2-Approximating Feedback Vertex Set in Tournaments

DANIEL LOKSHTANOV, University of California, USA
PRANABENDU MISRA, Max-Planck Institute for Informatics, SIC, Germany
JOYDEEP MUKHERJEE, Ramakrishna Mission Vivekananda Educational and Research Institute, India and Indian Statistical Institute, India
FAHAD PANOLAN, Department of Computer Science and Engineering, IIT Hyderabad, India
GEEVARGHESE PHILIP, Chennai Mathematical Institute, India and UMI ReLaX
SAKET SAURABH, Institute of Mathematical Sciences, HBNI, India and University of Bergen, Norway and UMI ReLaX

A *tournament* is a directed graph T such that every pair of vertices is connected by an arc. A *feedback vertex set* is a set S of vertices in T such that $T - S$ is acyclic. We consider the Feedback Vertex Set problem in tournaments. Here, the input is a tournament T and a weight function $w : V(T) \to \mathbb{N}$, and the task is to find a feedback vertex set S in T minimizing $w(S) = \sum_{v \in S} w(v)$. Rounding optimal solutions to the natural LP-relaxation of this problem yields a simple 3-approximation algorithm. This has been improved to 2.5 by Cai et al. [SICOMP 2000], and subsequently to 7/3 by Mnich et al. [ESA 2016]. In this article, we give the first polynomial time factor 2-approximation algorithm for this problem. Assuming the Unique Games Conjecture, this is the best possible approximation ratio achievable in polynomial time.

A preliminary version of this article appeared in the Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA 2020).
This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant Agreements No. 819416 and No. 715744), the IFCAM project "Applications of graph homomorphisms" (MA/IFCAM/18/39), the Bergens Forsknings Stiftelse (BFS) under the grant "Putting Algorithms Into Practice" (No. 810564) and the Norwegian Research Foundation (NRF) under the grant "Parameterized Complexity for Practical Computing" (No. 274526d). Saket Saurabh also acknowledges the support of the Swarnajayanti Fellowship Grant No. DST/SJF/MSA-01/2017-18.

Authors' addresses: D. Lokshtanov, Department of Computer Science, University of California, 2104 Harold Frank Hall, Santa Barbara, California, 93106-5110, USA; email: daniello@ucsb.edu; P. Misra, Max Planck Institute for Informatics, Campus E1-4, Saarland University, 66123 Saarbrucken, Germany; email: pmisra@mpi-inf.mpg.de; J. Mukherjee, Ramakrishna Mission Vivekananda Educational and Research Institute, PO Belur Math, Dist. Howrah 711202 West Bengal, India; email: joydeep.m1981@gmail.com; F. Panolan, Department of Computer Science and Engineering, Indian Institute of Technology Hyderabad, Kandi-502285, Sangareddy, Telangana, India; email: fahad@cse.iith.ac.in; G. Philip, Chennai Mathematical Institute, H1, SIPCOT IT Park, Siruseri, Kelambakkam 603103, India; email: gphilip@cmi.ac.in; S. Saurabh, The Institute of Mathematical Sciences, IV Cross Road, CIT Campus, Taramani, Chennai 600113, Tamil Nadu, India; email: saket@imsc.res.in.

1549-6325/2021/04-ART11 $15.00
https://doi.org/10.1145/3446969

CCS Concepts: • **Theory of computation → Graph algorithms analysis**; **Approximation algorithms analysis**; **Algorithm design techniques**;

Additional Key Words and Phrases: Approximation algorithm, local ratio, feedback vertex set, tournament, branching

ACM Reference format:
Daniel Lokshtanov, Pranabendu Misra, Joydeep Mukherjee, Fahad Panolan, Geevarghese Philip, and Saket Saurabh. 2021. 2-Approximating Feedback Vertex Set in Tournaments. *ACM Trans. Algorithms* 17, 2, Article 11 (April 2021), 14 pages.
https://doi.org/10.1145/3446969

1 INTRODUCTION

A *feedback vertex set* (FVS) in a graph G is a vertex subset S such that $G - S$ is acyclic. In the case of directed graphs, it means $G - S$ is a directed acyclic graph (DAG). In the (Directed) Feedback Vertex Set ((D)FVS) problem, we are given as input a (directed) graph G and a weight function $w : V(G) \to \mathbb{N}$. The objective is to find a minimum weight feedback vertex set S. Both the directed and undirected versions of the problem are NP-complete [14] and have been extensively studied from the perspective of approximation algorithms [3, 13], parameterized algorithms [7, 9, 18], exact exponential time algorithms [22, 26] as well as graph theory [12, 23].

In this article, we consider a restriction of DFVS, namely, the Feedback Vertex Set in Tournaments (TFVS) problem, from the perspective of approximation algorithms (we refer to the textbook of Williamson and Shmoys [25] for an introduction to approximation algorithms). A *tournament* is a directed graph G such that every pair of vertices is connected by an arc, and TFVS is simply DFVS when the input graph is required to be a tournament. Even this restricted variant of DFVS has applications in voting systems and rank aggregation and is quite well-studied [6, 11, 15, 19, 20, 21]. It is formally defined as follows.

Feedback Vertex Set in Tournaments (TFVS)
Input: A tournament G and a weight function $w : V(G) \to \mathbb{N}$.
Output: A minimum weight FVS of G.

It is well known that a tournament has a directed cycle if and only if there is a directed triangle [11]. Thus, the TFVS problem can be re-cast as a special case of the well-studied 3-Hitting Set problem (also known as Vertex Cover in 3-uniform hypergraphs). Here the input is a universe U, a weight function $w : U \to \mathbb{N}$ and a family $\mathbb{F}$ of subsets of U, each of size at most 3. The goal is to find a minimum weight subset S of the universe that intersects every set in $\mathbb{F}$. 3-Hitting Set (and therefore also TFVS) admits a simple 3-approximation algorithm: Taking the natural LP relaxation[1] and selecting all elements whose variable is set to at least 1/3 leads to a 3-approximate solution. For 3-Hitting Set this simple approximation algorithm is likely the best possible: assuming the Unique Games Conjecture (UGC) there is no c-approximation algorithm for $c < 3$ [17]. A c-approximation algorithm with $c < 2$ would imply P = NP [10].

Since TFVS is a special case of 3-Hitting Set, *algorithms* for 3-Hitting Set translate to algorithms for TFVS, but *lower bounds* for 3-Hitting Set do not translate to lower bounds for TFVS. Indeed, TFVS does admit c-approximation algorithms with $c < 3$. The first such algorithm was given by Cai et al. [6], who gave a 5/2-approximation algorithm using the local ratio technique of Bar-Yehuda and Even [5]. Recently, Mnich et al. [20] gave a 7/3-approximation algorithm using the

[1]There is a variable $0 \leq x_v \leq 1$ for every element v and a constraint $x_u + x_v + x_w \geq 1$ for every triple $\{u, v, w\} \in \mathbb{F}$. The objective is to minimize the sum of the variables.

iterative rounding technique. They also observe that the approximation-preserving reduction from Vertex Cover to TFVS of Speckenmeyer [24] implies that, assuming the Unique Games Conjecture (UGC) [17], TFVS cannot have an approximation algorithm with factor smaller than 2. Mnich et al. [20] state that their algorithm "*gives hope that a 2-approximation algorithm, that would be optimal under the UGC, might be achievable* (for TFVS)." In this article, we show that this is indeed the case, by giving a (randomized) 2-approximation algorithm for TFVS. More formally, we prove the following theorem.

Theorem 1. *There exists a randomized algorithm that, given a tournament G on n vertices and a weight function w on G, runs in time $O(n^{17})$ and outputs a feedback vertex set S of G. With probability at least $7/10$, S is a 2-approximate solution of (G, w). When the instance is unweighted, the running time can be improved to $O(n^{12})$.*

This algorithm can be easily de-randomized in quasi-polynomial time.

Overview of algorithm. We first give a high level overview of a 2-approximation algorithm for the unweighted case (when every vertex has weight 1). Let OPT be an optimal solution, if $|\text{OPT}| \geq n/2$ then every feasible solution (such as the entire vertex set!) is a 2-approximate solution. Assuming that $|\text{OPT}| < n/2$, a randomly chosen vertex p will be not in OPT with constant probability. Further, with constant probability such a vertex p will be "in the middle one-third" of the unique topological ordering of $G - \text{OPT}$. In other words, with constant probability p will have at least $(G - |\text{OPT}|)/3 \geq n/6$ in-neighbors and at least $(G - |\text{OPT}|)/3 \geq n/6$ out-neighbors. Crucially, both the number of in-neighbors and the number of out-neighbors will be *at most* $5n/6$. The idea is now to use p as a pivot in a "*quicksort-like*" procedure. This idea has been previously used in approximating Feedback Arc Set in Tournaments (FAST), and other related problems [1]. It is also a key component of the PTAS for FAST [16].

If there exists an arc uv from $N^+(p)$ to $N^-(p)$, then puv forms a directed triangle and hence, since $p \notin \text{OPT}$, OPT contains either u or v. We put *both* u and v into the solution, delete them from G (and OPT), and repeat as long as there are arcs from $N^+(p)$ to $N^-(p)$. Each iteration adds two vertices to the solution while decreasing $|\text{OPT}|$ by at least one. When the procedure terminates there are no arcs from $N^+(p)$ to $N^-(p)$. Hence, for the purposes of 2-approximation, we can assume without loss of generality that there are no arcs from $N^+(p)$ to $N^-(p)$.

When there are no arcs from $N^+(p)$ to $N^-(p)$ the problem breaks into two independent subinstances. Indeed, for every solution S^- to $G[N^-(p)]$ and solution S^+ to $G[N^+(p)]$, we have that $S^- \cup S^+$ is a solution to G. To see this, take the topological order of $G[N^-(p)] - S^-$, append p, then append the topological order of $G[N^+(p)] - S^+$ and observe that this is a topological order of $G - (S^- \cup S^+)$. The algorithm calls itself recursively on $G[N^-(p)]$ and $G[N^+(p)]$, obtains 2-approximate solutions S^- and S^+ and returns $S^- \cup S^+$ as its 2-approximate solution.

The algorithm thus makes two recursive calls to instances of size at most $5n/6$, leading to the recurrence $T(n) \leq 2T(5n/6)$, which solves to $T(n) = n^{O(1)}$ by the Master Theorem. This is the entire algorithm! Of course, when formulating the recurrence above, we silently assumed that the choice of p *always* succeeds, instead of succeeding with constant probability. To correct for this it is sufficient to repeat the experiment (pick a random p and run the algorithm recursively on $G[N^-(p)]$ and $G[N^+(p)]$) a constant number of times in each recursive call, leading to the recurrence $T(n) \leq O(1) \cdot T(5n/6)$, which still solves to $T(n) = n^{O(1)}$.

Derandomization. The only place where the algorithm uses randomness is the choice of the pivot p. The only properties we need from p is that it is not in OPT, and that its indegree and outdegree is at least $n/6$. We know that at least $n/6$ vertices of G have these properties. The deterministic algorithm replaces the step when p is selected at random with a loop that tries all

the n possible choices for p. This leads to the recurrence $T(n) \leq n \cdot 2T(5n/6)$, which solves to $T(n) \leq n^{O(\log n)}$.

Dealing with weights. There are two steps of the algorithm for unweighted graphs that do not work directly also for weighted graphs. The first problem is that we can no longer deal with the $|\text{OPT}| > n/2$ case by picking all the vertices into the solution (since their total weight can be more than twice the weight of OPT). The second problem is that when we pick a pivot vertex p and find an arc uv from $N^+(p)$ to $N^-(p)$, we can no longer pick both u and v into the approximate solution. Both problems are quite easily handled by "local ratio" arguments (Lemma 3 handles the first problem, while Lemma 4 handles the second).

2 PRELIMINARIES

In this article, we work with directed graphs (or *digraphs*) that do not contain any self loops or parallel arcs. We use $V(G)$ to denote the vertex set of a digraph G and $E(G)$ to denote the set of arcs of G. We use the notation uv to denote an arc from vertex u to vertex v in a digraph. Vertices u, v are *incident with* arc uv. A *tournament* is a digraph in which there is exactly one arc between any two vertices. The set of *out-neighbors* of a vertex v in a digraph G is defined to be $N^+(v) := \{u \mid vu \in E(G)\}$, and the set of *in-neighbors* of v in G is defined to be $N^-(v) := \{u \mid uv \in E(G)\}$. For an integer $\ell \geq 3$ a *directed cycle of length* ℓ in a digraph G is an alternating sequence $C = v_1a_1v_2a_2 \ldots v_\ell a_\ell$ where $\{v_1 \ldots, v_\ell\} \subseteq V(G)$ is a set of ℓ distinct vertices of G and $\{a_1 \ldots, a_\ell\} \subseteq E(G)$ is a subset of arcs of G where $a_i = v_iv_{i+1}; 1 \leq i < \ell$ and $a_\ell = v_\ell v_1$. A digraph is *acyclic* if it does not contain a directed cycle. A *triangle* in a digraph is a directed cycle of length three. In this article, we use the term "triangle" exclusively to denote directed triangles. A *topological sort* of a digraph G with n vertices is a permutation $\pi : V(G) \mapsto [n]$ of the vertices of the digraph such that for all arcs $uv \in E(G)$, it is the case that $\pi(u) < \pi(v)$. Such a permutation exists for a digraph G if and only if G is acyclic [4]. For an acyclic tournament, the topological sort is unique [4]. *Deleting* a vertex v from digraph G involves removing, from G, the vertex v and all those arcs in G with which v is incident in G. We use $G - v$ to denote the digraph obtained by deleting a vertex $v \in V(G)$ from digraph G. For a vertex set $S \subseteq V(G)$, we use $G - S$ to denote the digraph obtained from digraph G by deleting all the vertices of S.

A *feedback vertex set* (FVS) of a digraph G is a vertex set S such that $G - S$ is acyclic. A vertex set is a *feasible solution* if and only if it is an FVS. Given a weight function $w : V(G) \to \mathbb{N}$ the *weight* of a vertex set S is $w(S) = \sum_{v \in S} w(v)$. An FVS S_{OPT} of G is an *optimal* solution of the instance (G, w) if every other FVS S of G satisfies $w(S) \geq w(S_{OPT})$. An FVS S of G is called 2-*approximate solution* of the instance (G, w) if $w(S) \leq 2w(S_{OPT})$ for an optimal solution S_{OPT} of (G, w). An FVS S is called p-disjoint for a vertex p if $p \notin S$, and further, S is said to be an *optimal p-disjoint FVS of* (G, w) if, for every p-disjoint solution S' we have $w(S') \geq w(S)$. Note that an optimal p-disjoint solution of (G, w) is not necessarily an optimal solution of (G, w). However, if an optimal solution S_{OPT} of (G, w) happens to be p-disjoint, then S_{OPT} is also an optimal p-disjoint solution of G. A p-disjoint FVS S of G is called 2-*approximate p-disjoint solution* of the instance (G, w) if $w(S) \leq 2w(S')$ for an optimal p-disjoint solution S' of (G, w).

In the following, we will assume that G is a tournament on n vertices, and $w : V(G) \to \mathbb{N}$ is a weight function. Furthermore, for any induced subgraph H of G, we assume that w defines a weight function, when restricted to $V(H)$. We will frequently make use of the following lemma, which directly follows from the fact that acyclic digraphs are closed under vertex deletions.

Lemma 1. *Let S be an FVS of a digraph G and let X be a subset of the vertex set of G. Then $S \setminus X$ is an FVS of the digraph $G - X$. If $S^\star$ is an* optimal *solution of an instance (G, w) of TFVS and X is a subset of $S^\star$, then $S^\star \setminus X$ is an* optimal *solution of the instance $((G - X), w)$, of weight $w(S^\star) - w(X)$.*

We use the following lemma to prove the correctness our algorithm in the later section.

Lemma 2. *Let (G, w) be an instance of TFVS.*

(*i*) *A vertex $v \in G$ is not part of any triangle in G if and only if every arc between a vertex in $N^-(v)$ and a vertex in $N^+(v)$ is of the form xy ; $x \in N^-(v), y \in N^+(v)$.*
(*ii*) *Let $x \in V(G)$ be a vertex that is not part of any triangle in G. Let $H_{in} = G[N^-(x)]$ and $H_{out} = G[N^+(x)]$ be the subgraphs induced in G by the in- and out-neighborhoods of vertex x, respectively. A set S is an FVS of digraph G if and only if $S \cap V(H_{in})$ is an FVS of the subgraph H_{in} and $S \cap V(H_{out})$ is an FVS of the subgraph H_{out}.*

Proof. Suppose vertex v is not part of any triangle in G. If there is an arc st in G where vertex s is in the out-neighborhood $N^+(v)$ of vertex v and vertex t is in its in-neighborhood $N^-(v)$, then the vertices $\{s, v, t\}$ form a triangle containing vertex v, a contradiction. So every arc between vertices $x \in N^-(v)$ and $y \in N^+(v)$ is directed from x to y. Conversely, if vertices $\{v, s, t\}$ form a triangle and—without loss of generality—vs is an arc in G, then we have that both st and tv are arcs in G. Thus, $s \in N^+(v), t \in N^-(v)$, and arc st is not of the form xy ; $x \in N^-(v), y \in N^+(v)$.

Now prove statement (ii) of the lemma. Let S be an FVS of G. As $H_{in} - (S \cap V(H_{in}))$ and $H_{out} - (S \cap V(H_{out}))$ are subgraphs of $G - S$ (which is a DAG), we have that $S \cap V(H_{in})$ is an FVS of H_{in} and $S \cap V(H_{out})$ is an FVS of H_{out}. Now, we prove the other direction. Let $S \subseteq V(G)$ be such that $S \cap V(H_{in})$ is an FVS of H_{in} and $S \cap V(H_{out})$ is an FVS of H_{out}. Since $H_{in} - S$ is an acyclic tournament, there is a unique topological sort $u_1, \ldots, u_\ell$ of $H_{in} - S$, where $\{u_1, \ldots, u_\ell\} = V(H_{in}) \setminus S$. Also, since $H_{out} - S$ is an acyclic tournament, there is a unique topological sort $v_1, \ldots, v_{\ell'}$ of $H_{out} - S$, where $\{v_1, \ldots, v_{\ell'}\} = V(H_{out}) \setminus S$. Since x is not part of a triangle in G, by statement (i) of the lemma, there is no arc from a vertex in $\{v_1, \ldots, v_{\ell'}\}$ to a vertex in $\{u_1, \ldots, u_\ell\}$. This implies that $u_1, \ldots, u_\ell, x, v_1, \ldots, v_{\ell'}$ is a topological sort of $G - S$. Therefore, S is an FVS of G. □

3 THE ALGORITHM

In this section, we develop the required tools, and present an algorithm that illustrates our approach. In the following section, we present an algorithm with an improved running time, by fine-tuning our approach. The algorithm will distinguish between two cases: either the optimal solution contains many (more than some constant fraction of the) vertices, or it does not. The following lemma handles the case when the optimal solution contains many vertices.

Lemma 3. *Let (G, w) be an instance of TFVS where G has n vertices and that has an optimal solution $S^\star$ that contains at least αn vertices of G, where $\alpha > 1/2$ is a constant. Let $D \subseteq V(G)$ be a set of $n(\alpha - \frac{1}{2})$ vertices of the smallest weight in $V(G)$, ties broken arbitrarily, and let $\Delta = \max_{v \in D} w(v)$ be the weight of the heaviest vertex in D. Let $w' : V(G) \setminus D \to \mathbb{N}$ be the weight function that assigns the weight $w(v) - \Delta$ to each vertex v of $G - D$. If R_{apx} is a 2-approximate solution of the reduced instance $(G - D, w')$, then $R_{apx} \cup D$ is a 2-approximate solution of the instance (G, w).*

Proof. Let $R^\star$ be an optimum solution of the reduced instance $(G - D, w')$. Then $w'(R_{apx}) \leq 2w'(R^\star)$. From Lemma 1, we get that $S^\star \setminus D$ is a—not necessarily optimal—solution of the reduced instance $(G - D, w')$. Since $R^\star$ is an optimum solution of this instance, we have that $w'(S^\star \setminus D) \geq w'(R^\star)$. Since $w'(v) = (w(v) - \Delta)$ holds for each vertex $v \in (S^\star \setminus D)$, we get that $w'(S^\star \setminus D) = w(S^\star \setminus D) - |S^\star \setminus D| \cdot \Delta \leq w(S^\star) - |S^\star \setminus D| \cdot \Delta$. Since $|S^\star \setminus D| \geq \alpha n - (\alpha - \frac{1}{2})n = \frac{n}{2}$, we get that $w'(S^\star \setminus D) \leq w(S^\star) - \frac{\Delta}{2}n$. Hence, $w'(R^\star) \leq w'(S^\star \setminus D) \leq w(S^\star) - \frac{\Delta}{2}n$.

Thus, $w'(R_{apx}) \leq 2w'(R^\star) \leq 2w(S^\star) - \Delta n$. Since the set R_{apx} is disjoint from the deleted set D we have that $w'(v) = w(v) - \Delta$ holds for each vertex $v \in R_{apx}$. Hence, $w(R_{apx}) = w'(R_{apx}) + |R_{apx}| \cdot \Delta \leq (2w(S^\star) - \Delta n) + |R_{apx}| \cdot \Delta = 2w(S^\star) - \Delta(n - |R_{apx}|)$. Since $w(v) \leq \Delta$ holds for each

vertex $v \in D$ we have that $w(D) \leq |D| \cdot \Delta$. Hence,

$$\begin{aligned} w(R_{apx} \cup D) &= w(R_{apx}) + w(D) \\ &\leq 2w(S^\star) - \Delta(n - |R_{apx}|) + |D| \cdot \Delta \\ &= 2w(S^\star) - \Delta(n - |R_{apx}| - |D|) \\ &= 2w(S^\star) - \Delta(n - |R_{apx} \cup D|) \\ &\leq 2w(S^\star). \end{aligned}$$

Here the last inequality follows from the fact that $|R_{apx} \cup D| \leq n = |V(G)|$. □

We remark that, in Lemma 3, the instance $(G - D, w')$ contains at most $(\frac{3}{2} - \alpha)n$ vertices, where $n = |V(G)|$. This fact will be helpful in the analysis of our algorithm. The next lemma shows that given $\{p, u, v\}$, we can safely pick a lighter weight vertex of the two vertices u and v into a 2-approximate p-disjoint solution.

Lemma 4. *Let (G, w) be an instance of TFVS and $p \in V(G)$. Let $\{u, v\}$ be two vertices such that (i) $\{p, u, v\}$ form a triangle in G, and (ii) $w(v) \leq w(u)$. Let w' be the weight function defined by: (a) $w'(v) = 0$,(b) $w'(u) = w(u) - w(v)$, and $(c)w'(x) = w(x)$ for all vertices $x \notin \{u, v\}$. Then for every 2-approximate p-disjoint solution R_{apx} of the reduced instance $(G - v, w')$, we have $R_{apx} \cup \{v\}$ is a 2-approximate p-disjoint solution of the original instance (G, w).*

Proof. Since $(G - v) - R_{apx} = G - (R_{apx} \cup \{v\})$ and the former digraph is acyclic by assumption, we get that $R_{apx} \cup \{v\}$ is a FVS in the digraph G. We will show that $R_{apx} \cup \{v\}$ is a 2-approximate p-disjoint solution of (G, w). Since $p \notin R_{apx}$, $R_{apx} \cup \{v\}$ is a p-disjoint FVS of G. Let $S^\star$ be an optimal p-disjoint solution of (G, w). Notice that $S^\star \cap \{u, v\} \neq \emptyset$. Now to complete the proof, it remains to show that $w(R_{apx} \cup \{v\}) \leq 2w(S^\star)$. Let $\Delta = \min\{w(u), w(v)\}$, that is $w(v) = \Delta$. Now, we have the following:

$$\begin{aligned} w(R_{apx} \cup \{v\}) &\leq w'(R_{apx} \cup \{v\}) + 2\Delta && \text{(since } w(v) = \Delta \text{ and } w(u) = \Delta + w'(u)\text{)} \\ &= w'(R_{apx}) + 2\Delta && \text{(since } w'(v) = 0\text{)} \\ &\leq 2w'(S^\star \setminus \{v\}) + 2\Delta && \text{(since } S^\star \setminus \{v\} \text{ is an FVS of } G - v\text{)} \\ &= 2w'(S^\star) + 2\Delta && \text{(since } w'(v) = 0\text{)} \\ &= 2\Big(w(S^\star) - \Delta \cdot |S^\star \cap \{u, v\}|\Big) + 2\Delta \\ &\leq 2w(S^\star) && \text{(since } S^\star \cap \{u, v\} \neq \emptyset\text{)}. \end{aligned}$$

This completes the proof. □

Suppose that we have picked a pivot vertex p that is disjoint from an optimal solution. If there is an arc $xy \in E(G)$ such that $x \in N^+(p) \setminus D_i$ and $y \in N^-(p) \setminus D_i$, then the vertices $\{x, p, y\}$ form a triangle in G, and so at least one of the two vertices $\{x, y\}$ must be present in the solution $S^\star$. Let v be a vertex of the least weight among $\{x, y\}$, ties broken arbitrarily, and let u be the other vertex. Then Lemma 4 applies to the tuple $\{(G, w), p, \{u, v\}\}$.

Procedure Reduce(G, w, p) of Algorithm 1 applies Lemma 4 exhaustively until there are no arcs from $N^+(p)$ to $N^-(p)$: It starts by setting $D_0 = \emptyset$, $w_0 = w$, and $i = 0$. As long as there is an arc $xy \in E(G)$ such that $x \in N^+(p) \setminus D_i$ and $y \in N^-(p) \setminus D_i$ it finds vertices $\{u, v\}$ as described in the previous paragraph and computes a weight function w' as specified in Lemma 4 as applied to the collection $\{(G, w), p, \{u, v\}\}$. It sets $w_{i+1} = w'$, $D_{i+1} = D_i \cup \{v\}$, increments i by one, and repeats. When no such arc xy exists the procedure outputs the set $D = D_i$ and the weight function $\tilde{w} = w_i$.

Our next lemma states that procedure Reduce runs in polynomial time and correctly outputs a reduced instance. Recall that for an instance (G, w) of TFVS and a vertex $p \in V(G)$, a *p-disjoint solution* of (G, w) is an FVS of G, which does not contain vertex p.

Lemma 5. *Let (G, w) be an instance of TFVS and $p \in V(G)$. When given (G, w, p) as input, the procedure Reduce runs in $O(|V(G)|^2)$ time and outputs a vertex set $D \subseteq (V(G) \setminus \{p\})$ and a weight function $\tilde{w}$ with the following properties:*

(*i*) *there are no arcs from $N^+(p)$ to $N^-(p)$ in digraph $G - D$, and*
(*ii*) *for every 2-approximate p-disjoint solution S of $(G - D, \tilde{w})$, the set $S \cup D$ is a 2-approximate p-disjoint solution of (G, w).*

Proof. The check on line 3 of Algorithm 1 fails if and only if there are no arcs from $N^+(p)$ to $N^-(p)$ in the digraph $G - D_i$ for the value of i at that point. Since the assignment of D_i to D on line 13 happens only if this check fails, we get that there are no arcs from $N^+(p)$ to $N^-(p)$ in the digraph $G - D$. Let S be a 2-approximate p-disjoint solution of $(G - D, \tilde{w})$. Then by a simple induction on the number of iterations and Lemma 4, we obtain that $S \cup D$ is a 2-approximate p-disjoint solution of (G, w).

To complete the proof, we show that procedure Reduce runs in $O(n^2)$ time where $n = |V(G)|$. Let $V(G) = \{v_1, \ldots, v_n\}$. We assume that graph G is given as its $n \times n$ adjacency matrix M_G where $M_G[i, j] = 1$ if $v_i v_j$ is an arc in G and $M_G[i, j] = 0$ otherwise. We assume also that the weight function w is given as a $1 \times n$ array where $w[i]$ stores the weight of vertex v_i.

We compute the two neighborhoods $N^-(p)$ and $N^+(p)$ of the pivot vertex p by scanning the entries of the row $M_G[p]$; vertex $v_i \in N^+(p)$ if $M_G[p, i] = 1$, and $v_i \in N^-(p)$ if $v_i \neq p$ and $M_G[p, i] = 0$. This takes $O(n)$ time. Let $d_{in} = |N^-(p)|$, $d_{out} = |N^+(p)|$ be the in- and out-degrees of vertex p. We construct a $d_{out} \times d_{in}$ array $\mathcal{A}$ to store the neighborhood relation between the sets $N^+(p)$ and $N^-(p)$, and a $1 \times d_{out}$ array OD to store the out-degrees of vertices in $N^+(p)$ *into the set* $N^-(p)$. We initialize all entries of $\mathcal{A}$ and OD to zeroes. Now for each pair of vertices $v_i \in N^+(p), v_j \in N^-(p)$ we increment the entries $\mathcal{A}[i, j]$ and $OD[i]$ by 1 each if and only if $M_G[i, j] = 1$. Once this is done the cell $OD[i]$ holds the number of out-neighbors of vertex $v_i \in N^+(p)$ in the set $N^-(p)$, and $\mathcal{A}[i, j] = 1$ if and only if $v_i v_j$ is an arc in G for vertices $v_i \in N^+(p), v_j \in N^-(p)$. Since $|N^+(p)| + |N^-(p)| = (n - 1)$ all this can be done in $O(n^2)$ time.

To execute the test on line 3 of Algorithm 1, we scan the list OD for a non-zero entry. If all entries of OD are zeros, then there is no arc xy of the specified form and the test returns **False**. If $OD[i] > 0$ for some i, then we scan the row $\mathcal{A}[i]$ to find an index j such that $\mathcal{A}[i, j] = 1$. Then $x = v_i, y = v_j$ is a pair of vertices that satisfy the test. We use these vertices to execute lines 4 to 10 of the procedure. We effect the addition of vertex v to the set D_{i+1} on line 11 as follows: If $v = x = v_i \in N^+(p)$, then we set $OD[i] = 0$ and $\mathcal{A}[i, j] = 0$; $1 \leq j \leq d_{in}$. If $v = y = v_j \in N^-(p)$, then for each $1 \leq i \leq d_{out}$ such that $\mathcal{A}[i, j] = 1$, we decrement the cells $OD[i]$ and $\mathcal{A}[i, j]$ by 1.

Each line of Algorithm 1, except for line 11, takes constant time. Line 11—as described above—takes $O(n)$ time. Each execution of line 11 takes either a row or a column of $\mathcal{A}$, which has non-zero entries and sets all these entries to zero. Since the algorithm does not increment these entries in the loop, we get that the **while** loop of lines 3 to 12 is executed at most $|N^+(p)| + |N^-(p)| = (n - 1)$ times. Thus, the entire procedure runs in $O(n^2)$ time. □

Combining Lemmas 1, 2, and 5, we get

Corollary 1. *On input (G, w, p) the procedure Reduce runs in $O(n^2)$ time and outputs a vertex set $D \subseteq V(G) \setminus \{p\}$ and a weight function $\tilde{w}$ such that for every FVS S^- of $G[N^-(p) \setminus D]$ and every FVS S^+ of $G[N^+(p) \setminus D]$, we have that $S^- \cup S^+ \cup D$ is a p-disjoint FVS of G.*

Further, if S^- is a 2-approximate solution of $(G[N^-(p) \setminus D], \tilde{w})$ and S^+ is 2-approximate solution of $(G[N^+(p) \setminus D], \tilde{w})$, then $S^- \cup S^+ \cup D$ is a 2-approximate p-disjoint solution of (G, w).

Proof. The running time of procedure Reduce follows from Lemma 5. Let S^- be an FVS of $G[N^-(p) \setminus D]$ and S^+ be an FVS of $G[N^+(p) \setminus D]$. By Lemma 5, there are no arcs from $N^+(p)$ to $N^-(p)$ in digraph $G - D$. Then by statement (i) of Lemma 2, p is not part of any triangle in $G - D$. Thus, by statement (ii) of Lemma 2, $S^- \cup S^+$ is an FVS of $G - D$. Therefore, by Lemma 1, $S^- \cup S^+ \cup D$ is an FVS of G. Moreover, since $p \notin S^- \cup S^+ \cup D$, it is a p-disjoint FVS of G.

Suppose S^- is a 2-approximate solution of $(G[N^-(p) \setminus D], \tilde{w})$ and S^+ is a 2-approximate solution of $(G[N^+(p) \setminus D], \tilde{w})$. Now, we claim that $S^- \cup S^+$ is a 2-approximate p-disjoint solution of $(G - D, \tilde{w})$. Let R^- and R^+ be optimal solutions of $(G[N^-(p) \setminus D], \tilde{w})$ and $(G[N^+(p) \setminus D], \tilde{w})$, respectively. Then, we claim that $R^- \cup R^+$ is an optimal p-disjoint solution of $(G - D, \tilde{w})$. By statement (ii) of Lemma 2, $R^- \cup R^+$ is an FVS of $G - D$ and clearly it does not contain p. Suppose $R^- \cup R^+$ is not an optimal p-disjoint solution of $(G - D, \tilde{w})$. Let $R^\star$ be an optimal p-disjoint solution of $(G - D, \tilde{w})$ and $\tilde{w}(R^\star) < \tilde{w}(R^- \cup R^+)$. Then, either $\tilde{w}(R^\star \cap (N^-(p) \setminus D)) < \tilde{w}(R^-)$ or $\tilde{w}(R^\star \cap (N^+(p) \setminus D)) < \tilde{w}(R^+)$. Consider the case when $\tilde{w}(R^\star \cap (N^-(p) \setminus D)) < \tilde{w}(R^-)$. By Lemma 2, $R^\star \cap (N^-(p) \setminus D)$ is an FVS of $G[N^+(p) \setminus D]$. But this contradicts the assumption that R^- is an optimal solution of $(G[N^-(p) \setminus D], \tilde{w})$. The same arguments apply to the case when $\tilde{w}(R^\star \cap (N^+(p) \setminus D)) < \tilde{w}(R^+)$. Therefore, $R^- \cup R^+$ is an optimal p-disjoint solution of $(G - D, \tilde{w})$. Since S^- is a 2-approximate solution of $(G[N^-(p) \setminus D], \tilde{w})$ and S^+ is a 2-approximate solution of $(G[N^+(p) \setminus D], \tilde{w})$, we have that $\tilde{w}(S^- \cup S^+) = \tilde{w}(S^-) + \tilde{w}(S^+) \leq 2(\tilde{w}(R^-) + \tilde{w}(R^+)) \leq 2\tilde{w}(R^- \cup R^+)$. Hence, $S^- \cup S^+$ is a 2-approximate p-disjoint solution of $(G - D, \tilde{w})$. Then by Lemma 5, $S^- \cup S^+ \cup D$ is a 2-approximate p-disjoint solution of (G, w). This completes the proof of the corollary. □

We are now ready to prove the following theorem.

Theorem 2. *There exists a randomized algorithm that, given a tournament G on n vertices and a weight function w on G, runs in time $O(n^{34})$ and outputs a feedback vertex set S of G. With probability at least $1/2$, S is a 2-approximate solution of (G, w).*

Proof. We first describe the algorithm. On input (G, w), if G has at most 10 vertices, then the algorithm finds an optimal solution by exhaustively enumerating and comparing all potential solutions. Otherwise the algorithm iteratively computes at most 26 solutions of (G, w) by making recursive calls. It then outputs the least weight FVS among them. We now describe the iterations and the recursive calls. Let us index the iteration by $i \in \{0, 1, \ldots, 25\}$.

The first iteration is different from the other 25 iterations. In this iteration, the algorithm sets $D \subseteq V(G)$ to be the set of the $\frac{n}{6}$ vertices of smallest weight in $V(G)$ and $\Delta = \max_{v \in D} w(v)$. Let $w' : V(G) \setminus D \to \mathbb{N}$ be the weight function that assigns the weight $w(v) - \Delta$ to each vertex v of $G - D$. The algorithm calls itself recursively on $(G - D, w')$. The recursive call returns an FVS S of $G - D$, the algorithm constructs the FVS $S_0 = S \cup D$ of G.

We do the remaining 25 iterations only when the set $\{v : N^+(v) \leq 8n/9, N^-(v) \leq 8n/9\}$ is nonempty. For each of these 25 iterations (which we index by $i \in \{1, 2, \ldots, 25\}$), the algorithm picks a vertex p_i uniformly at random from the set of vertices $\{v : N^+(v) \leq 8n/9, N^-(v) \leq 8n/9\}$; p_i is the pivot vertex for the ith iteration. For each p_i the algorithm runs the procedure Reduce on G, p_i, and w and obtains a set D_i and a weight function $\tilde{w}_i$. It then makes two recursive calls, one on $(G[N^-(p_i) \setminus D_i], \tilde{w}_i)$, and the other on $(G[N^+(p_i) \setminus D_i], \tilde{w}_i)$. Let the sets returned by the two recursive calls be S_i^- and S_i^+, respectively. The algorithm constructs the set $S_i = S_i^- \cup S_i^+ \cup D_i$ as the FVS of G corresponding to i.

Finally, the algorithm outputs the minimum weight S_i, where the minimum is taken over $0 \leq i \leq 25$ as the solution. The algorithm terminates within the claimed running time, since

the running time is governed by the recurrence $T(n) \leq 26 \cdot T(8n/9) + 25 \cdot T(n/2) + O(n^2)$, which solves to $T(n) = O(n^{38})$ by the Master theorem [8]. Here, we rely on the fact that for each pivot-vertex v, the two recursive calls made on subgraphs induced by $N^+(v)$ and $N^-(v)$, which are disjoint, and hence one of them has size at most $n/2$. We now prove that in each iteration, the constructed solution S_i is indeed an FVS of G, and that the same holds for the solution returned by the algorithm. We apply an induction on the number of vertices in G. For $n \leq 10$ there are no recursive calls made, and the returned solution is an optimal solution, since it is computed by brute force. For $n > 10$ the returned solution is one of the S_i's and so it is sufficient to prove that all S_i's are in fact feedback vertex sets of G. For S_i, $i \geq 1$ this follows from Corollary 1 and the induction hypothesis. And for $i = 0$, we know that $S_0 = S \cup D$ and S is a vertex subset returned by the recursive call for the instance $(G - D, w')$, which is also an FVS of $G - D$, by the induction hypothesis. Since $G - S_0 = ((G - D) - S)$ and S is an FVS of $(G - D)$, clearly S_0 is an FVS of G.

Finally, we will show that with probability at least 1/2, the algorithm outputs a 2-approximate solution of (G, w). We prove this by induction on n, the number of vertices in G. Suppose that S_i is of the least weight among $S_0, S_1, \ldots, S_{25}$, for some $i \in \{0, 2, \ldots 25\}$, which is output by the algorithm. For $n \leq 10$ the returned solution is optimal, so assume $n > 10$. Let S_{OPT} be an optimal solution for (G, w). We distinguish between two cases, either $|S_{OPT}| \geq 2n/3$ or $|S_{OPT}| < 2n/3$. By the induction hypothesis the first iteration, the recursive call on $(G - D, w')$ returns a 2-approximate solution S for $(G - D, w')$ with probability at least 1/2. If $|S_{OPT}| \geq 2n/3$, then it follows from Lemma 3 (with $\alpha = 2/3$) that S_i for $i = 0$, is a 2-approximate solution for (G, w).

Suppose now that $|S_{OPT}| < 2n/3$. We will argue that in each of the 25 remaining iterations the probability that $p_i \notin S_{OPT}$ is at least 1/9. Indeed, $G - S_{OPT}$ is an acyclic tournament on at least $n/3$ vertices. Let R be the set of vertices in $V(G) \setminus S_{OPT}$ excluding the first $\lfloor n/9 \rfloor$ vertices and the last $\lfloor n/9 \rfloor$ vertices in the unique topological order of the acyclic tournament $G - S_{OPT}$. For each vertex v in R it holds that $|N^+(v)| \leq n - \lfloor n/9 \rfloor - 1 \leq 8n/9$ and similarly $|N^-(v)| \leq 8n/9$, i.e., $R \subseteq \{v : |N^+(v)| \leq 8n/9, |N^-(v)| \leq 8n/9\}$. Furthermore, $|R| \geq n/9$, since $|V(G) \setminus S_{OPT}| \geq n/3$. Hence, when we pick a random vertex p_i among all vertices with in-degree and out-degree at most $8n/9$, we have that with probability at least 1/9 the vertex p_i is in R, and therefore not in S_{OPT}.

We shall say that an iteration i with $i \geq 1$ is *good* if $p_i \notin S_{OPT}$ and the two solutions S_i^- and S_i^+ returned from the recursive calls on $(G[N^-(p_i) \setminus D_i], \tilde{w}_i)$ and $(G[N^+(p_i) \setminus D_i], \tilde{w}_i)$, are 2-approximate for their respective instances. Since $p_i \notin S_{OPT}$ with probability at least 1/9, and each of S_i^- and S_i^+ are 2-approximate with probability at least 1/2 (by the induction hypothesis), it follows that this iteration is good with probability at least $1/9 \cdot 1/2 \cdot 1/2 \geq 1/36$. Therefore, with probability at least

$$1 - (1 - 1/36)^{25} \geq 1/2,$$

there is at least one iteration i that is good. For this iteration it follows from Corollary 1 that $S_i = D_i \cup S_i^+ \cup S_i^-$ is 2-approximate p_i-disjoint solution of (G, w). Moreover, since $p_i \notin S_{OPT}$, S_{OPT} is also an optimal p_i-disjoint solution of (G, w). Hence, $w(S_i) \leq 2w(S_{OPT})$. Therefore, the solution output by the algorithm is a 2-approximate solution with probability at least 1/2. This concludes the proof. □

3.1 Deterministic 2-approximation in Quasi-polynomial Time

We can easily derandomize the above algorithm in quasi-polynomial time. Instead of randomly selecting the pivots p_i, we iterate over all the candidates in $\{v : N^+(v) \leq 8n/9, N^-(v) \leq 8n/9\}$. The correctness of this algorithm follows from the same arguments as above, and we obtain a deterministic 2-approximation algorithm for TFVS. To bound the running time, observe that the number of recursive calls will be at most $2n + 1$. Thus, the running time of the algorithm will be

governed by the recurrence $T(n) \leq (2n+1) \cdot T(8n/9) + O(n^2)$, which solves to $T(n) = n^{O(\log n)}$. Thus, we get the following theorem.

Theorem 1. *There exists an algorithm that given an instance (G, w) of TFVS on n vertices, runs in time $n^{O(\log n)}$ and outputs a 2-approximate solution of (G, w).*

4 IMPROVING THE RUNNING TIME

In this section, we present an improved implementation of the approximation algorithm in Theorem 2, that attains a running time of $O(n^{17})$ for general weighted instances. This running time can be further improved to $O(n^{12})$ for unweighted instances.

Theorem 3. *There exists a randomized algorithm that, given a tournament G on n vertices and a weight function v on $V(G)$, runs in time $O(n^{17})$ and outputs a feedback vertex set S of G. With probability at least $7/10$, S is a 2-approximate solution of (G, w).*

Proof. Our algorithm is identical to the one presented in the proof of Theorem 2 with a few updated parameters and an improved running time analysis. Let us fix an optimum solution $S^\star$ to (G, w). In particular, the following parameters of the algorithm are carefully chosen.

- Let $r \geq 1/2$ denote the minimum probability of the event where our algorithm finds a 2-approximate solution. Observe that r is a lower-bound on the probability of success for any recursive calls made by the algorithm. Note that in Theorem 2, we set $r = 1/2$.
- Let $\alpha \in (1/2, 1)$ denote the threshold on the cardinality of $S^\star$ beyond which it is considered large. That is, if $|S^\star| \geq \alpha n$, then we apply Lemma 3, and then recursively solve an instance on $(\frac{3}{2} - \alpha)n$ vertices in the (special) first iteration of the algorithm. Recall that in Theorem 2, we set $\alpha = 2/3$, and then recursively solved an instance on $5n/6$ vertices in the first iteration.
- Let $\beta \in (0, 1)$ denote the fraction of vertices in $G - S^\star$ that are suitable candidates for the pivot vertex (from the second iteration onward), where $S^\star$ is an optimum solution of (G, w). Let $R^\star \subseteq V(G) \setminus S^\star$ denote this subset of vertices. Note that, if $|S^\star| < \alpha n$, then $|V(G) \setminus S^\star| \geq (1-\alpha)n$ and $|R^\star| \geq \beta(1-\alpha)n$. Now, $G - S^\star$ is an acyclic tournament with a unique topological order, and let us define $R^\star$ to be the subset of vertices of $G - S^\star$ containing all but the first $\frac{(1-\alpha)(1-\beta)}{2}n$ and last $\frac{(1-\alpha)(1-\beta)}{2}n$ vertices in this topological order. Therefore, for any $v \in R$, $|N^+(v)|, |N^-(v)| \geq \frac{(1-\alpha)(1-\beta)}{2}n$, which implies that $|N^+(v)|, |N^-(v)| \leq (1 - \frac{(1-\alpha)(1-\beta)}{2})n$ for any vertex $v \in R$. Recall that we set $\beta = 1/3$ in Theorem 2.
- Finally, let k denote the number of iterations in our algorithm, and it will be chosen to ensure that we compute a 2-approximate solution to (G, w) with probability at least r. In Theorem 2, this was set to 26, which includes the special first iteration where only one recursive call is made, and the remaining 25 iterations where two recursive calls are made.

Let us now describe how we arrive at our choice of k, given r, α and β. The first iteration considers the case when $|S^\star| \geq \alpha n$. In the remaining $k-1$ iterations, we assume that $|S^\star| \leq \alpha n$. Recall that in each of the last $k-1$ iterations, we attempt to obtain a vertex $v \in R^\star$ as the pivot vertex, by sampling a vertex uniformly at random. The probability of success is at least $|R^\star|/n \geq (1-\alpha)\beta$.[2] We can improve this probability by slightly adjusting the sampling process and its analysis as follows. First, observe that we only require the following two properties of a pivot vertex v: (i) $v \notin S^\star$ and (ii) $|N^+(v)|, |N^-(v)| \leq (1 - \frac{(1-\alpha)(1-\beta)}{2})n$. Therefore, let $R_{\alpha,\beta}$ denote the subset of vertices in G with $|N^+(v)|, |N^-(v)| \leq (1 - \frac{(1-\alpha)(1-\beta)}{2})n$, and let $R = R_{\alpha,\beta} \setminus S^\star$. Observe that any vertex in R is

[2] In Theorem 2 for $\alpha = 2/3$, $\beta = 1/3$, and $r = 1/2$, this probability is at least $1/9$.

a suitable pivot vertex. Further note that $R^\star \subseteq R \subseteq R_{\alpha,\beta}$. Let v be a vertex sampled uniformly at random from $R_{\alpha,\beta}$. We claim that $v \in R$ with probability at least $\frac{(1-\alpha)\beta}{\alpha+(1-\alpha)\beta}$. Indeed, let $Z = R \setminus R^\star$ and let $|R^\star| = ((1-\alpha)\beta + \gamma)n$ for some $\gamma \geq 0$, and then we have

$$\begin{aligned} Pr_{v\sim R_{\alpha,\beta}}[v \in R] &= |R|/|R_{\alpha,\beta}| \\ &\geq |R|/|S^\star \cup R_{\alpha,\beta}| \\ &= \frac{|R^\star| + |Z|}{|S^\star| + |R^\star| + |Z|} \\ &\geq \frac{|R^\star|}{|S^\star| + |R^\star|} \\ &= \frac{(1-\alpha)\beta + \gamma}{\alpha + (1-\alpha)\beta + \gamma} \\ &\geq \frac{(1-\alpha)\beta}{\alpha + (1-\alpha)\beta}. \end{aligned}$$

In the above, we have used the fact $\frac{a+c}{b+c} \geq \frac{a}{b}$ whenever $a \leq b$ and $a, b, c \geq 0$.

Recall that in our algorithm, from the second iteration onward, we first sample a pivot vertex v. Then, we apply procedure Reduce to G, v, p to obtain $D \subseteq V(G)$ and a weight function w'. We then recursively solve two sub-instances (G_1, w') and (G_2, w'), where $V(G_1) = N^+(v) \setminus D$ and $V(G_2) = N^-(v) \setminus D$. These recursive calls produce solutions S_1 and S_2, and we produce $S = D \cup S_1 \cup S_2$ as an approximate solution to (G, w), in this iteration. Let us compute the probability that this iteration succeeds, i.e., the probability that S a 2-approximate solution. This is equal to the probability that $v \in R$ and then the two recursive calls also compute 2-approximate solutions to (G_1, w') and (G_2, w'), respectively. Hence, the probability that one iteration succeeds is at least $r^2 \frac{(1-\alpha)\beta}{\alpha+(1-\alpha)\beta}$. We need to boost this probability to at least r, and for this, we have $k-1$ such iterations. We choose the smallest value of k, which satisfies the following inequality:

$$\left(1 - r^2 \frac{(1-\alpha)\beta}{\alpha + (1-\alpha)\beta}\right)^{k-1} \leq 1 - r.$$

Given r, α, and β, we solve the above equation to compute the number of iterations k; which ensures that we compute a 2-approximate solution to (G, w) with probability at least r.

Next, let us compute the running time. Observe that in the first iteration, which considers the case when $|S^\star| \geq \alpha n$, we apply Lemma 3 and recursively solve an instance with at most $(\frac{3}{2} - \alpha)n$ vertices. In the remaining $k-1$ instances, we make two recursive calls, on graphs with at most $(1 - \frac{(1-\alpha)(1-\beta)}{2})n$ vertices. Further, note that the two instances have no common vertices. Hence, we can say that the recursive calls are made on two instance with δn and $(1-\delta)n$ vertices for some $1/2 \leq \delta \leq \frac{2-(1-\alpha)(1-\beta)}{2}$. Observe that $\delta \geq 1/2$ and $1 - \delta \leq 1/2$, and hence we make two recursive calls on instances with at most $\frac{2-(1-\alpha)(1-\beta)}{2}n$ vertices, and $\frac{n}{2}$ vertices, respectively. Therefore, the running time of our algorithm is

$$T(n) \leq T\left(\left(\frac{3}{2} - \alpha\right)n\right) + (k-1)T\left(\frac{2-(1-\alpha)(1-\beta)}{2}n\right) + (k-1)T\left(\frac{n}{2}\right) + O(n^2).$$

Here, the term $O(n^2)$ denotes the time spent by the algorithm in processing the current instance and preparing the sub-instances for the recursive calls (i.e in Lemma 3 and Corollary 1). In Theorem 2, we simplified and solved the above recurrence using the Master Theorem, which yielded a running time of $O(n^{34})$. Here, we apply the Akra-Bazzi method [2] to solve the above equation,

which yields the following bound:

$$T(n) \leq \Theta\left(n^p\left(1 + \int_1^n \frac{x^2}{x^{p+1}}dx\right)\right),$$

where p satisfies

$$\left(\frac{3}{2} - \alpha\right)^p + (k-1)\left(\frac{(1-\alpha)(1-\beta)}{2}\right)^p + (k-1)\left(\frac{1}{2}\right)^p = 1.$$

This leads to a running time of $O(n^p)$, assuming $p \geq 3$.

It can be determined that when we set $\alpha = 0.55, \beta = 0.1855$, and $r = 0.715$, we have $k = 19$ iterations and $p \sim 16.9$. Hence, the running time of the algorithm is $O(n^{17})$.[3] Since the probability of finding a 2-approximate solution to (G, w), denoted by r, is more than $7/10$, we have the claimed result. □

Next, we present an improved running time for unweighted instances.

Theorem 4. *There exists a randomized algorithm that, given a tournament G on n vertices, runs in time $O(n^{12})$ and outputs a feedback vertex set S of G. With probability at least $8/10$, S is a 2-approximate solution of G.*

Proof. Our algorithm and analysis is nearly the same as in Theorem 3. The main change is in the choice of the parameter α, which denotes the threshold at which $S^\star$ is considered large. In particular, we set $\alpha = \frac{1}{2}$. This corresponds to the case where there is an optimum solution $S^\star$ that contains at least half of the vertices of G, and hence $V(G)$ itself is a 2-approximate solution. Therefore, for $\alpha = \frac{1}{2}$, the first iteration of our algorithm does not lead to a recursive call. In the remaining iterations, our algorithm is the same as in Theorem 4; here it is helpful to assume that there is a weight function w that gives weight 1 to every vertex. From α, β, and r, we determine the number of iterations k, and then bound the running time of our algorithm as

$$T(n) \leq (k-1)T\left(\frac{2-(1-\alpha)(1-\beta)}{2}n\right) + (k-1)T\left(\frac{n}{2}\right) + O(n^2).$$

When we set $\alpha = 0.5, \beta = 0.223$, and $r = 0.8$, we get that the number of iterations is $k = 14$ (including the first iteration where no recursive calls are made), and $p \sim 11.9$. Hence, the running time of the algorithm is $O(n^{12})$. Since $r = 8/10$, we obtain the claimed result. □

Theorems 3 and 4 together give us Theorem 1.

5 CONCLUSIONS

We presented a simple randomized 2-approximation algorithm for Feedback Vertex Set in Tournaments. Assuming the Unique Games conjecture, the approximation ratio is optimal. It runs in time $O(n^{34})$. We then presented an improved algorithm, with a more sophisticated running time analysis that runs in time $O(n^{17})$. This running time can be improved to $O(n^{12})$ for unweighted instances. We also present a de-randomization of our algorithm that runs in quasi-polynomial time.

It would be interesting to see whether one can achieve the same approximation ratio can be obtained by an algorithm with a running time of $O(n^2)$ (i.e., linear in input size) or something

[3] These values were obtained by solving the above equations on a computer.

close to it. Another interesting open problems is to design a deterministic polynomial time 2-approximation algorithm. Finally, it would be interesting to see whether ideas from this article can be used to obtain improved approximation algorithms for other "structured hitting-set" problems.

REFERENCES

[1] Nir Ailon, Moses Charikar, and Alantha Newman. 2008. Aggregating inconsistent information: Ranking and clustering. *J. ACM* 55, 5 (2008), 23:1–23:27. DOI: https://doi.org/10.1145/1411509.1411513

[2] Mohamad Akra and Louay Bazzi. 1998. On the solution of linear recurrence equations. *Comput. Optimiz. Appl.* 10, 2 (1998), 195–210.

[3] Vineet Bafna, Piotr Berman, and Toshihiro Fujito. 1999. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM J. Discrete Math.* 12, 3 (1999), 289–297.

[4] Jørgen Bang-Jensen and Gregory Z. Gutin. 2009. *Digraphs—Theory, Algorithms and Applications*, 2nd ed. Springer.

[5] R. Bar-Yehuda and S. Even. 1981. A linear-time approximation algorithm for the weighted vertex cover problem. *J. Algor.* 2, 2 (1981), 198–203. DOI: https://doi.org/10.1016/0196-6774(81)90020-1

[6] Mao-cheng Cai, Xiaotie Deng, and Wenan Zang. 2000. An approximation algorithm for feedback vertex sets in tournaments. *SIAM J. Comput.* 30, 6 (2000), 1993–2007.

[7] Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. 2008. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM* 55, 5 (2008), 21:1–21:19.

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms (3rd ed.)*. MIT Press, Cambridge, MA.

[9] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. 2011. Solving connectivity problems parameterized by treewidth in single exponential time. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, Rafail Ostrovsky (Ed.). IEEE Computer Society, 150–159. DOI: https://doi.org/10.1109/FOCS.2011.23

[10] Irit Dinur, Venkatesan Guruswami, Subhash Khot, and Oded Regev. 2005. A new multilayered PCP and the hardness of hypergraph vertex cover. *SIAM J. Comput.* 34, 5 (2005), 1129–1146.

[11] Michael Dom, Jiong Guo, Falk Hüffner, Rolf Niedermeier, and Anke Truß. 2010. Fixed-parameter tractability results for feedback set problems in tournaments. *J. Discrete Algor.* 8, 1 (2010), 76–86.

[12] P. Erdős and L. Pósa. 1965. On independent circuits contained in a graph. *Can. J. Math.* 17 (1965), 347–352.

[13] Guy Even, Joseph Naor, Baruch Schieber, and Madhu Sudan. 1998. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica* 20, 2 (1998), 151–174.

[14] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co.

[15] Serge Gaspers and Matthias Mnich. 2013. Feedback vertex sets in tournaments. *J. Graph Theory* 72, 1 (2013), 72–89.

[16] Claire Kenyon-Mathieu and Warren Schudy. 2007. How to rank with few errors. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*. ACM, 95–103. DOI: https://doi.org/10.1145/1250790.1250806

[17] Subhash Khot and Oded Regev. 2008. Vertex cover might be hard to approximate to within 2-ε. *J. Comput. Syst. Sci.* 74, 3 (2008), 335–349.

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

[19] Mithilesh Kumar and Daniel Lokshtanov. 2016. Faster exact and parameterized algorithm for feedback vertex set in tournaments. In *Proceedings of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS'16) (LIPIcs)*, Nicolas Ollinger and Heribert Vollmer (Eds.), Vol. 47. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 49:1–49:13. DOI: https://doi.org/10.4230/LIPIcs.STACS.2016.49

[20] Matthias Mnich, Virginia Vassilevska Williams, and László A. Végh. 2016. A 7/3-approximation for feedback vertex sets in tournaments. In *Proceedings of the 24th Annual European Symposium on Algorithms (ESA'16) (LIPIcs)*, Piotr Sankowski and Christos D. Zaroliagis (Eds.), Vol. 57. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 67:1–67:14. DOI: https://doi.org/10.4230/LIPIcs.ESA.2016.67

[21] Venkatesh Raman and Saket Saurabh. 2006. Parameterized algorithms for feedback set problems and their duals in tournaments. *Theor. Comput. Sci.* 351, 3 (2006), 446–458.

[22] Igor Razgon. 2007. Computing minimum directed feedback vertex set in $O(1.9977^n)$. In *Proceedings of the 10th Italian Conference on Theoretical Computer Science (ICTCS'07)*, Giuseppe F. Italiano, Eugenio Moggi, and Luigi Laura (Eds.). World Scientific, Singapore, 70–81.

[23] Bruce Reed, Neil Robertson, Paul Seymour, and Robin Thomas. 1996. Packing directed circuits. *Combinatorica* 16, 4 (1996), 535–554.

[24] Ewald Speckenmeyer. 1989. On feedback problems in diagraphs. In *Proceedings of the 15th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'89) (Lecture Notes in Computer Science)*, Manfred Nagl (Ed.), Vol. 411. Springer, 218–231. DOI: https://doi.org/10.1007/3-540-52292-1_16

[25] David P. Williamson and David B. Shmoys. 2011. *The Design of Approximation Algorithms*. Cambridge University Press, Cambridge, UK.

[26] Mingyu Xiao and Hiroshi Nagamochi. 2015. An improved exact algorithm for undirected feedback vertex set. *J. Comb. Optim.* 30, 2 (2015), 214–241.

Received October 2019; revised September 2020; accepted January 2021