Detecting Feedback Vertex Sets of Size k in $O^\star(2.7^k)$ Time

Jason Li[*]   Jesper Nederlof[†]

November 5, 2019

Abstract

In the Feedback Vertex Set problem, one is given an undirected graph G and an integer k, and one needs to determine whether there exists a set of k vertices that intersects all cycles of G (a so-called feedback vertex set). Feedback Vertex Set is one of the most central problems in parameterized complexity: It served as an excellent test bed for many important algorithmic techniques in the field such as Iterative Compression [Guo et al. (JCSS'06)], Randomized Branching [Becker et al. (J. Artif. Intell. Res'00)] and Cut&Count [Cygan et al. (FOCS'11)]. In particular, there has been a long race for the smallest dependence $f(k)$ in run times of the type $O^\star(f(k))$, where the $O^\star$ notation omits factors polynomial in n. This race seemed to be run in 2011, when a randomized $O^\star(3^k)$ time algorithm based on Cut&Count was introduced.

In this work, we show the contrary and give a $O^\star(2.7^k)$ time randomized algorithm. Our algorithm combines all mentioned techniques with substantial new ideas: First, we show that, given a feedback vertex set of size k of bounded average degree, a tree decomposition of width $(1 - \Omega(1))k$ can be found in polynomial time. Second, we give a randomized branching strategy inspired by the one from [Becker et al. (J. Artif. Intell. Res'00)] to reduce to the aforementioned bounded average degree setting. Third, we obtain significant run time improvements by employing fast matrix multiplication.

1   Introduction

Feedback Vertex Set (FVS) is one of the most fundamental NP-complete problems; for example, it was among Karp's original 21 problems [Kar72]. In FVS we are given an undirected graph G and integer k, and are asked whether there exists a set F such that $G[V \setminus F]$ is a forest (i.e. F intersects all cycles of G). In the realm of parameterized complexity, where we aim for algorithms with running times of the type $O^\star(f(k))$[1] with $f(k)$ as small as possible (albeit exponential), FVS is clearly one of the most central problems: To quote [Cao18], to date the number of parameterized algorithms for FVS published in the literature exceeds the number of parameterized algorithms for any other single problem.

There are several reasons why FVS is the one of the most central problem in parameterized complexity: First and foremost, the main point of parameterized complexity, being that in many instance the parameter k is small, is very applicable for FVS: In the instances arising from e.g. resolving deadlocks in systems of processors [BGNR98], or from Bayesian inference or constraint satisfaction, one is only interested in whether small FVS's exist [BBG00, Dec90, WLS85]. Second, FVS is a very natural graph modification problems (remove/add few vertices/edges to make the graph satisfy a certain property) that serves as excellent starting point for many other graph modification problems such a planarization or treewidth-deletion (see e.g. [GLL+18] for a recent overview). Third, FVS and many of its variants (see e.g. [KK18]) admit elegant duality theorems such as the Erdös-Pósa property; understanding their use in designing algorithms can be instrumental to solve many problems different from FVS faster. The popularity of FVS also led to work on a broad spectrum of its variations such as Subset, Group, Connected, Simultaneous, or Independent FVS (see for example [AGSS16] and the references therein).

In this paper we study the most basic setting concerning the parameterized complexity of FVS, and aim to design an algorithm with runtime $O^\star(f(k))$ with $f(k)$ as small as possible.

One motivation for this study is that we want to get a better insight into the fine-grained complexity of computational problems: How hard is FVS really to solve in the worst-case setting? Can the current algorithms still be improved significantly or are they close to some computational barrier implied by some

---

[*]Carnegie Mellon University, `jmli@andrew.cmu.edu`.

[†]Eindhoven University of Technology, `j.nederlof@tue.nl`.

---

[1]The $O^\star()$ notation omits factors polynomial in n.

hypothesis or conjecture such as, for example, the Strong Exponential Time Hypothesis?

A second motivation is that, lowering the exponential factor $f(k)$ of the running time is a logical first step towards more practical algorithms. For example, the vertex cover problem[2] can be solved in $O(1.28^k + kn)$ time [CKX10], and a similar running time for FVS would be entirely consistent with our current knowledge. Algorithms with such run times likely outperform other algorithms for a wide variety of instances from practice. Note there already has been considerable interest in practical algorithms for FVS as it was the subject of the first Parameterized Algorithms and Computational Experiments Challenge (PACE, see e.g. [DHJ+16]).

For a third motivation of such a study, experience shows an improvement of the running time algorithms for well-studied benchmark problems as FVS naturally goes hand in hand with important new algorithmic tools: The *'race'* for the fastest algorithm for FVS and its variants gave rise to important techniques in parameterized complexity such as Iterative Compression [DFL+07, GGH+06, RSV04], Randomized Branching [BBG00] and Cut&Count [CNP+11].

The race for the fastest FVS algorithm. The aforementioned 'race' (see Figure 1) started in the early days of parameterized complexity (see e.g [AEFM89]) with an $O^\star((2k + 1)^k)$ time deterministic algorithm by Downey and Fellows [DF92]. We briefly discuss four relevant results from this race. A substantial improvement of the algorithm from [DF92] to an $O^\star(4^k)$ time randomized algorithm was obtained by Becker et al. [BBG00]. Their simple but powerful idea is to argue that, if some simple reduction rules do not apply, a random 'probabilistic branching' procedure works well. A few years later, in [DFL+07, GGH+06] it was shown how to obtain $O^\star(10.6^k)$ time in the deterministic regime using *Iterative Compression*. This technique allows the algorithm to assume a feedback vertex set of size $k + 1$ is given, which turns out to be useful for detecting feedback vertex sets of size k. The race however stagnated with the paper that introduced the Cut&Count technique [CNP+11] and gave a $O^\star(3^k)$ time randomized algorithm. In particular, the Cut&Count technique gave a $O^\star(3^{\mathrm{tw}})$ time algorithm for FVS if a *tree decomposition* (see Section 2 for definitions) of width tw is given, and this assumption can be made due to the iterative compression technique. After this result, no progress on randomized algorithms for FVS was made as it seemed that improvements over the $O^\star(3^{\mathrm{tw}})$ running time were not within reach: In [CNP+11] it was also proven that

---

[2]Given a graph G and integer k, find k vertices of G that intersect every edge of G.

any $O^\star((3-\epsilon)^{\mathrm{tw}})$ time algorithm, for some $\epsilon > 0$, would violate the SETH. It was therefore natural to expect the base 3 is also optimal for the parameterization by the solution size k. Moreover, the very similar $O^\star(2^k)$ time algorithm from [CNP+11] for the Connected Vertex Cover problem was shown to be optimal under the Set Cover Conjecture [CDL+16].

Our contributions. We show that, somewhat surprisingly, the $O^\star(3^k)$ time Cut&Count algorithm for FVS can be improved:

THEOREM 1.1. *There is a randomized algorithm that solves FVS in time $O^\star(2.69998^k)$. If $\omega = 2$, then the algorithm takes time $O^\star(2.6252^k)$.*

Here $2 \le \omega \le 2.373$ is the smallest number such that two n by n matrices can be multiplied in $O(n^\omega)$ time [Gal14]. Theorem 1.1 solves a natural open problem stated explicitly in previous literature [CFJ+14].

Using the method from [FGLS16] that transforms $O^\star(c^k)$ time algorithms for FVS into $O^\star((2 - 1/c)^n)$ we directly obtain the following improvement over the previously fastest $O^\star(1.67^n)$ time algorithm:

COROLLARY 1.1. *There is a randomized algorithm that solves FVS on an n-vertex graph in time $O^\star(1.6297^n)$.*

The above algorithms require space exponential in k, but we also provide an algorithm using polynomial space at the cost of the running time:

THEOREM 1.2. *There is a randomized algorithm that solves FVS in time $O^\star(2.8446^k)$ and polynomial space.*

Our Techniques. We build upon the $O^\star(3^{\mathrm{tw}})$ time algorithm from [CNP+11]. The starting standard observation is that a feedback vertex set of size k (which we can assume to be known to us by the iterative compression technique) gives a tree decomposition of treewidth $k + 1$ with very special properties. We show how to leverage these properties using the additional assumption that the average degree of all vertices in the feedback vertex set is constant:

LEMMA 1.1. *Let G be a graph and F be a feedback vertex set of G of size at most k, and define $\bar{d} := \deg(F)/k = \sum_{v \in F} \deg(v)/k$. There is an algorithm that, given G and F, computes a tree decomposition of G of width at most $(1 - 2^{-\bar{d}} + o(1))k$, and runs in polynomial time in expectation.*

To the best of our knowledge, Lemma 1.1 is new even for the special case where F is a vertex cover of G. We expect this result to be useful for other problems parameterized by the feedback vertex set or

Reference	Running Time	Deterministic?	Year
Downey and Fellows [DF92]	$O^\star((2k+1)^k)$	YES	1992
Bodlaender [Bod94]	$O^\star(17(k^4)!)$	YES	1994
Becker et al. [BBG00]	$O^\star(4^k)$	NO	2000
Raman et al. [RSS02]	$O^\star(12^k + (4\log k)^k)$	YES	2002
Kanj et al. [KPS04]	$O^\star((2\log k + 2\log\log k + 18)^k)$	YES	2004
Raman et al. [RSS06]	$O^\star((12\log k/\log\log k + 6)^k)$	YES	2006
Guo et al. [GGH$^+$06]	$O^\star(37.7^k)$	YES	2006
Dehne et al. [DFL$^+$07]	$O^\star(10.6^k)$	YES	2007
Chen et al. [CFL$^+$08]	$O^\star(5^k)$	YES	2008
Cao et al. [CCL15]	$O^\star(3.83^k)$	YES	2010
Cygan et al. [CNP$^+$11]	$O^\star(3^k)$	NO	2011
Kociumaka and Pilipczuk [KP14]	$O^\star(3.62^k)$	YES	2014
this paper	$O^\star(2.7^k)$, or $O^\star(2.6252^k)$ if $\omega = 2$	NO	2020

Figure 1: The 'race' for the fastest parameterized algorithm for Feedback Vertex Set.

vertex cover size (such parameterizations are studied in for example [JJ17]). Lemma 1.1 is proven via an application of the probabilistic method analyzed via proper colorings in a dependency graph of low average degree. It is presented in more detail in Section 3.

Lemma 1.1, combined with the $O^\star(3^{\mathrm{tw}})$ time algorithm from [CNP$^+$11], implies that we only need to ensure the feedback vertex set has constant average degree in order to get a $O^\star((3-\epsilon)^k)$ time algorithm for some $\epsilon > 0$. To ensure this property, we extend the randomized $O^\star(4^k)$ time algorithm of Becker et al. [BBG00]. The algorithm from [BBG00] first applies a set of reduction rules exhaustively, and then selects a vertex with probability proportional to its degree.[3] They show that this chosen vertex appears in an optimal feedback vertex set with probability at least $1/4$. To modify this algorithm, we observe that after applying the reduction rules in [BBG00], every vertex has degree at least 3, so one idea is to select vertices with probability proportional to $\deg(v) - 3$ instead.[4] It turns out that if $n \gg k$, then this biases us more towards selecting a vertex in an optimal feedback vertex set F. Indeed, we will show that if $n \geq 4k$, then we succeed to select a vertex of F with probability at least $1/2$. This is much better than even success probability $1/3$, which is what we need to beat to improve the $O^\star(3^k)$ running time.

Closer analysis of this process shows that even if $n < 4k$, as long as the graph itself has large enough average degree, then we also get success probability $\gg 1/3$. It follows that if the $\deg(v) - 3$ sampling does not give success probability $\gg 1/3$, then the graph has $n \leq 4k$ and constant average degree. Therefore, the graph has only $O(k)$ edges, and even if all of them are incident to the feedback vertex set of size k, the feedback vertex set still has constant average degree. Therefore, we can apply Lemma 1.1, which gives us a modest improvement of the $O^\star(3^k)$ running time to $O^\star(3^{(1-2^{-56})k})$ time.

To obtain improvements to a $O^\star(2.8446^k)$ time and polynomial space algorithm, we introduce the new case $n \ll 3k$, where we simply add a random vertex to the FVS F, which clearly succeeds with probability $\gg 1/3$. We then refine our analysis and apply the Cut&Count method from the $O^\star(3^{\mathrm{tw}})$ algorithm in a way similar to [CNP$^+$11, Theorem B.1].

To obtain Theorem 1.1 and further improve the above running times, we extend the proof behind Lemma 1.1 to decompose the graph using a "three-way separation" (see Definition 3) and leverage such a decomposition by combining the Cut&Count method with fast matrix multiplication. This idea to improve the running time is loosely inspired by previous approaches for MAX-SAT [CS15] and connectivity problems parameterized by branch-width [PBvR16].

Paper Organization. This paper is organized as follows: We first define notation and list preliminaries in Section 2. We present the proof of Lemma 1.1 in Section 3. In Section 4, we introduce a probabilistic reduction rule and its analysis. Subsequently we focus on improving the $O^\star(3^k)$ time algorithm for FVS in Section 5. The algorithm presented there only obtains a modest improvement, but illustrates our main ideas and uses previous results as a black box.

---

[3]The sampling is usually described as choosing a random edge and then a random vertex of this chosen edge, which has the same sampling distribution.

[4]Let us assume that the graph is not 3-regular, since if it were, then the feedback vertex set has constant average degree and we could proceed as before.

In the second half of the paper we show how to further improve our algorithms and prove our main theorems: Section 6 proves Theorem 1.2, and in Section 7 we prove Theorem 1.1. Both these sections rely on rather technical extensions of the Cut&Count method that we postpone to Section 8 to improve readability.

2 Preliminaries

Let G be an undirected graph. For a vertex v in G, $\deg(v)$ is the degree of v in G, and for a set S of vertices, we define $\deg(S) := \sum_{v \in S} \deg(v)$. If $S, T \subseteq V(G)$ we denote $E[S, T]$ for all edges intersecting both S, T, and denote $E[S] = E[T, T]$. For a set $\binom{A}{\cdot,\cdot,\cdot}$ denotes all partitions of A into three subsets. As we only briefly use tree-decompositions we refer to [CFK+15, Chapter 7] for its definitions and standard terminology.

Randomized Algorithms. All algorithms in this paper will be randomized algorithms for search problems with one-sided error-probability. The *(success) probability* of such an algorithm is the probability it will output the asked solution, if it exists. In this paper we define *with high probability* to be probability at least $1 - 2^{-c|x|}$ for some large c where x is the input, instead of the usual $1 - 1/|x|^c$. This is because FPT algorithms take more than simply $\text{poly}(|x|) = O^\star(1)$ time, so a probability bound of $1 - 2^{-c|x|}$ is more convenient when using an union bound to bound the probability any execution of the algorithm will fail.

Note that if the algorithm has constant success probability, we can always boost it to high probability using $O^\star(1)$ independent trials. For convenience, we record the folklore observation that this even works for algorithms with expected running time:

LEMMA 2.1. (FOLKLORE) *If a problem can be solved with success probability $1/S$ and in expected time T, and its solutions can be verified for correctness in polynomial time, then it can be also solved in $O^\star(S \cdot T)$ time with high probability.*

Proof. Consider $cS|x|$ independent runs of the algorithm for some large constant c, and if a run outputs a solution, we then verify that solution and output YES if this is successful. Given that a solution exists, it is not found and verified in any of $cS|x|$ rounds with probability at most $(1 - 1/S)^{c \cdot S|x|} \leq \exp(-cn)$. The expected running time of the $cS|x|$ independent runs is $c|x|ST$, and by Markov's inequality these jointly run in at most $2c|x|ST$ time with probability at least $3/4$. Therefore we can terminate our algorithm after $2c|x|ST$ time and by a union bound this gives and algorithm that solves the problem with constant success probability. To boost this success probability to high probability, simply use $|x|$ independent runs of the algorithm that reaches con-stant success probability. $\Box$

Using this lemma, we assume that all randomized algorithms with constant positive success probability actually solve their respective problems with high probability.

Separations. The following notion will be instrumental in our algorithms.

DEFINITION 1. (SEPARATION) *Given a graph $G = (V, E)$, a partition $(A, B, S) \in \binom{V(G)}{\cdot,\cdot,\cdot}$ of V is a separation if there are no edges between A and B.*

Reduction Rules. In the context of parameterized complexity, a *reduction rule* (for FVS) is a polynomial-time transformation of an input instance (G, k) into a different instance (G', k') such that G has a FVS of size k iff G' has a FVS of size k'. We state below the standard reduction rules for FVS, as described in [CFK+15], Section 3.3. For simplicity, we group all four of their reduction rules FVS.1 to FVS.4 into a single one.

REDUCTION 1. ([CFK+15], FOLKLORE) *Apply the following rules exhaustively, until the remaining graph has no loops, only edges of multiplicity at most 2, and minimum vertex degree at least 3:*

1. *If there is a loop at a vertex v, delete v from the graph and decrease k by 1; add v to the output FVS.*

2. *If there is an edge of multiplicity larger than 2, reduce its multiplicity to 2.*

3. *If there is a vertex v of degree at most 1, delete v.*

4. *If there is a vertex v of degree 2, delete v and connect its two neighbors by a new edge.*

3 Treewidth and Separators

In this section, we show how to convert an FVS with small average degree into a good tree decomposition. In particular, suppose graph G has a FVS F of size k with $\deg(F) \leq \bar{d}k$, where $\bar{d} = O(1)$. We show how to construct a tree decomposition of width $(1 - \Omega(1))k$. Note that a tree decomposition of width $k+1$ is trivial: since $G - F$ is a forest, we can take a tree decomposition of $G - F$ of width 1 and add F to each bag. To achieve treewidth $(1 - \Omega(1))k$, we will crucially use the fact that $\bar{d} = O(1)$.

We make the assumption that the algorithm already knows the small average degree FVS F. This reasoning may seem circular at first glance: after all, the whole task is finding the FVS in the first place. Nevertheless, we later show how to remove this assumption using the standard technique of *Iterative Compression*.

We now present a high level outline of our approach. Our goal is to compute a small set S of vertices—one of size at most $(1 - \Omega(1))k$—whose deletion leaves a graph of small enough treewidth. Then, taking the tree decomposition of $G - S$ and adding S to each bag gives the desired tree decomposition. Of course, settling for $|S| = (1 + o(1))k$ and treewidth 1 is easy: simply set $S = F$ so that the remaining graph is a forest, which has treewidth 1. Therefore, it is important that $|S| = (1 - \Omega(1))k$.

We now proceed with our method of constructing S. First, temporarily remove the FVS F from the graph, leaving a forest T. We first select a set S_ϵ of β vertices to remove from the forest, for some $\beta = o(k)$, to break it into connected components such that the edges between F and T are evenly split among the components. More precisely, we want every connected component of $T - S_\epsilon$ to share at most a $1/\beta$ fraction of all edges between F and T; we show in Lemma 3.1 below that this is always possible. The β vertices in S_ϵ will eventually go into every bag in the decomposition; this only increases the treewidth by $o(k)$, which is negligible. Hence, we can safely ignore the set S_ϵ.

Next, we perform a *random coloring procedure* as follows: randomly color every connected component of $T - S_\epsilon$ red or blue, uniformly and independently. Let A be the union of all components colored red, and B be the union of all components colored blue. For simplicity of exposition, we will assume here (*with* loss of generality) that F is an independent set: that is, there are no edges between vertices in the FVS. Then, if a vertex $v \in F$ has all its neighbors in $T - S_\epsilon$ belonging to red components, then v only has neighbors in A, so let us add v to A. Similarly, if all neighbors belong to blue components, then v only has neighbors in B, so let us add v to B. Observe that the new graphs $G[A]$ and $G[B]$ still have no edges between them, so every vertex addition so far has been "safe".

What is the probability that a vertex in F joins A or B? Recall that $d(F) = \bar{d}k$, and since F is an independent set, $|E[F, T - S_\epsilon]| \le |E[F, T]| = d(F) = \bar{d}k$. If a vertex in F has exactly $\bar{d}$ edges to $T - S_\epsilon$, then it has probability at least $2^{-\bar{d}}$ of joining A, with equality when all of these edges go to different connected components in $T - S_\epsilon$. Of course, we only have that vertices in F have at most $\bar{d}$ neighbors on average, but a convexity argument shows that in expectation, at least a $(2^{-\bar{d}} - o(1))k$ fraction of vertices in F join A. That is, $\mathbb{E}[|A \cap F|] \ge (2^{-\bar{d}} - o(1))k$. We can make a symmetric argument for vertices joining B. Of course, we need both events—enough vertices joining each of A and B—to hold simultaneously, which we handle with a concentration argument. From here, it

is straightforward to finish the treewidth construction. We now present the formal proofs.

We begin with the following standard fact on balanced separators of forests:

LEMMA 3.1. *Given a forest T on n vertices with vertex weights $w(v)$, for any $\beta > 0$, we can delete a set S of β vertices so that every connected component of $T - S$ has total weight at most $w(V)/\beta$.*

Proof. Root every component of the forest T at an arbitrary vertex. Iteratively select a vertex v of maximal depth whose subtree has total weight more than $w(V)/\beta$, and then remove v and its subtree. The subtrees rooted at the children of v have total weight at most $w(V)/\beta$, since otherwise, v would not satisfy the maximal depth condition. Moreover, by removing the subtree rooted at v, we remove at least $w(V)/\beta$ total weight, and this can only happen β times. □

LEMMA 3.2. (SMALL SEPARATOR) *Given an instance (G, k) and a FVS F of G of size at most k, define $\bar{d} := \deg(F)/k$, and suppose that $\bar{d} = O(1)$. There is a randomized algorithm running in expected polynomial time that computes a separation (A, B, S) of G such that:*

1. $|A \cap F|, |B \cap F| \ge (2^{-\bar{d}} - o(1))k$

2. $|S| \le (1 + o(1))k - |A \cap F| - |B \cap F|$

Proof. Fix a parameter $\epsilon := k^{-0.01}$ throughout the proof. Apply Lemma 3.1 to the forest $G - F$ with parameter ϵk, with vertex v weighted by $|E[v, F]|$, and let S_ϵ be the output. Observe that

$$|S_\epsilon| \le \epsilon k = o(k),$$

and every connected component C of $G - F - S_\epsilon$ satisfies

$$|E[C, F]| \le \frac{|E[\overline{F}, F]|}{\epsilon k} \le \frac{\deg(F)}{\epsilon k} = \frac{\bar{d}k}{\epsilon k} = \bar{d}/\epsilon.$$

Now form a bipartite graph H on vertex bipartition $F \uplus R$, where F is the FVS, and there are two types of vertices in R, the *component* vertices and the *subdivision* vertices. For every connected component C in $G - F - S_\epsilon$, there is a component vertex v_C in R that represents that component, and it is connected to all vertices in F adjacent to at least one vertex in C. For every edge $e = (u, v)$ in $E[F]$, there is a vertex v_e in R with u and v as its neighbors. Observe that (1) $|R| \le |E[\overline{F}, F]| + 2|E[F]| = \deg(F)$, (2) every vertex in R has degree at most $\bar{d}/\epsilon$, and (3) the degree of a vertex $v \in F$ in H is at most $\deg(v)$.

The algorithm that finds a separator works as follows. For each vertex in R, color it red or blue uniformly and independently at random. Every component C in $G - F - S_\epsilon$ whose vertex v_C is colored red is added to A in the separation (A, B, S), and every component whose vertex v_C is colored blue is added to B. Every vertex in F whose neighbors are all colored red joins A, and every vertex in F whose neighbors are all colored blue joins B. The remaining vertices in F, along with the vertices in S_ϵ, comprise S.

SUBCLAIM 1. (A, B, S) *is a separation.*

Proof. Suppose for contradiction that there is an edge connecting A and B. The edge cannot connect two distinct components of $G - F - S_\epsilon$, so it must have an endpoint in F. The edge cannot connect a vertex in F to a vertex in $G - F - S_\epsilon$, since a vertex in F only joins A or B if all of its neighbors in R are colored the corresponding color. Therefore, the edge e must connect two vertices in F. But then, v_e connects to both endpoints and is colored either red or blue, so it is impossible for one endpoint of e to have all neighbors colored red, and the other endpoint to have all neighbors colored blue, contradiction. □

We now show that with good probability both Conditions (1) and (2) hold. The algorithm can then repeat the process until both conditions hold.

SUBCLAIM 2. *With probability at least $1 - 1/\mathrm{poly}(k)$, Condition (1) holds for (A, B, S).*

Proof. There are at most $\epsilon|F|$ vertices in F with degree at least $\overline{d}/\epsilon$. Since they cannot affect condition (1) by an additive $\epsilon|F| \leq \epsilon k = o(k)$ factor, we can simply ignore them; let F' be the vertices with degree at most $\overline{d}/\epsilon$. Consider the *intersection graph* I on the vertices of F', formed by connecting two vertices in F' iff they share a common neighbor (in R). Since every vertex in F' and C has degree at most $\overline{d}/\epsilon$, the maximum degree of I is $(\overline{d}/\epsilon)^2$. Using the standard greedy algorithm, we color F' with $(\overline{d}/\epsilon)^2 + 1$ colors so that every color class forms an independent set in I. In particular, within each color class, the outcome of each vertex—namely, whether it joins A or B or S—is independent across vertices.

Let F'_i be the vertices colored i. If $|F'_i| < k^{0.9}$, then ignore it; since $\overline{d} \leq O(1)$ and $\epsilon = k^{-0.01}$, the sum of all such $|F'_i|$ is at most $((\overline{d}/\epsilon)^2 + 1)k^{0.9} = o(k)$, so they only affect condition (1) by an additive $o(k)$ factor. Henceforth, assume that $|F'_i| \geq k^{0.9}$. Each vertex $v \in F'_i$ has at most $\deg(v)$ neighbors in H, so it has independent probability at least $2^{-\deg(v)}$ of joining A. Let $X_i := |F'_i \cap A|$ be the number of vertices in F'_i

that join A; by Hoeffding's inequality[5],

$$\Pr[X_i \leq E[X] - k^{0.8}] \leq 2\exp(-2 \cdot (k^{0.8})^2/|F'_i|)$$
$$\leq 2\exp(-2 \cdot k^{0.6}) \leq 1/\mathrm{poly}(k)$$

for large enough k.

By a union bound over all $\leq k^{0.1}$ color classes F'_i with $|F'_i| \geq k^{0.9}$, the probability that $|F'_i \cap A| \geq E[|F'_i \cap A|] - k^{0.8}$ for each F'_i is $1 - 1/\mathrm{poly}(k)$. In this case,

$$|F \cap A| \geq \sum_{i:|F'_i| \geq k^{0.9}} \left(E[|F'_i \cap A|] - k^{0.8}\right)$$
$$\geq \sum_{i:|F'_i| \geq k^{0.9}} \sum_{v \in F'_i} 2^{-\deg(v)} - k^{0.1} \cdot k^{0.8}$$
$$= \sum_{v \in F'} 2^{-\deg(v)} - o(k)$$
$$\geq |F'| \cdot 2^{-\deg(F')/|F'|} - o(k),$$

where the last inequality follows from convexity of the function 2^{-x}. Recall that $|F'| \geq (1-o(1))k$, and observe that $\deg(F')/|F'| \leq \deg(F)/|F| = \overline{d}$ since the vertices in $F \backslash F'$ are precisely those with degree exceeding some threshold. It

$$|F \cap A| \geq (1 - o(1))k \cdot 2^{-\overline{d}},$$

proving condition (1) for $|A \cap F|$. Of course, the argument for $|B \cap F|$ is symmetric. □

SUBCLAIM 3. *With probability at least $1 - 1/\mathrm{poly}(k)$, Condition (2) holds for (A, B, S).*

Proof. At most $\epsilon k = o(k)$ vertices in S can come from S_ϵ, and the other vertices in S must be precisely $F \backslash ((A \cap F) \cup (B \cap F))$, which has size $k - |A \cap F| - |B \cap F|$. □

Hence, with at least constant probability, both Conditions (1) and (2) hold. Furthermore, whether or not they hold can be checked in polynomial time, so the algorithm can repeatedly run the algorithm until the separation satisfies both conditions. □

LEMMA 1.1. *Let G be a graph and F be a feedback vertex set of G of size at most k, and define $\overline{d} := \deg(F)/k = \sum_{v \in F} \deg(v)/k$. There is an algorithm that, given G and F, computes a tree decomposition of G of width at most $(1 - 2^{-\overline{d}} + o(1))k$, and runs in polynomial time in expectation.*

---

[5]If $a_1, \ldots, a_n$ are independent and Bernoulli and $X = a_1 + a_2 + \ldots + a_n$, then $\Pr[|X - E[x]| \geq t] \leq 2\exp(-2t^2/n)$.

Proof. Compute a separation (A, B, S) following Lemma 3.2. Since $(A \cap F) \cup S$ is a FVS of $A \cup S$ of size $(1 - 2^{-\overline{d}} + o(1))k$, we can compute a tree decomposition of $G[(A \cap F) \cup S]$ of width $(1 - 2^{-\overline{d}} + o(1))k$ as follows: start with a tree decomposition of width 1 of the forest $G[(A \cap F) \cup S] - (F \cup S)$, and then add all vertices in $(A \cap F) \cup S$ to each bag. Similarly, compute a tree decomposition of $G[(B \cap F) \cup S]$ in the same way. Finally, merge the two tree decompositions by adding an edge between an arbitrary node from each decomposition; since there is no edge connecting A to B, the result is a valid tree decomposition. □

4 Probabilistic Reduction

Whenever a reduction fails with a certain probability, we call it a *probabilistic reduction*. Our probabilistic reduction is inspired by the randomized $O^\star(4^k)$ FVS algorithm of [BBG00]. Whenever we introduce a probabilistic reduction, we include (P) in the header, such as in the reduction below.

REDUCTION 2. (P) *Assume that Reduction 1 does not apply and G has a vertex of degree at least 4. Sample a vertex $v \in V$ proportional to $w(v) := (\deg(v) - 3)$. That is, select each vertex v with probability $w(v)/w(V)$. Delete v, decrease k by 1.*

We say a probabilistic reduction *succeeds* if it selects a vertex in an optimal feedback vertex set.

OBSERVATION 1. *Let G be a graph F a FVS of G. Denoting $\overline{F} := V \setminus F$ we have that*

$$(4.1) \qquad \deg(\overline{F}) \leq \deg(F) + 2(|\overline{F}| - 1).$$

Proof. Since $G - F$ is a forest, there can be at most $|\overline{F}| - 1$ edges in $G - F$, each of which contributes 2 to the summation $\deg(\overline{F}) = \sum_{v \in \overline{F}} \deg(v)$. The only other edges contributing to $\deg(\overline{F})$ are in $E[F, \overline{F}]$, which contribute 1 to both $\deg(\overline{F})$ and $\deg(F)$. Therefore, we have the upper bound $\deg(\overline{F}) \leq$

$$2(|\overline{F}| - 1) + |E[F, \overline{F}]| \leq 2(|\overline{F}| - 1) + \deg(F).$$

□

LEMMA 4.1. *If $n \geq 4k$ and the instance is feasible, then Reduction 2 succeeds with probability at least $1/2$.*

Proof. Let $F \subseteq V$ be a FVS of size k.[6] We show that the probability of selecting a vertex in F is at least $1/2$. Define $\overline{F} := V \setminus F$, so that our goal is equivalent to showing that $w(F) \geq w(\overline{F})$.

[6]From any FVS of size less than k, we can arbitrarily add vertices until it has size k.

The value of $w(F)$ can be rewritten as

$$(4.2) \qquad w(F) = \sum_{v \in F} (\deg(v) - 3) = \deg(F) - 3|F|.$$

By Observation 1,

$$
\begin{aligned}
w(\overline{F}) &= \sum_{v \in \overline{F}} (\deg(v) - 3) \\
(4.3) \qquad &= \deg(\overline{F}) - 3|\overline{F}| \\
&\leq \deg(F) + 2(|\overline{F}| - 1) - 3|\overline{F}| \\
&\leq \deg(F) - |\overline{F}|,
\end{aligned}
$$

where the penultimate inequality is by 4.1. Therefore, $w(F) \geq w(\overline{F})$ is implied by

$$
\begin{aligned}
& \deg(F) - 3|F| \geq \deg(F) - |\overline{F}| \\
(4.4) \qquad &\Longleftrightarrow |\overline{F}| \geq 3|F| \\
&\Longleftrightarrow n \geq 4k,
\end{aligned}
$$

and the claim follows. □

Therefore, as long as $n \geq 4k$, we can repeatedly apply Reductions 1 and 2 until either $k = 0$, which means we have succeeded with probability at least $1/2^k$, or we have an instance (G, k) with $n \leq 4k$.

Later on, we will need the following bound based on the number of edges m. Informally, it says that as long as the average degree is large enough, Reduction 2 will still succeed with probability close to $1/2$ (even if $n < 4k$).

LEMMA 4.2. *Assume that $2m > 3n$. If the instance is feasible, then Reduction 2 succeeds with probability at least $\min\{\frac{1}{2}, \frac{m-n-2k}{2m-3n}\}$.*

Proof. There are at most $|\overline{F}| - 1$ edges not contributing to $\deg(F)$, so

$$(4.5) \qquad m \leq (|\overline{F}| - 1) + \deg(F) \leq (n - k) + \deg(F)$$

If $w(F)/w(\overline{F}) \geq 1$, then the success probability is at least $1/2$, so assume otherwise that $w(F) < w(\overline{F})$. Following the proof of Lemma 4.1, the contrapositive of (4.4) gives

$$(4.6) \qquad w(F) < w(\overline{F}) \implies |\overline{F}| < 3|F|,$$

so we have

$$
\begin{aligned}
\frac{w(F)}{w(\overline{F})} &\overset{(4.2)}{=} \frac{\deg(F) - 3|F|}{w(\overline{F})} \overset{(4.3)}{\geq} \frac{\deg(F) - 3|F|}{\deg(F) - |\overline{F}|} \\
&\geq \frac{(m - n + k) - 3|F|}{(m - n + k) - |\overline{F}|} = \frac{m - n - 2k}{m - 2n + 2k},
\end{aligned}
$$

where the penultimate inequality follows from (4.5, 4.6). Finally, as the Lemma statement is vacuous when $2k > m - n$, the Lemma follows. □

5 $O^\star((3-\epsilon)^k)$ Time Algorithm

In this section we present our simplest algorithm that achieves a running time of $O^\star((3-\epsilon)^k)$, for some $\epsilon > 0$. The improvement ϵ is very small, but we found this to be the simplest exposition that achieves the bound for any $\epsilon > 0$. We build on the following result:

LEMMA 5.1. (CYGAN ET AL. [CNP+11]) *There is an algorithm* treewidthDP *that, given a tree decomposition of the input graph of width* tw*, and parameter k outputs a FVS of size at most k with high probability if it exists. Moreover, the algorithm runs in $O^\star(3^{\text{tw}})$ time.*

First, we combine the tree decomposition from the previous section with the standard technique of *Iterative Compression* to build an algorithm that runs in time $O^\star((3-\epsilon)^k)$ time, assuming that $m = O(k)$ (recall m denotes the number of edges of the input graph). Then, we argue that by applying Reduction 2 whenever $m \gg k$, we can essentially "reduce" to the case $m = O(k)$. Combining these two ideas gives us the $O^\star((3-\epsilon)^k)$ algorithm.

The algorithm is introduced below in pseudocode. The iterative compression framework proceeds as follows. We start with the empty graph, and add the vertices of G one by one, while always maintaining a FVS of size at most k in the current graph. Maintaining a FVS of the current graph allows us to use the small tree decomposition procedure of Section 3. Then, we add the next vertex in the ordering to each bag in the tree decomposition, and then solve for a new FVS in $O^\star(3^{\text{tw}})$ time using Lemma 5.1. Of course, if there is no FVS of size k in the new graph, then there is no such FVS in G either, so the algorithm can terminate early.

LEMMA 5.2. *On input instance (G,k) with $m = O(k)$,* IC1(G,k) *runs in time $O^\star(3^{(1-2^{-2m/k}+o(1))k})$. Moreover, if there exists a FVS F of size at most k, then* IC1 *will return a FVS of size at most k with high probability.*

Proof. Suppose that there exists a FVS F^* of size at most k. Let $(v_1, \ldots, v_n)$ be the ordering from Line 1, and define $V_i := \{v_1, \ldots, v_i\}$. Observe that $F^* \cap V_i$ is a FVS of $G[V_i]$, so the FVS problem on Line 6 is feasible. By Lemma 5.1, Line 6 correctly computes a FVS with high probability on any given iteration. Therefore, after using $O^\star(1)$ independent trials, with high probability a FVS is returned successfully.

We now bound the running time. On Line 4, the current set F is a FVS of $G[V_{i-1}]$. To bound the value of $\overline{d}$ used in Lemma 1.1, we use the (rather crude) bound

$$\deg(F) \leq \deg(V) = 2m \implies \overline{d} = \frac{\deg(F)}{k} \leq \frac{2m}{k},$$

and moreover, $\overline{d} = O(1)$ since $m = O(k)$ by assumption. Therefore, Lemma 1.1 guarantees a tree decomposition of width at most $(1-2^{-2m/k}+o(1))k$, and adding v_i to each bag on Line 5 increases the width by at most 1. By Lemma 5.1, Line 6 runs in time $O^\star(3^{(1-2^{-2m/k}+o(1))k})$ time, as desired. □

We now claim below that if $m \geq \Omega(k)$ for a sufficiently large k, then Reduction 2 succeeds with good probability (in particular, with probability greater than $1/3$).

LEMMA 5.3. *If G has a FVS of size k and $m \geq 28k$, then Reduction 2 succeeds with probability at least $4/11$.*

Proof. We consider two cases. If $n \geq 4k$, then the success probability is at least $1/2$ by Lemma 4.1. Otherwise, if $n \leq 4k$, then $m \geq 28k \geq 7n$, and Lemma 4.2 and the trivial bound $k \leq n$ give a success probability of at least

$$\frac{m-n-2k}{2m-3n} \geq \frac{m-3n}{2m-3n} \geq \frac{7n-3n}{14n-3n} = \frac{4}{11}.$$

Hence, regardless of whether or not $n \geq 4k$, Reduction 2 succeeds with probability at least $4/11$. □

Below is the full randomized algorithm in pseudocode, which combines Reductions 1 and 2 with the iterative compression routine IC1. After a trivial check and reduction rule, Line 3 flips a coin that needs to be flipped Heads in order to proceed to the iterative compression step.

The motivation for this is that we want each iteration of FVS1 to run quickly in expectation—in particular, in $O^\star(3^{o(k)})$ time—for simplicity of analysis. This way, if the algorithm has success probability c^{-k} for some constant c, then we can repeat it $O^\star(c^k)$ times, succeeding with high probability and taking $O^\star(c^{(1+o(1))k})$ time in expectation. Since IC1 takes $O^\star(3^{(1-2^{-56}+o(1))k})$ time by Lemma 5.2, we should call IC1 with probability at most $3^{-(1-2^{-56})k}$, which is exactly the probability of the coin flipping Heads.

LEMMA 5.4. FVS1(G,k) *runs in expected $O^\star(3^{o(k)})$ time and has $\Omega(3^{-(1-2^{-56})k})$ success probability.*

Proof. For the running time, the computation outside of Line 5 clearly takes poly(n) time. For each $k' \in (k_0, k]$, Line 5 is executed with probability $3^{-(1-2^{-56})k'}$ and takes $O^\star(3^{(1-2^{-56}+o(1))k'})$ time, so in expectation, the total computation cost of Line 5 is $O^\star(2^{o(k)})$ per value of k', and also $O^\star(2^{o(k)})$ overall.

It remains to lower bound the success probability. Define $c := 3^{1-2^{-56}}$. We will prove by induction on k that FVS1(G,k) succeeds with probability at least

---

Algorithm 1 IC1(G, k)

Input: Graph $G = (V, E)$ and parameter k, with $m = O(k)$.

Output: FVS F of size at most k, or Infeasible if none exists.

1: Order the vertices V arbitrarily as $(v_1, \ldots, v_n)$
2: $F \leftarrow \emptyset$
3: **for** $i = 1, \ldots, n$ **do**   ▷ **Invariant:** F is a FVS of $G[\{v_1, \ldots, v_{i-1}\}]$
4:     Compute a tree decomposition of $G[\{v_1, \ldots, v_{i-1}\}]$ by applying Lemma 1.1 on input F
5:     Add v_i to each bag in the tree decomposition
6:     $F \leftarrow$ a FVS of $G[\{v_1, \ldots, v_i\}]$ with parameter k, computed using treewidthDP from Lemma 5.1
7:     **if** F is Infeasible **then**
8:         **return** Infeasible
9: **return** F

Algorithm 2 FVS1(G, k)

Input: Graph $G = (V, E)$ and parameter $k \leq n$.

Output: A FVS of size k with probability $3^{-(1-2^{-56})}$ if one exists; Infeasible otherwise.

1: **if** $k = 0$ **then return** $\emptyset$ if G is acyclic, and **return** Infeasible otherwise
2: Exhaustively apply Reduction 1 to (G, k) to get vertex set F and instance (G', k') with m' edges
3: Flip a coin with Heads probability $3^{-(1-2^{-56})k'}$
4: **if** $m' \leq 28k'$ and coin flipped Heads **then**
5:     $F' \leftarrow$ IC1(G', k')
6: **else**
7:     Apply Reduction 2 to (G', k') to get vertex $v \in V$ and instance $(G'', k' - 1)$
8:     $F' \leftarrow$ FVS1$(G'', k' - 1) \cup \{v\}$ ▷ Infeasible $\cup\, S =$ Infeasible *for any set* S
9: **return** $F \cup F'$

$c^{-k}/2$. This statement is trivial for $k = 0$, since no probabilistic reductions are used and FVS1(G, k) succeeds with probability 1. For the inductive step, consider an instance FVS1$(G, k+1)$. First, suppose that $m \leq 28k$. In this case, if IC1 in Line 5 is executed, then it will run in time $O^\star(3^{(1-2^{-2m/k}+o(1))k})$ by Lemma 5.2, and correctly output a FVS F of size at most k, with high probability. This happens with probability at least

$$3^{-(1-2^{-56})k} \cdot \left(1 - \frac{1}{\text{poly}(n)}\right) \geq c^{-k} \cdot \frac{1}{2},$$

as desired. If IC1 is not executed, then FVS1 can still succeed, but this only increases our overall success probability, so we disregard it.

Otherwise, suppose that $m > 28k$. Then, by Lemma 5.3, applying Reduction 2 succeeds with probability at least $4/11$. By induction, the recursive call on Line 8 succeeds with probability at least $c^{-(k-1)}/2$, so the overall probability of success is at least

$$\frac{4}{11} \cdot \frac{c^{-(k-1)}}{2} \geq c^{-1} \cdot \frac{c^{-(k-1)}}{2} = \frac{c^{-k}}{2},$$

as desired.   □

The claimed $O^\star((3-\epsilon)^k)$ time algorithm follows from Lemma 5.4 by boosting the success probability of Algorithm FVS1 according to Lemma 2.1.

6   Algorithms with Improved Space or Time

In this section, we present the $O^\star(2.8446^k)$ time algorithm promised by Theorem 1.2. At a high level, our goal is to obtain a tighter bound on $\overline{d} = \deg(F)/k$, which we only bounded loosely by $2m/k$ in Section 5. Recall that the treewidth bound of $(1 - 2^{-\overline{d}} + o(1))k$ from Lemma 1.1 has exponentially dependence on $\overline{d}$, so every constant factor savings in $\overline{d}$ is crucial.

First, we introduce another simple reduction step, which works well when $n \ll 3k$.

REDUCTION 3. (P) *Sample a uniformly random vertex v. Delete v and decrease k by 1.*

For the entire section, we will fix a constant $\epsilon > 0$ and obtain a running time that depends on ϵ. At the very end, we will optimize for ϵ and achieve the running time $O^\star(2.8446^k)$. For formality, we define the following assumption (A1) and state the corresponding direct claim.

(A1) $n \leq (3 - \epsilon)k$

CLAIM 1. *If (A1) is true, then Reduction 3 succeeds with probability at least $1/(3 - \epsilon)$.*

Now suppose that (A1) is false. Observe that Reduction 2 succeeds with probability at least $1/(3-\epsilon)$

precisely when

$$\frac{w(F)}{w(\overline{F})} \overset{(4.2)}{=} \frac{\deg(F) - 3|F|}{\deg(\overline{F}) - 3|\overline{F}|} \geq \frac{1}{2 - \epsilon}.$$

By Observation 1, we have

$$\frac{\deg(F) - 3|F|}{\deg(\overline{F}) - 3|\overline{F}|} \overset{(4.1)}{\geq} \frac{\deg(F) - 3|F|}{(\deg(F) + 2|\overline{F}|) - 3|\overline{F}|}$$
$$= \frac{\deg(F) - 3k}{\deg(F) - (n - k)},$$

and since (A1) is false,

$$\frac{\deg(F) - 3k}{\deg(F) - (n - k)} \geq \frac{\deg(F) - 3k}{\deg(F) - ((3 - \epsilon)k - k)}$$
$$= \frac{\deg(F) - 3k}{\deg(F) - (2 - \epsilon)k}.$$

We are interested in whether or not

$$\frac{\deg(F) - 3k}{\deg(F) - (2 - \epsilon)k} \overset{?}{\geq} \frac{1}{2 - \epsilon}$$
$$\iff (2 - \epsilon)(\deg(F) - 3k) \overset{?}{\geq} \deg(F) - (2 - \epsilon)k$$
$$\iff \deg(F) \overset{?}{\geq} \frac{4 - 2\epsilon}{1 - \epsilon} k,$$

which, if true, would imply that Reduction 2 succeeds with probability at least $1/(3 - \epsilon)$. Again, we present the assumption and corresponding claim:

(A2) $\deg(F) \geq \dfrac{4 - 2\epsilon}{1 - \epsilon} k$ for some FVS F of size k

CLAIM 2. *If (A1) is false and (A2) is true, then Reduction 2 succeeds with probability at least $1/(3 - \epsilon)$.*

An immediate issue in this assumption is that the algorithm does not know $\deg(F)$, so it cannot determine whether (A2) is true or not. This can be accomplished by designing an algorithm to find Feedback Vertex Sets with additional properties defined as follows:

DEFINITION 2. (BOUNDED TOTAL DEGREE FVS)
In the bounded total degree FVS (BFVS) *problem, the input is an unweighted, undirected graph G on n vertices, and parameters $k \leq n$ and $\overline{d} \leq O(1)$. The goal is to either output a FVS F of size at most k satisfying $\deg(F) \leq \overline{d}k$, or correctly conclude none exists.*

See Algorithm IC2 for an algorithm to solve BFVS. We remark that Lines 5 and 6 replace the tree decomposition and treewidthDP of IC1. Indeed. we need to solve the BFVS problem instead of FVS, and treewidthDP could be easily extended to solve this

problem as well. However, it treewidthDP crucially relies on exponential working space. In the new algorithm we circumvent this by exploiting special properties of the separation directly. The function of the new algorithm is described by the following lemma:

LEMMA 6.1. *There is an Algorithm BFVS1 that, given G, a FVS F of G of size k, parameter $\overline{d}$, and a separation (A, B, S) as given by Lemma 3.2, outputs a FVS of size at most $k - 1$ satisfying $\deg(F) \leq \overline{d}(k - 1)$, or Infeasible if none exists. The algorithm uses $O^\star(3^{(1 - 2^{-\overline{d}} + o(1))k})$ time and polynomial space.*

Because of its technical nature, we postpone the proof of this Lemma to Subsection 8.1.

LEMMA 6.2. *Algorithm IC2 solves the BFVS problem in $O^\star(3^{(1 - 2^{-\overline{d}} + o(1))k})$ time and polynomial space.*

Proof. Suppose that there exists a FVS F^* of size at most k satisfying $\deg(F^*) \leq \overline{d}k$. Let $(v_1, \ldots, v_n)$ be the ordering from Line 1, and define $V_i := \{v_1, \ldots, v_i\}$. Observe that $F^* \cap V_i$ is a FVS of $G[V_i]$ satisfying $\deg(F^* \cap V_i) \leq \overline{d}k$, so the FVS problem on Line 6 is feasible. By Lemma 6.1, Line 6 correctly computes a FVS with high probability on any given iteration. Therefore, with high probability, a FVS is returned successfully by a union bound.

We now bound the running time. On Line 4, the current set F is a FVS of $G[V_{i-1}]$ satisfying $\deg(F) \leq \overline{d}k$, so Lemma 1.1 guarantees a tree decomposition of width at most $(1 - 2^{-\overline{d}} + o(1))k$, and adding v_i to each bag on Line 5 increases the width by at most 1. By Lemma 6.1, Line 6 runs in time $O^\star(3^{(1 - 2^{-\overline{d}} + o(1))k})$ time, as desired. Lastly, the space bound follows clearly from the descriptions of IC2 and Lemma 6.1. □

LEMMA 6.3. *Fix the parameter $\epsilon \in (0, 1)$, and let $c_\epsilon := \max\{3 - \epsilon, 3^{1 - 2^{-(4 - 2\epsilon)/(1 - \epsilon)}}\}$. If $c_\epsilon \geq 2$, then FVS2(G, k) succeeds with probability at least c_ϵ^{-k}/k. Moreover, Algorithm FVS2(G, k) has $O^\star(3^{o(k)})$ expected running time.*

Proof. For the running time, the computation outside of Line 6 clearly takes poly(n) time. For each $k' \in (k_0, k]$, Line 6 is executed with probability $3^{-(1 - 2^{-\overline{d}})k'}$, and takes $O^\star(3^{(1 - 2^{-\overline{d}} + o(1))k'})$ time by Lemma 6.2. Therefore, in expectation, the total computation cost of Line 6 is polynomial per value of k', and also polynomial overall.

We continue with proving by induction on k that FVS2(G, k) succeeds with probability at least c^{-k}/k (we denote $c := c_\epsilon$). This statement is trivial for $k = 0$, since

---

Algorithm 3 $\mathtt{IC2}(G, k, \overline{d})$

Input: Graph $G = (V, E)$ and parameters $k \leq n$ and $\overline{d} = O(1)$.
Output: A FVS F of size at most k satisfying $\deg(F) \leq \overline{d}k$, or $\mathtt{Infeasible}$ if none exists.

1: Order the vertices V arbitrarily as $(v_1, \ldots, v_n)$
2: $F \leftarrow \emptyset$
3: **for** $i = 1, \ldots, n$ **do** $\triangleright$ ***Invariant:*** $\deg(F) \leq \overline{d}k$
4:   Compute a separation (A, B, S') of $G[\{v_1, \ldots, v_{i-1}\}]$ by Lemma 3.2 on input F
5:   $S \leftarrow S' \cup \{v_i\}$, so that (A, B, S) is a separation of $G[\{v_1, \ldots, v_i\}]$
6:   $F \leftarrow \mathtt{BFVS1}(G[\{v_1, \ldots, v_i\}], k+1, A, B, S)$
7:   **if** F is $\mathtt{Infeasible}$ **then**
8:     **return** $\mathtt{Infeasible}$
9: **return** F

Algorithm 4 $\mathtt{FVS2}(G, k)$

Input: Graph $G = (V, E)$ and parameter $k \leq n$.
Output: Either output a FVS F of size k, or (possibly incorrectly) conclude that one does not exist ($\mathtt{Infeasible}$).

1: **if** $k = 0$ **then return** $\emptyset$ if G is acyclic, and **return** $\mathtt{Infeasible}$ otherwise
2: Exhaustively apply Reduction 1 to (G, k) to get vertex set F and instance (G', k')
3: $\overline{d} \leftarrow (4 - 2\epsilon)/(1 - \epsilon)$
4: Flip a coin with $\mathtt{Heads}$ probability $3^{-(1-2^{-\overline{d}})k'}$
5: **if** coin flipped $\mathtt{Heads}$ **then**
6:   $F' \leftarrow \mathtt{IC2}(G', k', \overline{d})$
7: **else**
8:   **if** $n' \leq (3 - \epsilon)k'$ **then** $\triangleright$ *(A1) is true*
9:     Apply Reduction 3 to (G', k') to get vertex $v \in V$ and instance $(G'', k'-1)$
10:   **else** $\triangleright$ *(A1) is false*
11:     Apply Reduction 2 to (G', k') to get vertex $v \in V$ and instance $(G'', k'-1)$
12:   $F' \leftarrow \mathtt{FVS2}(G'', k'-1) \cup \{v\}$ $\triangleright$ *Denoting* $\mathtt{Infeasible} \cup S = \mathtt{Infeasible}$ *for any set* S
13: **return** $F \cup F'$

---

no probabilistic reductions are used and $\mathtt{FVS2}(G, k)$ succeeds with probability 1. For the inductive step, consider an instance $\mathtt{FVS2}(G, k+1)$. Let (G', k') be the reduced instance after Line 2. First, suppose that (A2) is false on instance (G', k'). That is, every FVS F of size at most k satisfies $\deg(F) \leq \frac{4-2\epsilon}{1-\epsilon}k'$; here, we will only need the existence of one such F. In this case, if $\mathtt{IC2}$ in Line 6 is executed, then it will correctly output a FVS F of size at most k, with high probability by Lemma 6.2. This happens with probability at least

$$3^{-(1-2^{-\overline{d}})k'} \cdot \left(1 - \frac{1}{\mathrm{poly}(n)}\right) \geq c^{-k'} \cdot \frac{1}{k} \geq \frac{c^{-k}}{k},$$

as desired.

Otherwise, suppose that (A2) is true on instance (G', k'). Then, by Claims 1 and 2, regardless of whether (A1) is true, the reduction applied succeeds with probability at least $1/(3 - \epsilon)$. This is assuming, of course, that Line 6 is not executed, which happens with probability $1 - c^{-k'} \geq 1 - 2^{-k'} \geq 1 - 1/k'$ since $c \geq 2$. By induction, the recursive call on Line 12 succeeds with probability at least $c^{-(k'-1)}/(k'-1)$, so the overall

probability of success is at least

$$\left(1 - \frac{1}{k'}\right) \cdot \frac{1}{3 - \epsilon} \cdot \frac{c^{-(k'-1)}}{k'-1} \geq \left(1 - \frac{1}{k'}\right) \cdot \frac{1}{c} \cdot \frac{c^{-(k'-1)}}{k'-1}$$
$$= \frac{c^{-k'}}{k'} \geq \frac{c^{-k}}{k},$$

as desired. $\quad\square$

To optimize for c_ϵ, we set $\epsilon \approx 0.155433$, giving $c_\epsilon \leq 2.8446$. Theorem 1.2 now follows by combining Lemma 6.3 with Lemma 2.1.

7 Improvement Using Matrix Multiplication

In this section, we further speed up the algorithm $\mathtt{IC2}$ that solves the BFVS problem. First, we open the Cut&Count black box, which essentially transforms the FVS (or BFVS) problem to counting the number of partitions of the graph that satisfy a particular constraint, modulo some integer. The transformation are similar to the presentation in [CNP+11], so we defer the details to Section 8. In [CNP+11], this counting problem is solved using dynamic programming on tree

decompositions in $O^\star(3^{\text{tw}})$ time, which can be turned in an $O^\star(3^k)$ time algorithm for BFVS.

As with most problems efficiently solvable on tree decompositions, the Cut&Count problem performs well when given small vertex separators. Indeed, we show in Subsection 8.1 that instead of calling the $O^\star(3^{\text{tw}})$ algorithm on the tree decomposition from Lemma 1.1, we can solve the problem by applying dynamic programming on the (A, B, S) separation from Lemma 3.2 directly in the same running time, and also in polynomial space. The resulting algorithm is the algorithm BFVS1 promised by Lemma 6.1.

How do we obtain an even faster running time, then? The main insight in this section is that the counting problem has a special arithmetic nature that also makes it amenable to *matrix multiplication* as well. Combining these two observations, we construct a *three-way vertex separation* of the graph G, defined as follows:

DEFINITION 3. (THREE-WAY SEPARATION)
Given a graph $G = (V, E)$, a partition $(S_1, S_2, S_3, S_{1,2}, S_{1,3}, S_{2,3}, S_{1,2,3})$ of V is a separation if there are no edges between any two sets S_I, S_J whose sets I and J are disjoint.

The construction of a good three-way separation is very similar to the "two-way separation" in Lemma 1.1: it also features a randomized coloring procedure and is proven using concentration arguments. We then apply a combination of dynamic programming and matrix multiplication on the three-way separator, which is presented as Algorithm BFVS2 in Subsection 8.2.

7.1 Three-Way Separator

LEMMA 7.1. (THREE-WAY SEPARATOR) *Given an instance (G, k) and a FVS F of size at most k, define $\overline{d} := \deg(F)/k$, and suppose that $\overline{d} = O(1)$. There is a polynomial time algorithm that computes a three-way separation $(S_1, S_2, S_3, S_{1,2}, S_{1,3}, S_{2,3}, S_{1,2,3})$ of G such that there exists values f_1, f_2 satisfying:*

1a. $f_1 \geq 3^{-\overline{d}}$

1b. $(f_1 - o(1))k \leq |S_i \cap F| \leq (f_1 + o(1))k$ *for all $i \in [3]$*

2a. $f_2 \geq (2/3)^{\overline{d}} - 2f_1$

2b. $(f_2 - o(1))k \leq |S_{i,j} \cap F| \leq (f_2 + o(1))k$ *for all $1 \leq i < j \leq 3$*

Proof. Our proof follows the outline of the proof of Lemma 3.2. Initially, we start out the same: fix $\epsilon := k^{-0.01}$, apply Lemma 3.1 on the same input (that is, $G - F$), and construct the bipartite graph H on the bipartition $F \uplus R$ in the same manner as in Lemma 3.2.

We recall that (1) $|R| \leq |E[\overline{F}, F]| + 2|E[F]| = \deg(F)$, (2) every vertex in R has degree at most $\overline{d}/\epsilon$, and (3) the degree of a vertex $v \in F$ in H is at most $\deg(v)$.

Now, instead of randomly two-coloring the vertex set R, the algorithm three-colors it. That is, for each vertex in R, color it with a color in $\{1, 2, 3\}$ chosen uniformly and independently at random. For each subset $I \subseteq 2^{[3]} \setminus \{\emptyset\}$, create a vertex set S_I consisting of all vertices $v \in F$ whose neighborhood in H sees the color set I precisely. More formally, let $c(v)$ and $N(v)$ be the color of $v \in R$ and the neighbors of v in H, and define $S_I = \{v \in R : \bigcup_{u \in N(v)} c(u) = I\}$. Furthermore, if I is a singleton set $\{i\}$, then add (to S_I) all vertices in the connected components C whose component vertex in R is colored i. From now on, we abuse notation, sometimes referring to sets $S_{\{1\}}, S_{\{1,2\}}$, etc. as $S_1, S_{1,2}$, etc.

The proof of the following easy Subclaim is essentially the same as the proof of Subclaim 1 (but with more cases), and therefore omitted.

SUBCLAIM 4. $(S_1, S_2, S_3, S_{1,2}, S_{1,3}, S_{2,3}, S_{1,2,3})$ *is a three-way separation.*

We start by proving Conditions (1a) and (1b) with the following strategy. First, we first present a value f_1 such that Condition (1b) holds with probability $1 - 1/\text{poly}(k)$. Then, we argue that actually, this value of f_1 satisfies Condition (1a) (with probability 1).

SUBCLAIM 5. *For $f_1 := \left(\sum_d p_d \cdot |F'_d|\right)/|F'|$, Condition (1b) holds with probability $1 - 1/\text{poly}(k)$.*

Proof. The proof uses similar concentration arguments as the proof of Subclaim 2. Again, fix a parameter $\epsilon := k^{-0.01}$ throughout the proof. Let F' be the vertices with degree at most $\overline{d}/\epsilon$, so that again, $|F'| \geq (1 - o(1))|F|$. Form the intersection graph I on the vertex set F' as in Subclaim 2, and color it with $(\overline{d}/\epsilon)^2 + 1$ colors with a standard greedy algorithm.

Let F'_d be the vertices in F' with degree $d \leq \overline{d}/\epsilon$ in H, and let $F'_{i,d}$ be the vertices colored i with degree d in H. If $|F'_{i,d}| < k^{0.9}$, then ignore it; since $\overline{d} \leq O(1)$ and $\epsilon = k^{-0.01}$, the sum of all such $F'_{i,d}$ is at most $((\overline{d}/\epsilon)^2 + 1) \cdot (\overline{d}/\epsilon) \cdot k^{0.9} = o(k)$, so they only affect condition (1b) by an additive $o(k)$ factor. Henceforth, assume that $|F'_i| \geq k^{0.9}$.

We only focus our attention on S_1; the claim for S_2 and S_3 are identical. The probability that a vertex $v \in F'_{i,d}$ joins S_1 is a fixed number p_d that only depends on d. Let $X := |F'_{i,d} \cap A|$ be the number of vertices in $F'_{i,d}$ that join S_1; we have $\mathbb{E}[X] = p_d \cdot |F'_{i,d}|$, and by

Hoeffding's inequality,

$$\Pr[|X - \mathbb{E}[X]| \geq k^{0.8}] \leq 2\exp(-2 \cdot (k^{0.8})^2/|F'_{i,d}|)$$
$$\leq 2\exp(-2 \cdot k^{0.6}) \leq 1/\mathrm{poly}(k)$$

for large enough k. Taking a union bound over all colors i and degrees d, we conclude that with probability $1 - 1/\mathrm{poly}(k)$,

$$||F' \cap S_1| - \mathbb{E}\left[|F' \cap S_1|\right]| \leq \left(\left(\frac{\overline{d}}{\epsilon}\right)^2 + 1\right)\frac{\overline{d}k^{0.8}}{\epsilon} + o(k),$$

which is $o(k)$. Moreover,

$$\mathbb{E}[|F' \cap S_1|] = \sum_d p_d \cdot |F'_d|,$$

and we see that

$$||S_1 \cap F| - f_1 k| = ||S_1 \cap F'| - f_1 \cdot |F'|| + o(k) = o(k),$$

which fulfills condition (1b). $\square$

SUBCLAIM 6. *For* $f_1 := \left(\sum_d p_d \cdot |F'_d|\right)/|F'|$, *Condition (1a) holds with probability* $1 - 1/\mathrm{poly}(k)$.

Proof. The number p_d equals 3^{-d}, so $f_1 = \left(\sum_d |F'_d| \cdot 3^{-d}\right)/|F'|$. Observe that $\deg(F')/|F'| \leq \deg(F)/|F| = \overline{d}$, since the vertices in $F\backslash F'$ are precisely those with degree exceeding some threshold. Therefore,

$$f_1 = \frac{1}{|F'|}\sum_d |F'_d| \cdot 3^{-d}$$
$$= \frac{1}{|F'|}\sum_{v \in F'} 3^{-\deg(v)}$$
$$\geq 3^{-\deg(F')/|F'|},$$

where the last inequality follows from convexity of the function 3^{-x}. $\square$

SUBCLAIM 7. *For* $f_2 := \left(\sum_d p_d \cdot |F'_d|\right)/k$, *Condition (2b) holds with probability* $1 - 1/\mathrm{poly}(k)$.

Proof. The proof is identical to that of Subclaim 5, except that p_d is now the probability that a vertex $v \in F'_{i,d}$ joins S_2. $\square$

SUBCLAIM 8. *The* $f_1 := \left(\sum_d p_d \cdot |F'_d|\right)/|F'|$ *and* $f_2 := \left(\sum_d p_d \cdot |F'_d|\right)/k$, *condition (2a) holds.*

Proof. Here, our strategy is slightly different. Let q_d be the probability that a vertex v of degree d in H joins one of S_1, S_2, and $S_{1,2}$. Since this is also the probability that no neighbor of v is colored 3, we have $q_d = (2/3)^d$. Let $p_{1,d}$ and $p_{2,d}$ be the value of p_d in the

proofs of Subclaim 5 and Subclaim 7, respectively, so that $q_d = 2p_{1,d} + p_{2,d}$. Therefore,

$$2f_1 + f_2 = 2 \cdot \frac{1}{|F'|}\sum_d p_{1,d} \cdot |F'_d| + \frac{1}{|F'|}\sum_d p_{2,d} \cdot |F'_d|$$
$$= \frac{1}{|F'|}\sum_d q_d \cdot |F'_d|$$
$$= \frac{1}{|F'|}\sum_d |F'_d| \cdot \left(\frac{2}{3}\right)^d$$
$$= \frac{1}{|F'|}\sum_{v \in F'}\left(\frac{2}{3}\right)^{\deg(v)}$$
$$\geq \left(\frac{2}{3}\right)^{\deg(F')/|F'|},$$

where the last inequality follows from convexity of the function $(2/3)^x$. Again, we have $\deg(F')/|F'| \leq \deg(F)/|F| = \overline{d}$, so

$$f_2 \geq \left(\frac{2}{3}\right)^{\deg(F')/|F'|} - 2f_1 \geq \left(\frac{2}{3}\right)^{\overline{d}} - 2f_1,$$

which fulfills condition (2a). $\square$

$\square$

7.2 Matrix Multiplication Algorithm In this section, we present the improved iterative compression algorithm IC3. It is mostly unchanged from IC2, except that the algorithm now computes a three-way separator and calls the faster BFVS algorithm BFVS2 on it. Because of its technical nature, the algorithm BFVS2 and its analysis are deferred to Subsection 8.2. Instead, we simply state its running time guarantee in Lemma 7.2 below.

LEMMA 7.2. *There is an Algorithm* BFVS2 *that, given* G, *a FVS* F *of* G *of size* k, *parameter* $\overline{d}$, *and a separation* $(S_1, S_2, S_3, S_{1,2}, S_{1,3}, S_{2,3}, S_{1,2,3})$ *as given by Lemma 7.1, outputs a FVS of size at most* $k - 1$ *satisfying* $\deg(F) \leq \overline{d}(k-1)$, *or* Infeasible *if none exists. The algorithm runs in time* $O^\star(3^{(1-\min\{(2/3)\overline{d}, (3-\omega)(2/3)\overline{d} + (2\omega-3)3^{-\overline{d}}\} + o(1))k})$.

Assuming Lemma 7.2, we prove our main result, Theorem 1.1, restated below.

THEOREM 7.1. *There is a randomized algorithm that solves FVS in time* $O^\star(2.69998^k)$. *If* $\omega = 2$, *then the algorithm takes time* $O^\star(2.6252^k)$.

Proof. We run FVS2, replacing every occurrence of IC2 with IC3. Following FVS2, we define $\overline{d} :=$

Algorithm 5 $\mathtt{IC3}(G, k, \overline{d})$

Input: Graph $G = (V, E)$ and parameters $k \leq n$ and $\overline{d} = O(1)$.
Output: A FVS F of size at most k satisfying $\deg(F) \leq \overline{d}k$, or `Infeasible` if none exists.

1: Order the vertices V arbitrarily as $(v_1, \ldots, v_n)$
2: $F \leftarrow \emptyset$
3: **for** $i = 1, \ldots, n$ **do**   ▷ *Invariant:* $\deg(F) \leq \overline{d}k$
4:   Compute a separation $(S_1, S_2, S_3, S_{1,2}, S_{1,3}, S_{2,3}, S_{1,2,3})$ of $G[\{v_1, \ldots, v_{i-1}\}]$ by Lemma 7.1 on input F
5:   $S_{1,2,3} \leftarrow S_{1,2,3} \cup \{v_i\}$, so that $(S_1, S_2, S_3, S_{1,2}, S_{1,3}, S_{2,3}, S_{1,2,3})$ is a three-way separation of $G[\{v_1, \ldots, v_i\}]$
6:   $F \leftarrow \mathtt{BFVS2}(G[\{v_1, \ldots, v_i\}], k+1, S_1, S_2, S_3, S_{1,2}, S_{1,3}, S_{2,3}, S_{1,2,3})$
7:   **if** F is `Infeasible` **then**
8:     **return** `Infeasible`
9: **return** F

$(4 - 2\epsilon)/(1 - \epsilon)$ for some $\epsilon > 0$ to be determined later; note that $\overline{d} \geq 4$ for any $\epsilon > 0$. Since $\omega < 2.3728639$ [Gal14], by Lemma 8.5, `IC3` runs in time $O^\star(3^{(1-((3-\omega)(2/3)^{\overline{d}}+(2\omega-3)\cdot 3^{-\overline{d}})+o(1))k})$, so `FVS2` runs in time $O^\star(c_\epsilon^k)$ for $c_\epsilon := \max\{3 - \epsilon, 3^{1-((3-\omega)(2/3)^{(4-2\epsilon)/(1-\epsilon)}+(2\omega-3)\cdot 3^{-(4-2\epsilon)/(1-\epsilon)})+o(1)}\}$. To optimize for c_ϵ, we set $\epsilon \approx 0.3000237$, giving $c_\epsilon \leq 2.699977$.

If $\omega = 2$, then by Lemma 8.5, `IC3` runs in time $O^\star(3^{(1-(2/3)^{\overline{d}}+o(1))k})$, so `FVS2` runs in time $O^\star(c_\epsilon^k)$ for $c_\epsilon := \max\{3 - \epsilon, 3^{1-(2/3)^{(4-2\epsilon)/(1-\epsilon)}+o(1)}\}$. To optimize for c_ϵ, we set $\epsilon \approx 0.3748068$, giving $c_\epsilon \leq 2.6252$.   □

8   Cut and Count

In this section we open the black box formed by the Cut&Count approach [CNP+11]. It should be noted that most of this section, except Subsection 8.2, is very similar to the methods from [CNP+11]. We need the following definition:

DEFINITION 8.1. ([CNP+11]) *Let G be a graph with weight function $\omega : V(G) \to \mathbb{N}$. Let s, m', W be integers. Define $\mathcal{C}_W^{\omega, s, m'}$ to be the set*

$$\left\{ (F, L, R) \in \binom{V(G)}{\cdot, \cdot, \cdot} \mid \omega(F) = W \wedge E[L, R] = \emptyset \right.$$
$$\left. \wedge\ |F| = s \wedge |E[L \cup R]| = m' \right\}.$$

In the above, $\binom{V(G)}{\cdot, \cdot, \cdot}$ denotes the set of all partitions of $V(G)$ into three sets (denoted by F for 'Feedback Vertex Set', L for 'left side of the cut', and R for 'right side of the cut'). In words, a partition (F, L, R) of the vertex set is an element of $\mathcal{C}_W^{\omega, s, m'}$ if the total weight of all vertices in F equals W, there are no edges between L and R, exactly m' edges with both endpoints either in L or in R, and $|F| = s$. The use of Definition 8.1 becomes clear in the following lemma. Intuitively, the crux is that F is a FVS of G if and only if for some

s, m', W the number of partitions (L, R) of $V \setminus F$ such that $|\mathcal{C}_W^{\omega, s, m'}|$ is odd; in this case $\deg(F)$ can be read off from W.

LEMMA 8.1. *Let G be a graph and d be an integer. Pick $\omega(v) \in_R \{1, \ldots, 2|V|\}$ uniformly and independent at random for every $v \in V(G)$, and define $\omega'(v) := |V|^2\omega(v) + d(v)$. The following statements hold:*

1. *If for some integers m', $W = i|V|^2 + d$ we have that $|\mathcal{C}_W^{\omega', k, m'}| \not\equiv 0 \pmod{2^{n-k-m'}}$, then G has a feedback vertex set F satisfying $\deg(F) = d$.*

2. *If G has a feedback vertex set F satisfying $\deg(F) = d$, then with probability at least $1/2$ for some m', $W = i|V|^2 + d$ we have that $|\mathcal{C}_W^{\omega', k, m'}| \not\equiv 0 \pmod{2^{n-k-m'}}$.*

Lemma 8.1 states that in order to solve the Feedback Vertex Set problem it is sufficient to compute $|\mathcal{C}_W^{\omega, n-k, m'}|$ for all setting of the parameters. Before proving the Lemma we need to recall some standard building blocks:

LEMMA 8.2. (LEMMA A.7 IN [CNP+11]) *A graph with n vertices and m edges is a forest iff it has at most $n - m$ connected components.*

DEFINITION 8.2. *A function $\omega : U \to \mathbb{Z}$ isolates a set family $\mathcal{F} \subseteq 2^U$ if there is a unique $S' \in \mathcal{F}$ with $\omega(S') = \min_{S \in \mathcal{F}} \omega(S)$, where $\omega(S') := \sum_{v \in S'} \omega(v)$.*

LEMMA 8.3. (ISOLATION LEMMA, [MVV87]) *Let $\mathcal{F} \subseteq 2^U$ be a non-empty set family over a universe U. For each $u \in U$, choose a weight $\omega(u) \in \{1, 2, \ldots, W\}$ uniformly and independently at random. Then $\Pr[\omega \text{ isolates } \mathcal{F}] \geq 1 - |U|/W$.*

Proof. [of Lemma 8.1] We first prove 1. Note that if $|\mathcal{C}_W^{\omega', k, m'}| \not\equiv 0 \pmod{2^{n-k-m'}}$, there must be some vertex subset F such that the number of choices L, R

with $(F, L, R) \in \mathcal{C}_W^{\omega,s,m'}$ is not a multiple of $2^{n-k-m'}$. As we can independently decide for each component of $G[V \setminus F]$ whether to include it in L, R $G[V \setminus F]$ thus must have at most $n - k - m'$ connected components. By Lemma 8.2 it therefore must be a FVS. The condition on the degree follows by the weighting.

For item 2. First apply Lemma 8.3 with $U = V$ and the set family $\mathcal{F}$ being the set of all feedback vertex sets F of G satisfying $\deg(F) = d$. With probability $1/2$, there will be some weight i such that there is a unique FVS F with $\deg(F) = d$ of weight i. By Lemma 8.2 this is the only F that has a contribution to $|\mathcal{C}_W^{\omega',k,m'}|$ that is not a multiple of $2^{n-k-m'}$ as the number of extension of F to an object of $\mathcal{C}_W^{\omega',k,m'}$ is exactly $2^{\mathrm{cc}(G[V \setminus F])}$,[7] assuming $\omega(F) = W$, $|F| = k$ and $|E[V \setminus F]| = m'$. $\square$

We now continue with a lemma that is useful towards computing $|\mathcal{C}_W^{\omega,s,m'}|$.

Definition 8.3. *If $F^0 \subseteq V(G)$ is a FVS of G and $(F, L, R) \in \binom{F^0}{\cdot,\cdot,\cdot}$, we denote $c_W^{\omega,s,m'}(F, L, R) :=$*

$$|\{(F', L', R') \in \mathcal{C}_W^{\omega,s,m'} : F' \cap F^0 = F$$
$$\wedge L' \cap F^0 = L \wedge R' \cap F^0 = R\}|.$$

Lemma 8.4. *There is a polynomial time algorithm* $\mathtt{forestDP}(G, \omega, F, L, R, s, m', W)$ *that, given a graph G, weight function $\omega : V(G) \to \mathbb{N}$, vertex sets F, L, R and integers s, k, m', W computes $c_W^{\omega,s,m'}(F, L, R)$ in* $\mathrm{poly}(n, W)$ *time, assuming that $F \cup L \cup R$ is an FVS of G.*

Proof. We denote $F_0 = F \cup L \cup R$ for the given FVS. We will use dynamic programming over the forest induced by $V \setminus (F \cup L \cup R)$, in a way very similar to the proof of [CNP+11, Theorem B.1]. We assign roots to each tree in the forest $V(G) \setminus F_0$ arbitrarily, so the standard relations parents, children, ancestors and descendants are well-defined. For a vertex v, we denote $T[v]$ for the tree induced by v and all its descendants. If v has d children (which we order in arbitrary fashion) and $i \leq d$, we also denote $T[v, i]$ for the tree induced by v and all descendants of its first i children.

For $x \in \{L', R', F'\}$, the table entries for the dynamic programming are defined as follows:

$$A_{W,s,m'}^{(x)}[v, i] := |\{(F', L', R') \in \binom{V(T[v,i]) \cup F_0}{\cdot,\cdot,\cdot} :$$
$$F' \cap F_0 = F \wedge L' \cap F_0 = L \wedge R' \cap F_0 = R$$
$$\wedge \omega(F') = W \wedge E[L', R'] = \emptyset$$
$$\wedge |F'| = s \wedge |E[L' \cup R']| = m' \wedge v \in x\}|.$$

---
[7] Here we let cc denote the number of connected components

For convenience we also denote $A_{W,s,m'}^{(x)}[v]$ for $A_{W,s,m'}^{(x)}[v, d]$, where d is number of children of v.

If v is a leaf of a tree in the forest $V \setminus F_0$, then it is easy to see that we have

$$A_{W,s,m'}^{(x)}[v, 0] = \begin{cases} 1, & \text{if } \omega(v) = W - \omega(F) \wedge |F| \\ & = s \wedge |E[L^* \cup R^*]| \\ & = m' \wedge E[L^*, R^*] = \emptyset \\ 0, & \text{otherwise,} \end{cases}$$

where L^* denotes $L' \cup \{v\}$ if $x = L'$ and L' otherwise, and similarly R^* denotes $R' \cup \{v\}$ if $x = R'$ and R' otherwise.

If v has children $v_1, \ldots, v_d$ in the forest $V \setminus F_0$, we have that

$$A_{W,s,m'}^{(F')}[v, 1] = \sum_{x' \in \{L', R', F'\}} A_{W-\omega(F'),s-1,m'}^{(x')}[v_1]$$

$$A_{W,s,m'}^{(L')}[v, 1] = [N(v) \cap R = \emptyset](A_{W,s,m'-|N(v)\cap L|-1}^{(L')}[v_1]$$
$$+ A_{W,s,m'-|N(v)\cap L|}^{(F')}[v_1])$$

$$A_{W,s,m'}^{(R')}[v, 1] = [N(v) \cap L = \emptyset](A_{W,s,m'-|N(v)\cap R|-1}^{(R')}[v_1]$$
$$+ A_{W,s,m'-|N(v)\cap R|}^{(F')}[v_1])$$

Here we use Iverson's bracket notation $[b]$ for a Boolean predicate b which denotes 1 if b is true and 0 otherwise.

To see that this holds, note we need to account for the possible contributions of v to $\omega(F')$, $|F'|$ and need to check whether $E[L', R'] = \emptyset$ is not violated and account for an increase of $E[L' \cup R']$ which may include the edge $\{v, v_1\}$.

Moreover, for $i > 1$ we have that

$$A_{W,s,m'}^{(F')}[v, i] =$$
$$\sum_{\substack{x' \in \{L', R', F'\} \\ W_1+W_2=W-|F| \\ s_1+s_2=m'-|F| \\ m'_1+m_2=m'-|E[L\cup R]|}} A_{W_1,s_1,m'_1}^{(x)}[v, i-1] * A_{W_2,s_2,m'_2}^{(x')}[v_i].$$

$$A_{W,s,m'}^{(L')}[v, i] =$$
$$\sum_{\substack{x' \in \{L', F'\} \\ W_1+W_2=W-|F| \\ s_1+s_2=m'-|F| \\ m'_1+m_2=m'-|E[L\cup R]|-[x'=L']}} A_{W_1,s_1,m'_1}^{(x)}[v, i-1] * A_{W_2,s_2,m'_2}^{(x')}[v_i].$$

$$A_{W,s,m'}^{(R')}[v, i] =$$
$$\sum_{\substack{x' \in \{R', F'\} \\ W_1+W_2=W-|F| \\ s_1+s_2=m'-|F| \\ m'_1+m_2=m'-|E[L\cup R]|-[x'=R']}} A_{W_1,s_1,m'_1}^{(x)}[v, i-1] * A_{W_2,s_2,m'_2}^{(x')}[v_i].$$

985

Similarly as before we need account for the possible contributions of v to $\omega(F')$, $|F'|$ and need to check whether $E[L', R'] = \emptyset$ is not violated and account for an increase of $E[L' \cup R']$ which may include the edge $\{v, v_1\}$. Note we compensate for double counting due to F, L, R.

Finally we can merge the counts stored at the roots of each tree of the forest to get the desired value. Specifically, if the the roots are $r_1, \ldots, r_c$ then
(8.7)
$$c_W^{\omega, s, m'}(F, L, R) = \sum_{\substack{x_1, \ldots, x_c \in \{L', R', F\} \\ W_1 + \ldots + W_d = W - (d-1)|F| \\ s_1 + \ldots + s_d = m' - (d-1)|F| \\ m'_1 + \ldots + m'_d = m' - (d-1)|E[L \cup R]|}} \prod_{i=1}^d A_{W_i, s_i, m'_i}^{(x_i)}[r_i].$$

Here we again compensate for double counting due to F, L, R. Given all entries $A_{W_i, s_i, m'_i}^{(x_i)}[r_i]$, we can combine (8.7) with standard dynamic programming to compute $c_W^{\omega, s, m'}(F, L, R)$ is polynomial time. $\square$

8.1 Cut and Count Using Simple Separation: Proof of Lemma 6.1

The algorithm promised by the lemma is listed in Algorithm BFVS1. For the claimed time bound, note all steps are polynomial time except Lines 5, 8 and 11, and these jointly give rise to $3^{(|S| + |A \cap F|)} + 3^{(|S| + |B \cap F|)}$ iterations. As the separation (A, B, S) was assumed to satisfy the properties $|A \cap F|, |B \cap F| \geq (2^{-\overline{d}} - o(1))k$ and $|S| \leq (1 - 2 \cdot 2^{-\overline{d}} + o(1))k$ from Lemma 3.2, the time bound follows.

For the correctness, we claim that at Line 14 count $= |\mathcal{C}_W^{\omega', k, m'}|$ for some m', $W = i|V|^2 + d$. The lemma follows from this by Lemma 8.1. To see the claim, observe that the algorithm iterates over all partitions (F, L, R) of the separator S in Line 5. For each partition, and every way to split W, k, m (Line 6), the algorithm computes the number countA (resp. countB) of "extensions" of the partition in $G[A \cup S]$ (resp. $G[B \cup S]$) that "respect" the split, and then multiples countA and countB. To see why the two counts are multiplied, observe that there are no edges between A and B in the separation (A, B, S), so extending into $G[A \cup S]$ is independent to extending into $G[B \cup S]$.

8.2 Cut and Count Using 3-way Separation: Proof of Lemma 7.2

We now present the improved BFVS algorithm below. First, we argue its correctness, that at Line 18, count $= |\mathcal{C}_W^{\omega', k, m'}|$. First, the algorithm iterates over partitions of $S_{1,2,3}$ in Line 6 the same way Algorithm BFVS1 iterates over partitions of S. The rest of the algorithm, which includes the matrix multiplication routine, seeks to compute the number of extensions in $S_1 \cup S_2 \cup S_3$ given each partition of

$S_{1,2} \cup S_{1,3} \cup S_{2,3}$ (and given the current partition of $S_{1,2,3}$, as well as a three-way split of W, k, m'). Like in the case of separator (A, B, S) in BFVS1, it is true that the extensions of S_1, S_2, and S_3 are independent given the partition of $S_{1,2} \cup S_{1,3} \cup S_{2,3}$, but in this case, the size of $|S_{1,2} \cup S_{1,3} \cup S_{2,3}|$ can be prohibitively large. Instead, to compute this quantity efficiently, first observe that since there are no edges between S_1 and $S_{2,3}$, the number of extensions of S_1 only depends on the partition of $S_{1,2} \cup S_{1,3}$, and not $S_{2,3}$. For each partition of $S_{1,2} \cup S_{1,3}$, take the graph H defined in Line 8, and imagine adding an edge between the respective partitions of $S_{1,2}$ and $S_{1,3}$, weighted by the number of extensions in S_1. We proceed analogously for extensions of S_2 and S_3. Finally, the total number of extensions (given the partition of $S_{1,2,3}$) amounts to computing, for all triangles in H, the product of the weights of the three edges (Line 16), which can be solved by a standard matrix multiplication routine.

Finally, the desired running time bound is more complicated for BFVS2. We prove Lemma 8.5 below which, together with Lemma 7.1, implies the running time bound of Lemma 7.2.

LEMMA 8.5. *For any constant $\epsilon > 0$, the BFVS problem with parameters k and $\overline{d}$ can be solved in time $O^\star(3^{(1 - \min\{(2/3)^{\overline{d}}, (3-\omega)(2/3)^{\overline{d}} + (2\omega-3)3^{-\overline{d}}\} + o(1))k})$.*

Proof. Let f_1, f_2 be the values from Lemma 7.1, and let $f_3 := 1 - 3f_1 - 3f_2$, so that $(f_3 - o(1))k \leq |S_{1,2,3}| \leq (f_3 + o(1))k$. For each of the $O^\star(3^{f_3 + o(1)})$ iterations on Line 6, building the graph H (Lines 8 to 15) takes time $O^\star(3^{2f_2 + f_1 + o(k)})$, and running matrix multiplication (Line 16) on a graph with $O^\star(3^{f_2 + o(k)})$ vertices to compute the sum over the product of the three edges of all triangles takes time $O^\star(3^{\omega f_2 + o(k)})$. Therefore, the total running time is
$$O^\star(3^{f_3 + o(k)}(3^{2f_2 + f_1 + o(k)} + 3^{\omega f_2 + o(k)}))$$
$$= O^\star(3^{f_3 + 2f_2 + f_1 + o(k)} + 3^{f_3 + \omega f_2 + o(k)})$$
$$= O^\star(3^{1 - f_2 - 2f_1 + o(k)} + 3^{1 - (3-\omega)f_2 - 3f_1 + o(k)})$$
$$\leq O^\star(3^{1 - \left(\frac{2}{3}\right)^{\overline{d}} + o(k)} + 3^{1 - (3-\omega)\left(\frac{2}{3}\right)^{\overline{d}} - \frac{2\omega-3}{3^{\overline{d}}} + o(k)}),$$

where the last inequality uses Conditions (1a) and (1b) of Lemma 7.1, and the fact that $2\omega - 3 \geq 0$. $\square$

References

[AEFM89] Karl R. Abrahamson, John A. Ellis, Michael R. Fellows, and Manuel E. Mata. On the complexity of fixed parameter problems (extended abstract). In *30th*

Algorithm 6 BFVS1(G, F, k, A, B, S)

Input: Graph $G = (V, E)$, FVS F of size k, parameters $k, \overline{d} \leq n$, and separation (A, B, S) from Lemma 3.2

Output: A FVS of size at most k satisfying $\deg(F) \leq \overline{d}k$, or `Infeasible` if none exists.

1: Pick $\omega \in_R \{1, \ldots, 2|V|\}$ uniformly and independently at random for every $v \in V(G)$
2: Set $\omega'(v) := |V|^2 \omega(v) + d(v)$
3: $\texttt{count} \leftarrow 0$
4: **for** m', W such that $0 \leq m' \leq m$, $W = |V^2|i + d \leq \omega(V)$ for some $d \leq \overline{d}k$ **do**
5:     **for** $(F_S, L_S, R_S) \in \binom{S}{\cdot,\cdot,\cdot}$ **do**
6:         **for** W_1, k_1, m_1' such that $0 \leq W_1 \leq W, 0 \leq k_1 \leq k, 0 \leq m_1' \leq m'$ **do**
7:             $\texttt{countA} \leftarrow 0$
8:             **for** $(F_A, L_A, R_A) \in \binom{A \cap F}{\cdot,\cdot,\cdot}$ **do**
9:                 $\texttt{countA} \leftarrow \texttt{countA} + \texttt{forestDP}(G[A \cup S], \omega, F_A \cup F_S, L_A \cup L_S, R_A \cup R_S, k_1, m_1', W_1)$
10:            $\texttt{countB} \leftarrow 0$
11:            **for** $(F_B, L_B, R_B) \in \binom{B \cap F}{\cdot,\cdot,\cdot}$ **do**
12:                 $\texttt{countB} \leftarrow \texttt{countB} + \texttt{forestDP}(G[B \cup S], \omega, F_B \cup F_S,$
                        $L_B \cup L_S, R_B \cup R_S, k + |F_S| - k_1, m' + |E[L_S \cup R_S]| - m_1', W + \omega(F_S) - W_1)$
13:            $\texttt{count} \leftarrow \texttt{count} + \texttt{countA} \cdot \texttt{countB}$
14: **if** $\texttt{count} \not\equiv 0 \pmod{2^{n-k-m'}}$ **then**
15:     **return** a FVS of $G[\{v_1, \ldots, v_i\}]$ of size $\leq k$ satisfying $\deg(F) \leq \overline{d}k$, constructed by self-reduction
16: **return** `Infeasible`

Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989, pages 210–215, 1989.

[AGSS16] Akanksha Agrawal, Sushmita Gupta, Saket Saurabh, and Roohani Sharma. Improved algorithms and combinatorial bounds for independent feedback vertex set. In *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, pages 2:1–2:14, 2016.

[BBG00] Ann Becker, Reuven Bar-Yehuda, and Dan Geiger. Randomized algorithms for the loop cutset problem. *J. Artif. Intell. Res.*, 12:219–234, 2000.

[BGNR98] Reuven Bar-Yehuda, Dan Geiger, Joseph Naor, and Ron M. Roth. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and bayesian inference. *SIAM J. Comput.*, 27(4):942–959, 1998.

[Bod94] Hans L. Bodlaender. On disjoint cycles. *Int. J. Found. Comput. Sci.*, 5(1):59–68, 1994.

[Cao18] Yixin Cao. A naive algorithm for feedback vertex set. In *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, pages 1:1–1:9, 2018.

[CCL15] Yixin Cao, Jianer Chen, and Yang Liu. On feedback vertex set: New measure and new structures. *Algorithmica*, 73(1):63–86, 2015.

[CDL+16] Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3):41:1–41:24, 2016.

[CFJ+14] Marek Cygan, Fedor Fomin, Bart M.P. Jansen, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx,

Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Open problems for fpt school 2014, 2014. http://fptschool.mimuw.edu.pl/opl.pdf.

[CFK+15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[CFL+08] Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. Improved algorithms for feedback vertex set problems. *J. Comput. Syst. Sci.*, 74(7):1188–1198, 2008.

[CKX10] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010.

[CNP+11] 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. *CoRR*, abs/1103.0534, 2011. Extended abstract appeared at FOCS 2011.

[CS15] Ruiwen Chen and Rahul Santhanam. Improved algorithms for sparse MAX-SAT and max-k-csp. In *Theory and Applications of Satisfiability Testing - SAT 2015 - 18th International Conference, Austin, TX, USA, September 24-27, 2015, Proceedings*, pages 33–45, 2015.

[Dec90] Rina Dechter. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artif. Intell.*, 41(3):273–312, 1990.

[DF92] Rodney G. Downey and Michael R. Fellows. Fixed parameter tractability and completeness. In *Complexity Theory: Current Research, Dagstuhl Workshop, February 2-8, 1992*, pages 191–225, 1992.

[DFL+07] Frank K. H. A. Dehne, Michael R. Fellows,

Algorithm 7 BFVS2$(G, k, S_1, S_2, S_3, S_{1,2}, S_{1,3}, S_{2,3}, S_{1,2,3})$

Input: Graph $G = (V, E)$, FVS F, parameters $k, \bar{d} \leq n$, and separation $(S_1, S_2, S_3, S_{1,2}, S_{1,3}, S_{2,3}, S_{1,2,3})$ from Lemma 7.1
Output: A FVS of size at most k satisfying $\deg(F) \leq \bar{d}k$, or `Infeasible` if none exists.

1: **for** poly(n) iterations **do**
2:      Pick $\omega \in_R \{1, \ldots, 2|V|\}$ uniformly and independently at random for every $v \in V(G)$
3:      Set $\omega'(v) := |V|^2 \omega(v) + d(v)$
4:      count $\leftarrow 0$
5:      **for** m', W such that $0 \leq m' \leq m$, $W = |V^2|i + d \leq \omega(V)$ for some $d \leq \bar{d}k$ **do**
6:          **for** $(F_0, L_0, R_0) \in \binom{S_{1,2,3}}{\cdot, \cdot, \cdot}$ **do**
7:              **for** nonnegative W_i, k_i, m'_i, $i \in [3]$ such that $\sum_i W_i = W$, $\sum_j k_i = k$, $\sum_i m'_i = m'$ **do**
8:                  $H \leftarrow$ an empty graph with vertices indexed by $\binom{S_1}{\cdot, \cdot, \cdot} \cup \binom{S_2}{\cdot, \cdot, \cdot} \cup \binom{S_3}{\cdot, \cdot, \cdot}$
9:                  **for** (i, j, k) in $\{(1, 2, 3), (2, 3, 1), (3, 1, 2)\}$ **do**
10:                      **for** $(F_1, L_1, R_1) \in \binom{S_{i,j}}{\cdot, \cdot, \cdot}$, $(F_2, L_2, R_2) \in \binom{S_{i,k}}{\cdot, \cdot, \cdot}$ **do**
11:                          count3 $\leftarrow 0$
12:                          **for** $(F_3, L_3, R_3) \in \binom{S_i \cap F}{\cdot, \cdot, \cdot}$ **do**
13:                              count3 $\leftarrow$ count3 $+$ forestDP$(G[S_i], \omega, F_3, L_3, R_3, k_i, m'_i, W_i)$
14:                      Add an edge e between vertices (F_1, L_1, R_1) and (F_2, L_2, R_2) of H
15:                      Assign weight count3 $(\bmod\ 2^{n-k-m'})$ to the edge e
16:              count0 $\leftarrow$ sum over the product of the three edges of all triangles in H
17:              count $\leftarrow$ count $+$ count0
18:      **if** count $\not\equiv 0$ $(\bmod\ 2^{n-k-m'})$ **then**
19:          **return** a FVS of $G[\{v_1, \ldots, v_i\}]$ of size $\leq k$ satisfying $\deg(F) \leq \bar{d}k$, constructed by self-reduction
20: **return Infeasible**

Michael A. Langston, Frances A. Rosamond, and Kim Stevens. An o($2^{\mathrm{o(k)}} n^3$) FPT algorithm for the undirected feedback vertex set problem. *Theory Comput. Syst.*, 41(3):479–492, 2007.

[DHJ$^+$16] Holger Dell, Thore Husfeldt, Bart M. P. Jansen, Petteri Kaski, Christian Komusiewicz, and Frances A. Rosamond. The first parameterized algorithms and computational experiments challenge. In *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, pages 30:1–30:9, 2016.

[FGLS16] Fedor V. Fomin, Serge Gaspers, Daniel Lokshtanov, and Saket Saurabh. Exact algorithms via monotone local search. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 764–775, 2016.

[Gal14] François Le Gall. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*, pages 296–303, 2014.

[GGH$^+$06] Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *J. Comput. Syst. Sci.*, 72(8):1386–1396, 2006.

[GLL$^+$18] Anupam Gupta, Euiwoong Lee, Jason Li, Pasin Manurangsi, and Michal Wlodarczyk. Losing treewidth by separating subsets. *CoRR*, abs/1804.01366, 2018. To appear at SODA 2019.

[JJ17] Lars Jaffke and Bart M. P. Jansen. Fine-grained parameterized complexity analysis of graph coloring problems. In *Algorithms and Complexity - 10th International Conference, CIAC 2017, Athens, Greece, May 24-26, 2017, Proceedings*, pages 345–356, 2017.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, pages 85–103, 1972.

[KK18] Eun Jung Kim and O-joung Kwon. Erdős-pósa property of chordless cycles and its applications. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1665–1684, 2018.

[KP14] Tomasz Kociumaka and Marcin Pilipczuk. Faster deterministic feedback vertex set. *Inf. Process. Lett.*, 114(10):556–560, 2014.

[KPS04] Iyad A. Kanj, Michael J. Pelsmajer, and Marcus Schaefer. Parameterized algorithms for feedback vertex set. In *Parameterized and Exact Computation, First International Workshop, IWPEC 2004, Bergen, Norway, September 14-17, 2004, Proceedings*, pages 235–247, 2004.

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[PBvR16] Willem J. A. Pino, Hans L. Bodlaender, and Johan M. M. van Rooij. Cut and count and representative sets on branch decompositions. In *11th Interna-*

tional Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark, pages 27:1–27:12, 2016.

[RSS02] Venkatesh Raman, Saket Saurabh, and C. R. Subramanian. Faster fixed parameter tractable algorithms for undirected feedback vertex set. In *Algorithms and Computation, 13th International Symposium, ISAAC 2002 Vancouver, BC, Canada, November 21-23, 2002, Proceedings*, pages 241–248, 2002.

[RSS06] Venkatesh Raman, Saket Saurabh, and C. R. Subramanian. Faster fixed parameter tractable algorithms for finding feedback vertex sets. *ACM Trans. Algorithms*, 2(3):403–415, 2006.

[RSV04] Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004.

[WLS85] Ching-Chy Wang, Errol L. Lloyd, and Mary Lou Soffa. Feedback vertex sets and cyclically reducible graphs. *J. ACM*, 32(2):296–313, 1985.