Linear Time Parameterized Algorithms for SUBSET FEEDBACK VERTEX SET

Daniel Lokshtanov, University of Bergen, Norway
M.S. Ramanujan, University of Bergen, Norway
Saket Saurabh, The Institute of Mathematical Sciences, HBNI, Chennai, India
University of Bergen, Norway

In the SUBSET FEEDBACK VERTEX SET (SUBSET FVS) problem, the input is a graph G on n vertices and m edges, a subset of vertices T, referred to as terminals, and an integer k. The objective is to determine whether there exists a set of at most k vertices intersecting every cycle that contains a terminal. The study of parameterized algorithms for this generalization of the FEEDBACK VERTEX SET problem has received significant attention over the last few years. In fact the parameterized complexity of this problem was open until 2011, when two groups independently showed that the problem is fixed parameter tractable (FPT). Using tools from graph minors Kawarabayashi and Kobayashi obtained an algorithm for SUBSET FVS running in time $\mathcal{O}(f(k) \cdot n^2 m)$ [SODA 2012, JCTB 2012]. Independently, Cygan et al. [ICALP 2011, SIDMA 2013] designed an algorithm for SUBSET FVS running in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$. More recently, Wahlström obtained the first single exponential time algorithm for SUBSET FVS, running in time $4^k \cdot n^{\mathcal{O}(1)}$ [SODA 2014]. While the $2^{\mathcal{O}(k)}$ dependence on the parameter k is optimal under the Exponential Time Hypothesis (ETH), the dependence of this algorithm as well as those preceding it, on the input size is at least quadratic. In this paper we design the first linear time parameterized algorithms for SUBSET FVS. More precisely, we obtain the following new algorithms for SUBSET FVS.

— A randomized algorithm for SUBSET FVS running in time $\mathcal{O}(25.6^k \cdot (n+m))$.
— A deterministic algorithm for SUBSET FVS running in time $2^{\mathcal{O}(k \log k)} \cdot (n+m)$.

Since it is known that assuming the Exponential Time Hypothesis, SUBSET FVS cannot have an algorithm running in time $2^{o(k)} n^{\mathcal{O}(1)}$, our first algorithm obtains the best possible asymptotic dependence on both the parameter as well as the input size. Both of our algorithms are based on "cut centrality", in the sense that solution vertices are likely to show up in minimum size cuts between vertices sampled from carefully chosen distributions.

CCS Concepts:•**Theory of computation** → **Fixed parameter tractability;**

Additional Key Words and Phrases: Feedback Vertex Set, Graph Separation Problems, Linear-time FPT algorithms

ACM Reference Format:
Daniel Lokshtanov, M. S. Ramanujan and Saket Saurabh, 2017. Linear Time Parameterized Algorithms for

---

This work is supported by funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no. 306992 and by Bergen Research Foundation under the grant "BeHard".
Author's addresses: Daniel Loksthanov, Department of Informatics, University of Bergen, daniello@ii.uib.no. M. S. Ramanujan, (Current address) Department of Computer Science, University of Warwick, R.Maadapuzhi-Sridharan@warwick.ac.uk. Saket Saurabh, The Institute of Mathematical Sciences, HBNI, Chennai, saket@imsc.res.in.

DOI: http://dx.doi.org/10.1145/0000000.0000000

SUBSET FEEDBACK VERTEX SET. *ACM Trans. Algor.* 0, 0, Article 0 (0), 40 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

FEEDBACK SET problems constitute one of the most important topics of research in parameterized algorithms [Cao et al. 2015; Chen et al. 2008; Chen et al. 2008; Chitnis et al. 2015; Cygan et al. 2011; Cygan et al. 2013; Kawarabayashi and Kobayashi 2012; Kakimura et al. 2012; Kociumaka and Pilipczuk 2014; Raman et al. 2006; Wahlström 2014]. Typically, in these problems, we are given an undirected graph G (or a directed graph) and a positive integer k, and the objective is to "hit" all cycles of the input graph using at most k vertices (or edges or arcs). Recently, there has been a lot of study on the *subset variant* of FEEDBACK SET problems. In these problems, the input also includes a terminal subset $T \subseteq V(G)$ and the goal is to detect the presence of a set, referred to as a *subset feedback vertex set*, that hits all T*-cycles*, that is, cycles whose intersection with T is non-empty. In this paper we consider the following problem.

SUBSET FEEDBACK VERTEX SET (SUBSET FVS)
Instance: A graph G on n vertices and m edges, a subset of T of $V(G)$, and a positive integer k.
Parameter: k
Question: Is there a k-sized vertex subset S that intersects every T-cycle?

SUBSET FVS generalizes FEEDBACK VERTEX SET as well as the well known MULTIWAY CUT problem. In this paper we explore parameterized algorithms for SUBSET FVS. In parameterized complexity each problem instance has an associated parameter k and a central notion in parameterized complexity is *fixed parameter tractability* (FPT). This means, for a given instance (x, k), solvability in time $\tau(k) \cdot |x|^{\mathcal{O}(1)}$, where τ is an arbitrary function of k.

The study of parameterized algorithms for the SUBSET FVS problem has received significant attention in the last few years. The existence of an FPT algorithm for SUBSET FVS was shown only in 2011, when two groups independently gave FPT algorithms for the problem. Using tools from graph minors Kawarabayashi and Kobayashi obtained an algorithm for SUBSET FVS with running time $\mathcal{O}(f(k) \cdot n^2 m)$ [Kawarabayashi and Kobayashi 2012] (also see [Kakimura et al. 2012]). Independently, Cygan et al. [Cygan et al. 2013] combined iterative compression [Niedermeier 2006] with Gallai's theorem [Diestel 2010] to obtain an algorithm for SUBSET FVS with running time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$. Cygan et al. asked whether it is possible to obtain an algorithm for SUBSET FVS running in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$. Wahlström [Wahlström 2014] resolved this question in the affirmative by giving an algorithm for SUBSET FVS, with running time $4^k \cdot n^{\mathcal{O}(1)}$. It is known that under the Exponential Time Hypothesis (ETH) [Impagliazzo et al. 2001], FEEDBACK VERTEX SET does not not admit an algorithm with running time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ [Cygan et al. 2014]. Since SUBSET FVS is a generalization of FEEDBACK VERTEX SET, it follows that assuming ETH, SUBSET FVS also does not admit an algorithm with running time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$.

The focus of this paper is the second component of the running time of parameterized algorithms, that is, the running time dependence on the input size n. This direction of research is as old as the existence of parameterized algorithms, with classic results, such as Bodlaender's linear time algorithm for treewidth [Bodlaender 1996] and the cubic time algorithm of Robertson and Seymour for the disjoint paths problem [Robertson and Seymour 1995]. A more recent phenomenon is that one strives for linear time parameterized algorithms that do not compromise too much on the dependence of the running time on the parameter k. The gold standard for these results are algorithms with linear dependence on input size as well as provably optimal (under ETH) dependence on the parameter. New results in this direction include parameterized algorithms for problems such as ODD CYCLE TRANSVERSAL [Iwata et al. 2014; Ramanujan and Saurabh 2017], SUBGRAPH ISOMORPHISM [Dorn 2010], PLANARIZATION [Jansen et al. 2014; Kawarabayashi 2009] as well as a single-exponential and linear time parameterized constant factor approximation algorithm for TREEWIDTH [Bodlaender et al. 2013]. Other recent results include parameterized algorithms with improved dependence on input size for a host of problems [Grohe et al. 2013; Kawarabayashi et al. 2012; Kawarabayashi and Mohar 2008; Kawarabayashi et al. 2008; Kawarabayashi and Reed 2009; 2010].

The running time dependence on the input size for all the previous algorithms for SUBSET FVS is quite far from being linear. Recently, the methods behind the $4^k \cdot n^{\mathcal{O}(1)}$ time algorithm of Wahlström have been applied to give linear time FPT algorithms [Iwata et al. 2016] for several problems, including the edge-deletion variant of UNIQUE LABEL COVER. In this paper we design the first linear time parameterized algorithms for SUBSET FVS. The first algorithm is randomized with one-sided error, and obtains linear dependence on n as well as single exponential dependence on k.

THEOREM 1.1. *There is an algorithm that, given an instance* (G, T, k) *of* SUBSET FVS *runs in time* $25.6^k k^{\mathcal{O}(1)}(m + n)$ *and either returns a subset feedback vertex set of size at most* k *or concludes correctly with probability at least* $1 - \frac{1}{e}$ *that no such set exists, where* $m = |E(G)|$ *and* $n = |V(G)|$.

The second algorithm is deterministic at the cost of a slightly worse dependence on the parameter k.

THEOREM 1.2. *There is an algorithm that given an instance* (G, T, k) *of* SUBSET FVS *runs in time* $2^{\mathcal{O}(k \log k)}(m+n)$ *and either returns a subset feedback vertex set of size at most* k*, or correctly concludes that no such set exists.*

Methodology.. Both algorithms begin by applying simple preprocessing rules to ensure that there are no irrelevant vertices or edges and that every vertex is sufficiently connected to the terminals. While the preprocessing rules are quite easy to state, applying some of the rules exhaustively in linear time is non-trivial. We achieve this by using a classic algorithm of Hopcroft and Tarjan [Hopcroft and Tarjan 1973a] to decompose a graph into its 3-connected pieces.

At this point the randomized algorithm of Theorem 1.1 exploits the following structural insight. Consider a graph G that does have a subset feedback vertex set S of size at most k. $G - S$ has no T-cycles, and a graph without any T-cycles is essentially a forest where some of the terminal-free regions have been replaced by arbitrary graphs. A terminal-free region here is a connected component of $G - T - S$. The terminal-free regions may only interact with neighboring terminals via single edges. That is, for every connected component C in $G - T - S$ and every $t \in T$, there is at most one edge between C and t. Since a forest has average degree at most 2, at least half the regions interact with at most two other terminals in this way. Any such "degree two" region can be separated from the terminals by removing the solution S, as well as the two edges leaving the region in $G - S$. On the other hand the preprocessing rules ensure that every vertex has sufficient flow to the terminals, in particular the rules ensure that each "degree two" region must have at least one neighbor in the solution. From this we infer that the vertices of S appear very frequently in small cuts between vertices in “degree two regions” and terminal vertices. Our algorithm is based on a random process which is likely to produce a vertex v which is in a “degree two region”. The algorithm then samples a small set A such that A separates v from the terminal, and with good probability A has a large intersection with the solution S. At this point the algorithm guesses the intersection of the set A with the solution S, removes $A \cap S$ from the graph and starts again. The difficult part of the analysis is to show that whenever the algorithm guesses that a set X is a subset of the solution, the algorithm is correct with probability at least $\frac{1}{2^{O(|X|)}}$.

The deterministic algorithm of Theorem 1.2 is based on the same ideas as the randomized algorithm, but is quite far from being a “direct derandomization”. An attempt at a “direct derandomization” of the algorithm of Theorem 1.1 could look like this. The randomized algorithm essentially selects a vertex and claims that this vertex is a part of the solution. The analysis basically shows that for any optimal solution S of size at most k, the probability that the randomized algorithm selects a vertex in S is at least $1/25.6$. Suppose that we could compute deterministically for each vertex v, the probability $p(v)$ with which v is selected. We know that $\sum_{v \in S} p(v) \geq \frac{1}{25.6}$. Thus there must be a vertex in $v \in S$ such that $p(v) \geq \frac{1}{25.6k}$. However, $\sum_{v \in V(G)} p(v) = 1$, implying that the number of vertices v such that $p(v) \geq \frac{1}{25.6k}$ is at most $25.6k$. This gives us a candidate set of size $25.6k$, out of which a vertex must be in the solution. We can now guess this vertex, decrease k by 1, and re-start. The main problem with this approach is that we are unaware of an algorithm to compute $p(v)$ for all vertices v in linear time. The engine behind the algorithm of Theorem 1.2 is a different random process which also ensures that solution vertices are picked with high probability, but for which the probabilities $p(v)$ are efficiently computable.

Related work:. Apart from SUBSET FVS, the subset variants of DIRECTED FEEDBACK VERTEX SET and ODD CYCLE TRANSVERSAL have also been studied. Chitnis et al. [Chitnis et al. 2015] showed that the SUBSET DIRECTED FEEDBACK VERTEX SET is FPT. Kakimura et al. [Kakimura et al. 2012] initiated the study of SUBSET ODD CYCLE TRANSVERSAL and proposed an FPT algorithm for this problem using graph minors theory. SUBSET FVS has also been studied in the realm of approximation algorithms [Even et al. 2000b; Even et al. 2000a]. The current best approximation algorithm for SUBSET FVS has factor 8 [Even et al. 2000a]. Kakimura et al. [Kakimura et al. 2011] and Pontecorvi and Wollan [Pontecorvi and Wollan 2012] gave an Erdős-Pósa type result for SUBSET FVS. In particular, Pontecorvi and Wollan proved that there exists a constant α such that for all graphs $G, T \subseteq V(G)$, and a positive integers k, either there exist k vertex disjoint T-cycles, or there exists a subset-fvs X with $|X| \leq \alpha k \log k$.

2. PRELIMINARIES

All graphs we consider are undirected and finite, unless explicitly stated otherwise. When considering multigraphs we allow self loops (but not more than one per vertex) and any number of multiple edges. We use $[n]$ as an abbreviation for $\{1, \ldots, n\}$. For a graph G we denote its vertex set by $V(G)$ and the edge set by $E(G)$. An edge between vertices u and v is denoted by uv, and is identical to the edge vu. If there are multiple edges between u and v, then uv denotes any one of these edges. Removing a vertex v from the graph means removing v and all edges incident to v. This graph is denoted by $G-v$. Removing a set Z of vertices is denoted by $G-Z$ and means removing all vertices in Z one by one. The subgraph $G[V']$ induced by a vertex set V' is $G - (V(G) \setminus V')$. It is possible to *add* a vertex v to a graph G. In this case the vertex v is added to the set

$V(G)$ and some specified edges incident to v are added to $E(G)$. We denote by $G+(v,E)$ the graph obtained from G by adding v to $V(G)$ and E to $E(G)$. Here all edges in E are required to be incident to v.

The open neighborhood of a vertex v in graph G contains the vertices adjacent to v, and is written as $N_G(v)$. The open neighborhood of a set $S \subseteq V(G)$ is defined as $\bigcup_{v\in S} N_G(v) \setminus S$. Thus, if G is simple then $d_G(v) = |N_G(v)|$. When the graph G is clear from context we omit the subscript.

A *cut vertex* is a vertex whose removal increases the number of connected components. A graph is *biconnected* (or 2-connected) if it is connected and does not contain any cut vertices. The *biconnected components* of a graph G are its maximal biconnected subgraphs. It is well known that the biconnected components of a graph form a partition of its edge set. Similarly, a connected graph is *triconnected* (or 3-connected) if there is no set of at most two vertices whose removal disconnects the graph.

A *separation* of a graph G is a pair (L,R) of subsets of $V(G)$ such that $L \cup R = V(G)$ and there are no edges between $L \setminus R$ and $R \setminus L$ in G. The intersection $L \cap R$ is called the *center* of the separation, and the size $|L \cap R|$ is the *order* of the separation.

Given a path P (cycle C), we refer to the number of edges in P (C) as the length of P (C) and denote it by $|P|$ (respectively $|C|$).

If P is a path from a vertex in X to a vertex in Y, we say that P is an X-Y path. If X contains a single vertex x, we say that P is an x-Y path. Two paths that do not share any vertices are called vertex disjoint. Two paths P_1 and P_2 such that the internal vertices of P_1 are disjoint from P_2 and vice versa are called internally vertex disjoint paths. For a graph G and $T \subseteq V(G)$, a path with both endpoints in T but no internal vertices in T is called a *T-path*. A cycle which intersects T is called a *T-cycle*. A *subset feedback vertex set* of a graph G and *terminal set* $T \subseteq V(G)$ is a set $S \subseteq V(G)$ such that $G - S$ does not contain any T-cycles.

Contracting an edge uv amounts to removing the vertices u and v from the graph and making a new vertex w which is adjacent to all vertices in $N(u) \cup N(v)$. When working with multigraphs the number of edges from w to a vertex x in $N(u) \cup N(v)$ is the number of edges from u to x plus the number of edges from v to x.

Tree decompositions and the Tutte Decomposition. A *tree decomposition* of a graph G is a pair (F,χ), where F is a tree and $\chi : V(F) \to 2^{V(G)}$ is a function such that the following conditions are satisfied:

(1) For each edge $uv \in E(G)$ there is a node $b \in V(F)$ such that $\{u,v\} \subseteq \chi(b)$.
(2) For each $v \in V(G)$ the nodes in $\{b \mid v \in \chi(b)\}$ induce a non-empty connected subtree of F.

The sets $\chi(b)$ for $b \in V(F)$ are called *bags* of the tree decomposition. We will assume that the reader is familiar with tree decompositions and their basic properties. For an introduction to tree decompositions, see [Diestel 2010]. We extend the definition of χ so that it also may take subsets of vertices of F as input. In particular, for $B \subseteq V(F)$ we define $\chi(B) = \bigcup_{b\in B} \chi(b)$. The *width* of a tree decomposition is the maximum size of a bag in the decomposition, minus 1. The tree-width of a graph G is the minimum width of any tree decomposition of G. A useful fact is that a graph G has tree-width at most 1 if and only if G is a forest [Diestel 2010].

It is possible to decompose any graph "in a tree-like fashion" into triconnected parts. While the idea of a triconnected part is similar in spirit to that of a biconnected component, a triconnected part in our context is not necessarily a subgraph of the input graph and is more conveniently stated using tree decompositions. The concepts of *torsos* and *adhesions* are needed to state this decomposition theorem. For a graph G and vertex set $M \subseteq V(G)$, the graph $\mathsf{torso}(G, M)$ has vertex set M. Two vertices u and v have an edge between each other in $\mathsf{torso}(G, M)$ if $uv \in E(G[M])$, or there is a path from u to v in G with all internal vertices in $V(G) \setminus M$. For a tree decomposition (F, χ) of G and edge $ab \in E(F)$ the set $\chi(a) \cap \chi(b)$ is called an *adhesion* of the tree decomposition (F, χ).

Definition 2.1. A *Tutte decomposition* of a connected graph G is a tree decomposition (F, χ) of G with the following properties.

(a) The size of the largest adhesion of (F, χ) is at most 2.
(b) For each $b \in V(F)$ the graph $\mathsf{torso}(G, \chi(b))$ is triconnected.
(c) For each set $B \subseteq V(F)$ such that at most one node in B has a neighbor outside of B, $G[\chi(B)]$ is connected.
(d) There is no pair of distinct nodes $b, b' \in V(F)$ such that $\chi(b) \subseteq \chi(b')$.

When representing a Tutte decomposition in memory, we store some additional information besides the tree and the bags corresponding to the nodes, to allow computations on the decomposition to be performed efficiently. For each edge $ab \in E(F)$, we store the adhesion $\chi(a) \cap \chi(b)$ between the two bags.

PROPOSITION 2.2 ([HOPCROFT AND TARJAN 1973A; GUTWENGER AND MUTZEL 2000]). *There is an algorithm that, given an n-vertex m-edge graph G, runs in time $\mathcal{O}(n + m)$ and outputs a Tutte decomposition (F, χ) of G, where $|V(F)| = \mathcal{O}(n)$.*

We will also need the following fact about Tutte decompositions. Although it follows implicitly from the decomposition given in [Diestel 2010], we give a proof here for the sake of completeness.

PROPOSITION 2.3. *Let (F, χ) be a Tutte decomposition of G and $b \in V(F)$ be a node such that $|\chi(b)| \geq 4$. For every pair of distinct vertices u, $v \in \chi(b)$, either uv are adjacent in G or there are at least 3 vertex disjoint paths from u to v in G.*

PROOF. Suppose u and v appear in the same bag $\chi(b)$ of size at least 4, but that there exists a separation L, R of order at most 2 such that $u \in L \setminus R$ and $v \in R \setminus L$. Pick u, v, L and R and such that $|L \cap R \setminus \chi(b)|$ is minimized. Let $Z = L \cap R$.

If u and v are non-adjacent in $\mathsf{torso}(G, \chi(b))$ then there are 3 vertex disjoint u-v paths in $\mathsf{torso}(G, \chi(b))$, and since $\mathsf{torso}(G, \chi(b))$ can also be obtained from G by vertex deletions, edge deletions and edge contractions, we infer that there are 3 vertex disjoint u-v paths in G as well. Thus u and v are adjacent in $\mathsf{torso}(G, \chi(b))$. Since u and v are non-neighbors in G it follows that there is an edge $bb' \in E(F)$, such that $\chi(b) \cap \chi(b') = \{u, v\}$. Let B be the vertex set of the connected component of $F - bb'$ that contains b', we have that $\chi(B) \cap \chi(b') = \{u, v\}$ and that $G[\chi(B)]$ is connected. Thus $Z \cap \chi(B) \neq \emptyset$.

Let $P = \chi(B)$ and $Q = \chi(V(G) \setminus B)$. Then, (P, Q) is a separation with $P \cap Q = \{u, v\}$ and $\chi(b) \subseteq Q$. Since $Z \cap \chi(B) \neq \emptyset$, it follows that $|Z \cap \chi(b)| \geq 1$ and that there exists a vertex $z \in \chi(b) \setminus (\{u, v\} \cup Z)$. The vertex z is either in $L \setminus R$ or $R \setminus L$. We assume without loss of generality that $z \in L \setminus R$. Consider now the separation $(L \cap Q, R \cup P)$. We have that $(L \cap Q) \cap (R \cup P) = \{u\} \cup (Z \setminus (P \setminus Q))$, and $Z \cap (P \setminus Q) \neq \emptyset$. Therefore, $(L \cap Q, R \cup P)$ is a separation of order at most 2. Furthermore $Z \cap (P \setminus Q) \cap \chi(b) = \emptyset$ while $u \in \chi(B)$. It follows that $|(L \cap Q) \cap (R \cup P) \setminus \chi(b)| < |(L \cap R) \setminus \chi(b)|$. Finally $z \in \chi(b) \cap ((L \cap Q) \setminus ((R \cup P))$ while $v \in \chi(b) \cap ((R \cup P) \setminus (L \cap Q))$. This contradicts the choice of u,v, L and R, thus completing the proof of the proposition. □

2.1. Important Separators

We review important separators, as well as some related results. Let G be a graph, let $X, S \subseteq V(G)$ be vertex subsets. We denote by $R_G(X, S)$ the set of vertices of G reachable from X in the graph $G - S$ and we denote by $NR_G(X, S)$ the set of vertices of $G - S$ which are not reachable from X in the graph $G - S$. We drop the subscript G if it is clear from the context. Let G be a graph and let $X, Y \subset V(G)$ be two disjoint vertex sets. A subset $S \subseteq V(G) \setminus (X \cup Y)$ is called an X-Y *separator* in G if $R_G(X, S) \cap Y = \emptyset$ or in other words there is no path from X to Y in the graph $G - S$. We denote by $\lambda_G(X, Y)$ the size of the smallest X-Y separator in G. If there is an edge in G with one endpoint in X and the other in Y, then $\lambda_G(X, Y) = \infty$. If G is clear from context we omit the subscript.

An X-Y separator S_1 is said to *cover* an X-Y separator S if $R(X, S_1) \supset R(X, S)$. Note that the definition of covering is asymmetric; the reachability set is taken from X and not from Y. We say that an X-Y separator S_1 *dominates* another X-Y separator S if S_1 covers S and $|S_1| \leq |S|$. An X-Y separator is said to be *inclusionwise minimal* if none of its proper subsets is an X-Y separator.

Definition 2.4. Let G be a graph, $X, Y \subset V(G)$ be disjoint vertex sets and $S \subseteq V(G)$ be an X-Y separator in G. We say that S is an *important* X-Y separator if it is inclusionwise minimal and there does not exist another X-Y separator S_1 such that S_1 dominates S.

Important separators were first defined by Marx [Marx 2006], and have found numerous applications since then [Chen et al. 2008; Lokshtanov and Ramanujan 2012; Marx and Razgon 2014; Razgon and O'Sullivan 2009]. The crucial property of important separators that has led to this widespread use is the fact that there are not too many important X-Y separators of small size. This is formalized in the following statement.

LEMMA 2.5 ([CHEN ET AL. 2009; MARX AND RAZGON 2014]). *The number of important X-Y separators of size at most k is at most $2^{2k-\lambda(X,Y)}$, where $\lambda(X, Y)$ denotes the size of the smallest X-Y separator in G. The set of important X-Y separators of size at most k can be enumerated in time $\mathcal{O}(4^k k(m + n))$.*

The following proposition is a crucial component of the proof of Lemma 2.5, and also features prominently in several arguments using important separators.

PROPOSITION 2.6 ([CHEN ET AL. 2009; MARX AND RAZGON 2014]). *For any disjoint X and Y such that there is no edge from X to Y there is a unique important X-Y separator A of size $\lambda(X, Y)$. Furthermore every important X-Y separator $A' \neq A$ covers A. Finally, there is an algorithm that given G, X, Y and k, runs in time $\mathcal{O}(k(n + m))$ and either correctly concludes that $\lambda(X, Y) > k$ or returns A.*

At certain points in our algorithm, we do not need to find the important separator of the minimum size and it is sufficient to compute *a* minimum separator. For this, we use the following proposition which follows from the classic Max-Flow algorithm of Ford and Fulkerson [Ford and Fulkerson 1956] and provides a witness that the given separator actually has the minimum possible size.

PROPOSITION 2.7. *Given a graph G with n vertices and m edges, disjoint sets $X, Y \subseteq V(G)$, and an integer k, there is an algorithm that runs in time $\mathcal{O}(k(n+m))$ and either correctly concludes that there is no (X, Y)-separator of size at most k, or returns a* *minimum (X,Y)-separator Δ and a collection of $|\Delta|$ pairwise internally vertex-disjoint $X-Y$ paths.*

We now prove a "sampling" analogue of Lemma 2.5. The proof of this lemma is an easy adaptation of the proof of Lemma 2.5.

LEMMA 2.8. *There is a randomized algorithm* SampleImp *that given as input a graph G, disjoint and non-adjacent vertex sets X and Y integer k, and rational* deletion probability $0 < p < 1$, *runs in time $k^{\mathcal{O}(1)}(m+n)$ and outputs an X-Y separator S of size at most k or* fail. *For each important X-Y separator A of size at most k, the probability that $S = A$ is at least $p^{|A|}(1-p)^{|A|-\lambda(X,Y)}$.*

PROOF. On input (G, X, Y, k, p) the algorithm first checks whether there is a path from X to Y in G. If no such path exists the algorithm returns $\emptyset$. If there is a path from X to Y but $k = 0$ the algorithm returns fail. Now the algorithm either correctly concludes that no X-Y separator S of size at most k exists, or computes the unique X-Y important separator Z of size $\lambda(X,Y)$ by invoking Proposition 2.6. Given Z the algorithm picks an arbitrary vertex $z \in Z$. With probability p the algorithm proceeds as follows: first it makes a recursive call to SampleImp$(G-z, X, Y, k-1, p)$. Let S' be the set returned by the recursive call, the algorithm returns $\{z\} \cup S'$. Otherwise (i.e with probability $1-p$) the algorithm proceeds as follows: first it makes a recursive call to SampleImp$(G-z, X \cup \{z\}, Y, k-1, p)$. Let S' be the set returned by the recursive call, the algorithm returns S'. If the recursive calls fail to return a set, the algorithm also fails.

First, observe that the algorithm terminates because in each recursive call either the number of vertices of G decreases or $n - |X|$ decreases. Now we show that if the algorithm returns a set S, then this set is always an X-Y separator in G. We show this by an induction on the recursion depth of the algorithm. In the base case the algorithm only returns $\emptyset$ if there are no X-Y paths. Consider now a recursive call to the algorithm that makes at least one recursive call and then returns a set S. If the algorithm returns $S = S'$ then we know by the induction hypothesis that $S = S'$ is an $X \cup \{z\}$-Y separator in G. If the algorithm returns $S = S' \cup \{z\}$ then it follows from the induction hypothesis that S' is an X-Y separator in $G - z$, implying that S is an X-Y separator in G.

Thus, what is left is to prove that for every important X-Y separator A of size at most k, the probability that $S = A$ is at least $p^{|A|}(1-p)^{|A|-\lambda(X,Y)}$. We prove the statement by induction on $2k - \lambda(X,Y)$. If $2k - \lambda(X,Y) < 0$ then $\lambda(X,Y) > k$ and there is no X-Y separator of size at most k. If $2k - \lambda(X,Y) = 0$ then it must be the case that $k = 0$ and the algorithm outputs $\emptyset$ which is the unique important X-Y separator (of any size). Consider now the execution of the algorithm on an instance (G, X, Y, k, p), let $\lambda = \lambda(X,Y)$ and assume that the statement holds for all smaller values of $2k - \lambda$. We distinguish between the following two cases. Either the vertex $z \in Z$ that the algorithm picks is in A or it is not.

If $z \in A$ then, with probability p the algorithm makes the recursive call SampleImp$(G-z, X, Y, k-1, p)$. Suppose that it does. In $G - z$, $A \setminus z$ is an important (X,Y) separator and $\lambda_{G-z}(X,Y) \geq \lambda_G(X,Y) - 1$. Thus, by the induction hypothesis the probability that the recursive call outputs $A \setminus z$ is at least $p^{k-1}(1-p)^{k-1-(\lambda-1)} = p^{k-1}(1-p)^{k-\lambda}$. Since the probability that G makes the recursive call is p it follows that the probability that G outputs A is at least $p \cdot p^{k-1}(1-p)^{k-\lambda} = p^k(1-p)^{k-\lambda}$.

If $z \notin A$ then, by Proposition 2.6 A covers Z and therefore z is reachable from X in $G - A$. Thus A is an important $(X \cup \{z\})$-Y separator. Furthermore, since Z is the unique important (X,Y) separator of size at most λ, it follows that Z covers all other (X,Y) separators of size at most λ. Thus there can not be any $(X \cup \{z\})$-Y separators in G of size at most λ, implying $\lambda_G(X \cup \{z\}, Y) \geq \lambda + 1$. Thus, by the induction hypothesis the recursive call outputs A with probability at least $p^k(1-p)^{k-(\lambda+1)}$. Since the algorithm makes the recursive call with probability $(1-p)$ it follows that the probability that G outputs A is at least $(1-p) \cdot p^k(1-p)^{k-(\lambda+1)}$.

All that remains is to bound the running time of the algorithm. The algorithm makes at most one recursive call, which means that the recursion tree is actually a path. In each node of the recursion tree the most expensive operation is the call to Proposition 2.6, which takes $\mathcal{O}(k(n+m))$ time. Furthermore the recursion depth is at most $2k - \lambda \leq 2k$, since in each recursive call either k decreases and λ decreases by at most 1, or λ increases by 1 while k remains unchanged. Thus the running time of the algorithm is $k^{\mathcal{O}(1)}(n+m)$, as claimed. □

3. PREPROCESSING

In this section we describe some reduction rules that are applied on the input instance (G, T, k) of SUBSET FVS. These rules will be useful both for the randomized and for the deterministic algorithms. For Rules 1, 2 and 3 we execute each rule exhaustively before proceeding to the application of the next rule. After a rule has been applied exhaustively and the reduction proceeds to the next rule, the procedure does *not* look for possibilities to apply the previous rules. After Rule 3 has been applied the reduction algorithm will try to apply either one of Rule 4, Rule 5 or Rule 6, until none of these rules can be applied. The algorithm will not look for opportunities to apply Rules 1, 2 and 3 after it is done with Rule 3. The only exception is that in some cases when Rule 6 is applied, the algorithm re-starts the reduction procedure from Rule 1. This is discussed in more detail later in the section.

PREPROCESSING RULE 1. *If an edge e is not contained in any T-cycle, then remove e.*

Any solution before e is deleted is a solution after, and vice versa. Hence Rule 1 is safe. It is not immediately obvious how to apply Rule 1 exhaustively in linear time. To that end we will use the following lemma.

LEMMA 3.1. *An edge e is contained in a T-cycle if and only if e is not a bridge and the biconnected component of G containing e contains a terminal.*

PROOF. For the forward direction, suppose $e = uv$ is contained in a T-cycle C. Then e is not a bridge, as we can go from u to v in $G - e$ along C. Further C contains a terminal t. If $u \neq t$ then C gives two disjoint paths from u to t. Similarly, if $v \neq t$ there are two disjoint paths from v to t. Hence u, v and t are in the same biconnected component.

For the reverse direction, suppose the edge uv is not a bridge, and the biconnected component of G containing e also contains a terminal t. If $t = u$ or $t = v$ then we can complete uv to a T-cycle by going back from v to u in $G - e$, since e is not a bridge. If $t \notin \{u, v\}$, then by Menger's theorem there are two paths from t to $\{uv\}$ that intersect only in t. These paths, together with the edge uv form a T-cycle. This completes the proof of the lemma. □

Using Lemma 3.1 we may apply Rule 1 exhaustively in linear time.

LEMMA 3.2. *Rule 1 can be applied exhaustively in time $\mathcal{O}(n+m)$.*

PROOF. We may partition $E(G)$ into the edge sets of its biconnected components in linear time [Hopcroft and Tarjan 1973b]. For each biconnected component C that does not contain any terminals, we remove all edges in C. If a biconnected component C contains exactly one edge e, then e is a bridge and can be removed. Note that removing all the edges from a biconnected component leaves all the other biconnected components untouched (and biconnected). Hence one pass of this algorithm over all biconnected components applies Rule 1 exhaustively. $\square$

Removing edges can create vertices of degree 0, which we can remove, since they do not appear in any T-cycle. Note that after we apply Rule 1 exhaustively, there are no vertices of degree 1, as these would be incident to a bridge.

PREPROCESSING RULE 2. *Delete vertices of degree* 0*.*

After Rule 1 and Rule 2 have been applied exhaustively, every vertex is in a T-cycle. Indeed, every vertex is in a T-cycle if and only if there are no vertices of degree 0 and every edge is in a T-cycle.

Observe that if every non-terminal vertex v is in a T-cycle, there cannot exist any separation (L, R) of order at most 1 such that $L \cap T = \emptyset$ and $v \in L \setminus R$. In such a case v cannot be in any T-cycle. We now give a reduction rule for such separations of order 2.

PREPROCESSING RULE 3. *If there exists a separation (L, R) of G such that $G - R$ and $G[L]$ are connected, $L \cap R = \{u, v\}$, $L \cap T = \emptyset$ and $L \setminus R \neq \emptyset$, then delete all vertices in $L \setminus R$ and add the edge uv.*

Observe that if a separation (L, R) of G satisfying the prerequisites of Rule 3 exists, and every vertex is in a T-cycle, then $G[R]$ must be connected (otherwise there is a vertex in L which is not part of any T-cycle).

LEMMA 3.3. *Rule 3 is safe.*

PROOF. Let G' be the graph output by Rule 3. Since G' can be obtained from G by contracting all edges in $G[L]$ (except uv), if (G, T, k) is a "yes" instance then so is (G', T, k).

For the reverse direction, assume there exists a subset feedback vertex set S such that $G' - S$ has no T-cycle. We claim that $G - S$ has no T-cycle either. Suppose for contradiction that $G \setminus S$ contains a T-cycle C. Then C must contain a terminal t, which is in $R \setminus L$ and a vertex $p \in (V(G') \setminus V(G))$, hence $p \in L \setminus R$. But then, since (L, R) has order 2, C must contain both u and v. In G' there is an edge from u to v and thus $G' \setminus S$ contains a T-cycle, a contradiction. $\square$

Observe that Rule 3 maintains the property that every vertex is in a T-cycle. Indeed, any T-cycle C which contains a deleted vertex must contain both u and v, which have an edge in $e \in E(G') \setminus E(G)$. Adding e to C and removing all vertices of $V(G) \setminus V(G')$ from C yields a T-cycle C' in G' which contains all vertices of $C \cap V(G')$.

We say that a non-terminal vertex v is *reducible* by Rule 3 if there exists a separation (L, R) satisfying the conditions of Rule 3, such that $v \in L \setminus R$. In other words v would be deleted if we applied Rule 3 on (L, R). A vertex which is not reducible by Rule 3 is *irreducible* by the same rule. Showing that Rule 3 can be applied exhaustively in linear time is non-trivial. To achive this goal we will exploit the Tutte decomposition.

LEMMA 3.4. *Rule 3 can be applied exhaustively in linear time.*

PROOF. From G, make a graph G' by adding a new vertex τ and making τ adjacent to all vertices in T. Let $T' = T \cup \{\tau\}$. Compute a Tutte decomposition (F, χ) of G' in linear time, using Proposition 2.2. Let B be the set of all nodes b in $V(F)$ such that $\chi(b) \cap T' \neq \emptyset$. Since $G'[T']$ is connected the properties of tree decompositions ensure that $F[B]$ is a connected subtree of F.

Consider an edge $ab \in E(F)$ such that $a \notin B$ and $b \in B$. Removing the edge ab from F produces two trees, F_a and F_b with $a \in V(F_a)$ and $b \in V(F_b)$. Set $L = \chi(V(F_a))$ and $R = \chi(V(F_b)) \setminus \{\tau\}$. We claim that Rule 3 can be applied on (L, R).

By the properties of tree decompositions $(L, R \cup \{\tau\})$ is a separation of G' and so (L, R) is a separation in G. Further, by the properties of Tutte decompositions, the order of (L, R) is at most 2. Also, since no bag in a Tutte decomposition is a subset of another bag, it must be the case that $L \setminus R \neq \emptyset$. Furthermore, since $B \cap V(F_a) = \emptyset$ it follows that $T \cap L = \emptyset$. Let $p \in L \setminus R$. Since p is in some T-cycle the order of the separation (L, R) must be at least 2 and hence it must be exactly 2. Thus (L, R) satisfies the prerequisites of Rule 3.

Let $L \cap R = \{u, v\}$. We apply Rule 3 on (L, R), i.e delete $L \setminus R$ and add the edge uv in G. When applying the reduction rule to (L, R) we update the graph G' as well by removing the vertices in $L \setminus R$ and adding the edge uv. Removing the subtree F_a from F yields a Tutte decomposition of the updated G' since adding the edge uv ensures that $\mathsf{torso}(G', \chi(b))$ stays triconnected, even after $L \setminus R$ is deleted. After performing the rule, updating G' and the Tutte decomposition of G', we may again look for an edge $ab \in V(F)$ such that $a \notin B$ and $b \in B$, and repeat the reduction process until no such edge exists. This can clearly be done in linear time.

Suppose now that the algorithm has reduced all edges $ab \in V(F)$ such that $a \notin B$ and $b \in B$, and that such edges no longer exist. Since $B \neq \emptyset$ it follows that $B = V(F)$. Define $Q = \{b \in V(F) \mid |\chi(b)| \geq 4\}$ and let $U = \bigcup_{b \in Q} \chi(b) \setminus T'$. We claim that all vertices in U are irreducible by Rule 3.

Suppose for contradiction that there exists a separation (L, R) of G that satisfies the conditions of Rule 3, but that $(L \setminus R) \cap U \neq \emptyset$. $(L, R \cup \{\tau\})$ is a separation of G' of order at most 2. Furthermore $T' \subseteq R \setminus L$. Let p be a vertex in $(L \setminus R) \cap U$. Since p is in U, it follows that there exists a node $b \in V(F)$ such that $\chi(b) \geq 4$ and $p \in \chi(b)$. Since every bag contains a node from T' there is a node $t' \in \chi(b) \cap T'$. Now, because $p \in L \setminus R$ and $t' \in (R \cup \{\tau\}) \setminus L$ it follows that p and t are non-adjacent, and hence, By Proposition 2.3 there are three vertex disjoint paths from p to t in G'. But each of these paths must intersect with $L \cap (R \cup \{\tau\}) = L \cap R$, and $|L \cap R| \leq 2$ yielding the desired contradiction. We conclude that for each vertex $p \in U$, p is irreducible.

Let $P = V(G) \setminus (U \cup N[T])$. Every reducible vertex is in P. We claim that $G[P]$ is a forest. We define a function $\chi' : V(F) \to 2^P$ as follows: $\chi'(b) = \chi(b) \cap P$. (F, χ') is a tree decomposition of $G[P]$. Since every node b of F such that $\chi(b) \cap P \neq \emptyset$ satisfies $|\chi(b)| \leq 3$, and $\chi(b) \cap T' \neq \emptyset$, it follows that $|\chi'(b)| \leq 2$. Thus the tree-width of $G[P]$ is at most 1 and hence $G[P]$ is a forest.

Next we show that every vertex $p \notin P$ has at most one neighbor in a connected component of $G[P]$. For a contradiction, suppose otherwise. Then $G'[P \cup p]$ contains a cycle C, and therefore the Tutte decomposition must contain a bag b such that $|\chi(b) \cap C| \geq 3$. But $\chi(b)$ contains a vertex from P, and hence $|\chi(b)| \leq 3$. But then $\chi(b)$ may not also contain a vertex in T', contradicting that all bags do.

We prove that at this point, every vertex with at least three neighbors is irreducible. Consider a vertex $p \in P$ such that $|N(P)| \geq 3$, and suppose that there is a separation (L, R) satisfying the conditions of Rule 3, such that $p \in L \setminus R$. Let $L \cap R = \{u, v\}$. Since

$L \setminus R \subseteq P$ it follows that $G[L \setminus R]$ is a tree. Let ℓ be the number of leaves of the tree except for p, and r be the number of neighbors of p in $L \cap R$. Since p has degree at least 3 it follows that $\ell + p \geq 3$. Each of the leaves of the tree are in some T-cycle, and hence they all are adjacent to at least one of u or v. Thus there are at least 3 edges between $\{u, v\}$ and $L \setminus R$. Without loss of generality u is incident to at least two of these edges. If $u \in P$ this contradicts that $G[P]$ is a forest, because $G \setminus R$ is connected. If $u \notin P$ this contradicts that a vertex outside of P may not have two neighbors in the same component of $G[P]$.

It follows that the only remaining reducible vertices have two neighbors, both non-terminals. For each such vertex p we can apply Rule 3 on the separation $(N[p], V(G) \setminus \{p\})$. This amounts to deleting p and adding an edge between p's neighbors. Note that this operation may never turn an irreducible vertex v reducible, since it cannot decrease the number of vertex disjoint paths from v to T. Applying Rule 3 on all non-terminals with two non-terminal neighbors can be done in linear time, and after this there are no reducible vertices left. This concludes the proof. $\square$

If Rule 3 cannot be applied it follows that every non-terminal vertex is either adjacent to a terminal or has at least three internally vertex-disjoint paths to T. All the following reduction rules maintain this property, unless explicitly stated otherwise.

PREPROCESSING RULE 4. *Short-circuit every degree 2 non-terminal and if a degree 2 terminal t has a terminal neighbor t', contract the edge tt'.*

The correctness of Reduction Rule 4 for non-terminals follows from the observation that if there is a vertex v of degree 2 then every cycle, and in particular every T-cycle must pass through both of v's neighbors. Thus there exists an optimal solution which avoids v, and hence it is safe to contract any edge incident to v. If t is a terminal of degree 2 and t has a terminal neighbor t', then any subset feedback vertex set that contains t could just as well have contained t' instead. Thus we can contract the edge tt'. Note that an application of Rule 4 may create multi-edges. We would like to get rid of multiple edges between the same two vertices. Though we cannot safely remove all of them, we can indeed remove the ones not incident to a terminal, and ensure that there are at most three edges between any pair of vertices.

PREPROCESSING RULE 5. *If there are at least 3 edges between any two vertices, delete all but two of the edges. If there are two edges between two non-terminals, delete one of the edges.*

The correctness of Reduction Rule 5 follows from the fact that no T-cycle contains both of the edges between two non-terminals, or more than two edges incident to the same vertex. The reason we keep double edges between a terminal and a non-terminal is that these constitute T-cycles all by themselves.

PREPROCESSING RULE 6. *Remove self-loops on non-terminals. If a terminal t has a self-loop remove t from the graph and decrease k by 1.*

A self-loop on a non-terminal is never a part of a T-cycle and may be removed. In addition, terminal with a self-loop must be in every subset feedback vertex set. It is important to note and briefly discuss the following possible consequence of an application of Rule 6.

Remark 3.5. An application of Rule 6 may cause a vertex to no longer be part of any T-cycle, or make a vertex which is irreducible by Rule 3, reducible. In this case we need to re-start the reduction procedure from Rule 1. However, an application of Rule 6 on a terminal decreases k. Therefore, such a re-start can only happen at most k times.

LEMMA 3.6. *There exists a $\mathcal{O}(k(n+m))$ time algorithm* **Reduce** *that given as input an instance (G,T,k), returns an equivalent instance (G',T',k') with the following properties.*

— $|V(G')| \leq |V(G)|$, $|E(G')| \leq |E(G)|$ *and* $k' \leq k$.
— *Every non-terminal vertex has degree at least* 3.
— *Every terminal vertex has degree at least* 2.
— *Every terminal vertex of degree* 2 *has only non-terminal neighbors.*
— *Every vertex is in a T-cycle.*
— *For every non-terminal v, either v is adjacent to a terminal or there are at least three internally vertex disjoint paths to T.*
— *Between any pair of vertices there are at most two edges.*
— *Between any pair of non-terminals there is at most one edge.*

PROOF. The algorithm applies the reduction rules in order. That is, it first applies 1 exhaustively using Lemma 3.2. Then it applies Rule 2 exhaustively in linear time. After this all vertices are in a T-cycle and all rules that will be applied, except possibly for Rule 6 maintain this property. The algorithm now applies Rule 3 exhaustively in linear time using Lemma 3.4. After this the algorithm applies Rules 4 and 5 exhaustively. Note that applications of Rule 5 may create new possibilities to apply Rule 4 and vice versa, but both rules maintain the two properties that (a) every vertex is in a T-cycle and (b) every vertex is irreducible by Rule 3. If at any point a vertex with a self-loop is created then the algorithm immediately executes Rule 6. If a self-loop is removed from a non-terminal, this preserves both property (a) and (b). If the rule removes a terminal then k is decreased and the algorithm restarts the reduction procedure from scratch, starting with Rule 1. Since this can happen at most k times the total running time of the algorithm is $\mathcal{O}(k(n+m))$ as claimed.

Consider now an instance after the reduction procedure is finished. Then, since every vertex is in a T-cycle every vertex has degree at least 2. Since non-terminals of degree 2 are reduced by Rule 4 there are no non-terminals of degree 2. Since terminals of degree 2 with at least one terminal neighbor are reduced by Rule 4 every terminal vertex of degree 2 has only non-terminal neighbors. After we have exaustively applied Rule 3 we have that every non-terminal v, either v is adjacent to a terminal or there are at least three internally vertex disjoint paths to T. All the rules after this maintain this property. The only exception is Rule 6 applied to terminals, but if this rule is ever applied the reduction restarts from scratch. Hence, for every non-terminal v, either v is adjacent to a terminal or there are at least three internally vertex disjoint paths to T. The two last properties are ensured by Rule 5. □

We will refer to instances satisfying the conclusions of Lemma 3.6 as *reduced instances*.

3.1. Structural Properties of Reduced Instances

Before commencing with the algorithms we need to make some observations on the structure of reduced instances that do not contain double edges. Even though reduced instances may contain double edges, these are always incident to a terminal and constitute T-cycles all by themselves. Since every subset feedback vertex set must contain at least one of the two endpoints of such a double edge, our algorithms will quickly get rid of double edges by branching. We start by inspecting the structure of instances that do not contain any T-cycles.

LEMMA 3.7. *Let G be a graph and let $T \subseteq V(G)$ be a set such that G has no T-cycles. Then (a) $G[T]$ is a forest, (b) for every connected component C of $G - T$ and connected component C_T of $G[T]$ there is at most one edge between C and C_T, and (c) contracting all edges of which both endpoints are non-terminals yields a forest.*

PROOF. For (a) observe that any cycle in $G[T]$ is a T-cycle. For (b) observe that if there are two edges between C and C_T then $G[C \cup C_T]$ contains a T-cycle.

For (c), suppose that contracting all edges with both endpoints non-terminals yields a graph containing a cycle C. Every edge of C must have at least one endpoint being a terminal, and thus C contains a terminal. But then C corresponds to a T-cycle in G, a contradiction. □

Let G be a graph and let T be a subset of $V(G)$ such that G does not contain a T-cycle. Next we define the notion of a terminal forest. While the definition might look technical at a first glance, a terminal forest is just the forest obtained from G by contracting all the edges with both endpoints non-terminals, rooting the trees in the forest at arbitrary roots and providing a function χ that maps each vertex v of the forest to the vertex set in G which was contracted into v. In order to keep the presentation simple, in the next definition, we misuse notation by treating the function χ as one that can output either vertex sets (non-terminals) or single vertices (terminals).

Definition 3.8. Let G be a graph and let T be a subset of $V(G)$ such that G does not contain a T-cycle. A *terminal forest* of G is a pair (F, χ), where F is a forest of rooted trees and $\chi : V(F) \to \{T\} \cup 2^{V(G)\setminus T}$ is a function with the following properties:

(a) Each tree T_i in F can be associated to a unique connected component C_i in G; for each $b \in V(T_i)$, $\chi(b) \subseteq C_i$.
(b) $\cup_{b\in V(F)}\chi(b) = V(G)$ and for any pair of vertices $b, b' \in V(F)$, $\chi(b) \cap \chi(b') = \emptyset$. That is, χ partitions the vertex set $V(G)$.
(c) For every $b \in V(F)$, $\chi(b) \neq \emptyset$ and the graph $G[\chi(b)]$ is connected.
(d) For every edge $uv \in E(G)$ such that $u, v \notin T$ there exists $b \in V(F)$ such that $u, v \in \chi(b)$.
(e) For every edge $tv \in E(G)$ such that $t \in T$ and $v \in V(G)$ there exists a $b_t \in V(F)$ and $b_v \in V(F)$ such that $\chi(b_t) = t$, $v \in \chi(b_v)$ and $b_t b_v \in E(F)$.

From Lemma 3.7 it follows directly that every graph G that does not have a T-cycle has a terminal forest. In addition, note that there can be multiple terminal forests for a single graph. This is because F is a forest of rooted trees and technically, this implies a different terminal forest for every different rooting of the trees in F.

For a terminal forest (F, χ) of G we define the function $\chi^- : V(G) \to V(F)$ so that $\chi^-(u)$ is the unique node $b \in V(F)$ so that $u \in \chi(b)$. We extend χ^- to vertex sets of G in the following manner: $\chi^-(S) = \bigcup_{u\in S} \chi^-(u)$.

As mentioned earlier, Definition 3.8 abuses notation - χ can output either vertex sets (non-terminals) or single vertices (terminals). When $\chi(b)$ outputs a terminal t it is sometimes convenient to treat it as the set $\{t\}$ containing the terminal. In the forest F the child-parent and descendant-ancestor relations are well defined. We can extend these relations to vertices in G in a natural way; u is a child/parent/ancestor/descendant of v if and only if $\chi^-(u)$ is a child/parent/ancestor/descendant of $\chi^-(v)$. We can also extend the relations to allow comparisons between a vertex u in the graph G and a node b in F, in particular u is a child/parent/ancestor/descendant of b if and only if $\chi^-(u)$ is. We will refer to the nodes b of F such that $\chi(b) \in T$ as *terminal nodes*, and to the other nodes as *non-terminal* nodes.

OBSERVATION 3.1. *Let (F, χ) be a terminal forest of a graph G that has no T-cycles. There is no $bb' \in E(F)$ such that both b and b' are non-terminal nodes.*

PROOF. Suppose such an edge bb' exists, then $\chi(b)$ and $\chi(b')$ are subsets of the same connected components of G, and therfore there must be an edge $uv \in E(G)$ such that $\chi^-(u) = b$ and $\chi^-(v) = b'$. But this contradicts property (d) of terminal forests. $\square$

OBSERVATION 3.2. *Let (F, χ) be a terminal forest of a graph G that has no T-cycles. Let $bb' \in E(F)$. There is exactly one edge between $\chi(b)$ and $\chi(b')$.*

PROOF. Suppose there is no edge between $\chi(b)$ and $\chi(b')$. Let C be the vertex set of the component of F containing bb', then $G[\chi(C)]$ is disconnected, contradicting Property (a) of terminal forests. Suppose there are at least two edges. By Observation 3.1 at least one of b and b' is a terminal node, without loss of generality it is b. Let $t = \chi(b)$. Suppose t has at least two edges to $\chi(b')$. Since $G[\chi(b')]$ is connected there is a path between the two edge endpoints in $G[\chi(b')]$. But this path together with t forms a T-cycle in G, yielding a contradiction. $\square$

Definition 3.9. Let G be a graph and $T \subseteq V(G)$ be such that G has no T-cycles. We say that a terminal $t' \in T$ is an *effective descendant* of $t \in T \setminus \{t'\}$ with respect to a terminal forest (F, χ) of G if we have that $\chi^-(t)$ is a descendant of $\chi^-(t')$ and there is a path from $\chi^-(t)$ to $\chi^-(t')$ in F with internal vertices disjoint from $\chi^-(T)$. We drop the explicit reference to (F, χ) if it is clear from the context.

To better understand the definition of effective descendants it is helpful to construct the *effective descendant graph*. This is a directed graph with vertex set T. Each vertex in T has arcs to all of its effective descendants.

It is easy to see that the effective descendant graph is obtained from F by contracting, for all terminals t, all edges to t's non-terminal children, and then orienting edges from parents to descendants in F. This leads to the following observation.

OBSERVATION 3.3. *The effective descendant graph is a forest of rooted trees.*

Definition 3.10. Let G be a graph with no T-cycles. A terminal t is called *good* with respect to a terminal forest (F, χ) if t has at most one effective descendant with respect to a terminal forest (F, χ). We drop the explicit reference to (F, χ) when it is clear from the context.

The following observation follows from the fact that any rooted tree has at least as many leaves as vertices with at least two children.

OBSERVATION 3.4. *Let G be a graph with no T-cycles. At least $|T|/2$ terminals are good.*

Finally we prove a lemma about the structure of reduced instances, and how a potential subset feedback vertex set interacts with the rest of the graph.

LEMMA 3.11. *Let (G, T, k) be a reduced instance and S be a subset fedback vertex set of G. Let (F, χ) be a terminal forest of $G \setminus S$. Then the following holds.*

— *For every leaf b of F, $\chi(b)$ has at least one neighbor in S.*
— *For every non-terminal leaf b in F such that $\chi(b)$ has exactly one neighbor in S, this neighbor is a terminal.*
— *For every non-terminal node $b \in V(F)$ that has degree 2 in F, $\chi(b)$ has at least one neighbor in S.*

PROOF. Consider a leaf b of F. If b is a terminal node then $\chi(b) = t$, and since t has degree at least 2, t must have a neighbor in S. If b is a non-terminal leaf then b either has a parent b' in F or it does not.

We first argue the case when b does not have a parent in F. Then, as G is connected, $\chi(b)$ has a neighbor in S. If $\chi(b)$ only has one neighbor v in S then v separates $\chi(b)$ from all terminals. Thus, unless v is a terminal none of the vertices in $\chi(B)$ are part of any T-cycles. We conclude that v is a terminal. On the other hand, suppose that b has a parent b' in F. Then, by Observation 3.1 b' must be a terminal node. Let $t = \chi(b')$. By Observation 3.2 t has a unique neighbor in $\chi(b)$, call this neighbor v. Since v is in a T-cycle, $\chi(b)$ does not contain any terminals and t only has one edge to $\chi(b)$, $\chi(b)$ must have at least one more neighbor. Moreover, this neighbor must be in S. Suppose now that $\chi(b)$ has exactly one neighbor in S say w. Suppose for contradiction that w is a non-terminal. The vertex v has degree 3, it has exactly one edge to t and at most one edge to w. Thus v must have at least one more neighbor $v' \in \chi(b)$. But then v and w are non-terminals that separate v' from T in G. Thus v' is reducible by Rule 3, contradicting that G is reduced. Thus w must be a terminal.

Finally consider a non-terminal node $b \in V(F)$ that has degree 2 in F, and let b_1 and b_2 be the neighbors of b. By Observation 3.1 both b_1 and b_2 are terminal nodes, and by Observation 3.2 there are exactly two edges from $\chi(b_1) \cup \chi(b_2)$ to $\chi(b)$. Suppose now that $\chi(b)$ contains a vertex u which is not incident to either of these edges. If u is adjacent to a terminal, this terminal is in S. Otherwise, since u is irreducible by Rule 3 it follows that there are three internally vertex disjoint paths from u to T. At most two of these paths intersect with $\chi(b_1) \cup \chi(b_2)$, implying that the third must intersect with $S \cap N(\chi(b))$ proving that $S \cap N(\chi(b))$ is non-empty. Suppose now that every vertex of $\chi(b)$ is incident to an edge to $\chi(b_1) \cup \chi(b_2)$. Then $|\chi(b)| \leq 2$. If $\chi(b) = \{v\}$ then v has degree at least 3 (by Lemma 3.6). At most two of the edges incident to v have an endpoint in $\chi(b_1) \cup \chi(b_2)$, which implies that at least one must have an endpoint in S. If $|\chi(b)| = 2$, let $\chi(b) = \{u, v\}$ where u is incident to the edge to $\chi(b_1)$ and v is incident to the edge to $\chi(b_2)$. Since u has degree at least 3, u must have a neighbor in S. $\square$

4. A RANDOMIZED LINEAR TIME ALGORITHM FOR SUBSET FVS

We now describe the randomized algorithm for SUBSET FVS. The algorithm, called SolveSFVS, is given in Algorithm 1. The algorithm runs in time $k^{\mathcal{O}(1)}(n + m)$ and has one-sided error. Except for applying Lemma 3.6, whenever the algorithm decreases k by x it also removes x vertices from the graph. Thus, whenever the algorithm outputs succeed, the input instance is a "yes" instane. The difficult part is to show that if the instance is a "yes" instance then the algorithm returns succeed with probability at least γ^k, for a constant $\gamma = \frac{1}{25.6}$. The remaining part of the analysis is essentially devoted to the following lemma and its proof.

LEMMA 4.1. *If (G, T, k) is a "yes" instance, then Algorithm* SolveSFVS *outputs* succeed *on (G, T, k) with probability at least γ^k for $\gamma = \frac{1}{25.6}$.*

The algorithm SolveSFVS makes use of some probability constants, $0 < \alpha_t, \alpha_v, \beta, p < 1$. These constants are set so as to maximize the success probability of the algorithm.

ALGORITHM 1: Algorithm SolveSFVS for SUBSET FVS

```
Input  : An instance $(G,T,k)$ of SUBSET FVS.
Output: succeed if the algorithm has found a subset feedback vertex set in $G$ of size
        at most $k$, or fail.
$(G,T,k) \leftarrow$ Reduce$(G,T,k)$

if $G$ has no $T$-cycles and $k \geq 0$ then return succeed
if $k \leq 0$ then return fail
if there exists a double edge $uv$ then
    pick $x$ from $\{u,v\}$ uniformly at random.

    return SolveSFVS$(G-x, T \setminus \{x\}, k-1)$

pick $t$ from $T$ uniformly at random.

with probability $(1-\alpha_t)$:
    return SolveSFVS$(G-t, T \setminus \{t\}, k-1)$


if $|N(t) \cap T| \geq 2$ then
    with probability $\frac{1}{2}$:
        pick $z$ from $N(t) \cap T$ uniformly at random.
        return SolveSFVS$(G-z, T \setminus \{z\}, k-1)$


    if $N(t) \setminus T = \emptyset$ then return fail
pick $v$ from $N(t) \setminus T$ uniformly at random.
with probability $(1-\alpha_v)$:
    return SolveSFVS$(G-v, T \setminus \{v\}, k-1)$


let $\bar{T} = \{t\}$
insert each $z \in T \setminus \{t\}$ with probability $1-\beta$.
let $G' = G - \{vt \in E(G) \mid t \in \bar{T}\} + (\tau, \{\tau t \mid t \in T\})$
$A^\star \leftarrow$ SampleImp$(G', v, \bar{T} \cup \{\tau\}, k+1, p)$
if $|N(v) \cap \bar{T}| \geq 2$ or $A^\star \setminus T = \emptyset$ or $|A^\star| = 1$ then
    return SolveSFVS$(G - A^\star, T \setminus A^\star, k - |A^\star|)$
pick $y^\star$ from $A^\star \setminus T$ uniformly at random.
return SolveSFVS$(G - (A^\star \setminus \{y^\star\}), T \setminus (A^\star \setminus \{y^\star\}), k - |A| + 1)$
```

The exact values of the constants are not important for any of the arguments, except for the final calculations. As a result, for the sake of simplicity, we provide the exact values only when we arrive at the final calculations and the reader may think of all the constants as $\frac{1}{2}$, and consider γ as 256 until arriving at the final calculations, where we give the exact values which are tweaked in order to optimize the success probability.

We will prove Lemma 4.1 by induction on k. Towards this we analyze the algorithm when run on a "yes" instance (G,T,k). Let S be a subset feedback vertex set of G of size at most k, and let (F,χ) be a terminal forest of $G \setminus S$.

The algorithm starts by reducing the instance using Lemma 3.6 and performing some sanity checks; if G already does not have any T-cycles we are done, on the other hand if there are T-cycles but $k \leq 0$ the input instance is a "no" instance (since we assumed the input instance was a "yes" instance, this does not happen). This lets us assume that the input instance is reduced, that $k \geq 1$ and that $S \neq \emptyset$.

Now, if G has a double edge uv then either u or v is a terminal, since G is reduced by Lemma 3.6. Thus at least one of u and v is in S. The algorithm selects x from $\{u, v\}$ uniformly at random. Therefore, the probability that x is in S is at least $\frac{1}{2}$. If $x \in S$ then the recursive call of the algorithm succeeds with probability at least γ^{k-1} by the induction hypothesis. Hence the probability that the algorithm outputs success is at least $\frac{1}{2} \cdot \gamma^{k-1} \geq \gamma^k$. It remains to consider the case when G has no double edges.

If G has no double edges the algorithm picks a random terminal $t \in T$. By Observation 3.4, at least half the terminals in $G \setminus S$ are good. Thus, with probability at least $\frac{1}{2}$, the vertex t is either in S or a good terminal of $G \setminus S$. Let T_g be the set of good terminals of $G \setminus S$. Let $\mathsf{success}$ be the event that the algorithm outputs $\mathsf{succeed}$. We have that

$$P[\mathsf{success}] \geq P[t \in S \cup T_g] \cdot P[\mathsf{success} \mid t \in S \cup T_g] \geq \frac{1}{2} P[\mathsf{success} \mid t \in S \cup T_g]$$

Now, $P[\mathsf{success} \mid t \in S \cup T_g]$ can be lower bounded as follows

$$P[\mathsf{success} \mid t \in S \cup T_g] \geq \min\left(P[\mathsf{success} \mid t \in S], P[\mathsf{success} \mid t \in T_g]\right).$$

With a constant probability $(1 - \alpha_t)$ the algorithm puts t in the solution and calls itself recursively on $G - t$ with terminal set $T \setminus t$ and budget of $k - 1$. If $t \in S$ then $G - t$ has a solution of size at most $k - 1$. By the induction hypothesis the recursive call will output success on $(G - t, T \setminus t, k - 1)$ with probability at least γ^{k-1}. Thus

$$P[\mathsf{success} \mid t \in S] \geq (1 - \alpha_t)\gamma^{k-1} \tag{1}$$

We now define a set of events related to the execution of algorithm $\mathsf{SolveSFVS}$. The event line_i is that the algorithm executes line i (if the algorithm reaches line i in some of the recursive calls this does not count). We have that

$$\begin{aligned} P[\mathsf{success} \mid t \in T_g] &\geq P[\mathsf{line}_{10} \mid t \in T_g] \cdot P[\mathsf{success} \mid \mathsf{line}_{10} \wedge t \in T_g] \\ &\geq \alpha_t \cdot P[\mathsf{success} \mid \mathsf{line}_{10} \wedge t \in T_g] \end{aligned} \tag{2}$$

Consider now the case that the algorithm reaches line 10, and that $t \in T_g$. By Lemma 3.6, t has degree at least two. Observe that since t is good, t has at most two neighbors in T which are not in S - the parent of t in F and at most one child of t. We consider the following four cases: (1a) $|N(t) \cap T| \geq 3$, (1b) $|N(t) \cap T| = 2$ and $N(t) \cap T \cap S \neq \emptyset$, (1c) $|N(t) \cap T| = 2$ and $N(t) \cap T \cap S = \emptyset$ and (1d) $|N(t) \cap T| \leq 1$. With each of the cases we associate the event that the case occurs.

We first consider case (1a). Starting from line 10, the algorithm reaches line 12 with probability $\frac{1}{2}$. Then, since $|N(t) \cap T \setminus S| \leq 2$ it follows that the probability that the vertex z, selected from $N(t) \cap T$ on line 13, is in S is at least $\frac{1}{3}$. Thus, we conclude that

$$P[\mathsf{success} \mid \mathsf{line}_{10} \wedge t \in T_g \wedge (1\mathsf{a})] \geq \frac{1}{6} \cdot \gamma^{k-1}. \tag{3}$$

Case (1b) is similar to case (1a), except that now the probability that z is in S is at least $\frac{1}{2}$, because at least one out of two vertices in $N(t) \cap T$ are in S. Thus,

$$P[\mathsf{success} \mid \mathsf{line}_{10} \wedge t \in T_g \wedge (1\mathsf{b})] \geq \frac{1}{4} \cdot \gamma^{k-1}. \tag{4}$$

In case (1c), line 11 is executed, and with probability $\frac{1}{2}$ the algorithm skips to line 14. In this case, $|N(t) \cap T| = 2$ and therefore t has at least one neighbor that is a terminal. Hence by Lemma 3.6 t has degree at least 3. Thus t has at least one non-terminal neighbor, implying that the algorithm does not return fail in line 14. Furthermore, since t's parent is a terminal, all non-terminal neighbors of t are either in S or children of t in F. Let C be the set of non-terminal children of t in F. We have that

$$P[\mathsf{success} \mid \mathsf{line}_{10} \wedge t \in T_g \wedge (\mathsf{1c})] \geq \frac{1}{2} \cdot P[\mathsf{success} \mid \mathsf{line}_{15} \wedge t \in T_g \wedge v \in S \cup C] \tag{5}$$

In case (1d) the algorithm skips directly to line 15. If t has exactly one terminal neighbor then by Lemma 3.6 t has degree at least 3, and so t has at least two non-terminal neighbors. If t has no terminal neighbors it also has at least two non-terminal neighbors. At most one of these non-terminal neighbors is the parent of t in F. Thus, when a non-terminal neighbor v of t is picked in line 15, the probability that v is in $S \cup C$ is at least $\frac{1}{2}$. It follows that

$$P[\mathsf{success} \mid \mathsf{line}_{10} \wedge t \in T_g \wedge (\mathsf{1d})] \geq \frac{1}{2} \cdot P[\mathsf{success} \mid \mathsf{line}_{15} \wedge t \in T_g \wedge v \in S \cup C] \tag{6}$$

Since the four cases (1a) ... (1d) are exhaustive, we may conclude that

$$P[\mathsf{success} \mid \mathsf{line}_{10} \wedge t \in T_g] \geq \min \begin{pmatrix} \frac{1}{6} \cdot \gamma^{k-1} \\ \frac{1}{2} \cdot P[\mathsf{success} \mid \mathsf{line}_{15} \wedge t \in T_g \wedge v \in S \cup C] \end{pmatrix}$$

The algorithm resolves that it "knows" that $v \in S \cup C$ but does not know whether $v \in S$ or $v \in C$ in exactly the same way that it resolved "knowing" $t \in S \cup T_g$ but not knowing whether $t \in S$ or $t \in T_g$. In particular, the algorithm recurses on $(G \setminus v, T, k-1)$ with probability $1 - \alpha_v$. This yields the following bounds.

$$P[\mathsf{success} \mid \mathsf{line}_{15} \wedge t \in T_g \wedge v \in S] \geq (1 - \alpha_v)\gamma^{k-1} \tag{7}$$

$$P[\mathsf{success} \mid \mathsf{line}_{15} \wedge t \in T_g \wedge v \in C] \geq \alpha_v \cdot P[\mathsf{success} \mid \mathsf{line}_{18} \wedge t \in T_g \wedge v \in C] \tag{8}$$

Plugging all of the above bounds together we get the following bound for $P[\mathsf{success}]$.

$$P[\mathsf{success}] \geq \min \begin{cases} \frac{1}{2} \cdot (1 - \alpha_t) \cdot \gamma^{k-1} \\ \frac{1}{12} \cdot \alpha_t \cdot \gamma^{k-1} \\ \frac{1}{4} \cdot \alpha_t \cdot (1 - \alpha_v) \cdot \gamma^{k-1} \\ \frac{1}{4} \cdot \alpha_t \cdot \alpha_v \cdot P[\mathsf{success} \mid \mathsf{line}_{18} \wedge t \in T_g \wedge v \in C] \end{cases} \tag{9}$$

At this point we need to lower bound $P[\mathsf{success} \mid \mathsf{line}_{18} \wedge t \in T_g \wedge v \in C]$. Starting at line 18 the algorithm does the following. First it makes some of the terminals "undeletable" by putting them into $\bar{T}$, samples an important separator from v to the set of terminals (but without deleting the undeletable ones), and either puts the entire separator into the solution, or puts the separator minus a randomly chosen vertex into the solution. The trickiest part of the proof is to show that with good probability the set which is put into the solution is either a subset of S or of another optimal solution to the instance. Towards this goal we need to prove a few "pushing lemmata".

Assume that $t \in T_g$ and that $v \in C$ (recall that C was the set of non-terminal children of t in F). Let $b = \chi^{-}(t)$ and $b' = \chi^{-}(v)$. Since t is good, t has at most one effective descendant. Thus b' is either a leaf or a degree 2 vertex of F. We define the event leaf as b' being a leaf of F, and the event deg_2 as b' having degree 2. If b' has degree 2 this means that t has exactly one effective descendant t' and that $\chi^{-}(b')$ is the parent of $\chi^{-}(t')$ in F. We distinguish between two cases, either $vt' \in E(G)$ or not. We define the event adj which is that $vt' \in E(G)$, and the event nadj which is that $vt' \notin E(G)$.

4.1. Pushing Lemmata

We now prove a series of lemmas about how one can modify a solution by "pushing" the solution vertices towards the terminals. In the following two lemma statements, G is a graph, T is a set of terminals and S is a subset feedback vertex set of G. Further, (F, χ) is a terminal forest of $G - S$, b is a terminal node in F and b' is a non-terminal child of b in F. Also, $t = \chi(b)$ and v is the unique neighbor of t in $\chi(b')$ (assuming the existence of b and b', the existence and uniqueness of v is guaranteed by Observation 3.2). Furthermore $A = N(\chi(b')) \cap S$ and $\bar{T} \subseteq T$ is a set of terminals such that $t \in \bar{T}$ and $\bar{T} \cap A = \emptyset$. Finally, G' is the graph obtained from G by adding a new vertex τ, making τ adjacent to all vertices of $\bar{T}$, and removing all edges between v and vertices in $\bar{T}$. Lemma 4.2 will be applied to analyze the event leaf, as well as in the deterministic algorithm of Section 5.

LEMMA 4.2. *If b' is a leaf node of F, then A is a v-τ separator in G'. Further, for any v-τ separator A' (in G') that dominates A, the set $(S \setminus A) \cup A'$ is a subset feedback vertex set of G.*

PROOF. To see that A is a v-τ separator in G' observe that $N_{G'}(\chi(b')) = A$ and that $\chi(b') \cap T = \emptyset$. We now move to the second statement, namely that $(S \setminus A) \cup A'$ is a subset feedback vertex set of G.

Suppose not. Then $G - ((S \setminus A) \cup A')$ contains a T-cycle C. Let L' be the set of vertices reachable from v in $G' - A'$, plus A'. Let $R' = (V(G') \setminus L') \cup A'$. (L', R') is a separation in G', since $L' \setminus R'$ are exactly the vertices reachable from v, while $R' \setminus L'$ are the vertices *not* reachable from v in $G' - A'$. A key observation is that since A' dominates A, we have that $\chi(b') \subseteq L' \setminus R'$ and that $A \setminus A' \subseteq L' \setminus R'$.

All edges of G that are not in G' are incident to v. Let $L = L' \cup \{v\}$ and $R = R' \cup \{v\}$, it follows that (L, R) is a separation in G. Since $G - S$ has no T-cycles it follows that C must contain some vertex of $A \setminus A'$. We know that $A \setminus A' \subseteq L' \setminus R' = L \setminus R$. Furthermore, since C is a T-cycle, C must contain a terminal z. Since A' separates v from τ it follows that $L' \cap \bar{T} \subseteq A'$. Since v is not a terminal it follows that $L \cap \bar{T} \subseteq A'$. Since C is disjoint from A' this means that $z \notin L$, implying that $z \in R \setminus L$. But then C contains a vertex in $L \setminus R$ and a vertex in $R \setminus L$, which implies that $|C \cap L \cap R| \geq 2$. However, the only vertex in $L \cap R$ which is not in A' is v, contradicting that $C \cap A' = \emptyset$. $\square$

Lemma 4.3 will be applied to analyze the event $\mathsf{deg}_2 \vee \mathsf{adj}$.

LEMMA 4.3. *If t has a unique effective descendant t', b' is adjacent to $\chi^{-}(t')$ (in F), v is adjacent to t' and t' is in $\bar{T}$, we conclude the following. A is a v-τ separator in G'. Further, for any v-τ separator A' (in G') that dominates A, $(S \setminus A) \cup A'$ is a subset feedback vertex set of G.*

The proof of Lemma 4.3 is identical (word by word!) to the proof of Lemma 4.2, and therefore omitted.

For the last lemma statement we need to change the definition of A. Suppose t has a unique effective descendant t', b' is adjacent to $\chi^{-}(t')$ (in F), v is non-adjacent to t'. Let y be the unique neighbor of t' in $\chi(b')$. We define Q to be the set of vertices reachable from v (including v) in $G - (S \cup \{y, t\})$, and define $A = N(Q) \setminus \{y, t\}$. Just as before, $\bar{T} \subseteq T$ is a set of terminals such that $t \in \bar{T}$ and $\bar{T} \cap A = \emptyset$. Finally, G' is the graph obtained from G by adding a new vertex τ, making τ adjacent to all vertices of $\bar{T}$, and removing all edges between v and vertices in $\bar{T}$. Lemma 4.4 will be applied to analyze the event $\mathsf{deg}_2 \vee \mathsf{nadj}$.

LEMMA 4.4. *If t has a unique effective descendant t', b' is adjacent to $\chi^{-}(t')$ (in F), v is non-adjacent to t' and t' is in $\bar{T}$, we conclude the following. Let y be the unique neighbor of t' in $\chi(b')$. Then $A \cup \{y\}$ is a v-τ separator in G'. Further, for any v-τ separator A' (in G') that dominates $A \cup \{y\}$, if $A' \cap \bar{T} = \emptyset$ then $y \in A'$ and $(S \setminus A) \cup (A' \setminus \{y\})$ is a subset feedback vertex set of G.*

PROOF. To see that $A \cup \{y\}$ is a v-τ separator in G' observe that the set of vertices reachable from v in $G' \setminus (A \cup \{y\})$ is exactly Q and that $Q \cap T = \emptyset$. We now show that $y \in A'$. We have that $G'[Q]$ is connected, and contains a neighbor of y, since $G'[\chi(b')]$ is connected. Since A' dominates $A \cup \{y\}$, all vertices in Q are reachable from v in $G - A'$. Since $t' \in \bar{T}$ we have that $t' \notin A'$. If $y \notin A'$ then there is a path from v to τ in $G' - A'$; via Q to y, then to t' by the edge yt' and finally to τ. This contradicts that A' separates v from τ. We conclude that $y \in A'$, and proceed to the last statement, namely that $(S \setminus A) \cup (A' \setminus \{y\})$ is a subset feedback vertex set of G.

Suppose not. Then $G - ((S \setminus A) \cup A')$ contains a T-cycle C. Let L' be the set of vertices reachable from v in $G' - A'$, plus A'. Let $R' = (V(G') \setminus L') \cup A'$. (L', R') is a separation in G', since $L' \setminus R'$ are exactly the vertices reachable from v, while $R' \setminus L'$ are the vertices *not* reachable from v in $G' - A'$. A key observation is that since A' dominates $A \cup \{y\}$, we have that $Q \subseteq L' \setminus R'$ and that $A \setminus A' \subseteq L' \setminus R'$.

All edges of G that are not in G' are incident to v. Let $L = L' \cup \{v\}$ and $R = R' \cup \{v\}$, it follows that (L, R) is a separation in G. Since $G - S$ has no T-cycles it follows that C must contain some vertex of $A \setminus (A' \setminus \{y\}) = A \setminus A'$. We know that $A \setminus A' \subseteq L' \setminus R' = L \setminus R$. Furthermore, since C is a T-cycle, it must contain a terminal z. Since A' separates v from τ in G' it follows that $L' \cap T \subseteq A'$. Since v is not a terminal it follows that $L \cap T \subseteq A'$. Since C is disjoint from $A' \setminus \{y\}$ and y is not a terminal, this means that $z \notin L$, and hence $z \in R \setminus L$. Thus C contains a vertex in $L \setminus R$ and a vertex in $R \setminus L$, implying that $|C \cap L \cap R| \geq 2$. However, the only two vertices in $L \cap R$ which are not in $A' \setminus \{y\}$ are v and y. Thus $C \cap L \cap R = \{v, y\}$.

It follows that C contains path P from v to y with at least one internal vertex, and all of its internal vertices in $R \setminus L$. The internal vertices of P are disjoint from $S \setminus A$ and disjoint from $A \setminus A'$, since $A \setminus A' \subseteq L$. Thus P is disjoint from S. However, all paths between v and y in $G \setminus S$ with at least one internal vertex must intersect $Q \setminus \{v\}$. But $Q \subseteq L$, contradicting that the internal vertices of P are disjoint from L. $\square$

4.2. Final Analysis of Algorithm SolveSFVS

Recall that in order to lower bound the success probability of Algorithm SolveSFVS, all that remained was to lower bound $P[\text{success} \mid \text{line}_{18} \wedge t \in T_g \wedge v \in C]$. Thus we consider the case that $t \in T_g$ and that $v \in C$ (C is the set of non-terminal children of t in F). This case is split in three subcases: leaf, $\text{deg}_2 \wedge \text{adj}$, and $\text{deg}_2 \wedge \text{nadj}$. In all cases the algorithm selects a subset $\bar{T} \subseteq T$ as follows, t is in $\bar{T}$, and every other terminal is put into $\bar{T}$ with probability $1 - \beta$. We now consider these cases one by one.

4.2.1. Handling the subcase $v \in C \wedge$ leaf. First we lower bound $P[\text{success} \mid \text{line}_{18} \wedge t \in T_g \wedge v \in C \wedge \text{leaf}]$. In this case, let $A = N(\chi(b')) \setminus \{t\}$. Since v is in a T-cycle we have that $|A| \geq 1$. The probability that $A \cap \bar{T} = \emptyset$ is $\beta^{|A \cap T|}$. Assume now that $A \cap \bar{T} = \emptyset$. The algorithm defines G', and G' is exactly the graph defined in the beginning of Section 4.1. By Lemma 4.2, A is a v-τ separator in G'. Furthermore $A \cap \bar{T} = \emptyset$ so A is a v-$(\{\tau\} \cup \bar{T})$ separator in G'. Hence, either there exists an important v-$(\{\tau\} \cup \bar{T})$ separator A' which dominates A, or A is itself an important v-$(\{\tau\} \cup \bar{T})$ separator in G'. In this case we let $A' = A$. In either case, by Lemma 4.2, $(S \setminus A) \cup A'$ is a subset feedback vertex set of G of size at most $|S| \leq k$. Furthermore, since A' dominates A and separates v from τ, it follows that $A \cap T \subseteq A'$.

The algorithm samples an important v-$(\{\tau\} \cup \bar{T})$ separator $A^\star$ using Lemma 2.8, with deletion probability p. By Lemma 2.8 the probability that $A^\star = A'$ is at least $p^{|A|}(1-p)^{|A|-\lambda}$, where $\lambda = \lambda_{G'}(v, \{\tau\} \cup \bar{T}) \geq 1$ since v is in a T-cycle.

At this point we distinguish between two cases, either $\lambda = 1$, or $\lambda \geq 2$. We consider first the case where $\lambda = 1$. In this case there exists a vertex z that separates v from τ in G'. Let C be the connected component of $G' - z$ that contains v. We prove that z is a terminal vertex. Suppose not, since v has degree at least 3 (in G) by Lemma 3.6, and the only potential neighbors of v outside of $\{t, z\}$ are in C, v has a neighbor u in C. But then u is separated (in G) from T by $\{v, z\}$ contradicting the conclusion of Lemma 3.6 that u has three internally vertex disjoint paths to T. We conclude that z is a terminal vertex.

The set $\{z\}$ is a v-$(\{\tau\} \cup \bar{T})$ separator, and since z is a terminal, $\{z\}$ is an important v-$(\{\tau\} \cup \bar{T})$ separator of size $\lambda_{G'}(v, \{\tau\} \cup \bar{T})$. By Lemma 2.6 every important $v-(\{\tau\} \cup \bar{T})$ separator $\hat{A}$ (except $\{z\}$) covers $\{z\}$. Consider an important $v - (\{\tau\} \cup \bar{T})$ separator $\hat{A}$. If $z \in \hat{A}$ then $\hat{A} = \{z\}$ since $\hat{A}$ is a minimal $v - (\{\tau\} \cup \bar{T})$ separator. But if $z \notin \hat{A}$ then z is reachable from v in $G' \setminus \hat{A}$, since $\hat{A}$ covers $\{z\}$. But z is adjacent to τ, contradicting that $\hat{A}$ separates v from τ. We conclude that $\{z\}$ is the only important $v - (\{\tau\} \cup \bar{T})$ separator, of any size.

Thus, if $\lambda = 1$ we have that $A^\star = A' = \{z\}$. By Lemma 4.2 the set $S^\star = (S \setminus A) \cup A^\star$ is a subset feedback vertex set of G, and $|S^\star| \leq |S| \leq k$. Since $|A^\star| = 1$ the algorithm calls itself recursively on $(G - A^\star, T \setminus A^\star, k - |A^\star|)$, which is a "yes" instance since $A^\star \subseteq S^\star$.

By the induction hypothesis the recursive call outputs $\mathsf{succeed}$ with probability at least γ^{k-1}. Hence,

$$P[\mathsf{success} \mid \mathsf{line}_{18} \wedge t \in T_g \wedge v \in C \wedge \mathsf{leaf} \wedge \lambda = 1] \geq \beta \cdot \gamma^{k-1} \tag{10}$$

We now consider the case that $\lambda \geq 2$. Thus the probability that $A^\star = A'$ is at least $p^{|A|}(1-p)^{|A|-2}$. If $A^\star = A'$ then we conclude that $S^\star = (S \setminus A) \cup A^\star$ is a subset feedback vertex set of size at most k. Assume now that $A^\star = A'$. Note that since $\lambda \geq 2$, $|A^\star| \geq 2$.

At this point we distinguish between two cases, either $A \subseteq T$ or not. We first consider the case that A is not a subset of T. If $A^\star \subseteq T$ then the algorithm calls itself recursively on $(G - A^\star, T \setminus A^\star, k - |A^\star|)$, which is a "yes" instance since $A^\star \subseteq S^\star$. By the induction hypothesis the recursive call outputs $\mathsf{succeed}$ with probability at least $\gamma^{k-|A^\star|}$. If $A^\star \setminus T \neq \emptyset$ then the algorithm picks a random $y^\star \in A^\star \setminus T$, and calls itself recursively on $(G - (A^\star \setminus \{y^\star\}), T \setminus (A^\star \setminus \{y^\star\}), k - |A^\star| + 1)$, which is a "yes" instance since $A^\star \subseteq S^\star$. By the induction hypothesis the recursive call outputs $\mathsf{succeed}$ with probability at least $\gamma^{k-|A^\star|+1}$. Thus, in either case ($A^\star \subseteq T$ or $A^\star \nsubseteq T$) the recursive call outputs $\mathsf{succeed}$ with probability at least $\gamma^{k-|A^\star|+1}$. Thus,

$$\begin{aligned} P[\mathsf{success} \mid \mathsf{line}_{18} \wedge t \in T_g \wedge v \in C \wedge \mathsf{leaf} \wedge \lambda \geq 2 \wedge A \nsubseteq T] \geq \\ \geq \beta^{|A \cap T|} \cdot p^{|A|}(1-p)^{|A|-2} \cdot \gamma^{k-|A|+1} \\ \geq \beta^{|A|-1} \cdot p^{|A|}(1-p)^{|A|-2} \cdot \gamma^{k-|A|+1} \end{aligned} \tag{11}$$

Note that in Equation 11 we have $|A| \geq 2$.

Consider now the case that $A \subseteq T$. Since $A^\star = A'$ dominates A and separates v from τ we have that $A \cap T = A \subseteq A^\star$. Since $|A^\star| \leq |A|$ it follows that $A^\star = A$, and therefore that $A^\star \subseteq T$. The algorithm calls itself recursively on $(G - A^\star, T \setminus A^\star, k - |A^\star|)$, which is a "yes" instance since $A^\star \subseteq S^\star$. By the induction hypothesis the recursive call outputs $\mathsf{succeed}$ with probability at least $\gamma^{k-|A^\star|}$. Hence,

$$P[\mathsf{success} \mid \mathsf{line}_{18} \wedge t \in T_g \wedge v \in C \wedge \mathsf{leaf} \wedge A \nsubseteq T] \geq \beta^{|A|} \cdot p^{|A|}(1-p)^{|A|-2} \cdot \gamma^{k-|A|} \tag{12}$$

In Equation 12 we have $|A| \geq 2$. To summarize the leaf case, we have:

$$P[\mathsf{success} \mid \mathsf{line}_{18} \wedge t \in T_g \wedge v \in C \wedge \mathsf{leaf}] \geq \min \begin{cases} \beta \cdot \gamma^{k-1} & \\ \beta^{a-1} \cdot p^a(1-p)^{a-2} \cdot \gamma^{k-a+1} & (a \geq 2) \\ \beta^a \cdot p^a(1-p)^{a-2} \cdot \gamma^{k-a} & (a \geq 2) \end{cases}$$

4.2.2. Handling the subcase $v \in C \wedge \mathsf{deg}_2 \wedge \mathsf{adj}$. Next we lower bound $P[\mathsf{success} \mid \mathsf{line}_{18} \wedge t \in T_g \wedge v \in C \wedge \mathsf{deg}_2 \wedge \mathsf{adj}]$. In this case t has an effective descendant t' and v is adjacent both to t and to t'. Further, b' has degree 2 in F, thus the neighbors of b' in F are exactly $\chi^-(t)$ and $\chi^-(t')$. Let $A = N(\chi(b')) \setminus \{t, t'\}$. We show that $A \neq \emptyset$.

For a contradiction, suppose $A = \emptyset$. Since v has degree at least 3 by Lemma 3.6, and the only potential neighbors of v outside of $A \cup \{t, t'\}$ are in $\chi(b')$, v has a neighbor $u \in \chi(b')$. But then u is separated (in G) from T by v, contradicting the conclusion of Lemma 3.6 that u has three internally vertex disjoint paths to T. We conclude that $A \neq \emptyset$.

The probability that $A \cap \bar{T} = \emptyset$ and that $t' \in \bar{T}$ is $(1-\beta)\beta^{|A \cap T|}$. Assume now that $A \cap \bar{T} = \emptyset$ and that $t' \in \bar{T}$. The algorithm defines G', and G' is exactly the graph defined in the beginning of Section 4.1. By Lemma 4.3, A is a v-τ separator in G'. Furthermore $A \cap \bar{T} = \emptyset$ so A is a v-$(\{\tau\} \cup \bar{T})$ separator in G'. Hence, either there exists an important v-$(\{\tau\} \cup \bar{T})$ separator A' which dominates A, or A is itself an important v-$(\{\tau\} \cup \bar{T})$ separator in G'. In this case we let $A' = A$. In either case, by Lemma 4.3, $(S \setminus A) \cup A'$ is a subset feedback vertex set of G of size at most $|S| \leq k$. Furthermore, since A' dominates A and separates v from τ, it follows that $A \cap T \subseteq A'$.

The algorithm samples an important v-$(\{\tau\} \cup \bar{T})$ separator $A^\star$ using Lemma 2.8, with deletion probability p. By Lemma 2.8 the probability that $A^\star = A'$ is at least $p^{|A|}(1-p)^{|A|-\lambda}$, where $\lambda = \lambda_{G'}(v, \{\tau\} \cup \bar{T})$. We show that $\lambda \geq 1$. Suppose not, then the connected component of $G' = G \setminus \{vt, vt'\}$ that contains v does not contain any terminals. Since the degree of v is at least 3 in G, v has a neighbor u in this component. But then v separates u from T contradicting that u is in a T-cycle. Thus the probability that $A^\star = A$ is at least $p^{|A|}(1-p)^{|A|-1}$. If $A^\star = A'$ then we conclude that $S^\star = (S \setminus A) \cup A^\star$ is a subset feedback vertex set of size at most k. Assume now that $A^\star = A'$

Since both t and t' are in $\bar{T} \cap N(v)$, the algorithm calls itself recursively on $(G - A^\star, T \setminus A^\star, k - |A^\star|)$, which is a "yes" instance since $A^\star \subseteq S^\star$. By the induction hypothesis the recursive call outputs $\mathsf{succeed}$ with probability at least $\gamma^{k-|A^\star|}$. Hence,

$$P[\mathsf{success} \mid \mathsf{line}_{18} \wedge t \in T_g \wedge v \in C \wedge \mathsf{deg}_2 \wedge \mathsf{adj}] \geq (1-\beta)\beta^a \cdot p^a(1-p)^{a-1} \cdot \gamma^{k-a} \quad (a \geq 1)$$

4.2.3. Handling the subcase $v \in C \wedge \mathsf{deg}_2 \wedge \mathsf{nadj}$. Now we lower bound $P[\mathsf{success} \mid \mathsf{line}_{18} \wedge t \in T_g \wedge v \in C \wedge \mathsf{deg}_2 \wedge \mathsf{nadj}]$. In this case t has an effective descendant t' and v is adjacent to t, but not to t'. Then by Observation 3.2 t' has exactly one neighbor in $\chi(b')$, call this neighbor y. Further, b' has degree 2 in F, thus the neighbors of b' in F are exactly $\chi^-(t)$ and $\chi^-(t')$. We define Q to be the set of vertices reachable from v (including v) in $G - (S \cup \{y, t\})$, and define $A = N(Q) \setminus \{y, t\}$. Observe that $Q \subseteq \chi(b')$. We show that $A \neq \emptyset$.

For a contradiction, suppose $A = \emptyset$. Since v has degree at least 3 by Lemma 3.6, and the only potential neighbors of v outside of $A \cup \{t, y\}$ are in Q, v has a neigbors $u \in Q$. We have that u is separated (in G) from T by $\{y, v\}$, contradicting the conclusion of

Lemma 3.6 that u has three internally vertex disjoint paths to T. We conclude that $A \neq \emptyset$.

The probability that $A \cap \bar{T} = \emptyset$ and that $t' \in \bar{T}$ is $(1-\beta)\beta^{|A \cap T|}$. Assume now that $A \cap \bar{T} = \emptyset$ and that $t' \in \bar{T}$. The algorithm defines G', and G' is exactly the graph defined right before the statement of Lemma 4.4. By Lemma 4.4, $A \cup \{y\}$ is a v-τ separator in G'. Furthermore $(A \cup \{y\}) \cap \bar{T} = \emptyset$ so $(A \cup \{y\})$ is a v-$(\{\tau\} \cup \bar{T})$ separator in G'. Hence, either there exists an important v-$(\{\tau\} \cup \bar{T})$ separator A' which dominates $(A \cup \{y\})$, or $(A \cup \{y\})$ is itself an important v-$(\{\tau\} \cup \bar{T})$ separator in G'. In this case we let $A' = (A \cup \{y\})$. In either case, by Lemma 4.4, $y \in A'$ and $(S \setminus A) \cup (A' \setminus \{y\})$ is a subset feedback vertex set of G of size at most $|S| \leq k$. Furthermore, since A' dominates $A \cup \{y\}$ and separates v from τ, it follows that $A \cap T \subseteq A'$.

The algorithm samples an important v-$(\{\tau\} \cup \bar{T})$ separator $A^\star$ using Lemma 2.8, with deletion probability p. By Lemma 2.8 the probability that $A^\star = A'$ is at least $p^{|A|}(1-p)^{|A|-\lambda}$, where $\lambda = \lambda_{G'}(v, \{\tau\} \cup \bar{T})$. We show that $\lambda \geq 2$.

Suppose not, then there exists a vertex z that separates v from $\{\tau\} \cup \bar{T}$ in G'. Since $t' \in \bar{T}$ and $G'[Q \cup \{y, t'\}]$ is a connected set containing both v and t' it follows that $z \in Q \cup \{y\}$, which means that z is not a terminal. Let C be the connected component of $G' \setminus z$ that contains v, since v has degree 3 or more in G (by Lemma 3.6), v has at least one neighbor $u \notin \{t, z\}$. Then $u \in C$ and so u is a non-terminal. But then u is a non-terminal separated in G from T by $\{v, z\}$ contradicting the conclusion of Lemma 3.6 that u has three internally vertex disjoint paths to T. We conclude that $\lambda \geq 2$. Thus the probability that $A^\star = A'$ is at least $p^{|A|+1}(1-p)^{|A|-1}$. If $A^\star = A'$ then we conclude that $S^\star = (S \setminus A) \cup (A^\star \setminus \{y\})$ is a subset feedback vertex set of size at most k. Assume now that $A^\star = A'$.

Since t is the only terminal neighbor of v except for the terminals in A, v has exactly one neighbor in $\bar{T}$. Further, $y \in A' = A^\star$ so $A^\star$ contains at least one non-terminal. Finally we showed that $\lambda_{G'}(v, \{\tau\} \cup \bar{T}) \geq 2$ and therefore $|A^\star| \geq 2$. It follows that the algorithm proceeds to line 24 where it picks a non-terminal vertex $y^\star \in A^\star \setminus T$. With probability $\frac{1}{|A^\star \setminus T|}$ the vertex $y^\star$ is equal to y. Since $A \cap T \subseteq A^\star$ it follows that $\frac{1}{|A^\star \setminus T|} \geq \frac{1}{|A \setminus T|+1}$. Assume now that $y^\star = y$.

At this point the algorithm calls itself recursively on $(G - (A^\star \setminus \{y^\star\}), T \setminus (A^\star \setminus \{y^\star\}), k - |A^\star| + 1)$, which is a "yes" instance since $(A^\star \setminus \{y\}) \subseteq S^\star$. By the induction hypothesis the recursive call outputs $\mathsf{succeed}$ with probability at least $\gamma^{k-|A^\star|+1} = \gamma^{k-|A|}$. Let $a = |A|$ and $r = |A \cap T|$, then

$$P[\mathsf{success} \mid \mathsf{line}_{18} \wedge t \in T_g \wedge v \in C \wedge \mathsf{deg}_2 \wedge \mathsf{nadj}] \geq (1-\beta)\beta^r \cdot p^{a+1}(1-p)^{a-1} \cdot \frac{\gamma^{k-a}}{a-r+1}$$

Here a and r must satisfy $a \geq 1$ and $a \geq r \geq 0$.

4.2.4. Completing the proof of Lemma 4.1 and the randomized algorithm for SUBSET FVS. Since the three cases leaf, $\mathsf{deg}_2 \vee \mathsf{adj}$ and $\mathsf{deg}_2 \vee \mathsf{nadj}$ are exhaustive if the algorithm reaches line 18, $t \in T_g$ and $v \in C$, the following inequalty summarizes the bounds for $P[\mathsf{success} \mid \mathsf{line}_{18} \wedge t \in T_g \wedge v \in C]$

$$P[\mathsf{success} \mid \mathsf{line}_{18} \wedge t \in T_g \wedge v \in C] \geq \min \begin{cases} \beta \cdot \gamma^{k-1} & \\ \beta^{a-1} \cdot p^a(1-p)^{a-2} \cdot \gamma^{k-a+1} & (a \geq 2) \\ \beta^a \cdot p^a(1-p)^{a-2} \cdot \gamma^{k-a} & (a \geq 2) \\ (1-\beta)\beta^a \cdot p^a(1-p)^{a-1} \cdot \gamma^{k-a} & \\ (1-\beta)\beta^r \cdot p^{a+1}(1-p)^{a-1} \cdot \frac{\gamma^{k-a}}{a-r+1} & (a \geq r \geq 0) \end{cases}$$

Here $a \geq 1$ in all inequalities, unless $a \geq 2$ is specified. Inserting this into the Equation 9 yields the following bounds for $P[\text{success}]$.

$$P[\text{success}] \geq \min \begin{cases} \frac{1}{2} \cdot (1-\alpha_t) \cdot \gamma^{k-1} & \\ \frac{1}{12} \cdot \alpha_t \cdot \gamma^{k-1} & \\ \frac{1}{4} \cdot \alpha_t \cdot (1-\alpha_v) \cdot \gamma^{k-1} & \\ \frac{1}{4} \cdot \alpha_t \cdot \alpha_v \cdot \beta \cdot \gamma^{k-1} & \\ \frac{1}{4} \cdot \alpha_t \cdot \alpha_v \cdot \beta^{a-1} \cdot p^a(1-p)^{a-2} \cdot \gamma^{k-a+1} & (a \geq 2) \\ \frac{1}{4} \cdot \alpha_t \cdot \alpha_v \cdot \beta^{a} \cdot p^a(1-p)^{a-2} \cdot \gamma^{k-a} & (a \geq 2) \\ \frac{1}{4} \cdot \alpha_t \cdot \alpha_v \cdot (1-\beta)\beta^{a} \cdot p^a(1-p)^{a-1} \cdot \gamma^{k-a} & \\ \frac{1}{4} \cdot \alpha_t \cdot \alpha_v \cdot (1-\beta)\beta^{r} \cdot p^{a+1}(1-p)^{a-1} \cdot \frac{\gamma^{k-a}}{a-r+1} & (a \geq r \geq 0) \end{cases}$$

Here $a \geq 1$ unless $a \geq 2$ is specified. In order to complete the inductive proof of Lemma 4.1 we need to prove that the right hand side is at least γ^k. At this point we plug in the following values for the probability constants in the algorithm: $\alpha_t = 0.92$, $\alpha_v = 0.83$, $\beta = 0.48$ and $p = 0.91$. Recall that $\gamma = \frac{1}{25.6}$. For the first four bounds, we get the following outcome.

$$\begin{aligned} \frac{1}{2} \cdot (1-\alpha_t) \cdot \gamma^{k-1} &\geq \gamma^k \cdot (25.6 \cdot \frac{1}{2} \cdot (1-0.92)) \geq \gamma^k \\ \frac{1}{12} \cdot \alpha_t \cdot \gamma^{k-1} &\geq \gamma^k \cdot (25.6 \cdot \frac{1}{12} \cdot 0.92) \geq \gamma^k \\ \frac{1}{4} \cdot \alpha_t \cdot (1-\alpha_v) \cdot \gamma^{k-1} &\geq \gamma^k \cdot (25.6 \cdot \frac{1}{4} \cdot 0.92 \cdot (1-0.83)) \geq \gamma^k \\ \frac{1}{4} \cdot \alpha_t \cdot \alpha_v \cdot \beta \cdot \gamma^{k-1} &\geq \gamma^k \cdot (25.6 \cdot \frac{1}{4} \cdot 0.92 \cdot 0.83 \cdot 0.48) \geq \gamma^k \end{aligned}$$

For the next bound, we have

$$\begin{aligned} &\frac{1}{4} \cdot \alpha_t \cdot \alpha_v \cdot \beta^{a-1} \cdot p^a(1-p)^{a-2} \cdot \gamma^{k-a+1} \geq \\ &\qquad \geq \gamma^k \cdot (25.6 \cdot 0.48 \cdot 0.91 \cdot (1-0.91))^{a-1} \cdot \frac{0.92 \cdot 0.83}{4} \cdot \frac{0.91}{1-0.91} \\ &\qquad \geq \gamma^k \end{aligned}$$

In the last transition we used that $a \geq 2$. Similarly, for the third last bound we have

$$\begin{aligned} &\frac{1}{4} \cdot \alpha_t \cdot \alpha_v \cdot \beta^{a} \cdot p^a(1-p)^{a-2} \cdot \gamma^{k-a} \geq \\ &\qquad \geq \gamma^k \cdot (25.6 \cdot 0.48 \cdot 0.91 \cdot (1-0.91))^{a} \cdot \frac{0.92 \cdot 0.83}{4} \cdot \frac{1}{(1-0.91)^2} \\ &\qquad \geq \gamma^k \end{aligned}$$

Again, in the last transition we used that $a \geq 2$. For the second to last bound have

$$\begin{aligned} &\frac{1}{4} \cdot \alpha_t \cdot \alpha_v \cdot (1-\beta)\beta^{a} \cdot p^a(1-p)^{a-1} \cdot \gamma^{k-a} \geq \\ &\qquad \geq \gamma^k \cdot (25.6 \cdot 0.48 \cdot 0.91 \cdot (1-0.91))^{a} \cdot \frac{0.92 \cdot 0.83}{4} \cdot \frac{1-0.48}{1-0.91} \\ &\qquad \geq \gamma^k \end{aligned}$$

For the last bound we obtain the following expression.

$$
\begin{aligned}
&\frac{1}{4}\cdot\alpha_t\cdot\alpha_v\cdot(1-\beta)\beta^r\cdot p^{a+1}(1-p)^{a-1}\cdot\frac{\gamma^{k-a}}{a-r+1}\geq\\
&\qquad\geq\gamma^k\cdot(25.6\cdot 0.91\cdot(1-0.91))^a\cdot\frac{0.48^r}{a+1-r}\cdot\frac{0.92\cdot 0.83}{4}\cdot\frac{(1-0.48)\cdot 0.91}{1-0.91}\\
&\qquad\geq\gamma^k\cdot(25.6\cdot 0.48\cdot 0.91\cdot(1-0.91))^a\cdot\frac{0.92\cdot 0.83}{4}\cdot\frac{(1-0.48)\cdot 0.91}{1-0.91}\\
&\qquad\geq\gamma^k
\end{aligned}
$$

We conclude that $P[\text{success}]\geq\gamma^k$, which proves Lemma 4.1 $\square$.

Having Lemma 4.1 at hand we are ready to prove the main result of this section.

THEOREM 4.5. *There is a $\mathcal{O}(25.6^k\cdot k^{\mathcal{O}(1)}\cdot(n+m))$ time algorithm for* SUBSET FVS. *If the input instance is a "no" instance the algorithm outputs* fail*, if the input instance is a "yes" instance the algorithm outputs* succeed *with probability at least $1-\frac{1}{e}$.*

PROOF. The algorithm simply runs Algorithm SolveSFVS 25.6^k times on the input instance. If either of these runs result in a succeed, then the algorithm outputs succeed. If all of the runs of SolveSFVS return fail, the algorithm returns fail. The running time bound for this algorithm follows directly from the running time bound of Algorithm SolveSFVS. Further, since SolveSFVS always returns fail on "no" instances, this algorithm will also always output fail on "no" instances.

Consider now a run of this algorithm on a "yes" instance. By Lemma 4.1 the probaility that a particular run of SolveSFVS returns fail is at most $(1-\frac{1}{\gamma^k})$, where $\gamma=25.6$. The probability that all 25.6^k runs return fail is at most $(1-\frac{1}{25.6^k})^{25.6^k}\leq\frac{1}{e}$. Thus the probability that the algorithm returns succeed is at least $1-\frac{1}{e}$, as claimed. $\square$

5. A DETERMINISTIC LINEAR TIME ALGORITHM FOR SUBSET FVS

In this section we design a deterministic algorithm for SUBSET FVS running in time $2^{\mathcal{O}(k\log k)}\cdot(m+n)$. This algorithm could be thought of as a weak derandomization of the algorithm presented in the previous section. The algorithm uses the structural properties developed in the previous sections for the randomized algorithm. The main ingredient of the algorithm is a "deterministic sampling subroutine", that outputs a set of size polynomial in k, and whose intersection with any subset-fvs of size at most k is non-empty.

5.1. Finding nice independent sets

In this section we introduce the notion of "nice independent sets" which form the crucial ingredient in the sampling lemma we prove in the next section. The independent sets that we define have the following important property. Either they intersect a solution or there are many vertex disjoint paths from them to some subset-fvs of size at most k avoiding a large piece of the graph, that themselves have large flow between vertices of the independent set. These are crucially used in the sampling algorithm.

Definition 5.1. Let G be a graph, $I\subseteq V(G)$ be an independent set and γ be a positive integer. We say that I is γ-separating if for any pair of vertices x,y in I, we have that $\lambda_G(x,y)\leq\gamma$.

Definition 5.2. Let G be a graph and $I \subseteq V(G)$. The *closure* of I in G is an induced subgraph, denoted by $\mathsf{closure}(I, G)$, obtained by taking the union of connected components of $G - I$ that have at least two neighbors in I.

Definition 5.3. Let G be a graph, $X \subseteq V(G)$, I be an independent set in $G - X$ and δ be a positive integer. We say that I is a δ-nice independent set with respect to X, if there exists a subset $I^* \subseteq I$ that satisfies the following properties.

— $|I^* = \{w_1, \ldots, w_\ell\}| \geq \delta$.
— There exists an induced subgraph H of $G - X$ containing I^* such that for any partition (A, B) of I^* we have that $\lambda_H(A, B) = \lambda_{G-X}(A, B)$.
— There exists a family of paths $P_i = w_i \cdots b_i$, $i \in \{1, \ldots, \ell\}$, such that $b_i \in X$. Let P_i' denote the subpath of P_i that contains all but the last vertex, then the family of paths P_i', $i \in \{1, \ldots, \ell\}$, are pairwise disjoint and $\mathcal{Z} = \cup_{i=1}^{\ell} V(P_i') \setminus I^*$ does not intersect $V(H)$. That is, $\mathcal{Z} \cap V(H) = \emptyset$.

We say that I^* *certifies* the niceness of I.

In what follows we prove the existence of big nice independent sets.

LEMMA 5.4. *Let (G, T, k) be an instance of* SUBSET FVS, *$\tilde{T} \subseteq T$ be a set of terminals with degree at least 3 in G and let G be reduced. Then in time $\mathcal{O}(|\tilde{T}|^2 k(m+n))$, either we output a $(k+1)$-separating independent set $I \subseteq \tilde{T}$, such that for any subset-fvs S of size at most k that is disjoint from $\tilde{T}$, I is a δ-nice independent set with respect to S; or report that every S intersects $\tilde{T}$. Here, $\delta = \frac{|\tilde{T}|}{4}$.*

PROOF. We first test whether $G[\tilde{T}]$ is a forest. If $G[\tilde{T}]$ is not a forest then we report that every S intersects $\tilde{T}$. So from now onwards we assume that $G[\tilde{T}]$ is a forest. We find a maximum independent set, say I, in $G[\tilde{T}]$. Since $G[\tilde{T}]$ is a forest, we can find I in linear time. Furthermore, since $G[\tilde{T}]$ is a forest, and hence bipartite, we have that $|I| \geq |\tilde{T}|/2$. We first show that I is a $(k+1)$-separating independent set or every S intersects $\tilde{T}$. Suppose there exists a subset-fvs S that is disjoint from $\tilde{T}$. Then for every pair of vertices $x, y \in I$, there is at most one internally vertex disjoint path between them in $G - S$ and thus x can be disconnected from y in $G - S$ with at most one vertex z (here $z \notin \{x, y\}$). Thus x can be disconnected from y in G by deleting $S \cup \{z\}$. This implies that $\lambda_G(x, y) \leq k + 1$. So if there exists a subset-fvs S that is disjoint from $\tilde{T}$ we have that I is a $(k+1)$-separating independent set. Thus, given I, we check for every pair of vertices $x, y \in I$ whether $\lambda_G(x, y) \leq k + 1$. If for any pair we have that $\lambda_G(x, y) > k + 1$ we report that every S intersects $\tilde{T}$. This task can be carried out in time $\mathcal{O}(|\tilde{T}|^2 k(m+n))$ using Proposition 2.7.

Let S be an arbitrary subset-fvs of size at most k that is disjoint from $\tilde{T}$. We will show that I is a δ-nice independent set with respect to S. Let $G^* = G - S$. Consider a terminal forest (F, χ) of G^* with respect to I. Let $Q \subseteq I$ be the set of good terminals, that is, a subset of I that have at most one effective descendant in F. By Observation 3.4 we have that $|Q| \geq |I|/2$. Let $\mathsf{closure}(Q, G - S)$ be the closure of Q in $G - S$. Define $H = G^*[V(\mathsf{closure}(Q, G - S) \cup Q]$. We need to show that for any partition (A, B) of Q, we have that $\lambda_H(A, B) = \lambda_{G^*}(A, B)$. To prove this it is sufficient to show that no vertex in $V(G^*) \setminus V(H)$ appears on any path between a vertex in A and a vertex in B in G^*. Suppose that this is not the case and let P be a path between a vertex in A and a vertex in B in G^* that violates this property. That is, P is a path from a vertex

x in A to a vertex $y \in B$, and there is a vertex w on P that is also in $V(G^*) \setminus V(H)$. Since $x, y \in Q$, there is a subpath P' of P such that the end vertices belong to Q and all the internal vertices belong to $V(G^*) \setminus V(H)$. But this implies that the internal vertices are connected and have two neighbours in Q and thus should be the part of $\mathsf{closure}(Q, G - S)$, a contradiction. This proves the claim that $\lambda_H(A, B) = \lambda_{G^*}(A, B)$.

For every vertex in $q \in Q$ we associate a set of vertices $A_q \subseteq V(G^*) \setminus V(\mathsf{closure}(Q, G - S))$ such that $G[A_q]$ is connected.

— If $|N(q) \cap S| \geq 1$ then $A_q = \{q\}$.
— So assume now that $N(q) \cap S = \emptyset$. Let $\tau = \chi^{-1}(q)$. Since q has at most one effective descendant in F and has degree at least three in G, there is one child of τ, say u_q, such that $\chi(u_q) \cap I = \emptyset$. We set $A_q = \chi(u_q)$. By the properties of terminal forests we have that $G[A_q]$ is connected. Furthermore, observe that $G[A_q]$ is a connected component of $G^* \setminus \{q\}$ and thus $V(A_q)$ does not intersect with $V(\mathsf{closure}(Q, G - S))$.

Observe that by our construction, the sets $\{A_q \mid q \in Q\}$ are pairwise disjoint. Next we claim that for every vertex $q \in Q$ there is a vertex v_q (could be equal to q itself) in A_q such that $|N(v_q) \cap S| \geq 1$. Towards a contradiction assume that $N(A_q) \cap S = \emptyset$. Let x be the unique neighbor of q in A_q (recall that if q will have two neighbours then it would imply a T-cycle in $G - S$). However, this implies that xq is a bridge in G and hence Reduction Rule 1 would have been applicable; a contradiction to the assumption that G is reduced. Now for every $q \in Q$, we define the path P_q as follows. Let b_q denote a neighbour of v_q in S and L_q denote the path from u_q to v_q in $G[A_q]$ and let the path $P_q = qL_qb_q$. Let P'_q, $q \in Q$, denote the subpath of P_q that contains all but the last vertex b_q, then the family of paths P'_q, $i \in \{1, \ldots, \ell\}$, are pairwise disjoint and $\mathcal{Z} = \cup_{q \in Q}^{\ell} V(P'_q) \setminus Q$ does not intersect $V(H)$. This completes the proof. $\square$

Now we prove the lemma for the case when the set of degree 2 terminals is large.

LEMMA 5.5. *Let (G, T, k) be an instance of* SUBSET FVS, *$\tilde{T} \subseteq T$ be a set of degree 2 terminals and let G be reduced. Then in time $2^{\mathcal{O}(|\tilde{T}|)}(m+n)$, either we output five $(k+1)$-separating independent sets I_1, I_2, I_3, I_4 and I_5 in $N(\tilde{T})$ such that for any subset-fvs S of size at most k that is disjoint from $N[\tilde{T}]$, there exists a $j \in \{1, 2, 3, 4, 5\}$ such that I_j is a δ-nice independent set with respect to S; or report that every S intersects $N[\tilde{T}]$. Here, $\delta = \frac{|\tilde{T}|}{40}$.*

PROOF. We first add an edge between a pair of vertices $a, b \in N(\tilde{T})$ if $\lambda_G(a, b) \geq k+2$. Let G' denote this supergraph. Observe that, using Proposition 2.7, we can construct G' in time $\mathcal{O}(|\tilde{T}|^2 k(m + n))$. The reason we construct G' is that it ensures that any independent set in $G'[N[\tilde{T}]]$ has the property that it is a $(k+1)$-separating independent set in G. We start with the following claim.

Let $G^* = G - S$. Consider a terminal forest (F, χ) of G^* with respect to $\tilde{T}$. Each tree F_i in F is rooted at $r_i \in V(F_i)$. Let $R = \{r_1, \ldots, r_\ell\}$ denote the set of roots. Let $\tilde{T}_{\text{good}}$ be the subset of $\tilde{T}$ that consists of good terminals (that is, they have at most one effective descendant in $\tilde{T}$). By Observation 3.4 we have that $|\tilde{T}_{\text{good}}| \geq \frac{|\tilde{T}|}{2}$. Let u_t and v_t denote the neighbors of $t \in \tilde{T}_{\text{good}}$. Note that $\chi^{-1}(u_t) \neq \chi^{-1}(v_t)$, else we will have T-cycle in $G[\{t\} \cup \chi(\chi^{-1}(u_t))] \subseteq G - S$. This implies that either $\chi^{-1}(u_t)$ is a child of $\chi^{-1}(t)$ or $\chi^{-1}(v_t)$ is a child of $\chi^{-1}(t)$. Now for every vertex $t \in \tilde{T}_{\text{good}}$ we associate a neighbor u_t such that $h = \chi^{-1}(u_t)$ is a child of $\chi^{-1}(t)$. We denote u_t by sp_h. Observe that if $\chi^{-1}(t)$ is not in R then it has a unique child. Let $X = \{u_t \mid t \in \tilde{T}_{\text{good}}\}$ and $I^* = \{\chi^{-1}(w) \mid w \in X\}$. Given Y, it would be easier to work with the following definition of X. That is, $I^* = \{\mathsf{sp}_h \mid h \in Y\}$. We first show that I^* is an independent set.

CLAIM 1. *$G'[I^*]$ is an independent set and $|I^*| \geq \frac{|\tilde{T}|}{2}$.*

PROOF. We first show that $G[I^*]$ is an independent set. Before prove the main claim, we collect two useful facts.

(1) The graph G is reduced and thus $N(\tilde{T}) \cap T = \emptyset$ and thus $I^* \cap T = \emptyset$.
(2) Since every child has a unique parent, for any pair of vertices $x, y \in I^*$, we have that $\chi^{-1}(x) \neq \chi^{-1}(y)$.

Next we show the desired claim. Suppose I^* is not an independent set, then this implies that there exist vertices $x, y \in I^*$ such that xy is an edge. Since, $x, y \notin T$, by the properties of (F, χ), there exists some $b \in V(H)$ such that $x, y \in \chi(b)$. Since χ partitions $V(G^*)$ this would imply that $\chi^{-1}(x) = \chi^{-1}(y)$, a contradiction. This implies that I^* is an independent set.

For every pair of vertices $x, y \in I^*$, there is at most one internally vertex disjoint path between them in G^* and thus we can disconnect x from y with at most one vertex, say z, that is not equal to x and y (this exists as there is no edge between x and y). Hence, we can disconnected x from y in G by deleting $S \cup \{z\}$. This implies that $\lambda_G(x, y) \leq k+1$, and thus $xy \notin E(G')$. From this we conclude that I is an independent set in G' as well.

The size bound on $|I^*|$ follows from the fact that $|\tilde{T}_{\text{good}}| \geq \frac{|\tilde{T}|}{2}$. □

For every, $h \in Y$, let $\mathsf{border}(h)$ denote the vertices in $\chi(h)$ that have neighbors in $V(G^*) \setminus \chi(h)$. By the properties of terminal forests and by the definitions of sp_h we have the following.

OBSERVATION 5.1. *For every $h \in Y$, $|\mathsf{border}(h)| \leq 2$ and $\mathsf{sp}_h \in \mathsf{border}(h)$.*

Let $h \in Y$, such that $|\mathsf{border}(h)| = 2$ and let $\delta_h = \mathsf{border}(h) \setminus \{\mathsf{sp}_h\}$. We now define an auxilliary graph G_h as follows. We start with the graph $G[\chi(h)]$. Let τ and τ_S be two new vertices. Add directed edge from a vertex in $N(S) \cap \chi(h)$ to τ_S, add the directed edge (δ_h, τ) and finally add the directed edge (τ_S, τ). Of course for edges, say uv, for which no direction is specified, we have directed edges (u, v) and (v, u). Now we have the following claim.

CLAIM 2. *Let $h \in Y$, such that $|\mathsf{border}(h)| = 2$. Then either $|N(\mathsf{sp}_h) \cap S| \geq 1$ or there are two vertex disjoint paths from sp_h to τ in G_h.*

PROOF. Assume that $N(\mathsf{sp}_h) \cap S = \emptyset$. Since $G[\chi(h)]$ is connected, there exists a path, say P, from sp_h to δ_h. Let (F_T, χ_T) denote a terminal forest of $G - S$ with respect to T. Of course, (F, χ) is a terminal forest of $G - S$ with respect to $\tilde{T}$. Let $h' \in V(F_T)$ such that $\mathsf{sp}_h \in \chi_T(h')$. Clearly, $\chi_T(h') \subseteq \chi(h)$. (It could be very well be true that $\chi_T(h') = \chi(h)$.) Observe that $\chi_T(h') \cap T = \emptyset$ and let $\delta_{h'}$ denote the last vertex on P from $\chi_T(h')$.

Let t denote the terminal in $\tilde{T}_{\text{good}}$ such that $\chi^{-1}(\mathsf{sp}_h)$ is a child of $\chi^{-1}(t)$. Since t is good, there exists at most one other terminal say t' (in our case exactly one) such that $\chi^{-1}(\mathsf{sp}_h)$ is a parent of $\chi^{-1}(t)$. Furthermore, t' is a neighbor to δ_h. Let h'' be the child of h' such that $\chi_T^{-1}(t')$ is a node in the subtree of F_T rooted at h''. We denote this tree by $F_T(h'')$.

Notice that $1 \leq \lambda_{G_h}(\mathsf{sp}_h, \tau) \leq 2$. Indeed, deleting δ_h and τ_S from G_h disconnects sp_h from τ in G_h. This implies that $\lambda_{G_h}(\mathsf{sp}_h, \tau) \leq 2$. For the other inequality notice that $P\tau$ is a path from sp_h to τ in G_h. For a contradiction assume that there are no two vertex disjoint paths from sp_h to τ in G_h. Then, by Menger's theorem this implies that there exists a vertex w such that deleting w from G_h disconnects sp_h from τ in G_h. Clearly, w is on the path P. Now we distinguish between the following two cases.

(1) Suppose $w \in \chi_T(h')$. Let $R_{G_h}(\mathsf{sp}_h, w)$ denote the set of vertices of G_h reachable from sp_h in the graph $G_h - \{w\}$. We first show that $\delta_{h'} \notin R_{G_h}(\mathsf{sp}_h, w)$. Suppose not, then take a path to $\delta_{h'}$ from sp_h using vertices from $R_{G_h}(\mathsf{sp}_h, w$ and then go along the path P and reach δ_h and then take the edge (δ_h, τ). This implies that we have shown path from sp_h to τ that avoids w, a contradiction. Thus, we have shown that $\delta_{h'} \notin R_{G_h}(\mathsf{sp}_h, w)$, which in turn implies that $R_{G_h}(\mathsf{sp}_h, w) \cap \chi_T(V(F_T(h'')) = \emptyset$. Next we show that $R_{G_h}(\mathsf{sp}_h, w) \subseteq \chi_T(h')$. Suppose not, then there exists another neighbour h^* (not equal to h'') of h' in F_T and $|\chi_T(h^*) \cap R_{G_h}(\mathsf{sp}_h, w)| \geq 1$. Let xy be the edge such that $x \in R_{G_h}(\mathsf{sp}_h, w) \cap \chi_T(h')$ and $y \in R_{G_h}(\mathsf{sp}_h, w) \cap \chi_T(h^*)$. We know that G is reduced and hence every edge participates in a T-cycle. In particular there is a T-cycle C that intersects S and use the edge xy. Let P be the path on C from y to a vertex in S such that it does not use the edge xy and all the internal vertices do not belong to S. Clearly, all the internal vertices of this path belong to $\chi_T(V(F_T(h^*)))$. Here, $F_T(h^*)$ denote the subtree of F_T rooted at h^*. This implies that there is a path from sp_h to τ in G_h that avoids w, a contradiction. Thus, we assume that $R_{G_h}(\mathsf{sp}_h, w) \subseteq \chi_T(h')$. We know that sp_h is a non-terminal and hence it has degree at least three in G. Furthermore, since none of its neighbours are in S, there exists a neighbor in $R_{G_h}(\mathsf{sp}_h, w)$, say z, such that z is not equal to t or w. But then w and sp_h disconnects z from all the terminals T. This leads to a contradiction that G is reduced, as Reduction Rule 3 is applicable now.

(2) Suppose $w \notin \chi_T(h')$. We will reduce this to the previous case. Since every path from sp_h to w must pass through $\delta_{h'}$, we have that $R_{G_h}(\mathsf{sp}_h, \delta_{h'}) \subseteq R_{G_h}(\mathsf{sp}_h, w)$ and thus $\delta_{h'}$ also separates sp_h from τ in G_h. However, $\delta_{h'} \in \chi_T(h')$ and thus we end up in the previous case.

This completes the proof of the claim. $\square$

Next we prove the following claim.

Claim 3. *For every $h \in Y$, we have that $|N(\chi(h)) \cap S| \geq 1$.*

Proof. We distinguish two cases. If $|\mathsf{border}(h)| = 2$ then the claim follows from Claim 2. So assume that $|\mathsf{border}(h)| = 1$. Let t denote the terminal in $\tilde{T}_{\text{good}}$ such that $\chi^{-1}(\mathsf{sp}_h)$ is a child of $\chi^{-1}(t)$. We know that G is reduced and hence every edge participates in a T-cycle. Consider the edge $t\ \mathsf{sp}_h$. In particular there is a T-cycle C that intersects S and uses the edge $t\ \mathsf{sp}_h$. Let P be the path on C from sp_h to a vertex in S such that it does not use the edge $t\ \mathsf{sp}_h$ and all the internal vertices do not belong to S. Since $|\mathsf{border}(h)| = 1$, all the internal vertices of P are in $\chi(h)$ and thus the last vertex on this path has a neighbour in S. This concludes the proof. $\square$

Now for every vertex $h \in H$ we define two paths $\mathfrak{P}_h$ and $\mathfrak{L}_h$.

— First we deal with the case when $|\mathsf{border}(h)| = 1$. Let $\mathsf{guard}(h)$ denote a vertex in $\chi(h)$ such that $\mathsf{guard}(h) \cap S \neq \emptyset$. In this case take $\mathfrak{P}_h = \mathsf{sp}_h$ and $\mathfrak{L}_h$ to be a path from $\mathsf{guard}(h)$ to sp_h with all the internal vertices in $\chi(h) \setminus \mathfrak{P}_h$.

— Now assume that $|\mathsf{border}(h)| = 2$. If $|N(\mathsf{sp}_h) \cap S| \geq 1$ then $\mathfrak{P}_h = \mathsf{sp}_h$ and $\mathfrak{L}_h = \mathsf{sp}_h$. So assume that $N(\mathsf{sp}_h) \cap S = \emptyset$. Now, by Claim 2 we know that there are two vertex disjoint paths from sp_h to τ in G_h. Let P_1 and P_2 denote these paths. By the construction of G_h one of these paths pass through δ_h, say P_1, define $\mathfrak{P}_h$ to be the subpath from sp_h to δ_h. Again, by the construction of G_h, the path P_2 must have τ_S on it. Let x and y be two neighbours of τ_S on P_2, clearly one of them is $\tau(= x)$ and the other is some vertex $y \in \chi(h)$ such that $|N(y) \cap S| \geq 1$. Also the subpath of P_2, say P_2', from sp_h to y is entirely contained in $\chi(h)$. We denote y by $\mathsf{guard}(h)$ and let $\mathfrak{L}_h = P_2'$.

Let

$$H = G^*[V(G^*) \setminus \big(\bigcup_{h \in Y} (\chi(h) \setminus \mathfrak{P}_h) \big)].$$

That is obtain the graph H by deleting vertices of $\chi(h)$, $h \in Y$ from G^* but leaving the vertices of the special path $\mathfrak{P}_h$. Next we have to prove the following.

CLAIM 4. *For any partition (A, B) of I^* we have that $\lambda_H(A, B) = \lambda_{G^*}(A, B)$.*

PROOF. Since H is a subgraph of G^*, we have that $\lambda_H(A, B) \leq \lambda_{G^*}(A, B)$. Next we show the other direction of the proof. That is, $\lambda_H(A, B) \geq \lambda_{G^*}(A, B)$. Let $\eta = \lambda_{G^*}(A, B)$. Let $\mathcal{P} = \{P_1, \ldots, P_\eta\}$ be a family of vertex disjoint paths in G^*, between A and B, such that the number of vertices outside $V(H)$ is minimised. That is $|(\cup_{i=1}^{\eta} V(P_i)) \setminus V(H)|$ is minimum among all η sized family of pairwise vertex disjoint paths between A and B in G^*. Since $\lambda_H(A, B) < \lambda_{G^*}(A, B)$, there exists a path, say $P \in \mathcal{P}$ such that there exists a vertex $w \in V(P) \setminus V(H)$. This implies that there exists a $h \in Y$ such that $w \in (\chi(h) \setminus \mathfrak{P}_h)$. Also, note that $w \notin \mathsf{border}(h)$. Thus, w is an internal vertex on P and both its neighbors belong to $\chi(h)$. This in turn implies that P contains both the vertices of $\mathsf{border}(h)$. In fact, P contains a subpath, say P', with its endpoints being the vertices in $\mathsf{border}(h)$ and w belongs to this path. Since the paths in $\mathcal{P}$ are pairwise vertex disjoint and $|\mathsf{border}(h)| = 2$, we have that P is the only path that contains vertices from $\chi(h)$. Now we replace the subpath P' by $\mathfrak{P}_h$ and obtain a path P^*. Clearly, $\mathcal{P}' = (\mathcal{P} \cup \{P^*\}) \setminus \{P\}$ is a family of pairwise vertex disjoint paths and the number of vertices from outside $V(H)$ is strictly smaller than $|(\cup_{i=1}^{\eta} V(P_i)) \setminus V(H)|$. This contradicts the minimality of our choice of family of paths. From this, we conclude that there exists a family of vertex disjoint paths between A and B in G^* of size $\lambda_{G^*}(A, B)$ such that all the vertices on these paths belong to $V(H)$. From this we conclude that $\lambda_H(A, B) \geq \lambda_{G^*}(A, B)$. This completes the proof. □

Clearly, by our construction the family of paths $\mathcal{Q} = \{\mathfrak{L}_h \mid h \in Y\}$ satisfies the last property in the definition of nice independent sets.

If there exists a subset-fvs S that is disjoint from $N[\tilde{T}]$ then by Claim 1 we know that $G'[N[\tilde{T}]]$ has $\frac{|\tilde{T}|}{2}$-nice independent set I^* with respect to S. Let $W = N(\tilde{T})$, we know that $|W| \leq 2|\tilde{T}|$. Next we give an algorithm that either reports that there is no subset-fvs that is disjoint from $N[\tilde{T}]$ or find an independent set that contains large fraction of I^*. Towards this we do as follows.

For $i = 1$ to 5 do as follows:

— If $W \neq \emptyset$, find a maximum independent set I_i in $G'[W]$ and set $W := W - I_i$; else return I_i as an empty set.

Set $|W| = \Delta$ and let $I_1, \ldots, I_4, I_5$ be the set of independent sets. If $|I_1| < \frac{\Delta}{4}$ then we conclude that there is no subset-fvs of size at most k that is disjoint from $N[\tilde{T}]$. This follows from Claim 1. Indeed, if there is a subset-fvs of size at most k then $G'[N[\tilde{T}]]$ has an independent set of size at least $|\tilde{T}|$ and hence of size at least $\frac{\Delta}{4}$. From now onwards we assume that $|I_1| \geq \frac{\Delta}{4}$. Next we have the following claim.

CLAIM 5. *There exists $j \in \{1, 2, 3, 4, 5\}$ such that $|I_j \cap I^*| \geq \frac{|\tilde{T}|}{40}$.*

PROOF. For $i \geq 2$, we say that I_i is good if $|I_i| \geq \frac{\Delta}{4} - \frac{(i-1)\Delta}{40}$. It could be the case that all the independent sets are good. We know that $|I^*| \geq \frac{\Delta}{4}$. Since every independent set is good, the number of vertices of W they cover is at least $\sum_{i=1}^{5}(\frac{\Delta}{4} - \frac{(i-1)\Delta}{40}) \geq \Delta$ and thus some independent sets cover at least $\frac{|I^*|}{5} \geq \frac{\Delta}{20} \geq \frac{|\tilde{T}|}{20}$ vertices of I^*. Now let $i \geq 2$ be the first integer such that $|I_i| < \frac{\Delta}{4} - \frac{(i-1)\Delta}{40}$. This implies that at least $\frac{\Delta}{4} - (\frac{\Delta}{4} - \frac{(i-1)\Delta}{40}) = \frac{(i-1)\Delta}{40}$ vertices of I^* are distributed among $I_1, \ldots, I_{i-1}$ and hence one of them contains at least $\frac{\Delta}{40} \geq \frac{|\tilde{T}|}{40}$ vertices of I^*. □

Thus, we have shown that one of the independent sets has larger intersection with I^*. We can find the desired independent sets in time $2^{\mathcal{O}(|\tilde{T}|)}$ by trying all possible subsets of $N(\tilde{T})$ and checking whether they are independent. This concludes the proof of Lemma 5.5.

5.2. Deterministic sampling

In this section we give an algorithm that, given an instance (G, T, k) of SUBSET FVS with sufficiently many terminals, finds a set whose size is upper bounded by a polynomial in k, that intersects with every subset-fvs of size at most k. We first prove a probabilistic statement that shows that if we have a nice independent set I of large size then for any random partition of I, any minimum vertex cut between these two parts intersects a subset-fvs of size at most k. Finally, we use this probabilistic statement to define a scoring scheme on the vertex set of G and show that the first $\mathcal{O}(k^4)$ vertices of highest score must intersect every solution of size at most k.

LEMMA 5.6. *Let (G, T, k) be an instance of* SUBSET FVS*, S be a subset-fvs of size at most k and I be an independent set such that I is a δ-nice independent set with respect to S. Furthermore, let (A, B) be a random partition of I. Then, with probability at least $\frac{1}{2}$ any minimum vertex cut C between A and B intersects S. Here $\delta = 1000k$.*

PROOF. To prove our claim, we first make an auxiliary bipartite graph H with the vertex set $S \cup I$. Now we know that I is a δ-nice independent set with respect to S and hence there exists a subset $I^* \subseteq I$ that satisfies the following properties.

— $|I^* = \{w_1, \ldots, w_\ell\}| \geq \delta$.
— There exists an induced subgraph $\tilde{H}$ of $G - S$ containing I^* such that for any partition (A, B) of I^* we have that $\lambda_{\tilde{H}}(A, B) = \lambda_{G-S}(A, B)$.
— There exists a family of paths $\mathcal{H} = \{P_i = w_i \ldots b_i \mid i \in \{1, \ldots, \ell\}\}$, such that $b_i \in S$. Let P'_i denote the subpath of P_i that contains all but the last vertex, then the family

of paths P'_i, $i \in \{1, \ldots, \ell\}$, are pairwise disjoint and $\mathcal{Z} = \cup_{i=1}^{\ell} V(P'_i) \setminus I^*$ does not intersect $V(\tilde{H})$. That is, $\mathcal{Z} \cap V(\tilde{H}) = \emptyset$.

We add an edge between w_i and b_i in the bipartite graph H. Clearly, the number of edges in H is at least $\ell \geq \delta$.

For subsets X, Y of I such that $X \cap Y = \emptyset$, we define

$$\mathsf{score}(X,Y) = \sum_{v \in S} \min\{|N_H(v) \cap X|, |N_H(v) \cap Y|\} = \sum_{v \in S} \min\{d_X(v), d_Y(v)\}.$$

Observe that if (A, B) is a random partition (that is, every vertex is placed in A with probability $1/2$) of I then $(A \cap I^*, B \cap I^*)$ is a random partition of I^*. We denote $(A \cap I^*, B \cap I^*)$ by (A^*, B^*). We first show that with probability at least $1/2$, $\mathsf{score}(A^*, B^*) \geq k+1$. This would immediately imply that with probability at least $1/2$, $\mathsf{score}(A, B) \geq k+1$.

Let $H' := H[S \cup I^*]$. With every vertex $v \in S$, we associate a random variable X_v and for every vertex $u \in I$ we associate a random variable Y_u. A random variable Y_u takes 1 if $u \in A^*$ and 0 otherwise. Let $d(v) = d_{H'}(v)$; that is, the degree of a vertex in the graph H'. Similarly define $N(v) = N_{H'}(v)$. Let $X_v = \sum_{u \in N(v)} Y_u$. Clearly, the expectation of X_v, $\mathbf{E}[X_v] = \frac{d(v)}{2}$. Now we do some filtering in S, let $\tilde{S} = \{v \mid d(v) \geq 100\}$. For every vertex $v \in \tilde{S}$, using Chernoff bounds (see [Mitzenmacher and Upfal 2005, page numbers 70-71, Corollaries 4.9 and 4.10]) we can show that

$$\begin{aligned} \mathbf{Pr}\left(X_v < \frac{d(v)}{10}\right) &\leq e^{\frac{-8d(v)^2}{25d(v)}} \\ &\leq e^{\frac{-8\times 100}{25}} = e^{-32}. \end{aligned}$$

We can similarly bound that $\mathbf{Pr}(X_v > \frac{9d(v)}{10}) \leq e^{-32}$. Let

$$J = \left\{ v \;\middle|\; v \in \tilde{S} \bigwedge \left(X_v < \frac{d(v)}{10} \text{ or } X_v > \frac{9d(v)}{10} \right) \right\}.$$

Let Z_v, $v \in \tilde{S}$ be a random variable that is 1 if $v \in J$ and 0 otherwise. Clearly,

$$\begin{aligned} \mathbf{Pr}(Z_v = 1) &= \mathbf{Pr}\left(X_v < \frac{d(v)}{10} \text{ or } X_v > \frac{9d(v)}{10}\right) \\ &\leq \mathbf{Pr}\left(X_v < \frac{d(v)}{10}\right) + \mathbf{Pr}\left(X_v > \frac{9d(v)}{10}\right) \\ &\leq 2e^{-32}. \end{aligned}$$

$Z_J = \sum_{v \in \tilde{S}} d(v) Z_v$ be a random variable. Then, the expectation of Z_J,

$$\mathbf{E}[Z_J] = \sum_{v \in \tilde{S}} d(v) \mathbf{E}[Z_v] \leq 2e^{-32} \sum_{v \in \tilde{S}} d(v)$$

Thus the expected sum of degrees in the set J is upper bounded by $\Lambda = 2e^{-32} \sum_{v \in \tilde{S}} d(v)$. Now by Markov's inequality we have that $\mathbf{Pr}(Z_J > 2\Lambda) \leq \frac{\mathbf{E}[Z_J]}{2\Lambda} \leq \frac{1}{2}$. Define $S^* = \tilde{S} \setminus J$. Then with probability at least $\frac{1}{2}$, we have that

$$\sum_{v \in S^*} d(v) = \sum_{v \in \tilde{S}} d(v) - \sum_{v \in J} d(v) \geq (1 - 2e^{-32}) \sum_{v \in \tilde{S}} d(v),$$

and for every vertex $v \in S^*$, we have that $\frac{d(v)}{10} \leq d_{A^*}(v) \leq \frac{9d(v)}{10}$. This implies that $\frac{d(v)}{10} \leq d_{B^*}(v) \leq \frac{9d(v)}{10}$. Hence,

$$\begin{aligned}\mathsf{score}(A^*, B^*) &= \sum_{v \in S} \min\{d_{A^*}(v), d_{B^*}(v)\} \\ &\geq \sum_{v \in S^*} \min\{d_{A^*}(v), d_{B^*}(v)\} \\ &\geq \frac{1}{10} \sum_{v \in S^*} d(v) \\ &\geq \frac{(1-2e^{-32})}{10} \sum_{v \in \tilde{S}} d(v)\end{aligned}$$

Observe that the number of edges in H and H' is the same, that is, at least $1000k$. Since every vertex in $S \setminus \tilde{S}$ has degree at most 100, we have that $\sum_{v \in \tilde{S}} d(v) \geq 900k$. Thus, we have that $\mathsf{score}(A^*, B^*) \geq (1-2e^{-32})90k \geq k+1$. This implies that $\mathsf{score}(A, B) \geq k+1$ which proves the claim.

Let C be a minimum vertex cut between A and B in G. We need to show that with probability at least $\frac{1}{2}$, $C \cap S \neq \emptyset$. With probability at least $\frac{1}{2}$, we know that $\mathsf{score}(A, B) \geq k+1$. For a contradiction assume that $C \cap S = \emptyset$. Let $G^* = G - S$. We know that $\lambda_{\tilde{H}}(A, B) = \lambda_{G^*}(A, B)$. By Menger's theorem we know that A can be disconnected from B in G^* by $\lambda_{G^*}(A, B)$ vertices. This implies that $\lambda_G(A, B) \leq \lambda_{G^*}(A, B) + |S| = \lambda_{\tilde{H}}(A, B) + |S| \leq \lambda_{\tilde{H}}(A, B) + k$. Next we show that if a minimum vertex cut between A and B does not include S then the size of any vertex set separating A from B in G has size at least $\lambda_{G^*}(A, B) + k + 1$. This will lead to our desired contradiction.

Let $\mathcal{P}$ be a family of vertex disjoint paths of size $\lambda_{G^*}(A, B) (= \lambda_H(A, B))$ between A and B in H. We also have a family of paths $\mathcal{H}$ whose internal vertices are disjoint from $V(H)$. For each $v \in S$, define $\mathsf{score}(A, B)[v] = \min\{d_A(v), d_B(v)\}$. It means that we have $\mathsf{score}(A, B)[v]$ number of paths in $\mathcal{H}$ that start at vertices in A and end at v – let $\mathcal{A}$ be a set of such paths of size $\mathsf{score}(A, B)[v]$. Similarly, we have $\mathsf{score}(A, B)[v]$ number of paths in $\mathcal{H}$ that start at vertices in B and end at v – let $\mathcal{B}$ be a set of such paths of size $\mathsf{score}(A, B)[v]$. We arbitrarily pair paths from $\mathcal{A}$ and $\mathcal{B}$ and get $\mathsf{score}(A, B)[v]$ paths from A to B that are internally pairwise vertex disjoint but for the vertex $v \in S$. Thus we can obtain a family of paths $\mathcal{D}$ of size $\mathsf{score}(A, B)$ from A to B that are internally pairwise vertex disjoint but for the vertices in S. Observe that no path in $\mathcal{D}$ intersects the set of internal vertices of any path in $\mathcal{P}$. Now any minimum vertex cut from A to B in G must intersects every path in $\mathcal{P}$ and $\mathcal{D}$. However, if it does not use vertices in S then it at least needs $|\mathcal{P}| + |\mathcal{D}| \geq \lambda_{G^*}(A, B) + \mathsf{score}(A, B)[v] \geq \lambda_{G^*}(A, B) + k + 1$ vertices. However, we know that $\lambda_G(A, B) \leq \lambda_{G^*}(A, B) + k$ and thus we have proved that any minimum vertex cut C between A and B in G must intersect S. This concludes the proof of the lemma. $\square$

Now we prove the main sampling lemma of this section.

LEMMA 5.7. *Let (G, T, k) be an irreducible instance of* SUBSET FVS *and $|T| \geq \alpha k$. There is an algorithm that runs in time $2^{\mathcal{O}(k)}(m+n)$ and either reports that the given instance is a* NO*-instance or outputs a set Z of size at most $\mathcal{O}(k^4)$ such that there exists a solution S of size at most k such that $|S \cap Z| \geq 1$. Here, $\alpha = 80 \times 10^3$.*

PROOF. The algorithm first tests whether it has a T-cycle of size 2. If yes, then it returns its vertices as the desired Z. So from now onwards we assume that there is no T-cycle of size 2 in G, which also implies that G does not have parallel edges. Let $\beta = 40 \times 10^3$. Since $|T| \geq \alpha k$ we have that either the number of terminals whose degree is at least 3 in G is at least βk or the number of terminals whose degree is equal to 2 in G is at least βk.

Number of terminals of degree at least 3 is at least βk.. Let $\tilde{T} \subseteq T$ be a set of terminals of size βk with degree at least 3 in G. We first add $\tilde{T}$ to the set Z we are constructing. This takes care of all subset-fvs of size at most k that intersects $\tilde{T}$. Now we need to take care of all those subset-fvs that are disjoint from $\tilde{T}$. Now we apply Lemma 5.4 and either obtain a $1000k$-nice independent set, I, that is $k+1$ separating or that there is no subset-fvs of size at most k that is disjoint from $\tilde{T}$. If it returns that there is no subset-fvs of size at most k that is disjoint from $\tilde{T}$ then the algorithm returns the current Z. So assume that we have $1000k$-nice independent set, I, that is $k+1$ separating. Let S be a subset-fvs of size at most k that is disjoint from $\tilde{T}$. Lemma 5.4 implies that I is a $1000k$-nice independent set with respect to S. Let (A, B) be a random partition of I. Then by Lemma 5.6 we know that with probability at least $\frac{1}{2}$ any minimum vertex cut C between A and B intersects S. We use this fact to find vertices in S. Towards this we do as follows.

(1) For every vertex $v \in V(G)$ define a variable $\mathsf{score}[v]$ which is initialized to 0.
(2) For every partition (A, B) of I, find a minimum sized vertex cut C and for every vertex $v \in C$ add 1 to $\mathsf{score}[v]$. Recall that since I is $k+1$ separating $|C| \leq 2500k^2(k+1) = \mathcal{O}(k^3)$. Thus we can find the desired C in time $\mathcal{O}(k^3(m+n))$ using Proposition 2.7.
(3) Sort the vertices of G in the decreasing order of their score. For any vertex $v \in V(G)$, $\mathsf{score}[v] \leq 2^{1000k}$ and thus we can sort the vertices in time $\mathcal{O}(2^{1000k}n)$ using bucket sort.
(4) Let X be the first $5000k^3(k+1)$ vertices of highest score.
(5) Output $Z := Z \cup X$.

Clearly, $|Z| = \mathcal{O}(k^4)$ as desired. What remains to prove is that $|Z \cap S| \geq 1$. Towards this we show that $|X \cap S| \geq 1$. Let (A, B) be a random partition of I. Then by Lemma 5.6 we know that with probability at least $\frac{1}{2}$ any minimum vertex cut C between A and B intersects S. This implies that for half of all possible partitions, the corresponding minimum sized vertex cut C intersects S and thus there is a vertex $v \in S$ such that $\mathsf{score}[v] \geq \frac{2^{|I|}}{2k}$. Since, $\Lambda = \sum_{v \in V(G)} \mathsf{score}[v] \leq 2^{|I|}2500k^2(k+1)$, we have that the number of vertices with score at least $\frac{2^{|I|}}{2k}$ is upper bounded by $5000k^3(k+1)$. This implies that indeed the first $5000k^3(k+1)$ vertices of highest score contain a vertex from S. This concludes the proof for this case.

Number of terminals of degree 2 is at least βk.. Let $\tilde{T} \subseteq T$ be a set of terminals of size $6000k$ of degree 2 in G. We first add $N[\tilde{T}]$ to the set Z we are constructing. Clearly, the size of $|Z| = \mathcal{O}(k)$ until now. This takes care of all the subset-fvs of size at most k that intersects $N[\tilde{T}]$. Now we need to take care of all those subset-fvs that are disjoint from $N[\tilde{T}]$. Now we apply Lemma 5.5 and either obtain five $(k+1)$-separating independent sets $I_1, I_2, I_3.I_4$ and I_5 in $N(\tilde{T})$ such that for any subset-fvs S of size at most k that is disjoint from $N[\tilde{T}]$ there exists a $j \in \{1, 2, 3, 4, 5\}$ such that I_j is a $1000k$-nice independent set with respect to S; or report that every S intersects $N[\tilde{T}]$. If it returns that there is no subset-fvs of size at most k that is disjoint from $N[\tilde{T}]$ then the algorithm returns the current Z. So assume that we are in the other case. From now onwards the proof for this case is identical to the case of number of terminals of degree at least 3 is at least βk. Thus, we only point out the differences. We know for every subset-fvs S of size at most k, there is some I_j that is $1000k$-nice independent set and $k+1$ separating. Now we apply the algorithm described in the case of number of terminals of degree at least 3 is at least βk on each I_j, $j \in \{1,2,3,4,5\}$ and obtain X_j. Let $X := \cup_{j=1}^{5} X_j$. As before we can show that $|S \cap X| \geq 1$. We output $Z := Z \cup X$. This concludes the proof. □

5.3. Deterministic algorithm for SUBSET FVS

In this section we design the claimed deterministic algorithm for SUBSET FVS. Observe that Lemma 5.7 from the previous section is useful when the number of terminals is at least αk. To deal with the case when the number of terminals are small we design an algorithm for SUBSET FVS running in time $|T|^{\mathcal{O}(k)}(m+n)$. We first give an algorithm for a slightly different version of the problem, namely, DISJOINT SUBSET FVS. An input to DISJOINT SUBSET FVS is an undirected graph G, a subset of terminals T, and a positive integer k and the objective is to determine whether G has a subset-fvs of size at most k that is disjoint from T.

LEMMA 5.8. DISJOINT SUBSET FVS *can be solved in time* $|T|^{\mathcal{O}(k)}(m+n)$.

PROOF. Let (G,T,k) be an instance to DISJOINT SUBSET FVS. We solve the problem by a recursive branching algorithm. If there is no T-cycle then we are done. Otherwise, we first apply Lemma 3.6 and obtain an equivalent instance (G',T',k) such that G' is irreducible. For clarity we also denote (G',T',k) by (G,T,k). If $G[T]$ has a cycle then return that the given instance is a NO-instance. Now, by Observation 3.4 we know that if there exists a subset-fvs S of size at most k then half of the terminals are good. In particular, we know that there exists a terminal $t \in T$ such that it does not have any effective descendant in some terminal forest of $G-S$. Furthermore, let u_t and v_t denote the two neighbours of t in G. Let $\bar{T} = T$. Finally, for a vertex v, $\tilde{G}_v$ is the graph obtained from G by adding a new vertex τ, making τ adjacent to all vertices of T, and removing all edges between v and vertices in $\bar{T}$. Then either $u_t \in S$ or $v_t \in S$, or for one of them, say u_t, by Lemma 4.2, there exists an important u_t-τ separator, disjoint from $\bar{T}$, that is contained in a subset-fvs of size at most k. Let $W = \{\{u_t\},\{v_t\} \mid t \in T\}$. For every $w_t \in W$, we use the algorithm of Lemma 2.5 to construct the set $\mathcal{S}_{w_t}$ consisting of every important $(\{w_t\},\tau)$-separator of size at most k in $\tilde{G}_{w_t}$. We now *clean* the family $\mathcal{S}_{w_t}$, by removing any set in $\mathcal{S}_{w_t}$ that intersects $\bar{T}$. Let

$$\mathcal{P}_k = \bigcup_{\{w_t\}\in W} \mathcal{S}_{w_t},$$

be those important separators that are disjoint from $\bar{T}$. We can construct $\mathcal{S}_k$ in time $\mathcal{O}(4^k k(m+n)|T|)$ using Lemma 2.5. Now we know that if there exists a subset-fvs of size at most k that is disjoint from T then it must contain either a set from W or a set from $\mathcal{P}_k$. Therefore, we branch on the sets in W or one of the vertex cuts in $\mathcal{P}_k$: for every $S' \in \mathcal{P}_k \cup W$, we recursively solve the DISJOINT SUBSET FVS instance $(G-S',T,k-|S'|)$. If one of these branches returns a solution S, then clearly $S \cup S'$ is a subset-fvs of size at most k in G.

The correctness of the algorithm is clear from Lemma 4.2. We claim that the search tree explored by the algorithm has at most $(2|T|)^k 8^k$ leaves. We prove this by induction on k, thus let us assume that the statement is true for every value less than k.

This means that we know that the recursive call $(G - S', T, k - |S'|)$ explores a search tree with at most $(2|T|)^{k-|S'|}8^{k-|S'|}$ leaves. Note that the value of $k - |S'|$ is always nonnegative and the claim is trivial for $k = 0$. Let $\mu(k)$ denotes the number of leaves in the search tree when the parameter is k. For our computation we will also need the following known result. Let G be an undirected graph and let $X, Y \subseteq V(G)$ be two disjoint sets of vertices. If $\mathcal{S}$ is the set of all important (X, Y)-separators, then $\sum_{S \in \mathcal{S}} 4^{-|S|} \leq 1$ [Cygan et al. 2014]. We have the following recurrence on $\mu(k)$.

$$\begin{aligned}
\mu(k) &= \sum_{\{w_t\} \in W} \mu(k-1) + \sum_{\{w_t\} \in W} \sum_{S' \in \mathcal{S}_{wt}} \mu(k - |S'|) \\
&= \sum_{\{w_t\} \in W} (2|T|)^{k-1}8^{k-1} + \sum_{\{w_t\} \in W} \sum_{S' \in \mathcal{S}_{wt}} (2|T|)^{k-|S'|}8^{k-|S'|} \\
&\leq (2|T|)^k 8^{k-1} + (2|T|)^{k-1}8^k \sum_{\{w_t\} \in W} \sum_{S' \in \mathcal{S}_{wt}} 8^{-|S'|} \\
&\leq (2|T|)^k 8^{k-1} + (2|T|)^{k-1}\frac{8^k}{2} \sum_{\{w_t\} \in W} \sum_{S' \in \mathcal{S}_{wt}} 4^{-|S'|} \\
&\leq (2|T|)^k 8^{k-1} + (2|T|)^k \frac{8^k}{2} \qquad (\text{since } \sum_{S' \in \mathcal{S}_{wt}} 4^{-|S'|} \leq 1) \\
&\leq (2|T|)^k 8^{k-1}(1+4) \leq (2|T|)^k 8^k.
\end{aligned}$$

As the height of the search tree is at most k, it has $\mathcal{O}(k(2|T|)^k 8^k)$ nodes. Due to Lemma 2.5, the time spent at each node is $\mathcal{O}(4^k k(m+n)|T|)$, and we conclude that the total time of the algorithm is upper bounded by $|T|^{\mathcal{O}(k)}(m+n)$. This completes the proof. □

Now we give an algorithm for SUBSET FVS running in time $|T|^{\mathcal{O}(k)}(m+n)$.

THEOREM 5.9. SUBSET FVS *can be solved in time* $|T|^{\mathcal{O}(k)}(m+n)$.

PROOF. Let (G, T, k) be an instance to SUBSET FVS. We solve the problem by first guessing the intersection of solution with T and then calling the algorithm for DISJOINT SUBSET FVS. In other words, for every $X \subseteq T$ of size at most k, we want to find a solution that include X and is disjoint from $T \setminus X$. To obtain the desired solution we delete X from G and obtain an instance $(G - X, T \setminus X, k - |X|)$ of DISJOINT SUBSET FVS. We finally solve the instance $(G - X, T \setminus X, k - |X|)$ of DISJOINT SUBSET FVS using Lemma 5.8 in time $|T|^{\mathcal{O}(k)}(m+n)$. If for any X, we get a solution S' of size at most $k - |X|$ for the corresponding instance $(G - X, T \setminus X, k - |X|)$ of DISJOINT SUBSET FVS, then clearly $X \cup S'$ is a subset-fvs of size at most k for G. Since the number of subsets of size at most k in T is upper bounded by $k|T|^k$, the claimed running time follows. □

Finally, we combine Lemma 5.7 and Theorem 5.9 and give the main theorem of this section. That is, a deterministic algorithm for SUBSET FVS running in time $2^{\mathcal{O}(k \log k)}(m+n)$.

THEOREM 5.10. SUBSET FVS *can be solved in time* $2^{\mathcal{O}(k \log k)}(m+n)$.

PROOF. Let (G, T, k) be an instance to SUBSET FVS. We first apply Lemma 3.6 and obtain an equivalent instance (G', T', k) such that G' is reduced. For brevity we denote

(G', T', k) by (G, T, k). Set $\alpha = 48000$. The problem is solved by a recursive branching algorithm. It distinguishes two cases based on whether $|T| \geq \alpha k$ or $|T| < \alpha k$.

In the former case we apply Lemma 5.7 and in time $2^{\mathcal{O}(k)}(m+n)$ either find out that the given instance is a NO-instance or obtain a set Z of size $\mathcal{O}(k^4)$ such that there exists a solution S of size at most k such that $|S \cap Z| \geq 1$. Therefore, we branch on the choice of one of these vertices: for every vertex $z \in Z$, we recursively solve the SUBSET FVS instance $(G - \{z\}, T \setminus \{z\}, k-1)$. If one of these branches returns a solution S, then clearly $S \cup \{z\}$ is a subset-fvs of size at most k in G. In the case when $|T| < \alpha k$, we apply Theorem 5.9 and solve the problem in time $|T|^{\mathcal{O}(k)}(m+n) = 2^{\mathcal{O}(k \log k)}(m+n)$.

In every branch of the algorithm k drops by 1 and thus the number of leaves in the search tree is upper bounded by $|Z|^k$ and since the height of the tree is at most k, the number of nodes in the search tree is upper bounded by $k|Z|^k$. The time we spend at every node is either $2^{\mathcal{O}(k)}(m+n)$ or $|T|^{\mathcal{O}(k)}(m+n) \leq 2^{\mathcal{O}(k \log k)}(m+n)$ time. In either case the total time spend at any node is upper bounded by $2^{\mathcal{O}(k \log k)}(m+n)$ and thus the running time of the algorithm is upper bounded by $k|Z|^k 2^{\mathcal{O}(k \log k)}(m+n) = 2^{\mathcal{O}(k \log k)}(m+n)$. This completes the proof. $\square$

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for several useful comments on the presentation of our results.

REFERENCES

Hans L. Bodlaender. 1996. A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM J. Comput.* 25, 6 (1996), 1305–1317.

Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. 2013. An $O(c^k n)$ 5-Approximation Algorithm for Treewidth. In *FOCS*. 499–508.

Yixin Cao, Jianer Chen, and Yang Liu. 2015. On Feedback Vertex Set: New Measure and New Structures. *Algorithmica* 73, 1 (2015), 63–86. DOI:http://dx.doi.org/10.1007/s00453-014-9904-6

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

Jianer Chen, Yang Liu, and Songjian Lu. 2009. An Improved Parameterized Algorithm for the Minimum Node Multiway Cut Problem. *Algorithmica* 55, 1 (2009), 1–13.

Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. 2008. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM* 55, 5 (2008).

Rajesh Hemant Chitnis, Marek Cygan, Mohammad Taghi Hajiaghayi, and Dániel Marx. 2015. Directed Subset Feedback Vertex Set Is Fixed-Parameter Tractable. *ACM Trans. Algorithms* 11, 4 (2015), 28:1–28:28. DOI:http://dx.doi.org/10.1145/2700209

Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. to appear in 2014. *Parameterized Algorithms*. Springer-Verlag, Berlin.

Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. 2011. Solving Connectivity Problems Parameterized by Treewidth in Single Exponential Time. In *FOCS*. 150–159.

Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. 2013. Subset Feedback Vertex Set Is Fixed-Parameter Tractable. *SIAM J. Discrete Math.* 27, 1 (2013), 290–309.

Reinhard Diestel. 2010. *Graph Theory* (4th ed.). Springer-Verlag, Heidelberg.

Frederic Dorn. 2010. Planar Subgraph Isomorphism Revisited. In *STACS*. 263–274.

Guy Even, Joseph Naor, Baruch Schieber, and Leonid Zosin. 2000b. Approximating Minimum Subset Feedback Sets in Undirected Graphs with Applications. *SIAM J. Discrete Math.* 13, 2 (2000), 255–267.

Guy Even, Joseph Naor, and Leonid Zosin. 2000a. An 8-Approximation Algorithm for the Subset Feedback Vertex Set Problem. *SIAM J. Comput.* 30, 4 (2000), 1231–1252.

L. R. Ford, Jr. and D. R. Fulkerson. 1956. Maximal flow through a network. 8 (1956), 399–404.

Martin Grohe, Ken-ichi Kawarabayashi, and Bruce A. Reed. 2013. A Simple Algorithm for the Graph Minor Decomposition - Logic meets Structural Graph Theory. In *SODA*. 414–431.

Carsten Gutwenger and Petra Mutzel. 2000. A Linear Time Implementation of SPQR-Trees. In *Proc. 8th GD*. 77–90. DOI:http://dx.doi.org/10.1007/3-540-44541-2_8

John E. Hopcroft and Robert Endre Tarjan. 1973a. Dividing a Graph into Triconnected Components. *SIAM J. Comput.* 2, 3 (1973), 135–158. DOI:http://dx.doi.org/10.1137/0202012

John E. Hopcroft and Robert Endre Tarjan. 1973b. Efficient Algorithms for Graph Manipulation [H] (Algorithm 447). *Commun. ACM* 16, 6 (1973), 372–378.

Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. 2001. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.* 63, 4 (2001), 512–530.

Yoichi Iwata, Keigo Oka, and Yuichi Yoshida. 2014. Linear-Time FPT Algorithms via Network Flow. In *SODA*. 1749–1761.

Yoichi Iwata, Magnus Wahlström, and Yuichi Yoshida. 2016. Half-integrality, LP-branching, and FPT Algorithms. *SIAM J. Comput.* 45, 4 (2016), 1377–1411. DOI:http://dx.doi.org/10.1137/140962838

Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. 2014. A Near-Optimal Planarization Algorithm. In *SODA*. 1802–1811.

Naonori Kakimura, Ken-ichi Kawarabayashi, and Yusuke Kobayashi. 2012. Erdös-Pósa property and its algorithmic applications: parity constraints, subset feedback set, and subset packing. In *SODA*. 1726–1736.

Naonori Kakimura, Ken-ichi Kawarabayashi, and Dániel Marx. 2011. Packing cycles through prescribed vertices. *J. Comb. Theory, Ser. B* 101, 5 (2011), 378–381.

Ken-ichi Kawarabayashi. 2009. Planarity Allowing Few Error Vertices in Linear Time. In *FOCS*. 639–648.

Ken-ichi Kawarabayashi and Yusuke Kobayashi. 2012. Fixed-parameter tractability for the subset feedback set problem and the S-cycle packing problem. *J. Comb. Theory, Ser. B* 102, 4 (2012), 1020–1034.

Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce A. Reed. 2012. The disjoint paths problem in quadratic time. *J. Comb. Theory, Ser. B* 102, 2 (2012), 424–435.

Ken-ichi Kawarabayashi and Bojan Mohar. 2008. Graph and map isomorphism and all polyhedral embeddings in linear time. In *STOC*. 471–480.

Ken-ichi Kawarabayashi, Bojan Mohar, and Bruce A. Reed. 2008. A Simpler Linear Time Algorithm for Embedding Graphs into an Arbitrary Surface and the Genus of Graphs of Bounded Tree-Width. In *FOCS*. 771–780.

Ken-ichi Kawarabayashi and Bruce A. Reed. 2009. A nearly linear time algorithm for the half integral parity disjoint paths packing problem. In *SODA*. 1183–1192.

Ken-ichi Kawarabayashi and Bruce A. Reed. 2010. An (almost) Linear Time Algorithm for Odd Cycles Transversal. In *SODA*. 365–378.

Tomasz Kociumaka and Marcin Pilipczuk. 2014. Faster deterministic Feedback Vertex Set. *Inf. Process. Lett.* 114, 10 (2014), 556–560.

Daniel Lokshtanov and M. S. Ramanujan. 2012. Parameterized Tractability of Multiway Cut with Parity Constraints. In *ICALP(1)*. 750–761. DOI:http://dx.doi.org/10.1007/978-3-642-31594-7_63

Dániel Marx. 2006. Parameterized graph separation problems. *Theoret. Comput. Sci.* 351, 3 (2006), 394–406.

Dániel Marx and Igor Razgon. 2014. Fixed-Parameter Tractability of Multicut Parameterized by the Size of the Cutset. *SIAM J. Comput.* 43, 2 (2014), 355–388.

Michael Mitzenmacher and Eli Upfal. 2005. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press.

Rolf Niedermeier. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and its Applications, Vol. 31. Oxford University Press, Oxford. xii+300 pages.

M. Pontecorvi and Paul Wollan. 2012. Disjoint cycles intersecting a set of vertices. *J. Comb. Theory, Ser. B* 102, 5 (2012), 1134–1141.

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

M. S. Ramanujan and Saket Saurabh. 2017. Linear-Time Parameterized Algorithms via Skew-Symmetric Multicuts. *ACM Trans. Algorithms* 13, 4, Article 46 (Sept. 2017), 25 pages. DOI:http://dx.doi.org/10.1145/3128600

Igor Razgon and Barry O'Sullivan. 2009. Almost 2-SAT is fixed-parameter tractable. *J. Comput. Syst. Sci.* 75, 8 (2009), 435–450. http://dblp.uni-trier.de/db/journals/jcss/jcss75.html#RazgonO09

Neil Robertson and Paul D. Seymour. 1995. Graph Minors .XIII. The Disjoint Paths Problem. *J. Comb. Theory, Ser. B* 63, 1 (1995), 65–110.

Magnus Wahlström. 2014. Half-integrality, LP-branching and FPT Algorithms. In *SODA*. 1762–1781.