Improved FPT Algorithms for Deletion to Forest-Like Structures

Kishen N. Gowda
IIT Gandhinagar, India
kishen.gowda@iitgn.ac.in

Aditya Lonkar
IIT Madras, India
laditya1235@gmail.com

Fahad Panolan
Department of Computer Science and Engineering, IIT Hyderabad, India
fahad@cse.iith.ac.in

Vraj Patel
IIT Gandhinagar, India
vraj.patel@iitgn.ac.in

Saket Saurabh
Institute of Mathematical Sciences, Chennai, India
saket@imsc.res.in

Abstract

The Feedback Vertex Set problem is undoubtedly one of the most well-studied problems in Parameterized Complexity. In this problem, given an undirected graph G and a non-negative integer k, the objective is to test whether there exists a subset $S \subseteq V(G)$ of size at most k such that $G - S$ is a forest. After a long line of improvement, recently, Li and Nederlof [SODA, 2020] designed a randomized algorithm for the problem running in time $\mathcal{O}^\star(2.7^k)$[1]. In the Parameterized Complexity literature, several problems around Feedback Vertex Set have been studied. Some of these include Independent Feedback Vertex Set (where the set S should be an independent set in G), Almost Forest Deletion and Pseudoforest Deletion. In Pseudoforest Deletion, each connected component in $G - S$ has at most one cycle in it. However, in Almost Forest Deletion, the input is a graph G and non-negative integers $k, \ell \in \mathbb{N}$, and the objective is to test whether there exists a vertex subset S of size at most k, such that $G - S$ is ℓ edges away from a forest. In this paper, using the methodology of Li and Nederlof [SODA, 2020], we obtain the current fastest algorithms for all these problems. In particular we obtain following randomized algorithms.

1. Independent Feedback Vertex Set can be solved in time $\mathcal{O}^\star(2.7^k)$.
2. Pseudo Forest Deletion can be solved in time $\mathcal{O}^\star(2.85^k)$.
3. Almost Forest Deletion can be solved in $\mathcal{O}^\star(\min\{2.85^k \cdot 8.54^\ell, 2.7^k \cdot 36.61^\ell, 3^k \cdot 1.78^\ell\})$.

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

Keywords and phrases Parameterized Complexity, Independent Feedback Vertex Set, PseudoForest, Almost Forest, Cut and Count, Treewidth

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2020.34

Related Version A full version of the paper is available at [15], `https://arxiv.org/abs/2009.13949`.

Funding This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 819416) and Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.

[1] Polynomial dependency on the input size is hidden in $\mathcal{O}^\star$ notation.

1 Introduction

Feedback Vertex Set (FVS) is a classical NP-complete problem and has been extensively studied in all subfields of algorithms and complexity. In this problem we are given an undirected graph G and a non-negative integer k as input, and the goal is to check whether there exists a subset $S \subseteq V(G)$ (*called feedback vertex set or in short fvs*) of size at most k such that $G - S$ is a forest. This problem originated in combinatorial circuit design and found its way into diverse applications such as deadlock prevention in operating systems, constraint satisfaction and Bayesian inference in artificial intelligence. We refer to the survey by Festa et al. [14] for further details on the algorithmic study of feedback set problems in a variety of areas like approximation algorithms, linear programming and polyhedral combinatorics.

FVS has been extensively studied in Parameterized Algorithms. FVS has played a pivotal role in the development of the field of Parameterized Complexity. The earliest known FPT algorithms for FVS go back to the late 80s and the early 90s [4, 13] and used the seminal Graph Minor Theory of Robertson and Seymour. These algorithms are quite impractical because of large hidden constants in the run-time expressions. Raman et al. [31] designed an algorithm with running time $\mathcal{O}^\star(2^{\mathcal{O}(k \log \log k)})$ which basically branched on short cycles in a bounded search tree approach. For FVS, the first deterministic $\mathcal{O}^\star(c^k)$ algorithm was designed only in 2005; independently by Dehne et al. [12] and Guo et al. [16]. It is important to note here that a randomized algorithm for FVS with running time $\mathcal{O}^\star(4^k)$ was known in as early as 1999 [3]. The deterministic algorithms led to the race of improving the base of the exponent for FVS algorithms and several algorithms [6, 7, 8, 9, 17, 22, 23], both deterministic and randomized, have been designed. Until few months ago the best known deterministic algorithm for FVS ran in time $\mathcal{O}^\star(3.619^k)$ [22], while the *Cut & Count* technique by Cygan et al. [9] gave the best known randomized algorithm running in time $\mathcal{O}^\star(3^k)$. However, just in the last few months both these algorithms have been improved; Iwata and Kobayashi [17, IPEC 2019] designed the fastest known deterministic algorithm with running time $\mathcal{O}^\star(3.460^k)$ and Li and Nederlof [23, SODA 2020] designed the fastest known randomized algorithm with running time $\mathcal{O}^\star(2.7^k)$. The success on FVS has led to the study of many variants of FVS in literature such as Connected FVS [9, 28], Independent FVS [1, 24, 27], Simultaneous FVS [2, 32], Subset FVS [11, 18, 19, 20, 26], Pseudoforest Deletion [5, 29], Generalized Pseudoforest Deletion [29], and Almost Forest Deletion [30, 25].

1.1 Our Problems, Results and Methods

In this paper we study three problems around FVS, namely, Independent FVS, Almost Forest Deletion, and Pseudoforest Deletion. We first define the generalizations of forests that are considered in these problems. We say that a graph F is an ℓ-forest, if we can delete at most ℓ edges from F to get a forest. That is, F is at most ℓ edges away from being a forest. On the other hand, a *pseudoforest* is an undirected graph, in which every connected component has at most one cycle. Now, we are ready to define our problems.

Independent FVS (IFVS): Given a graph G and a non-negative integer k, does there exist a fvs S of size at most k, that is also an *independent set* in G?

Almost Forest Deletion (AFD): Given a graph G and two non-negative integers k and ℓ, does there exists a vertex subset S of size at most k such that $G - S$ is an ℓ-forest?

Pseudoforest Deletion (PDS): Given a graph G and a non-negative integer k, does there exists a vertex subset S of size at most k such that $G - S$ is a pseudoforest?

Given an instance of FVS, by subdividing every edge we get an instance of Independent FVS, showing that it generalizes FVS. On the other hand setting $\ell = 0$ in Almost Forest Deletion results in FVS. The best known algorithms for Independent FVS, Almost Forest Deletion, and Pseudoforest Deletion are $\mathcal{O}^\star(3.619^k)$ [24], $\mathcal{O}^\star(5^k 4^\ell)$ [25], and $\mathcal{O}^\star(3^k)$ [5], respectively. Our main objective is to improve over these running times for the corresponding problems. In a nutshell our paper is as follows.

Motivated by the methodology developed by Li and Nederlof [23] for FVS, we relook at several problems around FVS, such as Independent FVS, Almost Forest Deletion, and Pseudoforest Deletion, and design the current fastest randomized algorithm for these problems. Our results show that the method of Li and Nederlof [23] is extremely broad and should be applicable to more problems.

To achieve improvements and tackle Independent FVS and Almost Forest Deletion at once, we propose a more generalized version of the Almost Forest Deletion problem.

Restricted Independent Almost Forest Deletion (RIAFD) **Parameter:** k and ℓ
Input: A graph G, a vertex set $R \subseteq V(G)$, and integers k and ℓ
Question: Does there exist a vertex set $S \subseteq V(G)$ of size at most k that does not contain any element from R, that is also an independent set in G, and $G - S$ is an ℓ-forest?

Setting $\ell = 0, R = \emptyset$ we get the Independent FVS problem. A simple polynomial time reduction, where we subdivide every edge and add all the subdivision vertices to R, yields an instance of RIAFD, given an instance of Almost Forest Deletion. The reduction leaves ℓ and k unchanged.

It is worth mentioning lower bounds for the above problems. Assuming the *Exponential Time Hypothesis* to be true, we know that no algorithm running in time $2^{o(k)}$ exists for either of PDS or RIAFD (for a fixed ℓ). To describe our results, we first summarize the method of Li and Nederlof [23] (for FVS) which we adopt accordingly. The main observation guiding the method is the fact that after doing some simple preprocessing on the graph, we can ensure that a *large fraction of edges are incident on every solution* to the problem. This leads to two-step algorithms, one for the dense case and the other for the sparse case. In particular, if we are aiming for an algorithm with running time $\mathcal{O}^\star(\alpha^k)$, then we do as follows.

Dense Case: In this case, the number of edges incident to any FVS is superlinear(in k), and we select a vertex into our solution with probability at least $\frac{1}{\alpha}$.

Sparse Case: Once the dense case is done, we know that we have selected vertices, say k_1, with probability $(\frac{1}{\alpha})^{k_1}$. Now, we know that the number of edges incident to an FVS of the graph is $\mathcal{O}(k)$ and the existence of solution S of size at most k, implies that the input graph has treewidth at most $k+1$. Now, using this fact and the fact that deleting the solution leaves a graph of constant treewidth, we can actually show that graph has treewidth $(1 - \Omega(1))k = \gamma k$. This implies that if we have an algorithm on graphs of treewidth $(\mathbf{tw})$ with running time $\beta^{\mathbf{tw}}$, such that $\beta^\gamma \leq \alpha$, then we get the desired algorithm with running time $\mathcal{O}^\star(\alpha^k)$.

So a natural approach for our problems which are parameterized by the solution size is to devise an algorithm using another algorithm parameterized by treewidth with an appropriate base in the exponent, along with probabilistic reductions with a good success probability.

However, to get the best out of methods of Li and Nederlof [23], it is important to have an algorithm parameterized by treewidth with an appropriate base in the exponent that is based on the *Cut & Count* method [10]. However, for all the algorithms for problems we consider, only non *Cut & Count* algorithms were known. Thus, our first result is as follows.

▶ **Theorem 1.1.** *There exists an* $\mathcal{O}^\star\left(3^{\mathbf{tw}}\right)$ *time Monte-Carlo algorithm that given a tree decomposition of the input graph of width* $\mathbf{tw}$ *solves the following problems:*
1. Restricted-Independent Almost Forest Deletion *in exponential space.*
2. Pseudoforest Deletion *in exponential space.*

Note that a yes-instance of RIAFD has treewidth $k + \ell + 1$. Thus as our first result, we design a randomized algorithm based on Theorem 1.1 and iterative compression with running time $\mathcal{O}^\star(3^k \cdot 3^\ell)$ for RIAFD. This yields $\mathcal{O}^\star(3^k)$ and $\mathcal{O}^\star(3^k \cdot 3^\ell)$ running time algorithms for Independent FVS and Almost Forest Deletion, respectively, which take polynomial space (though, these do not appear in literature). Next, we devise probabilistic reduction rules to implement the first step in the method of Li and Nederlof [23]. We analyze these rules by modifying the analysis of their lemmas to get an $\mathcal{O}^\star(2.85^k \cdot 8.54^\ell)$ time algorithm that takes polynomial space, and an $\mathcal{O}^\star(2.7^k \cdot 36.61^\ell)$ time algorithm that takes exponential space for solving RIAFD. All these algorithms while progressively improving the dependence on k slightly, significantly worsen the dependence on ℓ. Therefore, to obtain an algorithm with an improved dependence on ℓ we describe a procedure to construct a tree decomposition of width $k + \frac{3}{5.769}\ell + \mathcal{O}(\log(\ell))$ given a riafd-set of size k. This procedure when combined with an iterative compression routine yields an $\mathcal{O}^\star(3^k \cdot 1.78^\ell)$ algorithm for RIAFD. This brings us to the following result.

▶ **Theorem 1.2.** *There exist Monte-Carlo algorithms that solve* RIAFD *problem in*
1. $\mathcal{O}^\star(3^k \cdot 3^\ell)$ *time and polynomial space.*
2. $\mathcal{O}^\star(2.85^k \cdot 8.54^\ell)$ *time and polynomial space.*
3. $\mathcal{O}^\star(2.7^k \cdot 36.61^\ell)$ *time and exponential space.*
4. $\mathcal{O}^\star(3^k \cdot 1.78^\ell)$ *time and exponential space.*

As a corollary to Theorem 1.2, we get the following result about Independent FVS.

▶ **Theorem 1.3.** *There exist Monte-Carlo algorithms that solve* Independent FVS *in:*
1. $\mathcal{O}^\star(3^{\mathbf{tw}})$ *time, given a tree decomposition of width* $\mathbf{tw}$*.*
2. $\mathcal{O}^\star(2.85^k)$ *time and polynomial space*
3. $\mathcal{O}^\star(2.7^k)$ *time and exponential space*

Although we have a deterministic $\mathcal{O}^\star(3^k)$ algorithm for Pseudoforest deletion given by Bodlaender et al. [5] which runs in exponential space, to make use of the techniques from [23] we develop our *Cut & Count* algorithm which has the same asymptotic running time. This requires some work to apply *Cut & Count* here. However, even with our *Cut & Count* algorithm, we cannot make full use of the methods of Li and Nederlof [23] and only get the following improvement.

▶ **Theorem 1.4.** *There exists a Monte-Carlo algorithm that solves* Pseudoforest Deletion *in* $\mathcal{O}^\star(2.85^k)$ *time and polynomial space.*

Due to paucity of space we prove only Theorem 1.2 (2) and (4) in this paper. The reader is referred to the full version of the paper [15] for the details that have been skipped due to paucity of space.

2 Preliminaries

For a set A, $\binom{A}{\cdot,\cdot,\cdot}$ denotes the set of all partitions of A into three subsets.

Let $G(V, E)$ or $G = (V, E)$ be an undirected graph, where V is the set of vertices and E is the set of edges. We also denote $V(G)$ to be the vertex set and $E(G)$ to be the edge set of graph G. Also, $|V| = n$ and $|E| = m$. For a vertex subset $S \subseteq V(G)$, $G[S]$ denotes the subgraph induced on the vertex set S. For $S, T \subseteq V$, $E[S, T]$ denotes the edges intersecting both S and T. For a vertex subset V', the graph $G - V'$ denotes the graph $G[V \setminus V']$. For an edge subset E', the graph $G - E'$ denotes the graph $G' = (V, E \setminus E')$. For a vertex $v \in V$, $deg(v)$ denotes the degree of the vertex, i.e., the number of edges incident on v. For a vertex subset $S \subseteq V(G)$, $deg(S) = \sum_{v\in S} deg(v)$. Given an edge $e = (u, v)$, the subdivision of the edge e is the addition of a new vertex between u and v, i.e. the edge e is replaced by two edges (u, w) and (w, v), where w is the newly added vertex. Here, w is called a "subdivision vertex".

▶ **Definition 2.1.** *A* tree decomposition *of a graph $G = (V, E)$ is a pair $\mathbb{T} = (\{B_x \mid x \in I\}, T = (I, F))$ where T a tree and $\{B_x \mid x \in I\}$ is a collection of subsets (called bags) of V, such that*

1. $\bigcup_{x\in I} B_x = V$
2. *For all $(u, v) \in E$ there is an $x \in I$ with $\{u, v\} \subseteq B_x$*
3. *For all $v \in V$, the set of nodes $\{x \in I \mid v \subseteq B_x\}$ forms a connected subtree in $T = (V, I)$*

The width *of the tree decomposition $(\{B_x \mid x \in I\}, T = (I, F))$ $\mathbb{T}$ is $max_{x\in I} |B_x| - 1$. The* treewidth *of a graph G, denoted by* $\mathbf{tw}(G)$*, is the minimum width over all tree decompositions of G.*

We sometimes abuse notation and use $\mathbf{tw}(\mathbb{T})$ to denote the width of the tree decomposition $\mathbb{T}$. For the definition above, if there are parallel edges or self loops we can just ignore them, i.e., a tree decomposition of a graph with parallel edges and self loops is just the tree decomposition of the underlying simple graph (obtained by keeping only one set of parallel edges and removing all self loops).

There is also the notion of a *nice* tree decomposition, which is used in this paper. In literature, there are a few variants of this notion that differ in details. We use the one with *introduce edge* nodes and root bag and leaf bags of size zero. A nice tree decomposition is a tree decomposition $(\{B_x \mid x \in I\}, T = (I, F))$ where T is rooted tree and the nodes are one of the following five types. With each bag in the tree decomposition, we also associate a subgraph of G; the subgraph associated with bag x is denoted $G_x = (V_x, E_x)$. We give each type together with how the corresponding subgraph is formed.

- **Leaf** nodes x. x is a leaf of T; $|B_x| = 0$ and $G_x = (\emptyset, \emptyset)$ is the empty graph.
- **Introduce vertex** nodes x. x has one child, say y. There is a vertex v with $B_x = B_y \cup \{v\}$, $v \notin B_y$ and $G_x = (V_y \cup \{v\}, E_y)$, i.e, G_x is obtained by adding an isolated vertex v to G_y.
- **Introduce edge** nodes x. x has one child, say y. There are two vertices $v, w \in B_x$, $B_x = B_y$ and $G_x = (V_y, E_y \cup \{(v, w)\})$, i.e., G_x is obtained from G_y by adding an edge between these two vertices in B_x. If we have parallel edges, we have one introduce edge node for each parallel edge. A self loop with endpoint v is handled in the same way, i.e., there is an introduce edge node with $v \in B_x$ and G_x is obtained from G_y by adding the self loop on v.
- **Forget vertex** nodes x. x has one child, say y. There is a vertex v such that $B_x = B_y \setminus \{v\}$ and G_x and G_y are the same graph.

- **Join** nodes x. x has two children, say y and z. $B_x = B_y = B_z$, $V_y \cap V_z = B_x$ and $E_y \cap E_z = \varnothing$. $G_x = (V_y \cup V_z, E_y \cup E_z)$, i.e., G_x is the union of G_y and G_z, where the vertex set B_x is the intersection of the vertex sets of these two graphs.

In this paper, we will be dealing with randomized algorithms with one-sided error-probability, i.e. only *false negatives* are possible. The *success-probability* of an algorithm is the probability that the algorithm finds a solution, given that at least one such solution exists. We define *high-probability* to be probability at least $1 - \frac{1}{2^{c|x|}}$ or sometimes $1 - \frac{1}{|x|^c}$, where $|x|$ is the input size and c is a constant. Given an algorithm with constant success-probability, we can boost it to high-probability by performing $\mathcal{O}^\star(1)$ independent trials. We cite the following folklore observation:

▶ **Lemma 2.2** (Folklore, [23]). *If a problem can be solved with success probability $\frac{1}{S}$ and in expected time T, and its solutions can be verified for correctness in polynomial time, then it can be also solved in $\mathcal{O}^\star(S \cdot T)$ time with high probability.*

We will use the following notion of separation in a graph from [23]:

▶ **Definition 2.3** (Simple Separator, [23]). *Given a graph $G(V, E)$, a partition $(A, B, S) \in \binom{V(G)}{\cdot,\cdot,\cdot}$ of V is a separation if there are no edges between A and B.*

A β-separator for a graph $G(V, E)$ is a set of vertices whose removal from G leaves no connected component of size larger than $\frac{|V|}{\beta}$ vertices, where $\beta > 0$ is some constant. Thus, a β-separator is a balanced separator of the graph. More generally, one can define a β-separator with respect to a weight function on the vertices. We now give a method to construct a β-separator of a graph G given a tree decomposition (Lemma 2.4) and its proof can be found in the full version of the paper [15].

▶ **Lemma 2.4.** *Given a graph $G(V, E)$ on n vertices with vertex weights $\omega(v)$ and its tree decomposition $\mathbb{T}$ of width $\mathbf{tw}$, for any $\beta > 0$, we can delete a set S of $\beta(\mathbf{tw} + 1)$ vertices so that every connected component of $G - S$ has weight at most $\frac{\omega(V)}{\beta}$ in polynomial time.*

In [23], the authors presented a method involving randomized reductions and small separators to get faster randomized algorithms for FVS. It turns out that this method can be generalized to work for a certain set of "vertex-deletion problems". We will now describe the basic structure of this method and will follow this outline wherever this method is used in the rest of the paper.

Throughout this outline, assume that we are working on some vertex-deletion problem $\mathcal{P}$. Let $G(V, E)$ be the graph involved in a given instance of $\mathcal{P}$. A valid solution $S \subseteq V$ is a set of vertices of G which solves the given problem instance of $\mathcal{P}$.

The method is divided into two cases: A dense case and a sparse case.

Dense Case. The algorithm goes into this case when for a given problem all the existing solution sets are of high average degree. In formal terms, every set $S \subseteq V$ of size k which is a valid solution of the given instance satisfies $deg(S) > c \cdot k$, where $c = \Theta(1)$.

To handle this case, a vertex $v \in V$ is sampled randomly based on a weight function $\omega(v)$ which depends on $deg(v)$, deletes v and makes appropriate updates to the parameters. In this paper, we use $\omega(v) = deg(v) - 2$ for all the problems discussed. This process acts like a probabilistic reduction rule for the problem as it may fail with certain probability.

Sparse Case. The algorithm goes into this case when for a given problem there exists a solution set which has low average degree. In formal terms, there exists a vertex subset $S \subseteq V$ of size k which is a valid solution of the given instance and satisfies $deg(S) \leq c \cdot k$, where $c = \mathcal{O}(1)$. Due to this reason, the number of edges in the given graph can be bounded, thus the input graph G is sparse.

The proof for the small separator lemma in [23] doesn't require the remaining graph, i.e. the graph obtained by deleting the solution set, to be a forest only. As long as there is a good β-separator of the graph $G - S$, the proof works. Lemma 2.4 helps to construct such a β-separator of size $\beta(\mathbf{tw} + 1)$ for a graph with given tree decomposition of width $\mathbf{tw}$.

The small separator helps to construct a tree decomposition of small width, given a solution set with bounded degree. The idea suggested in [23] was to use iterative compression techniques to construct a solution utilizing the small separator. This also requires solving a bounded degree version of the problem, which can be done using *Cut & Count* based algorithms. Specific details for each problem will be explained in the corresponding sections in due course.

3 Proofs of Theorem 1.2 (2) and (4)

We use the method of [23] using the outline described in Section 2. We use the term riafd-set for a solution to the given instance of RIAFD and the term afd-set for a solution to the given instance of AFD. Following is a simple reduction rule for RIAFD.

▶ **Definition 3.1** (Reduction 1). *Apply the following rules exhaustively, until the remaining graph has minimum vertex degree at least 2:*

1. *Delete all vertices of degree at most one in the input graph.*
2. *If $k = 0$ and G is not an ℓ-forest, then we have a no instance. If $k \geq 0$ and G is an ℓ-forest, we have a yes instance.*

3.1 Dense Case

Now we give a probabilistic reduction for RIAFD that capitalizes on the fact that a large number of edges are incident to the riafd-set. In particular, for a yes instance we focus on obtaining a probabilistic reduction that succeeds with probability strictly greater than 1/3 so as to achieve a randomized algorithm running in time $\mathcal{O}^\star(3-\epsilon)^k$ with high probability.

▶ **Definition 3.2** (Reduction 2 (P)). *Assume that Reduction 1 does not apply and G has a vertex of degree at least 3. Sample a vertex $v \in V$ proportional to $\omega(v) := (deg(v) - 2)$ if $v \notin R$, else $\omega(v) := 0$. That is, select each vertex with probability $\frac{\omega(v)}{\omega(V)}$. Delete v and add its neighbours to R. Decrease k by 1.*

▷ Claim 3.3. Let F be an afd-set of (G, k, ℓ). Denote $\overline{F} := V(G) \setminus F$. We have that,

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

Proof. Notice that $deg(\overline{F}) = |2E[\overline{F}, \overline{F}]| + |E[\overline{F}, F]|$. As $G[\overline{F}]$ is an ℓ-forest, $|E[\overline{F}, \overline{F}]| \leq |\overline{F}| - 1 + \ell$. Also, $|E[\overline{F}, F]| \leq deg(F)$. Therefore,

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

▶ **Lemma 3.4.** *Given a graph G, if there exists a riafd-set F of size k such that $deg(F) \geq \frac{4-2\epsilon}{1-\epsilon}(k+\ell)$, then success of Reduction 2, which is essentially sampling a vertex $v \in F$, occurs with probability at least $\frac{1}{3-\epsilon}$.*

Proof. Let $F \subseteq V(G)$ be a riafd-set of G of size exactly k. For Reduction 2 to succeed with probability at least $\frac{1}{3-\epsilon}$, we need $\frac{\omega(F)}{\omega(\overline{F})} \geq \frac{1}{2-\epsilon}$.

The value of $\omega(F)$ can be rewritten as,

$$\omega(F) = \sum_{v \in F} (deg(v) - 2) = deg(F) - 2k.$$

By Claim 3.3 (as riafd-set is also an afd-set),

$$\omega(\overline{F}) \leq \sum_{v \in \overline{F}} (deg(v) - 2) = deg(\overline{F}) - 2|\overline{F}| \leq deg(F) + 2(|\overline{F}| - 1 + \ell) - 2|\overline{F}| \leq deg(F) + 2\ell.$$

$$\text{Therefore,} \quad \frac{\omega(F)}{\omega(\overline{F})} \geq \frac{deg(F) - 2k}{deg(F) + 2\ell} = 1 - \frac{2(k+\ell)}{deg(F) + 2\ell} \overset{(\ell \geq 0)}{\geq} 1 - \frac{2(k+\ell)}{deg(F)}.$$

$$\text{Hence, we need} \quad 1 - \frac{2(k+\ell)}{deg(F)} \geq \frac{1}{2-\epsilon} \iff deg(F) \geq \frac{4-2\epsilon}{1-\epsilon}(k+\ell).$$ ◀

3.2 Sparse Case

For the sparse case, we first construct a small separator. Due to the presence of two variables (k and ℓ), we have to modify the small separator lemma in [23] with a bivariate analysis. Also, though we are discussing RIAFD, we will show how to construct a small separator assuming that we are given an afd-set, as a riafd-set is also an afd-set.

Small Separator. The main idea, as presented in [23], is to convert an afd-set with small average degree into a good tree decomposition. In particular, suppose a graph G has an afd-set F of size k with $deg(F) \leq \overline{d}(k+\ell)$, where $\overline{d} = \mathcal{O}(1)$. We show how to construct a tree decomposition of width $(1-\Omega(1))k + (2-\Omega(1))\ell$. Note that $\overline{d}$ is not exactly the average degree of F. This definition helps us to bound the width of the tree decomposition well.

Before constructing this separator, we will first see a construction of a β-separator of an ℓ-forest. We could use Lemma 2.4, but the size of the separator obtained would be $\ell \cdot o(k)$ which is huge (treewidth $\leq \ell$). We now give a method to construct a β-separator of size $\ell + o(k)$.

▶ **Lemma 3.5.** *Given an ℓ-forest $T(V,E)$ on n vertices with vertex weights $\omega(v)$, for any $\beta > 0$, we can delete a set S of $\beta + \ell$ vertices in polynomial time so that every connected component of $T - S$ has total weight at most $\frac{\omega(V)}{\beta}$.*

Proof. Construct some spanning tree for each connected component of T, call this resulting forest T'. Let X be the set of remaining edges which are not in T'. For each edge in X, delete one vertex from T'. As $|X| \leq \ell$, we will delete at most ℓ vertices. The resulting graph will still be a forest, call it T''.

Now, root every component of the forest T'' at an arbitrary vertex. Iteratively select a bag x of maximal depth whose subtree has total weight more than $\frac{\omega(V)}{\beta}$, add all vertices in B_x to S and then remove x and its subtree. The subtrees rooted at the children of v have total weight at most $\frac{\omega(V)}{\beta}$, since otherwise, v would not satisfy the maximal depth condition. Moreover, by removing the subtree rooted at v, we remove at least $\frac{\omega(V)}{\beta}$ total weight, and this can only happen β times. Thus, we delete at most $\beta + \ell$ vertices overall. ◀

With the help of Lemma 3.5, we will now proceed to the small separator lemma.

▶ **Lemma 3.6** (Small Separator). *Given an instance (G, k, ℓ) and an* afd-set *F of G of size k, define $\overline{d} := \frac{deg(F)}{k+\ell}$, and suppose that $\overline{d} = \mathcal{O}(1)$. There is a randomized algorithm running in expected polynomial time that computes a separation (A, B, S) of G such that:*

1. $|A \cap F|, |B \cap F| \geq (2^{-\overline{d}} - o(1))(k + \ell) - \ell$
2. $|S| \leq (1 + o(1))(k + \ell) - |A \cap F| - |B \cap F|$

Proof. The proof will be similar to [23, Lemma 4]. First, we fix a parameter $\epsilon := (k+\ell)^{-0.01}$ throughout the proof. Apply Lemma 3.5 to the ℓ-forest $G - F$ with $\beta = \epsilon(k + \ell)$ and vertex v weighted by $|E[v, F]|$. Let S_ϵ be the output. Observe that:

$$|S_\epsilon| \leq \ell + \epsilon(k + \ell) = \ell + o(k + \ell),$$

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

$$|E[C, F]| \leq \frac{|E[\overline{F}, F]|}{\epsilon(k + \ell)} \leq \frac{deg(F)}{\epsilon(k + \ell)} \leq \frac{\overline{d}(k + \ell)}{\epsilon(k + \ell)} = \frac{\overline{d}}{\epsilon}.$$

Now form a bipartite graph H, as in [23], i.e., on the vertex bipartition $F \uplus \mathcal{R}$, where F is the afd-set, and there are two types of vertices in $\mathcal{R}$, the component vertices and the subdivision vertices. For every connected component C in $G - F - S_\epsilon$, there is a component vertex v_C in $\mathcal{R}$ that represents that component, and it is connected to all vertices in F adjacent to at least one vertex in C. For every edge $e = (u, v)$ in $E[F, F]$, there is a vertex v_e in $\mathcal{R}$ with u and v as its neighbours. Observe that:

- $|\mathcal{R}| \leq |E[\overline{F}, F]| + 2|E[F, F]| = deg(F)$
- every vertex in $\mathcal{R}$ has degree at most $\frac{\overline{d}}{\epsilon}$
- the degree of a vertex $v \in F$ in H is at most $deg(v)$.

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

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

▷ Claim 3.7. With probability at least $\left(1 - \frac{1}{k^{\mathcal{O}(1)}}\right)$ condition (1) holds for (A, B, S).

Proof. Firstly, notice that F has at most $\epsilon(k + \ell)$ vertices with degree at least $\frac{\overline{d}}{\epsilon}$. These can be ignored as they affect condition (1) only by an additive $\epsilon(k + \ell) = o(k + \ell)$ factor. Let F' be the vertices with degree at most $\frac{\overline{d}}{\epsilon}$. Now, consider the *intersection graph* I on vertices of F' formed by connecting two vertices if they share a common neighbour (in $\mathcal{R}$). Since every vertex in F' and all the component vertices have degree at most $\frac{\overline{d}}{\epsilon}$, the maximum degree of I is at most $\left(\frac{\overline{d}}{\epsilon}\right)^2$. Color the vertices of F' with $\left(\frac{\overline{d}}{\epsilon}\right)^2 + 1$ colors such that the vertices of the same color class form an independent set in I, using the standard greedy algorithm. Note that, within each color class, the outcome of each vertex whether it joins A, B or S is independent across vertices.

Let F'_i be the set of vertices colored i. If $|F'_i| \leq k^{0.9}$, then this color class can be ignored since the sum of all such $|F'_i|$ is at most $\left(\left(\frac{\overline{d}}{\epsilon}\right)^2 + 1\right) k^{0.9} = o(k)$ and this affects condition (1) by an additive $o(k)$ factor. Henceforth, assume $|F'_i| \geq k^{0.9}$. Each vertex $v \in F'_i$ has at most $deg(v)$ neighbours in H. So, it can join A with an independent probability of $2^{-deg(v)}$. Let $X_i = |F'_i \cap A|$, then by Hoeffding's inequality [2],

$$\Pr[X_i \leq E[X_i] - k^{0.8}] \leq 2 \cdot exp\left(-2 \cdot \frac{\left(k^{0.8}\right)^2}{|F'_i|}\right) \leq 2 \cdot exp\left(-2 \cdot \frac{k^{1.6}}{k}\right) \leq \frac{1}{k^{\mathcal{O}(1)}}$$

for large enough k.

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

$$\begin{aligned} |F \cap A| &\geq \sum_{i:|F'_i| \geq k^{0.9}} \left(E[|F'_i \cap A|] - k^{0.8}\right) \\ &\geq \sum_{i:|F'_i| \geq k^{0.9}} \sum_{v \in F'_i} \left(2^{-deg(v)}\right) - k^{0.1} \cdot k^{0.8} \\ &= \sum_{v \in F'} 2^{-deg(v)} + \sum_{j=1}^{\ell} 2^0 - \ell - o(k) \\ &\geq (|F'| + \ell) \cdot 2^{-\frac{deg(F')}{|F'|+\ell}} - \ell - o(k), \end{aligned}$$

where the last inequality follows from convexity of the function 2^{-x}. Recall that $|F'| \geq k - o(k + \ell)$, and observe that $\frac{deg(F')}{|F'|+\ell} \leq \frac{deg(F)}{k+\ell} = \overline{d}$ since the vertices in $F \setminus F'$ are vertices with degree greater than some threshold. Thus,

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

proving condition (1) for A. The argument for $|B \cap F|$ is symmetric. ◁

▷ Claim 3.8. With probability at least $1 - \frac{1}{k^{\mathcal{O}(1)}}$ condition (2) holds for (A, B, S).

Proof. Note that at most $\ell + o(k + l)$ vertices in S are from S_ϵ, and the other vertices in S are from the set $F \setminus ((F \cap A) \cup (F \cap B))$ which has size $k - |A \cap F| - |B \cap F|$. Thus, the overall size of $S \leq (1 + o(1))(k + \ell) - |A \cap F| - |B \cap F|$. ◁

This completes the proof. ◀

▶ **Lemma 3.9.** *Let G be a graph and F be a* afd-set *of G of size k, and define $\overline{d} := \frac{deg(F)}{k+\ell}$. There is a randomized algorithm that, given G and F, computes a tree decomposition of G of width at most $(1 - 2^{-\overline{d}} + o(1))k + (2 - 2^{-\overline{d}} + o(1))\ell$, and runs in polynomial time in expectation.*

Proof. Compute a separation (A, B, S) following Lemma 3.6. Conditions (1) and (2) can be checked easily in polynomial time, so we can repeatedly compute a separation until they both hold. Notice that $G[A \cup S] - (F \cup S)$ is a forest, as S_ϵ includes the ℓ vertices corresponding to the ℓ extra edges of the ℓ-Forest $G - F$. Thus, $(A \cap F) \cup S$ is a fvs of $A \cup S$. The size of this fvs is,

$$|(A \cap F) \cup S| = |A \cap F| + |S| \leq (1 + o(1))(k + \ell) - |B \cap F| \leq (1 - 2^{-\overline{d}} + o(1))k + (2 - 2^{-\overline{d}} + o(1))\ell.$$

[2] We use the notation $exp(x)$ to denote the function e^x.

Therefore, we can compute a tree decomposition of $G[A \cup S]$ of width $(1 - 2^{-\overline{d}} + o(1))k + (2 - 2^{-\overline{d}} + o(1))\ell$ as follows: start with a tree decomposition of width 1 of the forest $G[A \cup S] - (F \cup S)$, and then add all vertices in $(A \cap F) \cup S$ to each bag. Call this tree decomposition of $G[A \cup S]$ as $\mathbb{T}_1$. Similarly, compute a tree decomposition of $G[B \cup S]$ in the same way, call it $\mathbb{T}_2$.

Since there is no edge connecting A to B, we can construct the tree decomposition $\mathbb{T}$ of G by simply adding an edge between an arbitrary node from $\mathbb{T}_1$ and $\mathbb{T}_2$. It is evident from the construction procedure that $\mathbb{T}$ is a valid tree decomposition of G and it takes polynomial time to compute it. ◀

As we are in the sparse case, there exists a riafd-set F of size k with bounded degree, i.e., $deg(F) \leq \overline{d}k$. We call this bounded version of the problem BRIAFD. As we saw, the small separator helps in constructing a tree decomposition of small width, but requires that we are given an afd-set of size k and bounded degree. To attain this, we use an *Iterative Compression* based procedure which at every iteration constructs a riafd-set of size at most k with bounded degree and uses it to construct the small separator. Using this small separator we construct a tree decomposition of small width and run a *Cut & Count* based procedure to solve bounded RIAFD problem for the current induced subgraph, i.e, get a riafd-set of size at most k with bounded degree. Further, we make use of the observation that since each bag of the tree decomposition consists primarily of vertices from the given afd-set, as seen in proof of Lemma 3.9, if we fix the states of these vertices in the *Cut & Count* algorithm, then there exists only a constant number of vertices in each bag for which we need to compute all states, and hence reduces the space requirement considerably from exponential (corresponding to all states of **tw** number of vertices in each bag) to polynomial space (corresponding to all states of constant number of vertices in each bag). Due to paucity of space, the details of the algorithm are deferred to the full version of the paper [15]. We state the result here.

▶ **Lemma 3.10.** *There is an algorithm* `RIAFD_IC1` *that solves* BRIAFD *in* $\mathcal{O}^\star(3^{(1-2^{-\overline{d}}+o(1))k} \cdot 3^{(2-2^{-\overline{d}}+o(1))\ell})$ *time and polynomial space.*

3.3 Combining dense and sparse cases: Algorithm for RIAFD

Having described the *Dense* and the *Sparse* Cases, we now combine them to give the final randomized algorithm. Now, we give the Algorithm `RIAFD1`(G, k, ℓ), which is the complete randomized algorithm combining the Dense and the Sparse Cases (small separator).

▶ **Lemma 3.11.** *Fix the parameter* $\epsilon \in (0,1)$ *and* $\overline{d} := \frac{4-2\epsilon}{1-\epsilon}$*, let* $c_k := max\{3 - \epsilon, 3^{1-2^{-\overline{d}}}\}$ *and* $c_\ell := 3^{2-2^{-\overline{d}}}$*. Then* `RIAFD1`$(G, R, k, \ell)$ *succeeds with probability at least* $\frac{c_k^{-k} c_\ell^{-\ell}}{k}$ *and has* $\mathcal{O}^\star(3^{o(k+\ell)})$ *expected running time.*

Proof. We will focus on running time for each iteration of the outer loop. The computation till line 6 takes $n^{\mathcal{O}(1)}$ time. For each $k'' \in (0, k']$, Line 7 is executed with probability $3^{-(1-2^{\overline{d}})k''} \cdot 3^{-(2-2^{\overline{d}})\ell}$ and takes time $\mathcal{O}^\star(3^{(1-2^{\overline{d}}+o(1))k''} \cdot 3^{(2-2^{\overline{d}}+o(1))\ell})$. So, in expectation, the total computation cost of Line 7 is $\mathcal{O}^\star(3^{o(k+\ell)})$ per value of k'', and also $\mathcal{O}^\star(3^{o(k+\ell)})$ overall. Note here that for all values of $\epsilon \in (0,1)$, $c_k \geq 2$ and $c_l \geq 1$.

Now, we prove that `RIAFD1`(G, k, ℓ) succeeds with probability $\frac{c_k^{-k} \cdot c_\ell^{-\ell}}{k}$. For simplicity of calculations, we replace k' with k. Moreover, as each iteration is an independent trial, k is an upper bound for any k' that succeeds. We use Induction on k. The statement is trivial when $k = 0$, since no probabilistic reduction is used and hence it succeeds with probability 1. For

Algorithm 1 RIAFD1(G, R, k, ℓ).

```
Input  : Graph $G = (V, E)$, a set $R$, two parameters $k \leq n$ and $\ell \leq m$
Output : Either output a riafd-set $F$ of size at most $k$, or (possibly incorrectly)
         conclude that one does not exist (Infeasible)
1  begin
2      for $0 \leq k' \leq k$ do
3          Exhaustively apply Reduction 1 to $(G, R, k', \ell)$ to get vertex set $F'$ and the
            instance $(G', R, k'', \ell)$
4          $\overline{d} \leftarrow (4 - 2\epsilon)/(1 - \epsilon)$
5          Flip a coin with Heads probability $3^{-(1-2^{\overline{d}})k''} \cdot 3^{-(2-2^{\overline{d}})\ell}$
6          if coin flipped Heads then
7              $F \leftarrow$ RIAFD_IC1($G'$,$R$,$k''$,$\ell$, $\overline{d}$)
8          else
9              Apply Reduction 2 to $(G', R', k'', \ell)$ to get vertex $v \in V$ and instance
                $(G'', R'', k'' - 1, \ell)$
10             $F \leftarrow$ RIAFD1($G''$,$R''$, $k'' - 1$, $\ell$) $\cup \{v\}$
11         if $F \cup F'$ is not Infeasible then
12             return $F \cup F'$
13     return Infeasible
```

the inductive step, consider an instance RIAFD1$(G, k+1, \ell)$. Let (G', k'', ℓ) be the reduced instance after Line 3. Suppose that every riafd-set F of G of size k'' satisfies the condition $deg(F) \leq \overline{d}k''$; here, we only need the existence of one such F. In this case, if Line 7 is executed, then it will correctly output a riafd-set F of size at most k'', with high probability by Lemma 3.10. This happens with probability at least

$$3^{-(1-2^{\overline{d}})k''} \cdot 3^{-(2-2^{\overline{d}})\ell} \cdot \left(1 - \frac{1}{n^{\mathcal{O}(1)}}\right) \geq c_k^{-k''} \cdot c_\ell^{-\ell} \cdot \frac{1}{k} \geq \frac{c_k^{-k} \cdot c_\ell^{-\ell}}{k}.$$

as desired.

Otherwise, suppose that the above condition doesn't hold for every riafd-set F of G' of size k''. This means that there exists a riafd-set F of size k'' such that $deg(F) \geq \overline{d}k''$. In this case, by Lemma 3.4, Reduction 2 succeeds with probability at least $\frac{1}{3-\epsilon}$. This is assuming, of course, that Line 7 is not executed, which happens with probability $1 - c_k^{-k''} \cdot c_\ell^{-\ell} \geq 1 - c_k^{-k''} \geq 1 - 2^{-k''} \geq 1 - \frac{1}{k''}$, since $c_l \geq 1$ and $c_k \geq 2$. By Induction, the recursive call on Line 10 succeeds with probability at least $\frac{c_k^{-(k''-1)} \cdot c_\ell^{-\ell}}{(k''-1)}$. So, the overall probability of success is at least,

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

as desired. ◀

To optimize for c_k, we set $\epsilon \approx 0.155433$, giving $c_k \leq 2.8446$ and $c_\ell \leq 8.5337$. Theorem 1.2 (2) now follows by combining Lemma 3.11 and Lemma 2.2.

3.4 Improving the Dependence on ℓ

In this subsection, we will try to reduce the dependence on ℓ in the *Cut & Count* algorithm. To achieve this, we will construct a tree decomposition with reduced dependence on ℓ (Lemma 3.13). We use the following result to prove Lemma 3.13.

▶ **Proposition 3.12** ([21, Theorem 4.7]). *Given a graph $G(V, E)$, we can obtain a tree decomposition of G of width at most $|E|/5.769 + \mathcal{O}(log(|V|))$ in polynomial time.*

▶ **Lemma 3.13.** *Given a graph $G(V, E)$ with $tw(G) > 2$ and a riafd-set F of size k, there exists a tree decomposition of width $k + \frac{3}{5.769}\ell + \mathcal{O}(\log(\ell))$ for G and it can be constructed in polynomial time.*

Proof. Given $G(V, E)$ with n vertices and m edges, we define the graph $G'(V', E') := G[V \setminus F]$. G' is an ℓ-forest from the definition of riafd-set. We apply the following reduction rules exhaustively on G':

- R_0: If there is a $v \in V'$ with $deg(v) = 0$, then remove v.
- R_1: If there is a $v \in V'$ with $deg(v) = 1$, then remove v.
- R_2: If there is a $v \in V'$ with $deg(v) = 2$, then contract v, i.e. remove v and insert a new edge between its two neighbors, if no such edge exists.

For the safeness of the above reduction rules refer to [21]. Let the reduced graph be called $G''(V'', E'')$. It is trivial to see that after applying these rules the G'' we get is also an ℓ-Forest. Therefore, after removing at most ℓ edges from G'', we are left with at most $|V''| - 1$ edges (since the remaining graph is a forest). Therefore, we get that $|E''| \leq |V''| + \ell - 1$. Since the degree of each vertex in G'' is at least 3, $|E''| \geq 3|V''|/2$. Therefore, $1.5|V''| \leq |V''| + \ell - 1$ from which we obtain the bounds $|V''| \leq 2\ell$ and $|E''| \leq 3\ell$. Proposition 3.12 implies that, G'' has a tree decomposition of width at most $\frac{3}{5.769}\ell + \mathcal{O}(\log(\ell))$ which can be computed in polynomial time.

▷ Claim 3.14 ([21, Lemma 4.2]). Given a connected graph G, with $\mathbf{tw}(G) > 2$ and let G' be a graph obtained from G by applying R_0, R_1 and R_2 then $\mathbf{tw}(G) = \mathbf{tw}(G')$

Also, from proof of Lemma 4.2 of [21], it's easy to see that this also works on graphs which might not be connected. Given these facts, we see that we can obtain a tree decomposition of G' with width at most $\frac{3}{5.769}\ell + \mathcal{O}(\log(\ell))$ in polynomial time from the tree decomposition of G''. Now to get the tree decomposition of the given graph instance G, add F (of size k which we removed) to all the bags of the tree decomposition of G'. This finally gives the required tree decomposition of G of width at most $k + \frac{3}{5.769}\ell + \mathcal{O}(\log(\ell))$. ◀

We combine the treewidth bound that can be obtained from Lemma 3.13 with *Iterative Compression*, together with the $3^{\mathbf{tw}}$ algorithm to obtain an $\mathcal{O}^{\star}(3^k 1.78^{\ell})$ algorithm for RIAFD.

We now describe the working of the routine `RIAFD_IC3`. The iterative compression routine proceeds as follows. We start with an empty graph, and add the vertices of G one by one, while always maintaining an riafd-set of size at most k in the current graph. Maintaining an RIAFD for the current graph helps us utilize Lemma 3.13 to obtain a small tree decomposition (of size $k + \frac{3}{5.769}\ell + \mathcal{O}(\log(\ell))$). Then we can add the next vertex in the ordering to all the bags in the tree decomposition to get a new riafd-set of size k in $\mathcal{O}^{\star}(3^{\mathbf{tw}})$. If we are unable to find such an riafd-set in a particular iteration, we can terminate the algorithm early.

Now we restate Theorem 1.2 (4) and prove it.

▶ **Theorem 1.2 (4).** `RIAFD_IC3` *solves* RIAFD *problem in time $\mathcal{O}^{\star}(3^k 1.78^{\ell})$ and exponential space with high probability.*

Algorithm 2 RIAFD_IC3(G, R, k, ℓ).

```
Input  : Graph $G = (V, E)$, a set $R$ and parameters $k \le n$ and $\ell = \mathcal{O}(n^2)$
Output : A RIAFD $F$ of size at most $k$ or Infeasible
1  begin
2      Order the vertices $V$ arbitrarily as $(v_1, v_2, \ldots, v_n)$
3      $F \leftarrow \varnothing$
4      for $i = 1, 2, \ldots, n$ do
5          $\mathbb{T} \leftarrow$ Compute the tree decomposition of $G[\{v_1, \ldots, v_{i-1}\}]$ by Lemma 3.13
6          Add $v_i$ to all bags of $\mathbb{T}$
7          $F \leftarrow$ a riafd-set of $G[\{v_1, \ldots, v_i\}]$ with parameters $k$ and $\ell$, computed using
            RIAFDCutandCount on $\mathbb{T}$
8          if $F$ is Infeasible then
9              return Infeasible
10     return $F$
```

Proof. Suppose that there exists a riafd-set $F^\star$ of size at most k. Let $(v_1, v_2, \ldots, v_n)$ be the ordering from Line 2, and define $V_i := \{v_1, \ldots, v_i\}$. We note that $F^\star \cap V_i$ is an RIAFD of $G[V_i]$ so RIAFD problem on Line 7 will be feasible in each iteration (and will be computed correctly with high probability in every iteration). Therefore, with high probability, an RIAFD is returned successfully (by union bound).

We now bound the running time. On Line 5, the current set F is a riafd-set of $G[V_i]$, so Lemma 3.13 guarantees a tree decomposition of width at most $k+\frac{3}{5.769}\ell+\mathcal{O}(\log(\ell))$ and adding v_i to each bag on Line 6 increases the width by at most one. By the *Cut & Count* algorithm for RIAFD from Theorem 1.1, Line 6 runs in time $\mathcal{O}^\star(3^{\left(k+\frac{3}{5.769}\ell+\mathcal{O}(\log(\ell))\right)}) = \mathcal{O}^\star(3^{\left(k+\frac{3}{5.769}\ell\right)})$ (since $\ell = \mathcal{O}(|V|^2)$ for non-trivial instance). This gives the desired time of $\mathcal{O}^\star(3^k 1.78^\ell)$ on simplification. The space bound follows directly from the description of RIAFD_IC3, Lemma 3.13 and the space bound of the *Cut & Count* algorithm. ◀

4 Conclusion

In this paper, we applied the technique of Li and Nederlof [23] to other problems around the FEEDBACK VERTEX SET problem. The technique of Li and Nederlof is inherently randomized, and it uses the *Cut & Count* technique, which is also randomized. Designing matching deterministic algorithms for these problems, as well as for FEEDBACK VERTEX SET, is a long standing open problem. However, there is a deterministic algorithm for PSEUDOFOREST DELETION running in time $\mathcal{O}^\star(3^k)$ [5]. So obtaining a deterministic algorithm for PSEUDOFOREST DELETION running in time $\mathcal{O}^\star(c^k)$ for a constant $c < 3$ is an interesting open question. Further, can we design an algorithm for PSEUDOFOREST DELETION running in time $\mathcal{O}^\star(2.7^k)$, by designing a different *Cut & Count* based algorithm for this problem? Finally, could we get a $\mathcal{O}^\star(c^k 2^{o(\ell)})$ algorithm for ALMOST FOREST DELETION, for a constant c possibly less than 3?

References

1 Akanksha Agrawal, Sushmita Gupta, Saket Saurabh, and Roohani Sharma. Improved algorithms and combinatorial bounds for independent feedback vertex set. In *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, volume 63, pages 2:1–2:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

2 Akanksha Agrawal, Daniel Lokshtanov, Amer E. Mouawad, and Saket Saurabh. Simultaneous feedback vertex set: A parameterized perspective. *TOCT*, 10(4):18:1–18:25, 2018.

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

4 Hans L. Bodlaender. On disjoint cycles. In Gunther Schmidt and Rudolf Berghammer, editors, *WG '91*, volume 570 of *LNCS*, pages 230–238. Springer, 1992.

5 Hans L. Bodlaender, Hirotaka Ono, and Yota Otachi. A faster parameterized algorithm for pseudoforest deletion. *Discret. Appl. Math.*, 236:42–56, 2018. doi:10.1016/j.dam.2017.10.018.

6 Yixin Cao. A naive algorithm for feedback vertex set. In *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, volume 61, pages 1:1–1:9. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

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

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

9 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159. IEEE Computer Society, 2011.

10 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. *CoRR*, abs/1103.0534, 2011. arXiv:1103.0534.

11 Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. Subset feedback vertex set is fixed-parameter tractable. *SIAM J. Discrete Math.*, 27(1):290–309, 2013.

12 Frank K. H. A. Dehne, Michael R. Fellows, Michael A. Langston, Frances A. Rosamond, and Kim Stevens. An $O(2^{O(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem. *Theory Comput. Syst.*, 41(3):479–492, 2007.

13 Rodney G. Downey and Michael R. Fellows. Fixed parameter tractability and completeness. In *Complexity Theory: Current Research*, pages 191–225. Cambridge University Press, 1992.

14 Paola Festa, Panos M. Pardalos, and Mauricio G.C. Resende. Feedback set problems. In *Handbook of Combinatorial Optimization*, pages 209–258. Kluwer Academic Publishers, 1999.

15 Kishen N. Gowda, Aditya Lonkar, Fahad Panolan, Vraj Patel, and Saket Saurabh. Improved fpt algorithms for deletion to forest-like structures, 2020. arXiv:2009.13949.

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

17 Yoichi Iwata and Yusuke Kobayashi. Improved analysis of highest-degree branching for feedback vertex set. In *14th International Symposium on Parameterized and Exact Computation, IPEC 2019, September 11-13, 2019, Munich, Germany*, pages 22:1–22:11, 2019.

18 Yoichi Iwata, Magnus Wahlström, and Yuichi Yoshida. Half-integrality, LP-branching, and FPT algorithms. *SIAM J. Comput.*, 45(4):1377–1411, 2016.

19 Yoichi Iwata, Yutaro Yamaguchi, and Yuichi Yoshida. 0/1/all CSPs, half-integral a-path packing, and linear-time FPT algorithms. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 462–473. IEEE Computer Society, 2018.

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

21 Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. A bound on the pathwidth of sparse graphs with applications to exact algorithms. *SIAM J. Discrete Math.*, 23:407–427, January 2009. doi:10.1137/080715482.

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

23 Jason Li and Jesper Nederlof. Detecting feedback vertex sets of size k in $O^*(2.7^k)$ time. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 971–989, 2020.

24 Shaohua Li and Marcin Pilipczuk. An improved FPT algorithm for independent feedback vertex set. In *Graph-Theoretic Concepts in Computer Science - 44th International Workshop, WG 2018, Cottbus, Germany, June 27-29, 2018, Proceedings*, volume 11159, pages 344–355. Springer, 2018.

25 Mugang Lin, Qilong Feng, Jianxin Wang, Jianer Chen, Bin Fu, and Wenjun Li. An improved FPT algorithm for almost forest deletion problem. *Inf. Process. Lett.*, 136:30–36, 2018. doi:10.1016/j.ipl.2018.03.016.

26 Daniel Lokshtanov, M. S. Ramanujan, and Saket Saurabh. Linear time parameterized algorithms for subset feedback vertex set. *ACM Trans. Algorithms*, 14(1):7:1–7:37, 2018.

27 Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, and Saket Saurabh. On parameterized independent feedback vertex set. *Theor. Comput. Sci.*, 461:65–75, 2012.

28 Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, Saket Saurabh, and Somnath Sikdar. FPT algorithms for connected feedback vertex set. *J. Comb. Optim.*, 24(2):131–146, 2012.

29 Geevarghese Philip, Ashutosh Rai, and Saket Saurabh. Generalized pseudoforest deletion: Algorithms and uniform kernel. *SIAM J. Discret. Math.*, 32(2):882–901, 2018. doi:10.1137/16M1100794.

30 Ashutosh Rai and Saket Saurabh. Bivariate complexity analysis of almost forest deletion. *Theor. Comput. Sci.*, 708:18–33, 2018. doi:10.1016/j.tcs.2017.10.021.

31 Venkatesh Raman, Saket Saurabh, and C. R. Subramanian. Faster fixed parameter tractable algorithms for undirected feedback vertex set. In *ISAAC*, volume 2518 of *Lecture Notes in Computer Science*, pages 241–248. Springer, 2002.

32 Junjie Ye. A note on finding dual feedback vertex set. *CoRR*, abs/1510.00773, 2015. arXiv:1510.00773.