coloring, e.g. $f : V \to [3]$. Let $S' = \{u_j^i \in V' \mid f(u_i) = j\} \cup S$ be a set of size $n + 3m$. S' is a feedback vertex set of G'. Indeed, since it contains all vertices e_j^i, the only remaining cycles are due to the vertices of U_i and w, for every $i \in [n]$, but $|S' \cap U_i| = 1$, for every i. It remains to show that S' is minimal. $S_1 = S' \setminus \{u_j^i\}$ is not a feedback vertex set, for any $u_j^i \in S'$, since then $w, u_j^i \notin S_1$, for $j \in [3]$. Additionally, $S_2 = S' \setminus \{e_j^i\}$ is not a feedback vertex set, for any $e_j^i \in S'$. Assume that $e_i = \{u_p, u_q\}$. Then, since $f(u_p) \neq f(u_q)$, it holds that at least one of u_j^p, u_j^q does not belong to S'. Name this vertex v_j, and notice that since $|S' \cap U_j| = 1$, there exists a path from v_j to w containing only vertices of U_j. In that case, since e_j^i has an edge with w and v_j is a neighbor of e_j^i, it follows that S_2 is not a feedback vertex set.

Assume that G' has a minimal feedback vertex set $S' \supseteq S$, where $S' \cap F = \emptyset$ and $|S'| \geq n + 3m$. Then, if $u_k^i, u_\ell^i \in S'$ for some i and some $k \neq \ell \in [3]$, S' is not minimal, since $S' \setminus \{u_k^i\}$ remains a feedback vertex set. Consequently, S' contains a single element from each U_i. Now consider the coloring $f : V \to [3]$ where $f(u_i) = j$ if $u_j^i \in S'$. In that case, for f to be a valid coloring, it suffices to prove that if $\{u_i, u_j\} \in E$, then $u_k^i, u_\ell^j \in S'$ for $k \neq \ell$. Assume that this is not the case, i.e. there exist $u_k^i, u_k^j \in S'$ and $e = \{u_i, u_j\} \in E$. In that case, $S' \setminus \{e_k\}$ remains a feedback vertex set, since e_k only has a single neighbor not belonging to S', contradiction. ◀

▶ **Theorem 7.** *Let $\mathcal{I} = (G, S, F, k)$ be a path restricted instance of* Annotated MMFVS*, and let g denote the number of its good vertices. There is an algorithm running in time $O(3^{k-g} n^{O(1)})$ which either returns a minimal feedback vertex set $S' \subseteq S \cup U$ of G of size at least k or concludes that $\mathcal{I}$ is a No instance of* Annotated MMFVS*.*

Proof. The main idea of the algorithm lies on the fact that we can efficiently handle instances where either $k = 0$ or $S = \emptyset$. Towards this, we will employ a branching strategy that, as long as S remains non empty, new instances with reduced k are produced. Prior to performing branching, we first observe that we can efficiently deal with the good vertices. Afterwards, by employing said branching strategy, in every step we decide which vertex will be counted towards the k required, thereby reducing parameter k on each iteration. If at some point $k = 0$ or $S = \emptyset$, it remains to decide whether this comprises a viable solution S'. Notice that S' may not be a solution for the annotated instance, since even if $|S'| \geq k$, it does not necessarily hold that $S' \supseteq S$.

We first show that indeed, the case where either $k = 0$ or $S = \emptyset$ can be efficiently decided. Afterwards, we present the algorithm and finally we argue about its correctness.

▶ **Lemma 9.** *Let $\mathcal{I} = (G, S, F, k)$ be a path restricted instance of* Annotated MMFVS *and $S^* \subseteq S \cup U$ a minimal feedback vertex set of G, where $F^* = V(G) \setminus S^*$ denotes the corresponding forest.*

(i) *From every path of $G[U]$, at most one vertex belongs to S^*.*

(ii) *Let $u, v \in F^*$. Then, u and v are in the same connected component of $G[F \cup U]$ if and only if they are in the same connected component of $G[F^*]$.*

Proof. For the first statement, suppose there exist $u_1, u_2 \in S^* \cap U$ belonging to the same connected component of $G[U]$, and let $P_u \subseteq U$ denote the set of the vertices belonging to said component. In that case, $G[F^* \cup \{u_1\}]$ must contain a cycle involving u_1. Since $\mathcal{I}$ is a path restricted instance, it holds that $\forall v \in P_u$, $\deg_{F \cup U}(v) = 2$, and since $F^* \cup \{u_1\} \subseteq F \cup U$, $\deg_{F^* \cup \{u_1\}}(v) \leq 2$ follows. Therefore, for $G[F^* \cup \{u_1\}]$ to contain a cycle it holds that $F^* \supseteq P_u \setminus \{u_1\}$, contradiction.

For the second statement, first consider the case when $u, v \in F$, both belonging to the same connected component of $G[F \cup U]$. If u, v are connected in $G[F^*]$, we are done. Suppose that this is not the case. Assume there exists a path of U the endpoints of which have an edge towards both u and v. Then, either this path belongs entirely to F^*, or one of its vertices, say w, is in S^*. In the first case, u and v are in the same connected component of F^* due to said path. In the latter case, the private cycle of w in $G[F^* \cup \{w\}]$ contains both u and v, thus they are in the same connected component of F^*. Therefore, the statement holds. If no such path connecting u and v exists in U, let P be the path of $G[F \cup U]$ connecting u and v, where $f_1, \ldots, f_j$ are the vertices of P belonging to F in the order that they appear in P. Then, due to the previous arguments, any consecutive vertices f_i, f_{i+1} are in the same connected component of F^*. Lastly, due to transitivity of connectivity, the statement follows.

In case at least one of u, v belongs to U, let $F' = F \cup \{u, v\}$ and consider the instance $\mathcal{I}' = (G, S, F', k)$. Obviously, u, v are in the same connected component of $G[F \cup U]$ if and only if they are in the same connected component of $G[F' \cup U']$, where $U' = U \setminus \{u, v\}$. Moreover, any $S^* \not\ni u, v$ is a solution of instance $\mathcal{I}$ if and only if it is a solution of $\mathcal{I}'$. Thus, the statement follows.

Since $F^* \subseteq F \cup U$, the converse direction also holds. Consequently, if $u, v \in F^*$, then u, v are in the same connected component of $G[F \cup U]$ if and only if u, v are in the same connected component of $G[F^*]$. ◀

Due to Lemma 9, we can therefore infer the connected components of any forest F^* corresponding to a minimal feedback vertex set $S^* \subseteq S \cup U$ of G. Based on this property, we will establish the following reduction rules.

Rule (i). Let $\mathcal{I} = (G, S, F, k)$ be a path restricted instance of Annotated MMFVS, and $u \in U$ such that the connected components of $G[(F \cup U) \setminus \{u\}]$ are more than the connected components of $G[F \cup U]$. Then, replace $\mathcal{I}$ with $\mathcal{I}' = (G, S, F \cup \{u\}, k)$.

▶ **Lemma 10.** *Applying rule (i) does not change the outcome of the algorithm.*

Proof. Let $\mathcal{I} = (G, S, F, k)$ be a path restricted instance of Annotated MMFVS and $\mathcal{I}' = (G, S, F \cup \{u\}, k)$ be the path restricted instance of Annotated MMFVS resulting from applying Rule (i) to $\mathcal{I}$, where $u \in U$. In that case, the connected components of $G[(F \cup U) \setminus \{u\}]$ are more than the connected components of $G[F \cup U]$. We will show that if $S' \subseteq S \cup U$ is a minimal feedback vertex set of G, then $u \notin S'$. Let $F' = V(G) \setminus S'$ be the corresponding forest. Suppose that $u \in S'$. Then u must have a private cycle in $G[F' \cup \{u\}]$. However, both neighbors of u are in different connected components of $G[(F \cup U) \setminus \{u\}]$, and since $F' \subseteq (F \cup U) \setminus \{u\}$, its neighbors are in different connected components of $G[F']$, hence contradiction. ◀

Rule (ii)**.** Let $\mathcal{I} = (G, S, F, k)$ be a path restricted instance of Annotated MMFVS, and $u \in S$ such that it has two edges towards F such that they are in the same connected component of $G[F \cup U]$. Then, replace $\mathcal{I}$ with $\mathcal{I}' = (G - u, S \setminus \{u\}, F, k - 1)$.

▶ **Lemma 11.** *Applying rule* (ii) *does not change the outcome of the algorithm.*

Proof. Let $\mathcal{I} = (G, S, F, k)$ be a path restricted instance of Annotated MMFVS and $\mathcal{I}' = (G', S \setminus \{u\}, F, k-1)$ be the path restricted instance of Annotated MMFVS resulting from applying Rule (ii) to $\mathcal{I}$, where $u \in S$ and $G' = G - u$. In that case, u has two edges towards F such that they are in the same connected component of $G[F \cup U]$.

Let $S_1 \supseteq S$ be a minimal feedback vertex set of G of size at least k, where $S_1 \cap F = \emptyset$. Then, $S_1 \setminus \{u\}$ is a minimal feedback vertex set of $G - u$, since the private cycles of the vertices of S_1 remain unaffected by the deletion of u, and $\text{ammfvs}(\mathcal{I}) \leq \text{ammfvs}(\mathcal{I}')$ follows.

Let $S' \subseteq (S \cup U) \setminus \{u\}$ be a minimal feedback vertex set of $G - u$, where $F' = V(G-u) \setminus S'$. Since the neighbors of u in F are in the same connected component of $G[F \cup U]$, they are in the same connected component of $G'[F \cup U]$. Consequently, due to Lemma 9, u has two edges towards F such that they are in the same connected component of $G'[F']$, therefore, $S' \cup \{u\}$ is a minimal feedback vertex set of G, since u has a private cycle in $G[F' \cup \{u\}]$. Notice that this also implies that $\text{ammfvs}(\mathcal{I}') \leq \text{ammfvs}(\mathcal{I})$. ◀

Note that, if applying rule (ii) to $\mathcal{I} = (G, S, F, k)$ results in $\mathcal{I}' = (G - u, S \setminus \{u\}, F, k)$, and the algorithm returns a minimal feedback vertex set S' of $G - u$, where $S' \cap F = \emptyset$, then this can be extended to a minimal feedback vertex set $S' \cup \{u\}$ of G, although S' might not be a solution to the annotated instance $\mathcal{I}'$, since $S' \supseteq S$ does not necessarily hold.

Utilizing Lemma 9, we now prove that any instance where either $S = \emptyset$ or $k = 0$ can be solved in polynomial time.

▶ **Lemma 12.** *Let $\mathcal{I} = (G, S, F, k)$ be a path restricted instance of Annotated MMFVS. If $k = 0$ or $S = \emptyset$, we can determine whether G has a minimal feedback vertex set $S' \subseteq S \cup U$ of size at least k in time $n^{O(1)}$.*

Proof. Due to Lemma 9, it holds that for any minimal feedback vertex set $S' \subseteq S \cup U$, if $u, v \in F'$, where $F' = V(G) \setminus S'$, then u and v are in the same connected component if and only if that is the case in $G[F \cup U]$. We will say that a path of U belongs to F' when all of its vertices belong to F'.

Notice that the vertices of F can be partitioned into equivalence classes, depending on their connectivity in $G[F \cup U]$. For $u, v \in F$, let them belong to the same equivalence class C_i if they are in the same connected component of $G[F \cup U]$. Let p denote the number of equivalence classes, where $p \leq |F|$. Now, for each C_i, let c_i be equal to the number of connected components $G[C_i]$. Since every path of U has exactly 2 edges towards F, it holds that the number of paths belonging to F' will be exactly $c_i - 1$ per equivalence class C_i. Intuitively, since all components of $G[C_i]$ must be connected in the final forest, the number of paths required is $c_i - 1$, per equivalence class C_i. Therefore, it suffices to greedily add each path to the final forest F', as long as no cycle is formed. If that is not the case, it suffices to add one of its vertices to S', since it has two edges towards the same connected component of F'. In the end, $G[F']$ has the connectivity dictated by $G[F \cup U]$, while $S' \subseteq S \cup U$ is a minimal feedback vertex set, since all of its elements have a private cycle. If $k = 0$, we are done. Alternatively, if $S = \emptyset$, notice that, due to Lemma 9, S' is a maximum minimal feedback vertex set of G such that $S' \cap F = \emptyset$. In that case, we can determine whether $\mathcal{I}$ is a Yes or No instance, depending on whether $|S'| \geq k$ holds. ◀

Armed with the previous lemmas, we are now ready to describe our algorithm. Let $\mathcal{I} = (G, S, F, k)$ be a path restricted instance of Annotated MMFVS. Notice that if at any point of execution of our algorithm there exists some vertex $s \in S$ which does not have two edges towards the same connected component of $G[F \cup U]$, then this is a No instance of Annotated MMFVS and we discard it. Moreover, we exhaustively apply rules (i) and (ii) in every produced instance. Note that this induces a polynomial time overhead.

Regarding our branching strategy, we consider the different cases for vertices of U. When these vertices are moved from U to S, it is imperative that the connectivity of the vertices belonging to the forest remains the same in any final forest. Since we have assumed that rule (i) has been exhaustively applied, that is indeed the case. We will firstly do some preprocessing and afterwards describe a branching strategy which, as long as S remains non empty, produces instances with reduced k.

Preprocessing. Assume that rule (ii) has already been applied exhaustively. Suppose there still exists some good vertex $h \in H$. Recall that h has at most one neighbor in U. In that case, for h to have a private cycle, it is necessary that its neighbor $u \in N(h) \cap U$ belongs to the forest. Also, u must be in the same connected component of $G[F \cup U]$ as one of the other neighbors of h in F. Therefore, we consider the instance $\mathcal{I}' = (G, S, F \cup \{u\}, k)$ in which rule (ii) can be applied due to h. Therefore, we replace the current instance with the instance $\mathcal{I}'' = (G - h, S \setminus \{h\}, F \cup \{u\}, k - 1)$. Note that the preprocessing can be done in polynomial time while for the resulting instance $\mathcal{I}^* = (G^*, S^*, F^*, k^*)$ it holds that $k^* \leq k - g$.

Branching. Let $s \in S$, where $\mathcal{I} = (G, S, F, k)$ is the instance after the preprocessing. For $u \in U$, let $T_u \subseteq U \setminus \{u\}$ denote the vertices in the same connected component as u in $G[U]$. Consider the following cases: either there exists $u \in N(s) \cap U$ such that u is in the same connected component of $G[F \cup U]$ as some $f \in N(s) \cap F$ or not.

- In the first case, we branch depending on whether u is in the feedback vertex set or not. Notice that if u is in the feedback vertex, then all vertices of T_u must be in the forest due to Lemma 9. Therefore, we replace our current instance with the following two:
  - $\mathcal{I}_1 = (G, S \cup \{u\}, F \cup T_u, k)$, and
  - $\mathcal{I}_2 = (G, S, F \cup \{u\}, k)$

  In both instances we can apply Rule (ii). In particular, in $\mathcal{I}_1$, u has two neighbors in $F \cup T_u$ which are in the same component of $G[F \cup U]$, therefore applying rule Rule (ii) gives $\mathcal{I}_1' = (G - u, S, F \cup T_u, k - 1)$. Also, in $\mathcal{I}_2$, s has two neighbors in $F \cup \{u\}$ which are in the same component of $G[F \cup U]$, therefore, applying Rule (ii) gives $\mathcal{I}_2' = (G - s, S \setminus \{s\}, F \cup \{u\}, k - 1)$.
- In the latter case, two vertices $a, b \in N(s) \cap U$ that belong to the same connected component of $G[F \cup U]$ must exist. For these vertices we branch on the following 3 cases: $a, b \in F$, or $a \in S$, or $b \in S$. Therefore, we replace the current instance with the following three:
  - $\mathcal{I}_1 = (G, S, F \cup \{a, b\}, k)$,
  - $\mathcal{I}_2 = (G, S \cup \{a\}, F \cup T_a, k)$,
  - $\mathcal{I}_3 = (G, S \cup \{b\}, F \cup T_b, k)$.

  Now, in each one of these instances we can apply Rule (ii). Indeed, vertices s, a and b can be used to apply rule (ii) and obtain instances
  - $\mathcal{I}_1' = (G - s, S \setminus \{s\}, F \cup \{a, b\}, k - 1)$,

- $\mathcal{I}'_2 = (G - a, S, F \cup T_a, k - 1)$,
- $\mathcal{I}'_3 = (G - b, S, F \cup T_b, k - 1)$

respectively.

Complexity. The preprocessing part of the algorithm, as well as the application of the rules requires polynomial time. The branching strategy previously described results in at most 3^{k-g} instances, since on every step at most 3 instances may be produced, each with reduced k. Lastly, due to Lemma 12, the case when $S = \emptyset$ or $k = 0$ is solvable in polynomial time. Therefore, the final running time is $3^{k-g} n^{O(1)}$. ◀

▶ **Lemma 5.** *Let $\mathcal{I} = (G, S, F, k)$ be an instance of* Annotated *MMFVS, where $\mu(\mathcal{I}) \leq 1$. Then, G has a minimal feedback vertex set $S' \subseteq S \cup U$ of size at least k.*

Proof. Since F is a forest, $S \cup U$ comprises a valid feedback vertex set of G. Let S' be a minimal feedback vertex set obtained in polynomial time from $S \cup U$, while $F' = V \setminus S'$ denotes the forest resulting from the vertices belonging to F plus the vertices of $(S \cup U) \setminus S'$.

Let a *loss* be when either a good vertex of S, or the entirety of an interesting path belongs to $F' = V \setminus S'$. Notice that both good vertices and interesting paths have at least 2 edges to some vertices of F. Consequently, for every loss, the connected components of F reduce by at least 1: in order to move a good vertex or an interesting path to the forest, no cycles should be formed, i.e. all of their neighbors are in distinct connected components of F, thus the connected components of the forest will be reduced. Therefore, it follows that at most $cc(F) - 1$ losses may happen, which means that S' contains at least $g + p - (cc(F) - 1)$ vertices; each of those corresponds to either a good vertex or belongs to an interesting path which has not moved entirely to F. In that case however, $|S'| \geq g + p - (cc(F) - 1) \geq k$, since $\mu(\mathcal{I}) \leq 1$. ◀

▶ **Lemma 6.** *Let $G = (V, E)$ be a (multi)graph and $uv \in E(G)$. Then, G is acyclic if and only if G/uv is acyclic.*

Proof. First we consider the case that there more that one edges between u and v. In this case, G has a cycle that uses these edges. Therefore, contracting one of these edges results in a self loop in G' and the statement holds. So, we only need to consider the case where there is only one edge between u and v and w does not have a self loop in G/uv.

Suppose that uv is part of a cycle in G. Since G does not include any edges parallel to uv, this cycle has at least three vertices. This means that there exists a path from u to v which does not include the edge uv. Then, in G/uv, this path is a cycle as we have replaced u and v with a single vertex. Moreover, any cycles not including edge uv are not affected by its contraction.

For the other direction, assume that G/uv has a cycle C and let w be the vertex that has replaced u and v in G/uv. There are two cases, either $w \notin C$ or $w \in C$. In the first case notice that C is also a cycle in G therefore the statement holds. In the latter, since we know that w does not have a self loop, there is a path P of size at least 1 such that, the starting and the ending vertices of this path are adjacent to w. Let v_s and v_t be these (not necessarily distinguished) vertices. If there is $v' \in \{u, v\}$ such that $v' \in N(v_s) \cap N(v_t)$ then the path P together with v' comprises a cycle in G. Otherwise, one of v_s, v_t is adjacent to u and the other to v. W.l.o.g. let $v_s u, v_t v \in E(G)$. Notice that there is a path in G that starts with u, ends with v, and uses the vertices in P. Consequently, this path does not include the edge uv. Adding the edge uv to this path results in a cycle in G. ◀

▶ **Lemma 7.** *Applying rules 1, 2 and 3 does not change the outcome of the algorithm and does not increase the measure of progress.*

Proof. We will prove each rule in a distinct paragraph.

Rule 1. Let $\mathcal{I} = (G, S, F, k)$ be an instance of Annotated MMFVS and $\mathcal{I}' = (G', S, F', k)$ be the instance of Annotated MMFVS resulting from applying Rule 1 to $\mathcal{I}$, where $G' = (V', E')$ occurs from the contraction of u and v into w (i.e. $G' = G/uv$), while $F' = (F \cup \{w\}) \setminus \{u, v\}$.

We will show that $\text{ammfvs}(\mathcal{I}') = \text{ammfvs}(\mathcal{I})$ and $\mu(\mathcal{I}') \le \mu(\mathcal{I})$.

Let $S_1 \supseteq S$ be a minimal feedback vertex set of G, such that $S_1 \cap F = \emptyset$. We claim that S_1 is a minimal feedback vertex set of G'. Indeed, $G'[V' \setminus S_1]$ is obtained from $G[V \setminus S_1]$ by contracting uv, so both are acyclic due to Lemma 6. Furthermore, for all $z \in S_1$, $G'[(V' \setminus S_1) \cup \{z\}]$ is obtained from $G[(V \setminus S_1) \cup \{z\}]$ by contracting uv, therefore both have a cycle due to Lemma 6, hence no vertex of S_1 is redundant in G'. Consequently, $\text{ammfvs}(\mathcal{I}) \le \text{ammfvs}(\mathcal{I}')$.

For the other direction, let $S_2 \supseteq S$ be a minimal feedback vertex set of G', such that $S_2 \cap F' = \emptyset$, which implies that $w \notin S_2$. We claim that S_2 is a minimal feedback vertex set of G. Let $F_1 = V \setminus S_2$ and $F_2 = V' \setminus S_2$. By definition, $G'[F_2]$ is acyclic. $G[F_1]$ is also a forest due to Lemma 6 and the fact that $G'[F_2]$ is obtained from $G[F_1]$ by contracting uv. To see that S_2 is minimal, let $z \in S_2$ and consider the graphs $G_1 = G[(V \setminus S_2) \cup \{z\}]$ and $G_2 = G'[(V' \setminus S_2) \cup \{z\}]$. We see that G_2 can be obtained from G_1 by contracting uv. But G_2 must have a cycle, by the minimality of S_2, so, by Lemma 6, G_1 also has a cycle. Thus, S_2 is minimal in G, and $\text{ammfvs}(\mathcal{I}) \ge \text{ammfvs}(\mathcal{I}')$ follows.

Moreover, it holds that $\mu(\mathcal{I}') = \mu(\mathcal{I})$, since $cc(F) = cc(F')$, while p and g are not affected.

Rule 2. Let $\mathcal{I} = (G, S, F, k)$ be an instance of Annotated MMFVS and $\mathcal{I}' = (G', S, F, k)$ be the instance of Annotated MMFVS we take by applying Rule 2 to $\mathcal{I}$, where $G' = (V', E')$ occurs from the deletion of some $u \in U$ such that $\deg_{F \cup U}(u) = 0$ (i.e. $G' = G - u$). We will show that $\text{ammfvs}(\mathcal{I}') = \text{ammfvs}(\mathcal{I})$ and $\mu(\mathcal{I}') \le \mu(\mathcal{I})$.

Let $S_1 \supseteq S$ be a minimal feedback vertex set of G, such that $S_1 \cap F = \emptyset$. Since $N(u) \subseteq S \subseteq S_1$, it follows that $u \notin S_1$, since $S_1 \setminus \{u\}$ remains a feedback vertex set. Then, S_1 is a feedback vertex set of $G - u$. To see that S_1 is also minimal in $G - u$, note that any private cycle of G also exists in $G - u$, since no private cycle contains u. Therefore, $\text{ammfvs}(\mathcal{I}) \le \text{ammfvs}(\mathcal{I}')$.

For the other direction, let $S_2 \supseteq S$ be a minimal feedback vertex set of $G - u$, such that $S_2 \cap F = \emptyset$. We observe that $S_2 \cup \{u\}$ is a feedback vertex set of G. If $S_2 \cup \{u\}$ is minimal, we are done. Alternatively, we delete vertices from it until it becomes minimal. We now note that the only vertex which may be deleted in this process is u, since all vertices of S_2 have a private cycle in $G - u$. Therefore, $\text{ammfvs}(\mathcal{I}) \ge \text{ammfvs}(\mathcal{I}')$.

Lastly, $\mu(\mathcal{I}') \le \mu(\mathcal{I})$, since the deletion of u does not affect $cc(F)$ and p, while g could potentially increase.

Rule 3. Let $\mathcal{I} = (G, S, F, k)$ be an instance of Annotated MMFVS and $\mathcal{I}' = (G', S, F', k)$ be the instance of Annotated MMFVS we take by applying Rule 3 to $\mathcal{I}$, where $G' = (V', E')$ occurs from the contraction of u and v into w (i.e. $G' = G/uv$), for some $u \in U$ such that $\deg_{F \cup U}(u) = 1$, and $v \in N(u) \cap (F \cup U)$. Moreover, it holds that $F' = (F \cup \{w\}) \setminus \{v\}$ if $v \in F$, and $F' = F$ otherwise. We will show that $\text{ammfvs}(\mathcal{I}') = \text{ammfvs}(\mathcal{I})$ and $\mu(\mathcal{I}') \le \mu(\mathcal{I})$.

Notice that, since $\deg_{F\cup U}(u) = 1$, $u \notin S_1$ for any minimal feedback vertex set $S_1 \supseteq S$ of G such that $S_1 \cap F = \emptyset$. We will continue by considering the two cases separately.

First, assume that $v \in F$. Since $u \notin S_1$, we have that $\mathcal{I}$ is a Yes instance of Annotated MMFVS if and only if $\mathcal{J} = (G, S, F \cup \{u\}, k)$ is a Yes instance of Annotated MMFVS. Notice that, by applying Rule 1 on $\mathcal{J}$, the resulting instance is $\mathcal{I}'$. Therefore the statement holds in this case.

It remains to prove the statement when both $u, v \in U$. Assume that this is the case. Let $S_1 \supseteq S$ be a minimal feedback vertex set of G, such that $S_1 \cap F = \emptyset$. We consider two cases: either $v \notin S_1$ or $v \in S_1$.

If $v \notin S_1$, then we claim that S_1 is also a minimal feedback vertex set of G'. Indeed, $G'[V' \setminus S_1]$ is obtained from $G[V \setminus S_1]$ by contracting uv, so, by Lemma 6, both are acyclic. Furthermore, for all $z \in S_1$, $G'[(V' \setminus S_1) \cup \{z\}]$ is obtained from $G[(V \setminus S_1) \cup \{z\}]$ by contracting uv, therefore, by Lemma 6, both have a cycle. So, S_1 is minimal feedback vertex set of G'.

If $v \in S_1$, then we claim that $S^* = (S_1 \setminus \{v\}) \cup \{w\}$ is a minimal feedback vertex set of G'. It is not hard to see that S^* is a feedback vertex set of G', since it corresponds to deleting $S_1 \cup \{u\}$ from G. To see that it is minimal, for all $z \in S^* \setminus \{w\}$ we observe that $G'[(V' \setminus S^*) \cup \{z\}]$ is obtained from $G[(V \setminus S_1) \cup \{z\}]$ by deleting u, which has degree at most 1 due to z. Therefore, this deletion strongly preserves acyclicity. Finally, to see that w is not redundant for S^*, we observe that $G[(V \setminus S_1) \cup \{v\}]$ has a cycle, and a corresponding cycle must be present in $G'[(V' \setminus S^*) \cup \{w\}]$, which is obtained from the former graph by contracting uv.

Consequently, $\text{ammfvs}(\mathcal{I}) \leq \text{ammfvs}(\mathcal{I}')$ follows. For the other direction, let $S_1 \supseteq S$ be a minimal feedback vertex set of G', such that $S_1 \cap F' = \emptyset$. Recall that we consider the case where $u, v \in U$, so $F' = F$. We consider two cases, either $w \in S_1$ or $w \notin S_1$.

If $w \in S_1$, we claim that $S_2 = (S_1 \cup \{v\}) \setminus \{w\}$ is a minimal feedback vertex set of G. Let $F_1 = V' \setminus S_1$ and $F_2 = V \setminus S_2$. Notice that in $G[F_2]$, u is an isolated vertex since $\deg_{F\cup U}(u) = 1$ and $v \in S_2$. Also, $G[F_2 \setminus \{v\}]$ is acyclic since it is the same as $G'[F_1]$. Therefore S_2 is a feedback vertex set of G. We need to show that S_2 is minimal. Let $x \in S_1 \setminus \{w\}$. Notice that in $G[F_2 \cup \{x\}]$, u has degree at most 1 due to x, therefore, it cannot be included in any cycle of $G[F_2 \cup \{x\}]$. This means that $G[F_2 \cup \{x\}]$ has a cycle if and only if $G[(F_2 \setminus \{u\}) \cup \{x\}]$ has a cycle. However, $G[(F_2 \setminus \{u\}) \cup \{x\}]$ has a cycle because it is the same as $G'[F_1 \cup \{x\}]$ and $x \in S_1$. It remains to show that $G[F_2 \cup \{v\}]$ has a cycle. Notice that $G'[F_1 \cup \{w\}]$ can be obtained from $G[F_2 \cup \{v\}]$ by contracting uv. Therefore $G[F_2 \cup \{v\}]$ has a cycle and S_2 is minimal.

If $w \notin S_1$ we claim that S_1 is a minimal feedback vertex set of G. Notice that we can obtain $G'[V' \setminus S_1]$ by contracting uv in $G[V \setminus S_1]$, therefore $G[V \setminus S_1]$ is acyclic and S_1 a feedback vertex set of G. We also need to show minimality. Assume that $x \in S_1$. Since we can obtain $G'[(V' \setminus S_1) \cup \{x\}]$ by contracting uv in $G[(V \setminus S_1) \cup \{x\}]$, we have that $G[(V \setminus S_1) \cup \{x\}]$ has a cycle. Therefore, S_1 is a minimal feedback vertex set of G.

Consequently, $\text{ammfvs}(\mathcal{I}') = \text{ammfvs}(\mathcal{I})$ follows. Lastly, we need to show that $\mu(\mathcal{I}') \leq \mu(\mathcal{I})$ in the case where $u, v \in U$. Indeed, if both of them are in U the contraction does not change the number of components in F, or the number of interesting paths or the number of good vertices in S. ◀

▶ **Lemma 8.** *The branching strategy produces instances of reduced measure of progress, without reducing the number of good vertices.*

Proof. Let $\mathcal{I} = (G, S, F, k)$ be an instance of Annotated MMFVS and $\mathcal{I}_1 = (G, S', F, k)$,

$\mathcal{I}_2 = (G, S, F', k)$ the instances produced by the branching strategy, where $S' = S \cup \{v\}$ and $F' = F \cup \{v\}$ for $v \in U$. Moreover, let g, g_1 and g_2 denote the number of good vertices of each instance respectively. Notice that $g \leq g_1$ and $g \leq g_2$. We assume that none of $\mathcal{I}_1, \mathcal{I}_2$ has been discarded, i.e. $G[F']$ is a forest. Notice that then, if $\deg_F(v) \geq 2$, it follows that v has at least two neighbors in distinct connected components of $G[F]$. We will prove that $\mu(\mathcal{I}_1) < \mu(\mathcal{I})$ and $\mu(\mathcal{I}_2) < \mu(\mathcal{I})$.

We will distinguish between three different cases.

Case 1. $\deg_U(v) = 0$ and $\deg_F(v) \geq 3$, i.e. v is an isolated vertex of $G[U]$ with multiple edges to F. On $\mathcal{I}_1$, it holds that $\mu(\mathcal{I}_1) \leq \mu(\mathcal{I}) - 1$, since $g_1 \geq g + 1$. On the other hand, on $\mathcal{I}_2$, it holds that $\mu(\mathcal{I}_2) \leq \mu(\mathcal{I}) - 2$, since $cc(F') \leq cc(F) - 2$, otherwise $G[F']$ contains a cycle.

Case 2. $\deg_U(v) = 1$ and $\deg_F(v) \geq 2$. On $\mathcal{I}_1$, it holds that $\mu(\mathcal{I}_1) \leq \mu(\mathcal{I}) - 1$, since $g_1 \geq g+1$. On the other hand, on $\mathcal{I}_2$, it holds that $\mu(\mathcal{I}_2) \leq \mu(\mathcal{I}) - 1$, since $cc(F') \leq cc(F) - 1$, otherwise $G[F']$ contains a cycle. As a matter of fact, the number of interesting paths might also increase.

Case 3. Lastly, either (i) $\deg_U(v) = 2$ and $\deg_F(v) \geq 1$, or (ii) $\deg_U(v) \geq 3$. Since v is an interesting vertex of maximum height, for all of its descendants w in its corresponding tree in $G[U]$, it holds that $\deg_{F \cup U}(w) = 2$. On $\mathcal{I}_1$, for any child u of v, it holds that $\deg_{V \setminus S'}(u) = 1$. In that case, by exhaustively applying Rule 3 and producing an instance $\mathcal{I}_1^* = (G', S', F^*, k)$, it follows that v has an additional edge to F^* for each such child. In total, v has at least 2 edges towards F^* in both (i) and (ii), either due to the children or preexisting edges. Consequently, $g_1 \geq g + 1$ and $\mu(\mathcal{I}_1) \leq \mu(\mathcal{I}) - 1$. Note that the number of interesting paths might also increase in the new instance.

For $\mathcal{I}_2$, we consider (i) and (ii) separately.

- In (i), $cc(F') \leq cc(F)$ since v has at least 1 neighbor in F, while p is increased by at least 1. Indeed, since v has at least one child u in U, all of the descendants of which have degree 2 in $G[V \setminus S]$, this means that we have increased the interesting paths by at least 1.
- In (ii), since v does not necessarily have a neighbor in F, it holds that $cc(F') \leq cc(F) + 1$. However, v has at least 2 children in U, all of the descendants of which have degree 2 in $G[V \setminus S]$, therefore the number of interesting paths p increase by at least 2.

Consequently, $\mu(\mathcal{I}_2) \leq \mu(\mathcal{I}) - 1$. ◀