Parameterized Approximation Scheme for Feedback Vertex Set

Satyabrata Jana
Institute of Mathematical Sciences, HBNI, Chennai, India

Daniel Lokshtanov
Department of Computer Science, University of California Santa Barbara, Santa Barbara, CA, USA

Soumen Mandal
Department of Mathematics, IIT Delhi, New Delhi, India

Ashutosh Rai
Department of Mathematics, IIT Delhi, New Delhi, India

Saket Saurabh
Institute of Mathematical Sciences, HBNI, Chennai, India
University of Bergen, Bergen, Norway

Abstract

Feedback Vertex Set (FVS) is one of the most studied vertex deletion problems in the field of graph algorithms. In the decision version of the problem, given a graph G and an integer k, the question is whether there exists a set S of at most k vertices in G such that $G-S$ is acyclic. It is one of the first few problems which were shown to be NP-complete, and has been extensively studied from the viewpoint of approximation and parameterized algorithms. The best-known polynomial time approximation algorithm for FVS is a 2-factor approximation, while the best known deterministic and randomized FPT algorithms run in time $\mathcal{O}^*(3.460^k)$ and $\mathcal{O}^*(2.7^k)$ respectively.[1]

In this paper, we contribute to the newly established area of parameterized approximation, by studying FVS in this paradigm. In particular, we combine the approaches of parameterized and approximation algorithms for the study of FVS, and achieve an approximation guarantee with a factor better than 2 in randomized FPT running time, that improves over the best known parameterized algorithm for FVS. We give three simple randomized $(1+\epsilon)$ approximation algorithms for FVS, running in times $\mathcal{O}^*(2^{\epsilon k} \cdot 2.7^{(1-\epsilon)k})$, $\mathcal{O}^*\left(\left(\left(\frac{4}{1+\epsilon}\right)^{(1+\epsilon)} \cdot \left(\frac{\epsilon}{3}\right)^{\epsilon}\right)^k\right)$, and $\mathcal{O}^*(4^{(1-\epsilon)k})$ respectively for every $\epsilon \in (0,1)$. Combining these three algorithms, we obtain a factor $(1+\epsilon)$ approximation algorithm for FVS, which has better running time than the best-known (randomized) FPT algorithm for every $\epsilon \in (0,1)$. This is the first attempt to look at a parameterized approximation of FVS to the best of our knowledge. Our algorithms are very simple, and they rely on some well-known reduction rules used for arriving at FPT algorithms for FVS.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Feedback Vertex Set, Parameterized Approximation

Digital Object Identifier 10.4230/LIPIcs.MFCS.2023.56

Funding *Soumen Mandal*: Supported by Council of Scientific and Industrial Research, India.
Ashutosh Rai: Supported by Science and Engineering Research Board (SERB) Grant SRG/2021/002412.
Saket Saurabh: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819416), and Swarnajayanti Fellowship (No. DST/SJF/MSA01/2017-18).

[1] here the $\mathcal{O}^\star()$ notations hides the polynomial factors in the running time of the algorithm.

1 Introduction

Vertex deletion problems are one of the most basic and well-studied classes of graph editing problems. In the decision version of these problems, given a graph G and integer k, we are asked whether we can delete at most k vertices from G such that the resulting graph satisfies certain properties. When we want the resulting graph to be empty or acyclic, then the vertex deletion problem corresponds to the well-known Vertex Cover and Feedback Vertex Set problems respectively. These were two of the first few problems to be shown NP-complete and appear in Karp's list of 21 NP-complete problems. In this paper, we concentrate on the Feedback Vertex Set problem, in the realm of parameterized approximation. The problem is formally defined as follows.

Feedback Vertex Set (FVS)	**Parameter:** k
Input: A graph $G = (V, E)$ and a positive integer k.	
Question: Does there exist a set $S \subseteq V(G)$ such that $\lvert S\rvert \leq k$ and $G - S$ is acyclic?	

Parameterized approximation combines the fields of parameterized and approximation algorithms, where we try to get the best of both worlds by achieving a better guarantee than that of the best-known approximation algorithm (running in polynomial time), while the algorithm takes FPT time, but beats the best known exact algorithm for the problem. Such algorithms are termed "parameterized approximation" algorithms in the literature. An even more ambitious goal can be to get a parameterized approximation scheme, where for every $\epsilon \in (0, \alpha - 1)$ (where α is the best-known approximation factor for polynomial-time algorithms), we get a factor $(1 + \epsilon)$ approximation algorithm, that runs in FPT time and is faster than the best known parameterized algorithm for the problem.

Probably the best example for this approach is the Min k-Cut problem (delete a minimum number of edges to get at least k connected components), which is not expected to be FPT when parameterized by k since it is W[1]-hard. The best-known approximation factor, that is possible in polynomial time, for this is 2 [20]. Gupta et al. [11] showed that we can get an approximation ratio better than 2 by allowing FPT running time, and in a recent breakthrough result, Lokshtanov et al. [19] designed a *parameterized approximation scheme* for the problem, which means that for every $\epsilon > 0$, there is an algorithm running in FPT time which gives a factor $(1 + \epsilon)$ approximation for Min k-Cut. There are a few of lower bounds results in the field as well [3, 5, 6, 17], which show that this approach does not work for certain problems.

In addition to problems that are not expected to be FPT, researchers have also studied problems that are FPT from the lens of parameterized approximation. For these problems, we try to get a better approximation factor (than what is known in polynomial time) by allowing faster FPT running time (as compared to the best FPT algorithm for the problem). This approach has been applied to many problems including Vertex Cover, d-Hitting Set [10, 4, 15], classical cut problems like Directed-FVS, Multicut [18] etc. For a comprehensive overview of the current state of parameterized approximation, we refer to the recent survey by Feldmann et al. [9], as well as the surveys conducted by Kortsarz [14] and Marx [21].

In this paper, we look at Feedback Vertex Set (FVS), which is one of the most studied problems in the field of parameterized complexity, from the parameterized approximation lens. For FVS, the best-known approximation factor that can be achieved in polynomial time is 2 [1], and this is the best approximation factor that we can hope for FVS in polynomial time under the famous Unique Games Conjecture [13]. On the other side, the fastest known deterministic and randomized FPT algorithms for FVS run in times $\mathcal{O}^*(3.460^k)$ [12] and

$\mathcal{O}^*(2.7^k)$ [16] respectively. We want to design a parameterized approximation scheme for FVS. This turns out to be a harder task than designing such schemes for problems like Vertex Cover. For Vertex Cover, there is a simple branching algorithm, that picks an edge and branches on its endpoints, but no such simple algorithm exists for FVS because the forbidden subgraph for FVS, a cycle, can be very large. For designing a good parameterized approximation scheme for Vertex Cover or d-Hitting Set, there are two ways to achieve it: i) pick disjoint copies of edges or sets in the solution and then run the fastest FPT algorithm on the remaining instance, and ii) do branching for some steps, and then run an approximation algorithm on the remaining instance. These seem difficult to achieve for FVS because the cycles can be very big, and hence the approximation guarantee in i) and the branching factor in ii) are not bounded. In this paper, we overcome these difficulties using ideas from known randomized algorithms and obtain the following result.

▶ **Theorem 1.** *There exists a randomized algorithm that, given an instance (G, k) of* FVS *and $\epsilon \in (0, 1)$, either reports a failure or finds a feedback vertex set in G of size at most $(1+\epsilon)k$ in time*

$$\mathcal{O}^*\Big(\min\Big\{2^{\epsilon k} \cdot 2.7^{(1-\epsilon)k}, \left(\left(\frac{4}{1+\epsilon}\right)^{(1+\epsilon)} \cdot \left(\frac{\epsilon}{3}\right)^{\epsilon}\right)^{k}, 4^{(1-\epsilon)k}\Big\}\Big).$$

Moreover, if G has a feedback vertex set of size at most k, the algorithm returns a solution of size at most $(1+\epsilon)k$ with probability at least $1/e$.

Our methods. For proving Theorem 1, we make use of two randomized algorithms, the first being one of the oldest algorithms for FVS that runs in time $\mathcal{O}^*(4^k)$ [2], and the other being the currently best known randomized algorithm for FVS running in time $\mathcal{O}^*(2.7^k)$ [16]. We give three different algorithms to get the running time mentioned in Theorem 1, where each of the three algorithms outperforms the other two in some range of ϵ in the interval $[0, 1]$. For a clearer view, we provide a graph depicting the running times of the three algorithms for different values of ϵ in Figure 1. We observe that for every $\epsilon \in (0, 1)$, the algorithm of Theorem 1 gives a randomized $(1+\epsilon)$ approximation and runs in time better than $\mathcal{O}^*(2.7^k)$. All our algorithms make use of some simple reduction rules for FVS, which have been extensively applied to obtain FPT algorithms for the problem. The useful property of the rules is that if none of them are applicable, then the graph has a minimum degree of at least 3.

The first algorithm that we design uses the property that in a graph with a minimum degree of at least 3, at least half the edges are incident on any feedback vertex set of the graph [2]. It picks some edges randomly, adds the endpoints of the edges to the solution, and then it runs the $\mathcal{O}^*(2.7^k)$ time algorithm of [16] on the remaining graph. This gives an algorithm running in time $\mathcal{O}^*(2.7^{(1-\epsilon)k})$ and succeeding (giving a factor $(1+\epsilon)$ approximation for every $\epsilon \in (0, 1)$) with probability $c \cdot 2^{-\epsilon k}$ for some $c \geq \frac{1}{2}$. We repeat this algorithm $\frac{1}{c} \cdot 2^{\epsilon k}$ times to get a constant probability of success, and the final running time is $\mathcal{O}^*(2^{\epsilon k} \cdot 2.7^{(1-\epsilon)k})$.

The second algorithm that we design also uses the same property, but it picks one of the endpoints randomly from a randomly picked edge in the solution (instead of both endpoints in the first algorithm). We keep doing that till we either exhaust our budget or the graph becomes acyclic. Using the techniques of [15], we show that this algorithm succeeds with probability $\left(\left(\frac{4}{1+\epsilon}\right)^{(1+\epsilon)} \cdot \left(\frac{\epsilon}{3}\right)^{\epsilon}\right)^{-k}$. Repeating this $\left(\left(\frac{4}{1+\epsilon}\right)^{(1+\epsilon)} \cdot \left(\frac{\epsilon}{3}\right)^{\epsilon}\right)^{k}$ times, we get constant success probability. The running time of this algorithm is better than the first algorithm for $\epsilon \in (0.176, 1)$.

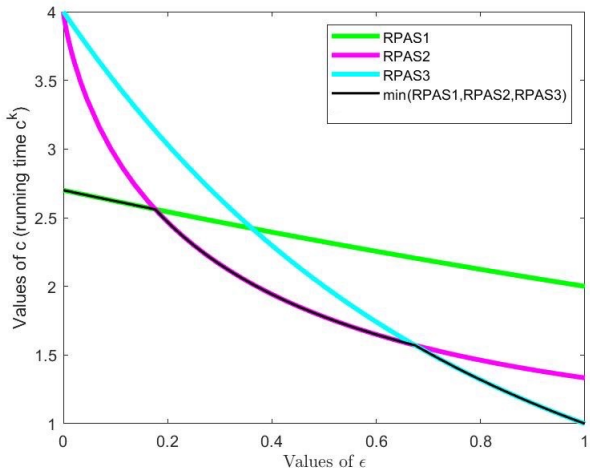

Figure 1 A graph showing running times of the three algorithms, which we name RPAS1, RPAS2, and RPAS3.

The third algorithm, instead of picking the vertices like the second algorithm does till the budget is exhausted, picks vertices up to a certain threshold, and then applies the 2-approximation algorithm of [1] on the remaining graph. This algorithm succeeds with probability $4^{-(1-\epsilon)k}$, which gives the running time of $\mathcal{O}^*(4^{(1-\epsilon)k})$ for constant success probability. This algorithm performs better than the first two algorithms for $\epsilon \in (0.674, 1)$.

The main purpose of this article is to put forward a proof of concept for designing FPT approximation algorithms for problems for which the forbidden sets are of unbounded size. All our algorithms are randomized, and getting a deterministic FPT approximation scheme for FVS is an interesting open problem.

2 Preliminaries

In this section, we give the notations and definitions, along with some known results and reduction rules which are used in the paper.

For a graph $G = (V, E)$, we denote the set of vertices of the graph by $V(G)$ and the set of edges of the graph by $E(G)$. For a set $S \subseteq V(G)$, the subgraph of G induced by S is denoted by $G[S]$ and it is defined as the subgraph of G with vertex set S and edge set $\{\{u, v\} \in E(G) : u, v \in S\}$ and the subgraph obtained after deleting S (and the edges incident to the vertices in S) is denoted as $G - S$. We say a graph G is acyclic if there is no cycle in the graph. For ease of notation, we will use uv to denote an edge of a graph instead of $\{u, v\}$. We denote the degree of a vertex $v \in V(G)$ as $d(v)$ and it is equal to the number of edges incident on v. In case of a self-loop on v, the self-loop contributes 2 to $d(v)$. The minimum degree of a graph G is denoted as $\delta(G)$ and it is defined as $\delta(G) = \min\{d(v) : v \in V(G)\}$. For a graph G, a set $S \subseteq V(G)$ is called a *feedback vertex set* of G if $G - S$ is acyclic. An instance (G, k) of FVS is said to be a `Yes` instance if there is a feedback vertex set of size at most k in G. Given a `Yes` instance (G, k) of FVS and an $\epsilon \in (0, 1)$ as an input, we say that an algorithm $\mathcal{A}$ *succeeds* if it outputs a feedback vertex set of G of size at most $(1 + \epsilon)k$. We denote the size of a minimum-sized feedback vertex set of a graph G by $\mathsf{fvs}(G)$.

Let us state the following reduction rules that we will use in this paper to make the minimum degree of the graph at least 3.

1. [8] If there is a self-loop at a vertex v, delete v from the graph and decrease k by 1.
2. [8] If there is an edge of multiplicity larger than 2, reduce its multiplicity to 2.
3. [8] If there is a vertex v of degree at most 1, delete v.
4. [8] If there is a vertex v of degree 2, delete v and connect its two neighbors by a new edge.

If none of the above four reduction rules are applicable to a graph G then we can assume that $\delta(G) \geq 3$. We will also use the following lemma to ensure that if $\delta(G) \geq 3$ then any random endpoint of a random edge of the graph is part of any feedback vertex set of G with probability at least $\frac{1}{4}$.

▶ **Lemma 2** ([8]). *Let G be a multigraph on n vertices, with minimum degree at least 3. Then, for every feedback vertex set X of G, at least half of the edges of G have at least one endpoint in X.*

3 Algorithm I

In this section, we present the first randomized $(1+\epsilon)$ approximation algorithm for FVS for every $\epsilon \in (0,1)$. Given an instance (G,k) of FVS, we first apply reduction rules 2-4 on the graph. After applying the reduction rules, we pick all the vertices having a self-loop into the set S_1 and we decrease the parameter by the number of vertices picked into S_1. If no vertex has a self-loop, then we pick an edge uv uniformly at random and add both u and v into S_1 if none of the reduction rules are applicable and $G - S_1$ is not acyclic. We decrease the parameter by 1 in this case. We do that because with good probability that one of these vertices belongs to any feedback vertex set of the graph, and hence we decrease $\mathsf{fvs}(G)$ by one with good probability. Then we delete S_1 from the graph and repeat the same process until $G - S_1$ becomes acyclic or the parameter decreases by at least ϵk. Next, we check whether $G - S_1$ is acyclic and $|S_1| \leq (1+\epsilon)k$. If yes, then we just return S_1 as a solution. Otherwise, we apply the randomized FPT algorithm of [16] for FVS on the graph $G - S_1$ with the remaining parameter. If the randomized FPT algorithm of [16] returns a solution S_2, then we return $S_1 \cup S_2$ as a solution, otherwise, we return `No`. We describe the algorithm formally in Algorithm 1. Now we state the main result of this section.

▶ **Theorem 3.** *There exists a randomized algorithm running in $\mathcal{O}^*(2^{\epsilon k} \cdot 2.7^{(1-\epsilon)k})$ time such that, given an* FVS *instance (G,k) and $\epsilon \in (0,1)$, it either reports a failure or finds a feedback vertex set of G of size at most $(1+\epsilon)k$. Moreover, if the algorithm is given a `Yes`-instance, it returns a solution of size at most $(1+\epsilon)k$ with probability at least $1/e$.*

Let us call the algorithm of Theorem 3 RPAS1. It is obtained by repeating Algorithm 1 multiple times to get a constant success probability. So before we give the proof of Theorem 3, we prove a couple of lemmas about Algorithm 1.

▶ **Lemma 4.** *If Algorithm 1 returns a set S then S is a feedback vertex set in G of size at most $(1+\epsilon)k$.*

Proof. The returned solution is either of the form S_1 or of the form $S_1 \cup S_2$ for some vertex sets S_1 and S_2. If it consists of only S_1, then by line 15 of the algorithm it is clear that $G - S_1$ is acyclic and $|S_1| \leq (1+\epsilon)k$.

Now, if the returned solution is of the form $S_1 \cup S_2$, then by the correctness of the randomized FPT algorithm of [16] for FVS, we can say that S_2 is a feedback vertex set of $G - S_1$, and this implies that $S = S_1 \cup S_2$ is a feedback vertex set of G. Also, if the parameter is decreased by β inside the while loop when Algorithm 1 returns a solution of the form $S_1 \cup S_2$, then observe that $|S_1| \leq \beta + \epsilon k$ and $|S_2| \leq k - \beta$. Thus, we get $|S| = |S_1| + |S_2| \leq \beta + \epsilon k + k - \beta = (1+\epsilon)k$. ◀

Algorithm 1 First randomized $(1+\epsilon)$ approximation algorithm for FVS.

Input : An instance (G, k) of FVS and a $\epsilon \in (0, 1)$.
Fix $l' = k$;
Fix $l = (1-\epsilon)k$;
Initialize, $S_1 \leftarrow \emptyset$;
while $k > l$ & $G - S_1$ *is not acyclic* **do**
  apply reduction rules 2-4 exhaustively to $G - S_1$;
  if *there is a self-loop* **then**
    $S_1 = S_1 \cup \{v : \text{there is a loop on } v\}$;
    $k \leftarrow k - |\{v : \text{there is a loop on } v\}|$;
  else
    pick an edge $e = uv$ uniformly at random from $E(G - S_1)$;
    $S_1 = S_1 \cup \{u, v\}$;
    $k \leftarrow k - 1$;
  end
end
if $G - S_1$ *is acyclic* & $|S_1| \leq (1+\epsilon)l'$ **then**
  return S_1;
end
apply the randomized FPT algorithm of [16] for FVS on $(G - S_1, k)$;
if *Above algorithm returns a solution* S_2 **then**
  return $S_1 \cup S_2$;
else
  return `No`;
end

▶ **Lemma 5.** *Given an* `Yes`*-instance of* FVS*, Algorithm 1 returns a solution of size at most* $(1+\epsilon)k$ *with probability at least* $c \cdot 2^{-\epsilon k}$ *for some constant* $c \geq \frac{1}{2}$.

Proof. Let (G, k) be a given `Yes`-instance for FVS. Notice that, inside the while loop, the parameter decreases when we pick vertices having a self-loop on them or when we choose an edge uniformly at random and add both of its endpoints to S_1. We do the latter only if none of the reduction rules are applicable.

Now, when we pick vertices having a self-loop into the set S_1, by the correctness of reduction rule 1, there must exist a feedback vertex set F of G of size at most k (as the given instance is a `Yes`-instance) containing those vertices having self-loops. Then we choose an edge uv uniformly at random and add both its endpoints u and v to S_1 only if none of the reduction rules are applicable and thus the minimum degree of the graph is at least 3. Then by Lemma 2, at least one of u and v is in F with probability at least $\frac{1}{2}$. Thus, when the parameter decreases by one, we add at least one vertex of F to S_1 with probability at least $\frac{1}{2}$ (the statement is true with probability 1 when we are adding the vertices having self-loops to S_1). If the parameter decreases by β (note that $\beta \geq \epsilon k$) inside the while loop, then S_1 contains at least β vertices of F with probability at least $\frac{1}{2^{\epsilon k}}$ (as we choose an edge uniformly at random inside the while loop for at most ϵk steps). Also, if S_1 contains at least β vertices from F, then $(G - S_1, k - \beta)$ is a `Yes`-instance and the randomized FPT algorithm of [16] will find feedback vertex set in $G - S_1$ of size at most $k - \beta$ with probability at least c for some constant $c \geq \frac{1}{2}$. Thus, given a `Yes`-instance (G, k) of FVS, Algorithm 1 returns a feedback vertex set of size at most $(1+\epsilon)k$ with probability at least $c \cdot 2^{-\epsilon k}$. ◀

Algorithm 2 Second randomized $(1+\epsilon)$ approximation algorithm for FVS.

Input : An instance (G, k) of FVS and $\epsilon \in (0, 1)$.
Initialize, $S \leftarrow \emptyset$;
while $|S| < (1+\epsilon)k$ & $G - S$ *is not acyclic* **do**
  apply reduction rules 2-4 exhaustively to $G - S$;
  if *there is a self-loop* **then**
    $S = S \cup \{v$: there is a self-loop on $v\}$;
  else
    pick an edge e u.a.r from $E(G - S)$;
    pick a vertex v u.a.r. from the endpoints of e ;
    $S = S \cup \{v\}$;
  end
  if $G - S$ *is acyclic* & $|S| \leq (1+\epsilon)k$ **then**
    return S;
  end
end
return `No`;

Proof of Theorem 3. The randomized FPT algorithm of [16] for FVS runs in $\mathcal{O}^*(2.7^{k'})$ time for an instance (G, k') of FVS. Notice that, when Algorithm 1 calls the algorithm of [16], the parameter is at most $(1-\epsilon)k$. Thus Algorithm 1 takes $\mathcal{O}^*(2.7^{(1-\epsilon)k})$ time when it calls the algorithm of [16]. Except for this step, all other steps in Algorithm 1 can be done in polynomial time. So overall Algorithm 1 runs in time $\mathcal{O}^*(2.7^{(1-\epsilon)k})$. Also, due to Lemma 5, given a `Yes`-instance of FVS it outputs a solution of size at most $(1+\epsilon)k$ with probability at least $c \cdot 2^{-\epsilon k}$. The algorithm of Theorem 3 (which we call RPAS1) repeats Algorithm 1 at most $\frac{1}{c} \cdot 2^{\epsilon k}$ times. The first time Algorithm 1 returns a solution, RPAS1 returns the same solution and stops. Otherwise, if all the $\frac{1}{c} \cdot 2^{\epsilon k}$ runs of Algorithm 1 return `No`, then RPAS1 returns `No` as well. This gives the running time of RPAS1 to be $\mathcal{O}^*(2^{\epsilon k} \cdot 2.7^{(1-\epsilon)k})$. We have already seen in Lemma 4 that Algorithm 1 either fails or returns a solution of size at most $(1+\epsilon)k$, so this is true for RPAS1 as well. For showing the second part of Theorem 3, we observe that RPAS1 fails if and only if each of the runs of Algorithm 1 fails. Hence, RPAS1 succeeds with probability at least $1 - (1 - c \cdot 2^{-\epsilon k})^{\frac{1}{c} \cdot 2^{\epsilon k}} \geq 1/e$. This finishes the proof of the theorem. ◀

4 Algorithm II

In this section, we present our second randomized $(1+\epsilon)$ approximation algorithm for FVS for every $\epsilon \in (0, 1)$. Given an instance (G, k) of FVS, like before, we first apply reduction rules 2-4 on the graph. When none of the reduction rules 2-4 are applicable, we pick all the vertices with self-loops into the set S. If no vertex has a self-loop, then we pick an edge uniformly at random and then we pick an endpoint (say v) of this edge uniformly at random. We add this vertex v into S. Then we delete S from the graph and repeat the same process until $G - S$ becomes acyclic or the size of S crosses $(1+\epsilon)k$. If $G - S$ becomes acyclic before the size of S crosses $(1+\epsilon)k$, then we return S as a solution, otherwise, we return `No`. We describe the algorithm formally in Algorithm 2. Now we state the main result of this section.

▶ **Theorem 6.** *There exists a randomized algorithm running in* $\mathcal{O}^*\left(\left(\left(\frac{4}{1+\epsilon}\right)^{(1+\epsilon)} \cdot \left(\frac{\epsilon}{3}\right)^{\epsilon}\right)^{k}\right)$ *time such that, given an* FVS *instance* (G, k) *and* $\epsilon \in (0, 1)$*, it either reports a failure or finds a feedback vertex set of* G *of size at most* $(1+\epsilon)k$*. Moreover, if the algorithm is given a* `Yes`*-instance, it returns a solution of size at most* $(1+\epsilon)k$ *with probability at least* $1/e$*.*

Let us call the algorithm of Theorem 6 RPAS2. Just like RPAS1, it is obtained by repeating Algorithm 2 multiple times to get a constant success probability. So before we give the proof of Theorem 6, we need to show some results about the success probability of Algorithm 2. We first make the following observation which follows directly from line 11 of the algorithm.

▶ **Observation 7.** *If Algorithm 2 returns a solution* S *for a given instance* (G, k) *of* FVS *then* S *is a feedback vertex set in* G *of size at most* $(1+\epsilon)k$*.*

Let $T(b, k, n) : \mathbb{Z} \times \mathbb{N} \times \mathbb{N} \to [0, 1]$ be defined as follows.

$$T(b,k,n) := \min_{G:\, |V(G)| \leq n,\, \mathsf{fvs}(G) \leq k} \{\rho \mid \text{Algorithm 2 returns a feedback vertex set of } G \text{ of size at most } b \text{ with probability } \rho\}$$

If we can show a lower bound on $T(b, k, n)$ where $b = (1+\epsilon)k$, and k and ϵ are as in the input of Algorithm 2, then that will give a lower bound for the success probability of Algorithm 2. We first make the following observation.

▶ **Observation 8.** $T(b, k, n) \leq T(b, k, n-1)$ *and* $T(b, k, n) \leq T(b, k-1, n)$*.*

Proof. The first part of the observation follows from the fact that the set of graphs considered for $T(b, k, n-1)$ while taking the minimum in the definition is a subset of the set of graphs considered for $T(b, k, n)$. The second part also follows by a similar reasoning on the size of the feedback vertex set. ◀

Now, let us see what happens when Algorithm 2 adds v_i to S. Let E_1 denote the event when this v_i belongs to some feedback vertex set of G of size at most k, and let E_2 denote the event when v_i is not part of any feedback vertex set of G of size at most k. The probability of success of Algorithm 2 on G with parameter k and budget b is at least $T(b-1, k-1, n-1)$ and $T(b-1, k, n-1)$ respectively in case E_1 or E_2 happens. This gives us the following.

$$T(b,k,n) \geq \Pr[E_1] \cdot T(b-1, k-1, n-1) + \Pr[E_2] \cdot T(b-1, k, n-1).$$

Now, if we add v_i to S from line 5 of Algorithm 2 and G has a feedback vertex set of size at most k, then there must exists a feedback vertex set F of G of size at most k that contains v_i. Else, if v_i is added to S from line 9, then none of the reduction rules are applicable and by Lemma 2, we have that, $\Pr[E_1] \geq \frac{1}{4}$. That is, $\Pr[E_1] = \frac{1}{4} + c_i$ for some $c_i \geq 0$. We also know that $\Pr[E_1] + \Pr[E_2] = 1$. Hence, the recurrence relation of success probability is

$$T(b,k,n) \geq (\frac{1}{4} + c_i) \cdot T(b-1, k-1, n-1) + (\frac{3}{4} - c_i) \cdot T(b-1, k, n-1).$$

Now, using Observation 8, we can write

$$T(b,k,n) \geq \frac{1}{4} \cdot T(b-1, k-1, n) + \frac{3}{4} \cdot T(b-1, k, n). \tag{1}$$

Since n is an invariant in the above recurrence relation, we can rewrite 1 as

$$T(b,k) \geq \frac{1}{4} \cdot T(b-1,k-1) + \frac{3}{4} \cdot T(b-1,k). \tag{2}$$

It can be easily seen that two trivial base cases of the recurrence 2 are the following.
i) $T(b,k) = 0$ when $b < 0$, and **ii)** $T(b,k) = 1$ when $k = 0$ and $b \geq 0$.

4.1 Solution of recurrence relation 2 and proof of Theorem 6

To solve the recurrence 2, we make use of the results in [15]. Let, a recurrence relation define a function $p : \mathbb{Z} \times \mathbb{N} \to [0,1]$ satisfying the following equations.

$$\begin{aligned} p(b,k) &= \min_{\{1 \leq j \leq N \mid \bar{k}^j \leq k\}} \sum_{i=1}^{r_j} \bar{\gamma}_i^j \cdot p\left(b - \bar{b}_i^j, k - \bar{k}_i^j\right) \\ p(b,k) &= 0 \qquad \forall b < 0, k \in \mathbb{N} \\ p(b,0) &= 1 \qquad \forall b \geq 0, \end{aligned} \tag{3}$$

where $N \in \mathbb{N}$, and for any $1 \leq j \leq N$ the following hold: $\bar{b}^j \in \mathbb{N}_+^{r_j}, \bar{k}^j \in \mathbb{N}^{r_j}$ and $\bar{\gamma}^j \in \mathbb{R}_+^{r_j}$ with $\sum_{i=1}^{r_j} \bar{\gamma}_i^j = 1$. We say that $\bar{k}^j \leq k$ if $\bar{k}_i^j \leq k, \forall 1 \leq i \leq r_j$. We refer to the recurrence relation in 3 as the composite recurrence of $\left\{\left(\bar{b}^j, \bar{k}^j, \bar{\gamma}^j\right) \mid 1 \leq j \leq N\right\}$. Note that for the recurrence to be properly defined, there must be $1 \leq j \leq N$ such that $\bar{k}^j \leq 1$ (otherwise the min operation in 3 may be taken over an empty set). Also notice that the recurrence 2 is a composite recurrence with $N = 1$ (and thus $j = 1$), $r_j = r = 2$, $\bar{\gamma}_1 = \frac{1}{4}$, $\bar{\gamma}_2 = \frac{3}{4}$, $\bar{b}_1 = 1$, $\bar{b}_2 = 1$, $\bar{k}_1 = 1$ and $\bar{k}_2 = 0$. Throughout this subsection, we use the word term when referring to triples of the form $\left(\bar{b}^j, \bar{k}^j, \bar{\gamma}^j\right)$.

We say that a vector $\bar{q} \in \mathbb{R}_{\geq 0}^r$ is a distribution if $\sum_{i=1}^r \bar{q}_i = 1$ and use $D(\cdot\|\cdot)$ to denote **Kullback-Leibler divergence** [7] [2] [3].

To state the main result of [15], we need the next definition. For short, associate the term $(\bar{b}, \bar{k}, \bar{\gamma})$ with the expression $\sum_{i=1}^r \bar{\gamma}_i \cdot p\left(b - \bar{b}_i, k - \bar{k}_i\right)$.

▶ **Definition 9.** *Let $\bar{b} \in \mathbb{N}_+^r, \bar{k} \in \mathbb{N}^r$ and $\bar{\gamma} \in \mathbb{R}_{\geq 0}^r$ with $\sum_{i=1}^r \bar{\gamma}_i = 1$. Then for $\alpha > 0$, the α branching number of the term $(\bar{b}, \bar{k}, \bar{\gamma})$ is the optimal value M^* of the following minimization problem over $\bar{q} \in \mathbb{R}_{\geq 0}^r$:*

$$M^* = \min \left\{ \frac{1}{\sum_{i=1}^r \bar{q}_i \cdot \bar{k}_i} D(\bar{q}\|\bar{\gamma}) \mid \sum_{i=1}^r \bar{q}_i \cdot \bar{b}_i \leq \alpha \sum_{i=1}^r \bar{q}_i \cdot \bar{k}_i, \bar{q} \text{ is a distribution} \right\}.$$

If the optimization above does not have a feasible solution then $M^* = \infty$.

The main result of [15] is the following.

▶ **Theorem 10** ([15]). *Let p be the composite recurrence of $\left\{\left(\bar{b}^j, \bar{k}^j, \bar{\gamma}^j\right) \mid 1 \leq j \leq N\right\}$, and $\alpha > 0$. Denote by M_j the α-branching number of $\left(\bar{b}^j, \bar{k}^j, \bar{\gamma}^j\right)$, and let $M = \max\{M_j \mid 1 \leq j \leq N\}$. If $M < \infty$ then*

$$\lim_{k \to \infty} \frac{\log p(\lfloor \alpha k \rfloor, k)}{k} = -M.$$

[2] Formally, for $\bar{c}, \bar{d} \in \mathbb{R}^k$ define $D(\bar{c}\|\bar{d}) = \sum_{i=1}^k \bar{c}_i \log \frac{\bar{c}_i}{\bar{d}_i}$.
[3] Throughout the paper we refer by log to the natural logarithm.

As said earlier, for the recurrence 2, we have $N = 1$, and thus, j will always have only one value, so we just ignore j. Also, we have $r_j = r = 2$, $\bar{\gamma}_1 = \frac{1}{4}$, $\bar{\gamma}_2 = \frac{3}{4}$, $\bar{b}_1 = 1$, $\bar{b}_2 = 1$, $\bar{k}_1 = 1$ and $\bar{k}_2 = 0$. The α-branching number of $(\bar{b}, \bar{k}, \bar{\gamma})$ is,

$$M = \min \left\{ \frac{1}{\sum_{i=1}^{2} \bar{q}_i \cdot \bar{k}_i} D(\bar{q} \| \bar{\gamma}) \mid \sum_{i=1}^{2} \bar{q}_i \cdot \bar{b}_i \leq \alpha \sum_{i=1}^{2} \bar{q}_i \cdot \bar{k}_i, \bar{q} \text{ is a distribution } \right\}.$$

So, we have to minimize

$$\frac{1}{\sum_{i=1}^{2} \bar{q}_i \cdot \bar{k}_i} D(\bar{q} \| \bar{\gamma}) \tag{4}$$

with respect to the constraint,

$$\sum_{i=1}^{2} \bar{q}_i \cdot \bar{b}_i \leq \alpha \sum_{i=1}^{2} \bar{q}_i \cdot \bar{k}_i \tag{5}$$

Also, since $\bar{q}$ is a distribution, we have

$$\bar{q}_1 + \bar{q}_2 = 1 \tag{6}$$

So, using 6 in 5 we get,

$$\bar{q}_1 + \bar{q}_2 \leq \alpha \bar{q}_1 \implies \bar{q}_1 \geq \frac{1}{\alpha} \tag{7}$$

Now, from 4, we get,

$$\frac{1}{\sum_{i=1}^{2} \bar{q}_i \cdot \bar{k}_i} D(\bar{q} \| \bar{\gamma}) = \frac{1}{\bar{q}_1} \left(\bar{q}_1 \log \frac{\bar{q}_1}{\bar{\gamma}_1} + \bar{q}_2 \log \frac{\bar{q}_2}{\bar{\gamma}_2} \right) = \left(\log 4\bar{q}_1 + \frac{(1 - \bar{q}_1)}{\bar{q}_1} \log \frac{4(1 - \bar{q}_1)}{3} \right).$$

Notice that, the above expression is a function of $\bar{q}_1$ only. Let us consider the function, $\mathsf{f} : [\frac{1}{\alpha}, 1] \to \mathbb{R}$ such that,

$$\mathsf{f}(x) = \left(\log 4x + \frac{1 - x}{x} \log \frac{4(1 - x)}{3} \right).$$

Since we want a factor $(1 + \epsilon)$ approximation for every $\epsilon \in (0, 1)$, we are interested for $1 < \alpha < 2$ only and in this range of α, $\frac{1}{\alpha} > \frac{1}{2}$. Now, to compute M, we need to minimize $\mathsf{f}(x)$.

Here, derivative of $\mathsf{f}(x)$,

$$\mathsf{f}'(x) = -\frac{1}{x^2} \log \frac{4(1 - x)}{3} \geq 0, \forall x \in \left[\frac{1}{4}, 1 \right].$$

This implies that the function f is a monotonically increasing function on the domain of our interest (i.e., where $\frac{1}{\alpha} > \frac{1}{2}$). Hence, $\mathsf{f}(x)$ is minimum at $x = \frac{1}{\alpha}$ and this implies

$$M = \mathsf{f}\left(\frac{1}{\alpha} \right) = \log \left(\frac{4}{\alpha} \left(\frac{4}{3\alpha} (\alpha - 1) \right)^{(\alpha - 1)} \right).$$

By Theorem 10,

$$p(\alpha k, k) = \left(\frac{4}{\alpha} \left(\frac{4}{3\alpha} (\alpha - 1) \right)^{(\alpha - 1)} \right)^{-k}.$$

Now, from recurrence relation 2 and putting $\alpha = (1 + \epsilon)$, we get,

$$T((1 + \epsilon)k, k) \geq \left(\left(\frac{4}{1 + \epsilon} \right)^{(1 + \epsilon)} \cdot \left(\frac{\epsilon}{3} \right)^{\epsilon} \right)^{-k}.$$

Proof of Theorem 6. Showing that $T(b,k,n) \geq \left(\left(\frac{4}{1+\epsilon}\right)^{(1+\epsilon)} \cdot \left(\frac{\epsilon}{3}\right)^{\epsilon}\right)^{-k}$ shows that given a Yes-instance (G,k) of FVS, Algorithm 2 succeeds with probability at least $\left(\left(\frac{4}{1+\epsilon}\right)^{(1+\epsilon)} \cdot \left(\frac{\epsilon}{3}\right)^{\epsilon}\right)^{-k}$. Now, the proof follows along the lines of the proof of Theorem 3, and by repeating Algorithm 2 $\left(\left(\frac{4}{1+\epsilon}\right)^{(1+\epsilon)} \cdot \left(\frac{\epsilon}{3}\right)^{\epsilon}\right)^{k}$ times, we get success probability at least $1/e$. Since Algorithm 2 runs in polynomial time, the running time of RPAS2 is $\mathcal{O}^*\left(\left(\left(\frac{4}{1+\epsilon}\right)^{(1+\epsilon)} \cdot \left(\frac{\epsilon}{3}\right)^{\epsilon}\right)^{k}\right)$. ◀

5 Algorithm III

In this section, we present the third randomized $(1+\epsilon)$ approximation algorithm for FVS for every $\epsilon \in (0,1)$. Given an instance (G,k) of FVS, we first apply reduction rules 2-4 on the graph. After this, we pick all the vertices with self-loops into the set S_1. If no vertex has a self-loop, then we pick an edge uniformly at random, select an endpoint v of this edge uniformly at random, and add this vertex v into S_1. Then we delete S_1 from the graph and repeat the same process until $G - S_1$ becomes acyclic or the size of S_1 crosses $(1-\epsilon)k$. Next, we check whether $G - S_1$ is acyclic and $|S_1| \leq (1+\epsilon)k$. If yes, then we just return S_1 as a solution. Otherwise, we apply a 2-approximation algorithm [1] for FVS on the graph $G - S_1$. If the 2-approximate solution, say S_2, of $G - S_1$ is of size at most $2(k - |S_1|)$ then we return $S_1 \cup S_2$ as a solution, otherwise we return No. We describe the algorithm formally in Algorithm 3. Now we state the main result of this section.

▶ **Theorem 11.** *There exists a randomized algorithm running in $\mathcal{O}^*\left(4^{(1-\epsilon)k}\right)$ time such that, given an* FVS *instance (G,k) and $\epsilon \in (0,1)$, it either reports a failure or finds a feedback vertex set of G of size at most $(1+\epsilon)k$. Moreover, if the algorithm is given a* Yes*-instance, it returns a solution of size at most $(1+\epsilon)k$ with probability at least $1/e$.*

Let us call the algorithm of Theorem 6 RPAS3. Just like RPAS1 and RPAS2, it is obtained by repeating Algorithm 3 multiple times to get a constant success probability. So before we give the proof of Theorem 11, we need to prove some lemmas about Algorithm 3.

▶ **Lemma 12.** *If Algorithm 3 returns a set S then S is a feedback vertex set in G of size at most $(1+\epsilon)k$.*

Proof. The returned solution will be either of the form S_1 or of the form $S_1 \cup S_2$ for some vertex set S_1, and S_2. If it is only S_1, then by the algorithm, it is clear that $G - S_1$ is acyclic and $|S_1| \leq (1+\epsilon)k$.

Now, if the returned solution is of the form $S_1 \cup S_2$, then S_2 is a 2-approximate solution for the graph $G - S_1$. Thus, by the correctness of 2-approximation algorithm for Feedback Vertex Set, we can say S_2 is a feedback vertex set in $G - S_1$, and this implies, $S = S_1 \cup S_2$ is a feedback vertex set in G.

Now from Algorithm 3, we can see, $|S_1| \geq (1-\epsilon)k$. Again, Algorithm 3 returns a solution of the form $S_1 \cup S_2$ only when $|S_2| \leq 2(k - |S_1|)$. Thus, we get,

$$|S| = |S_1| + |S_2| \leq |S_1| + 2(k - |S_1|) = 2k - |S_1| \leq 2k - (1-\epsilon)k = (1+\epsilon)k.$$ ◀

▶ **Lemma 13.** *Given an* Yes*-instance of* FVS*, Algorithm 3 returns a solution of size at most $(1+\epsilon)k$ with probability at least $4^{-(1-\epsilon)k}$.*

Algorithm 3 Third randomized $(1+\epsilon)$-approximation algorithm for FVS.

```
Input : An instance (G, k) of FVS and ϵ ∈ (0, 1).
1  Initialize, S_1 ← ∅;
2  while |S_1| < (1 − ϵ)k & G − S_1 is not acyclic do
3  |  apply reduction rules 2-4 exhaustively to G − S_1;
4  |  if there is a self-loop then
5  |  |  S_1 = S_1 ∪ {v : there is a self-loop on v};
6  |  else
7  |  |  pick an edge e u.a.r from E(G − S_1);
8  |  |  pick a vertex v u.a.r. from the endpoints of e;
9  |  |  S_1 = S_1 ∪ {v};
10 |  end
11 |  if G − S_1 is acyclic & |S_1| ≤ (1 + ϵ)k then
12 |  |  return S_1;
13 |  end
14 end
15 apply 2-approximation for FVS on G − S_1;
16 Let S_2 be returned solution;
17 if |S_2| > 2(k − |S_1|) then
18 |  return No;
19 else
20 |  return S_1 ∪ S_2;
21 end
```

Proof. Let the given instance (G, k) be a `Yes` instance, that is, there is a feedback vertex set in G of size at most k. Let S_1^1 and S_1^2 be the set of vertices we add in S by the line number 5 and 9 of Algorithm 3, respectively. Notice that we add vertices to S_1^1 only when there are some self-loops in the graph (i.e., reduction rule 1 is applicable). By the correctness of reduction rule 1, if (G, k) is a `Yes` instance, then there exists a feedback vertex set of size at most k containing S_1^1. Now, when we add a vertex in S_1^2, from the pseudocode of Algorithm 3, we can see none of the reduction rules are applicable and thus, we can assume the minimum degree of the graph to be at least 3. Now if we pick an edge uniformly at random and then we pick one of its endpoints, v uniformly at random into S_1^2 then Lemma 2 says that, v is a part of any feedback vertex set with probability at least $\frac{1}{4}$. Thus, for some feedback vertex set F of size at most k, $pr[S_1 = S_1^1 \cup S_1^2 \subseteq F] \geq \frac{1}{4} \cdot \frac{1}{4} \cdot \frac{1}{4} \cdots \frac{1}{4}(|S_1^2| times) = \frac{1}{4^{|S_1^2|}}$. Since, Algorithm 3 adds at most one vertex to S_1^2 at every execution of the while loop and the while loop runs for at most $(1-\epsilon)k$ steps, so,$|S_1^2| \leq (1-\epsilon)k$. Therefore, $pr[S_1 = S_1^1 \cup S_1^2 \subseteq F] \geq \frac{1}{4^{(1-\epsilon)k}}$. If $S_1 = F$, then S_1 is a feedback vertex set in G and $|S_1| \leq k \leq (1+\epsilon)k$ and Algorithm 3 will return a solution. Else if $S_1 \subset F$ then $F \setminus S_1$ is a feedback vertex set in $G - S_1$ of size $k - |S_1|$. Hence, the size of the 2-approximate solution S_2 will be at most $2(k - |S_1|)$ and Algorithm 3 will return a solution $S = S_1 \cup S_2$. So, if the given instance (G, k) is a yes-instance then for some feedback vertex set F in G of size at most k, $S_1 \subseteq F$ with probability at least $\frac{1}{4^{(1-\epsilon)k}}$ and when $S_1 \subseteq F$, Algorithm 3 always returns a solution. Hence the proof. ◀

Proof of Theorem 11. There is a polynomial time 2-approximation algorithm for FVS that can be found in [1]. Also, every step under the while loop in Algorithm 3 takes only polynomial time and the while loop runs for at most $(1-\epsilon)k$ times. So, overall Algorithm 3

runs in polynomial time. RPAS3 calls Algorithm 3 for $4^{(1-\epsilon)k}$ times to achieve constant success probability, and the remainder of the proof follows from Lemma 12 and Lemma 13 along the lines of proofs of Theorem 3 and Theorem 6. ◀

6 Comparison of RPAS1, RPAS2, and RPAS3, and proof of Theorem 1

In this section we compare the running times of the algorithms in Theorem 3, Theorem 6, and Theorem 11, which we have named RPAS1, RPAS2, and RPAS3 respectively. We observe that each of RPAS1, RPAS2 and RPAS3 performs better than the other two when ϵ lies in the intervals $(0, 0.176)$, $(0.176, 0.674)$, and $(0.674, 1)$ respectively. For $\epsilon = 0.176$, the running times of RPAS1 and RPAS2 are the same, while for $\epsilon = 0.674$, the running times of RPAS2 and RPAS3 turn out to be the same. The following table gives the running time of the three algorithms for different values of ϵ for comparison. Now we are ready to give the proof of Theorem 1.

No.	ϵ	Approximation factor	RPAS1 run time	RPAS2 run time	RPAS3 run time	Better run time
1	0.05	1.05	2.66^k	3.319^k	3.732^k	2.66^k
2	0.10	1.10	2.62^k	2.945^k	3.482^k	2.62^k
3	0.15	1.15	2.581^k	2.676^k	3.482^k	2.581^k
4	0.176	1.176	2.561^k	2.561^k	3.134^k	2.561^k
5	0.20	1.20	2.543^k	2.467^k	3.031^k	2.467^k
6	0.25	1.25	2.505^k	2.3^k	2.828^k	2.3^k
7	0.30	1.30	2.468^k	2.16^k	2.639^k	2.16^k
8	0.35	1.35	2.431^k	2.043^k	2.462^k	2.043^k
9	0.40	1.40	2.395^k	1.942^k	2.297^k	1.942^k
10	0.45	1.45	2.359^k	1.855^k	2.144^k	1.855^k
11	0.50	1.50	2.324^k	1.778^k	2^k	1.778^k
12	0.55	1.55	2.289^k	1.71^k	1.866^k	1.71^k
13	0.60	1.60	2.255^k	1.649^k	1.741^k	1.649^k
14	0.65	1.65	2.222^k	1.595^k	1.624^k	1.595^k
15	0.674	1.674	2.206^k	1.571^k	1.571^k	1.571^k
16	0.70	1.70	2.188^k	1.546^k	1.516^k	1.516^k
17	0.75	1.75	2.156^k	1.502^k	1.414^k	1.414^k
18	0.80	1.80	2.124^k	1.462^k	1.32^k	1.32^k
19	0.85	1.85	2.092^k	1.426^k	1.231^k	1.231^k
20	0.9	1.90	2.061^k	1.392^k	1.149^k	1.149^k
21	0.95	1.95	2.03^k	1.362^k	1.072^k	1.072^k

Proof of Theorem 1. Given an instance (G, k) and an $\epsilon \in (0, 1)$, the algorithm runs one of RPAS1, RPAS2, and RPAS3 depending upon which of them performs the best for that value of ϵ (for $\epsilon = 0.176$, we can choose either of RPAS1 or RPAS2 and for $\epsilon = 0.674$, we can choose either of RPAS2 or RPAS3). The running time and the correctness follows from Theorem 3, Theorem 6, and Theorem 11. ◀

References

1 Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM J. Discret. Math.*, 12(3):289–297, 1999. doi:10.1137/S0895480196305124.

2 Ann Becker, Reuven Bar-Yehuda, and Dan Geiger. Randomized algorithms for the loop cutset problem. *J. Artif. Intell. Res.*, 12:219–234, 2000. doi:10.1613/jair.638.

3 Arnab Bhattacharyya, Suprovat Ghoshal, Karthik C. S., and Pasin Manurangsi. Parameterized intractability of even set and shortest vector problem from gap-eth. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 17:1–17:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.17.

4 Ljiljana Brankovic and Henning Fernau. A novel parameterised approximation algorithm for minimum vertex cover. *Theor. Comput. Sci.*, 511:85–108, 2013. doi:10.1016/j.tcs.2012.12.003.

5 Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From gap-eth to fpt-inapproximability: Clique, dominating set, and more. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 743–754. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.74.

6 Yijia Chen and Bingkai Lin. The constant inapproximability of the parameterized dominating set problem. *SIAM J. Comput.*, 48(2):513–533, 2019. doi:10.1137/17M1127211.

7 Thomas M. Cover and Joy A. Thomas. *Elements of information theory (2. ed.)*. Wiley, 2006. URL: http://www.elementsofinformationtheory.com/.

8 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

9 Andreas Emil Feldmann, Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6):146, 2020. doi:10.3390/a13060146.

10 Michael R. Fellows, Ariel Kulik, Frances A. Rosamond, and Hadas Shachnai. Parameterized approximation via fidelity preserving transformations. *J. Comput. Syst. Sci.*, 93:30–40, 2018. doi:10.1016/j.jcss.2017.11.001.

11 Anupam Gupta, Euiwoong Lee, and Jason Li. An FPT algorithm beating 2-approximation for k-cut. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2821–2837. SIAM, 2018. doi:10.1137/1.9781611975031.179.

12 Yoichi Iwata and Yusuke Kobayashi. Improved analysis of highest-degree branching for feedback vertex set. *Algorithmica*, 83(8):2503–2520, 2021. doi:10.1007/s00453-021-00815-w.

13 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008. doi:10.1016/j.jcss.2007.06.019.

14 Guy Kortsarz. Fixed-parameter approximability and hardness. In *Encyclopedia of Algorithms*, pages 756–761. Springer, 2016. doi:10.1007/978-1-4939-2864-4_763.

15 Ariel Kulik and Hadas Shachnai. Analysis of two-variable recurrence relations with application to parameterized approximations. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 762–773. IEEE, 2020. doi:10.1109/FOCS46700.2020.00076.

16 Jason Li and Jesper Nederlof. Detecting feedback vertex sets of size k in $O^\star$ $(2.7k)$ time. *ACM Trans. Algorithms*, 18(4):34:1–34:26, 2022. doi:10.1145/3504027.

17 Bingkai Lin. The parameterized complexity of the k-biclique problem. *J. ACM*, 65(5):34:1–34:23, 2018. doi:10.1145/3212622.

18 Daniel Lokshtanov, Pranabendu Misra, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Fpt-approximation for FPT problems. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 199–218. SIAM, 2021. doi:10.1137/1.9781611976465.14.

19 Daniel Lokshtanov, Saket Saurabh, and Vaishali Surianarayanan. A parameterized approximation scheme for min k-cut. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 798–809. IEEE, 2020. doi:10.1109/FOCS46700.2020.00079.

20 Pasin Manurangsi. Inapproximability of maximum biclique problems, minimum k-cut and densest at-least-k-subgraph from the small set expansion hypothesis. *Algorithms*, 11(1):10, 2018. doi:10.3390/a11010010.

21 Dániel Marx. Parameterized complexity and approximation algorithms. *Comput. J.*, 51(1):60–78, 2008. doi:10.1093/comjnl/bxm048.