2-Approximating Feedback Vertex Set in Tournaments

Daniel Lokshtanov* Pranabendu Misra† Joydeep Mukherjee‡ Fahad Panolan§
Geevarghese Philip¶ Saket Saurabh‖

Abstract

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 paper 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.

*University of California, Santa Barbara, USA. daniello@ucsb.edu
†University of Bergen, Bergen, Norway. pranabendu.misra@uib.no
‡Indian Statistical Institute, Kolkata, India. joydeep.m1981@gmail.com
§University of Bergen, Bergen, Norway. fahad.panolan@uib.no
¶Chennai Mathematical Institute, India. gphilip@cmi.ac.in
‖The Institute of Mathematical Sciences, HBNI, Chennai, India. saket@imsc.res.in

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 version of the problem are NP-complete [14] and have been extensively studied from the perspective of approximation algorithms [1, 12], parameterized algorithms [6, 8, 19], exact exponential time algorithms [23, 29] as well as graph theory [11, 24].

In this paper 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 [28] 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 DFVS has applications in voting systems and rank aggregation and is quite well-studied [5, 10, 15, 22, 21, 20]. 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 [10]. 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 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 $c < 3$ [18]. A c-approximation algorithm with $c < 2$ implies $\mathsf{P} \neq \mathsf{NP}$ [9].

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. [5], who gave a $5/2$-approximation algorithm using the local ratio technique of Bar-Yehuda and Even [3]. Recently, Mnich et al. [21] gave a $7/3$-approximation algorithm using the iterative rounding technique. They also observe that the approximation-preserving reduction from Vertex Cover to TFVS of Speckenmeyer [26] implies that, assuming the Unique Games Conjecture (UGC) [18], TFVS cannot have an approximation algorithm with factor smaller than 2. Mnich et al. [21] 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 paper 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 1.1. *There exists a randomized algorithm that, given a tournament G on n vertices and a weight function w on G, runs in time $\mathcal{O}(n^{34})$ and outputs a feedback vertex set S of G. With probability at least $1/2$, S is a 2-approximate solution of (G, w).*

This algorithm can be easily derandomized in quasi-polynomial time.

[1] There is a variable $0 \leq x_v \leq 1$ for every element v and 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.

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 $|\mathsf{OPT}| \geq n/2$ then every feasible solution (such as the entire vertex set!) is a 2-approximate solution. Assuming that $|\mathsf{OPT}| < n/2$, a randomly chosen vertex p will be not in OPT with constant probability. Further, with constant probability p will be "in the middle one third" of the unique topological ordering of $G - \mathsf{OPT}$. In other words, with constant probability p will have at least $(G - |\mathsf{OPT}|)/3 \geq n/6$ in-neighbors and at least $(G - |\mathsf{OPT}|)/3 \geq n/6$ 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 is a pivot in a "*quicksort-like*" procedure.

If there exists an arc uv from $N^+(p)$ to $N^-(p)$ then puv forms a directed triangle and hence, since $p \notin \mathsf{OPT}$, 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 $|\mathsf{OPT}|$ 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 sub-instances. 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 $T(n) = n^{\mathcal{O}(1)}$ 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* succeds, 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 $T(n) \leq \mathcal{O}(1) \cdot T(5n/6)$, which still solves to $T(n) = n^{\mathcal{O}(1)}$.

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 $T(n) \leq n^{\mathcal{O}(\log n)}$.

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 $|\mathsf{OPT}| > n/2$ 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 paper 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 paper 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 [2]. For an acyclic tournament, the topological sort is unique [2]. *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 denotes 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 $w(S) \geq w(S_{OPT})$. A 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). On the other hand 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 which 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

The algorithm will distinguish between two cases: either the optimal solution contains many (more than two thirds 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 which has an optimal solution $S^\star$ that contains at least $2n/3$ vertices of G. Let $D \subseteq V(G)$ be a set of $\frac{n}{6}$ 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 $w' \colon V(G) \setminus D \to \mathbb{N}$ be the weight function which assigns the weight $w(v) - \Delta$ to each vertex v of $G - D$. If R_{approx} is a 2-approximate solution of the reduced instance $(G - D, w')$ then $R_{approx} \cup D$ 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 $w'(R_{approx}) \leq 2w'(R^\star)$. 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 $|S^\star \setminus D| \geq (\frac{2n}{3} - \frac{n}{6}) = \frac{n}{2}$ we get that $w'(S^\star \setminus D) \leq (w(S^\star) - \frac{\Delta \cdot n}{2})$. Hence $w'(R^\star) \leq w'(S^\star \setminus D) \leq (w(S^\star) - \frac{\Delta \cdot n}{2})$.

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

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

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

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_{approx} of the reduced instance $(G - v, w')$, we have $R_{approx} \cup \{v\}$ is a 2-approximate p-disjoint solution of the original instance (G, w).*

Proof. Since $(G - v) - R_{approx} = G - (R_{approx} \cup \{v\})$ and the former digraph is acyclic by assumption, we get that $R_{approx} \cup \{v\}$ is a FVS in the digraph G. We will show that $R_{approx} \cup \{v\}$ is a 2-approximate p-disjoint solution of (G, w). Since $p \notin R_{approx}$, $R_{approx} \cup \{v\}$ 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 $w(R_{approx} \cup \{v\}) \leq 2w(S^\star)$. Let $\Delta = \min\{w(u), w(v)\}$, that is $w(v) = \Delta$. Now we have the following.

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

This completes the proof. □

Suppose that we have picked a pivot vertex p that is disjoint from an optmal 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 $\tilde{w} = w_i$.

Algorithm 1 The reduction procedure.

```
1:  procedure Reduce( G, w, p )
2:      D_0 ← ∅; w_0 ← w; i ← 0
3:      while G has an arc xy ; x ∈ (N^+(p) \ D_i), y ∈ (N^-(p) \ D_i) do
4:          if w_i(x) ≤ w_i(y) then                ▷ definition of the vertices u and v
5:              v ← x; u ← y
6:          else
7:              v ← y; u ← x
8:          w_i(u) ← w_i(u) − w_i(v)
9:          w_i(v) ← 0
10:         w_{i+1} ← w_i                          ▷ w_{i+1} is now the weight function w' from the discussion
11:         D_{i+1} ← D_i ∪ {v}
12:         i ← i + 1
13:     D ← D_i; w̃ ← w_i
14:     return (D, w̃)
```

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 $\mathcal{O}(|V(G)|^2)$ 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 $\mathcal{O}(n^2)$ 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 $\mathcal{O}(n)$ 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 $\mathcal{O}(n^2)$ 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 which 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 $\mathcal{O}(n)$ 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 $\mathcal{O}(n^2)$ time. □

Combining Lemma 1, Lemma 2, and Lemma 5 we get

Corollary 1. *On input (G, w, p) the procedure* Reduce *runs in $\mathcal{O}(n^2)$ 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 our main theorem.

Theorem 1.1. *There exists a randomized algorithm that, given a tournament G on n vertices and a weight function w on G, runs in time $\mathcal{O}(n^{34})$ 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 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 $w' \colon V(G) \setminus D \to \mathbb{N}$ be the weight function which 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 non-empty. 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 $\{v : N^+(v) \leq 8n/9, N^-(v) \leq 8n/9\}$. 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 $T(n) \leq 51 \cdot T(8n/9) + \mathcal{O}(n^2)$ which solves to $T(n) = \mathcal{O}(n^{34})$ by the Master theorem [7]. 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, 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 $|S_{OPT}| < 2n/3$. If $|S_{OPT}| \geq 2n/3$ then, 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 $1/2$. In this case it follows from Lemma 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 $(G[N^+(p_i) \setminus D_i], \tilde{w}_i)$, respectively 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 which 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 $T(n) \leq (2n + 1) \cdot T(8n/9) + \mathcal{O}(n^2)$ which solves to $T(n) = n^{\mathcal{O}(\log n)}$. Thus we get the following theorem.

Theorem 3.1. *There exists an algorithm that given an instance (G, w) of TFVS on n vertices, runs in time $n^{\mathcal{O}(\log n)}$ and outputs a 2-approximate solution of (G, w).*

4 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. However there is still some room for improvement. First and foremost, is it possible to obtain a deterministic 2-approximation algorithm? Further, for the sake of clarity of presentation we did not attempt at all to optimize the running time of the algorithm. The exponent 34 can be brought down substantially by implementing the following.

1. Changing the threshold $2n/3$ for when $|S_{OPT}|$ is considered big (and the first recursive call returns an optimal solution) to αn. In this case the set D must be chosen to be the set of $(\alpha - 1/2)n$ vertices of smallest weight.
2. Changing the success probability with which the algorithm returns a solution from $1/2$ to some constant r. This allows to reduce the number of iterations.
3. Changing the maximum indegree and outdegree of the sampled vertices p_i from $8n/9$ to βn. This gives a trade-off between the probability that each iteration is good, and the upper bound on the size of the digraphs $G[N^-(p_i) \setminus D_i]$ and $G[N^+(p_i) \setminus D_i]$ in the recursive calls.
4. Instead of computing the probability that the pivot p_i is in R, computing the probability that p_i is not in S_{OPT}. In particular vertices in $V(G) \setminus (S_{OPT} \cup R)$ either have both indegree and outdegree at most $\lfloor 8n/9 \rfloor$, in which case they contribute equally to the numerator and the denominator of the probability, or they do not, in which case they contribute to neither the numerator nor the denominator. The worst probability is achieved in the latter case, making the probability that $p_i \notin S_{OPT}$ be at least $1/7$ (instead of the lower bound of $1/9$ of being in R).
5. Not using the same upper bound on the number of vertices in all recursive calls. The first recursive call is made on an instance with (potentially) fewer vertices. More importantly, in each of the remaining iterations the algorithm makes two recursive calls, one with $\gamma_i n$ vertices and the other with $(1 - \gamma_i)n$ vertices. In our analysis we just used that $\gamma_i \leq 8/9$ and $(1 - \gamma_i) \leq 8/9$ without also using that in the worst case when $\gamma_i = 8/9$ we have $1 - \gamma_i = 1/9$.
6. Taking point 5 one step further, after the algorithm has sampled p_i it can observe what γ_i is. It may then make several recursive calls on $G[N^-(p_i) \setminus D_i]$ and on $G[N^+(p_i) \setminus D_i]$, this gives another tradeoff between the time spent and the success probability that a particular iteration is good. Note that the number of recursive calls on $G[N^-(p_i) \setminus D_i]$ and on $G[N^+(p_i) \setminus D_i]$ need not be the same - indeed it pays off to make more recursive call to the smaller instance, since that provides the best trade-off between running time and success probability. In particular the number of calls on $G[N^-(p_i) \setminus D_i]$ and on $G[N^+(p_i) \setminus D_i]$ should be chosen as a function of γ_i.

Nevertheless this is still a far cry from a practical running time, and 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 $\mathcal{O}(n^2)$ (i.e. linear in input size) or something close.

Finally it would be interesting to see whether ideas from this algorithm can be used to improve approximation algorithms for other "structured hitting-set" problems. Here the Cluster Vertex Deletion problem is a possible candidate.

References

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

[2] Jørgen Bang-Jensen and Gregory Z. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer Publishing Company, Incorporated, 2nd edition, 2008.

[3] R Bar-Yehuda and S Even. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198 – 203, 1981.

[4] Reuven Bar-Yehuda and Dror Rawitz. On the equivalence between the primal-dual schema and the local ratio technique. *SIAM Journal on Discrete Mathematics*, 19(3):762–797, 2005.

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

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

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

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

[9] Irit Dinur, Venkatesan Guruswami, Subhash Khot, and Oded Regev. A new multilayered pcp and the hardness of hypergraph vertex cover. *SIAM Journal on Computing*, 34(5):1129–1146, 2005.

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

[11] P Erdős and L Pósa. On independent circuits contained in a graph. *Canad. J. Math*, 17:347–352, 1965.

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

[13] Guy Even, Joseph Seffi Naor, Satish Rao, and Baruch Schieber. Divide-and-conquer approximation algorithms via spreading metrics. *Journal of the ACM (JACM)*, 47(4):585–616, 2000.

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

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

[16] Venkatesan Guruswami, Johan Håstad, Rajsekar Manokaran, Prasad Raghavendra, and Moses Charikar. Beating the random ordering is hard: Every ordering csp is approximation resistant. *SIAM Journal on Computing*, 40(3):878–914, 2011.

[17] Venkatesan Guruswami and Euiwoong Lee. Simple proof of hardness of feedback vertex set. *Theory of Computing*, 12(1):1–11, 2016.

[18] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2- ε. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

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

[20] Mithilesh Kumar and Daniel Lokshtanov. Faster exact and parameterized algorithm for feedback vertex set in tournaments. In *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, pages 49:1–49:13, 2016.

[21] Matthias Mnich, Virginia Vassilevska Williams, and László A Végh. A 7/3-approximation for feedback vertex sets in tournaments. In *24th Annual European Symposium on Algorithms (ESA 2016)*. Schloss Dagstuhl, 2016.

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

[23] Igor Razgon. Computing minimum directed feedback vertex set in $o(1.9977^n)$. In *Theoretical Computer Science, 10th Italian Conference, ICTCS 2007, Rome, Italy, October 3-5, 2007, Proceedings*, pages 70–81, 2007.

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

[25] Paul D. Seymour. Packing directed circuits fractionally. *Combinatorica*, 15(2):281–288, 1995.

[26] Ewald Speckenmeyer. On feedback problems in digraphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 218–231. Springer, 1989.

[27] Ola Svensson. Hardness of vertex deletion and project scheduling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 301–312. Springer, 2012.

[28] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.

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