Rigging Nearly Acyclic Tournaments Is Fixed-Parameter Tractable

M. S. Ramanujan and **Stefan Szeider**
Algorithms and Complexity Group, TU Wien, Vienna, Austria
[ramanujan,sz]@ac.tuwien.ac.at

Abstract

Single-elimination tournaments (or knockout tournaments) are a popular format in sports competitions that is also widely used for decision making and elections. In this paper we study the algorithmic problem of manipulating the outcome of a tournament. More specifically, we study the problem of finding a seeding of the players such that a certain player wins the resulting tournament. The problem is known to be NP-hard in general. In this paper we present an algorithm for this problem that exploits structural restrictions on the tournament. More specifically, we establish that the problem is *fixed-parameter tractable* when parameterized by the size of a smallest feedback arc set of the tournament (interpreting the tournament as an oriented complete graph). This is a natural parameter because most problems on tournaments (including this one) are either trivial or easily solvable on *acyclic* tournaments, leading to the question—what about *nearly acyclic* tournaments or tournaments with a small feedback arc set? Our result significantly improves upon a recent algorithm by Aziz et al. (2014) whose running time is bounded by an exponential function where the size of a smallest feedback arc set appears in the exponent and the base is the number of players.

Introduction

Single-elimination tournaments, also known as knockout tournaments, are a popular format in sports competitions. It is used, for instance, at the Wimbledon tennis tournament and is widely applied in decision making and elections. As a result, the study of *manipulating* the results of knockout tournaments in various ways has received a lot of attention.

In the TOURNAMENT FIXING PROBLEM (TFP), we are given the results of a round-robin tournament in the form of a tournament graph on n vertices, and a special player. The objective is to decide whether there is a seeding of the n players such that the special player is the winner of the resulting knockout tournament, given the same match outcomes between each pair of players.

Aziz et al. (2014) showed that this problem is NP-hard in general while there are several results (Vassilevska Williams 2010; Stanton and Vassilevska Williams 2011; Kim and Vassilevska Williams 2015; Kim, Suksompong, and Vassilevska Williams 2016) identifying certain structural properties of the input tournament which guarantee that the required seeding exists. We take the search for tractable instances of this problem in a different direction by studying the *parameterized complexity* of this problem.

A parameterized problem $\mathcal{P}$ is a problem whose instances are tuples (I, k), where $k \in \mathbb{N}$ is called the parameter. The central notion of tractability in parameterized complexity is that of *fixed parameter tractability*. We say that a parameterized problem is *fixed parameter tractable* (FPT in short) if it can be solved by an algorithm which runs in time $f(k) \cdot |I|^{\mathcal{O}(1)}$ for some computable function f; algorithms with running time of this form are called FPT algorithms. We refer the reader to other sources (Downey and Fellows 2013; Flum and Grohe 2006; Cygan et al. 2015) for an in-depth introduction into parameterized complexity.

Numerous graph problems, when studied in the parameterized complexity setting with parameter k, turn out to be easily solvable in time $n^{f(k)}$ for some function f and the main goal in these situations is to achieve an FPT running time. However, this is not true in the case of 'graph layout' problems where the goal is to compute a permutation of the vertex set which optimizes an objective function. Even on tournaments, an algorithm with running time $n^{f(k)}$ for some of these problems is very non-trivial and highly technical (see, e.g., Fradkin and Seymour 2013).

The TFP problem is yet another graph layout problem where it is not at all obvious how one would go about designing even an algorithm that runs in time $n^{f(k)}$ where k is the size of the smallest feedback arc set in the tournament. Recently, Aziz et al. (2014) gave a clever dynamic programming algorithm with this running time, which relies on the bounded index of a certain equivalence relation induced by the feedback arc set on the set of players. In this paper, we show that the TFP problem is in fact *fixed-parameter tractable* parameterized by the size of the smallest feedback arc set by proving the following theorem.

Theorem 1. *The* TOURNAMENT FIXING PROBLEM *can be solved in time* $2^{\mathcal{O}(k^2 \log k)} n^{\mathcal{O}(1)}$ *on tournaments of size* n *with a feedback arc set of size* k.

This theorem implies that when the given tournament has a feedback arc set of size k, where $k^2 \log k = \mathcal{O}(\log n)$ then it can be decided in polynomial time whether the tournament can be fixed to make a given player win. Our algorithm relies on a combination of certain new structural properties of knockout tournaments with a small feedback arc set and an appropriate decomposition of arborescences. We believe that the structural results proved in this paper will have applications in further fixed-parameter algorithms for this problem.

Related Work The computational problem of manipulation knockout tournaments was first stated by Vu, Altman, and Shoham (2009) and has received a lot of attention (Kim, Suksompong, and Vassilevska Williams 2016; Kim and Vassilevska Williams 2015; Aziz et al. 2014; Stanton and Vassilevska Williams 2011; Vassilevska Williams 2010). Over recent years, the framework of fixed-parameter algorithms and parameterized complexity has been shown adequate and suitable for problems in computational social choice which includes questions related to tournaments and manipulations (Betzler et al. 2012; Lindner and Rothe 2008). Fixed-parameter tractability results for voting manipulations have recently been given by Hemaspaandra, Lavaee, and Menton (2016). Somewhat related is the computational problem of controlling candidates in elections, whose parameterzed complexity was recently studied in the AI context (Chen et al. 2015). Several other recent parameterized complexity results on manipulations of elections and rankings include (Yang 2014; Dey, Misra, and Narahari 2015; Bevern et al. 2016).

Preliminaries

We begin by recalling the definition of seedings of a tournament. Let D be a round robin tournament on n players. That is, for every pair of players, exactly one is picked to be the winner of the match between them (depicted in D by an arc from the winner to the loser). A knockout tournament of D is defined by a binary tree T with n leaves $L(T)$ and a bijective function $S : V(D) \to L(T)$ called the *seeding*, mapping the n players to the n leaves. Then the winner of the knockout tournament corresponding to this seeding is determined recursively: the winner at a leaf l is the player j with $l = S(j)$, and the winner of the subtree rooted at a node v is the winner of the match between the winners of the two subtournaments rooted at the children of v.

Binomial arborescences. An arborescence is a rooted directed tree such that all arcs are directed away from the root.

Definition 1 (see also Vassilevska Williams 2010). *Let D be a tournament. A* **binomial arborescence** *T rooted at $a \in V(D)$ is defined as follows.*

- *a single node a is a binomial arborescence rooted at a,*
- *if $|V(T)| = 2^i$ for some $i > 0$, then T is a binomial arborescence if a has a child b such that if T_b is the subarborescence of T rooted at b and $T_a = T \setminus T_b$, then T_a and T_b are 2^{i-1}-node binomial arborescences rooted at a and b respectively.*

If $V(T) = V(D)$, then we say that T is a spanning binomial arborescence of D.

In the rest of the paper, we will refer to a spanning binomial arborescence simply as an *s.b.a.* The relevance of binomial arborescences comes from the following statement.

Proposition 1 (Vassilevska Williams 2010). *Let D be a tournament with a special vertex $v^\star \in V(D)$. Then, there is a seeding of the vertices in D such that the resulting knockout tournament is won by $v^\star$ if and only if D has an s.b.a rooted at $v^\star$.*

Let T be a (not necessarily binomial) arborescence. For a vertex $u \in V(T)$, we denote by $\mathsf{Child}_T(u)$ the set of children of u in T, by $\mathsf{Desc}_T(u)$ the set of descendants of u in T and by $\mathsf{Ansc}_T(u)$ the set of ancestors of u in T. Note that $u \in \mathsf{Desc}_T(u) \cap \mathsf{Ansc}_T(u)$. Observe that if T is an s.b.a of D, then for every $v \in V(T)$, $|\mathsf{Desc}(v)| = 2^i$ for some $i \in [\log n] \cup \{0\}$ and v is the winner of a subtournament played by the players in $\mathsf{Desc}(v)$. For an arborescence T, we say that another arborescence T' is a *leaf-subtree of* T if T' is the subtree of T induced on the set $\mathsf{Desc}(v)$ for some $v \in V(T)$. We will use the terms vertex and player interchangeably.

ILP-feasibility. We will use as a subroutine the well-known FPT algorithm for the ILP-FEASIBILITY problem. The ILP-FEASIBILITY problem is defined as follows. The input is a pair of matrices $A \in \mathbb{Z}^{m\times p}$ and $b \in \mathbb{Z}^{m\times 1}$ and the objective is to check whether there exists a vector $\bar{x} \in \mathbb{Z}^{p\times 1}$ satisfying the m inequalities, that is, $A \cdot \bar{x} \leq b$.

Proposition 2 (Lenstra and Jr. 1983, Kannan 1987, Frank and Tardos 1987). ILP-FEASIBILITY *can be solved using $\mathcal{O}(p^{2.5p+o(p)}\cdot L)$ arithmetic operations and space polynomial in L, where L is the number of bits in the input and p is the number of variables.*

Lemma 1. *For every $k, n \in \mathbb{N}$ where $k \leq n$, $(\log n)^k \leq (4k \log k)^k + n^2$.*

The FPT algorithm for TFP

Definition 2. *Let D be a tournament with a special player $v^\star$. Let $F \subseteq A(D)$ be a smallest feedback arc set of D and let $\pi : [n] \to V(D)$ be the linear ordering of the players in decreasing order of strength obtained by flipping the arcs in F. In this ordering, player u appears before player v if and only if (u, v) is an arc. Then, we say that π* **witnesses** *F. We call the vertices in $\{v^\star\} \cup V(F)$,* **affected** *vertices and denote this set by $\mathbf{A}_F$.*

We now define the notion of a permutation γ of $X \subseteq V(D)$ 'respecting' the permutation π witnessing F. Essentially, we say that γ respects π if the vertices of X appear in the same order (relative to each other) in γ as they do in π. The formal definition follows.

Definition 3. *Let D be a tournament and $X \subseteq V(D)$. Let $\pi : [n] \to V(D)$ be a linear ordering of $V(D)$. Let $\gamma : [|X|] \to X$ be an ordering of the vertices in X such that for every $1 \leq i < j \leq |X|$, $\pi^{-1}(\gamma(i)) < \pi^{-1}(\gamma(j))$. Then, we say that γ is an ordering of X that* **respects** *π.*

We will now define a 'type' function that partitions the vertices in $V(D) \setminus \mathbf{A}_F$ into at most $|\mathbf{A}_F| + 1$ partitions using the vertices in $\mathbf{A}_F$ as 'breakpoints'. We first define the set $\mathsf{Types} = [|\mathbf{A}_F| + 1] \cup \{\mathbf{A}_F\}$.

Definition 4. *Let D be a tournament with a special player $v^\star$. Let $F \subseteq A(D)$ be a smallest feedback arc set of D and let $\pi : [n] \to V(D)$ be the linear ordering witnessing F. Let $\gamma : [|\mathbf{A}_F|] \to \mathbf{A}_F$ be the ordering of $\mathbf{A}_F$ that respects*

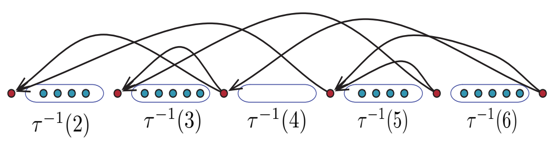

Figure 1: An illustration of the partition of the vertices of D induced by the vertices in $\mathbf{A}_F$.

π. We define the type function $\tau_\gamma^\pi : V(D) \to$ Types as follows. For every $v \in \mathbf{A}_F$, $\tau_\gamma^\pi(v) = v$. For every $v \in V(D) \setminus \mathbf{A}_F$, $\tau_\gamma^\pi(v) = i$ where i is the smallest index in $[|\mathbf{A}_F|]$ such that $\pi(v) < \pi(\gamma(i))$. If there is no such index, that is, $\pi(v) > \pi(\gamma(|\mathbf{A}_F|))$, then we set $\tau_\gamma^\pi(v) = |\mathbf{A}_F| + 1$. For each $i \in$ Types, we denote by $\mathcal{P}_i(\tau_\gamma^\pi)$ the set of vertices in the pre-image of i under τ_γ^π.

For an example of the way the type function τ_γ^π partitions $V(D) \setminus \mathbf{A}_F$, see Figure 1. When the permutations π and γ are clear from the context, we will simply refer to τ_γ^π as τ. When τ is clear from the context, we say that a vertex v has type $\tau(v)$ without explicitly referring to τ. Note that for some $i \in [|\mathbf{A}_F| + 1]$, the set $\tau^{-1}(i)$ could be empty (see Figure 1). We now observe that vertices in the same type have essentially the same behaviour with respect to every vertex *of a different type* in the graph.

Lemma 2. *Let τ be the type function as described earlier. For all $i \in$ Types, and distinct vertices $u, v \in \tau^{-1}(i)$, for any vertex $w \in V(D)$ such that $\tau(u) \neq \tau(w)$, w beats u if and only if w beats v.*

Motivated by the above lemma, for every $u \in V(D)$, we define the set $\mathsf{Types}_{\geq}(u)$ as the set $\tau(u) \cup \{i \in \mathsf{Types} \setminus \mathbf{A}_F | \forall v \in \tau^{-1}(i), (u, v) \in A(D)\}$. That is $\mathsf{Types}_{\geq}(u)$ is $\tau(u)$ plus the set of those types in $\mathsf{Types} \setminus \mathbf{A}_F$ whose 'pre-image-elements' are all beaten by u which by the above lemma also means precisely those types in $\mathsf{Types} \setminus \mathbf{A}_F$ which have at least one element which appears after u in the ordering π. We denote by $\mathsf{Types}_{<}(u)$ the set $\mathsf{Types} \setminus \mathsf{Types}_{\geq}(u)$. For a set $Z \subseteq V(D)$, we say that a vertex $v \in Z$ is the *strongest* vertex of type $\tau(v)$ in Z if v beats every other vertex $w \in Z$ such that $\tau(w) = \tau(v)$. Similarly, we say that a vertex $v \in Z$ is the *weakest* vertex of type $\tau(v)$ in Z if v is beaten by every other vertex $w \in Z$ such that $\tau(w) = \tau(v)$. If there is no $u, v \in Z$ such that $(u, v) \in F$, then the subtournament $D[Z]$ induced on D is acyclic and we have a natural notion of *the strongest* vertex in Z. This is simply the vertex of Z which beats every other vertex in Z.

Let $Z' \subseteq Z$ be a set of size ℓ such that $\tau(u) = \tau(v) = i$ for every $u, v \in Z'$. We say that Z' is the set of the *strongest* ℓ vertices of type $\tau(u)$ in Z if there is no vertex $w \in Z \setminus Z'$ such that $\tau(w) = i$ and w beats a vertex in Z'. Similarly, we say that Z' is the set of the *weakest* ℓ vertices of type $\tau(u)$ in Z if there is no vertex $w \in Z \setminus Z'$ such that $\tau(w) = i$ and w is beaten by a vertex in Z'.

We now show that the fact that the types themselves have a natural linear ordering is of great consequence.

Lemma 3. *Let D be a tournament with a special player $v^\star$, $F \subseteq A(D)$ a smallest feedback arc set of D and let π, γ, τ be as defined earlier. Then, the following statements hold.*

- *For any $u \in V(D)$ and $Z \subseteq \bigcup_{i \in \mathsf{Types}_{\geq}(u)} \tau^{-1}(i)$ such that $|Z| = 2^j$ for some $j \in [\log n]$, let v be the strongest vertex of type $\tau(u)$ in Z. Then, there is a binomial arborescence rooted at v and spanning the set Z.*
- *Let $Z_0, Z_1, \ldots, Z_\ell$ be disjoint non-empty sets of vertices such that $\sum_{j=0}^{\ell} |Z_j| = 2^p$ for some $p \in [\log n]$. Let $T_1, \ldots, T_\ell$ be binomial arborescences such that for each $j \in [\ell]$, T_j is rooted at r_j and spans Z_j. Suppose that there is no $(u, v) \in F$ such that $u, v \in Z_0$ and the strongest vertex q in Z_0 beats r_j for every $j \in [\ell]$. Then, any binomial arborescence spanning the set $\mathbf{Z}$ and containing the binomial arborescences $T_1, \ldots, T_\ell$ as leaf-subtrees, must be rooted at q.*

Proof. For the first statement, observe that by the definition of $\pi, \mathbf{A}_F, \gamma$ and hence τ, along with the set $\mathsf{Types}_{\geq}(u)$, the vertex v beats *every* other vertex in the set Z. Hence, v wins every tournament played on the vertices in Z, implying that there is a binomial arborescence rooted at v and spanning the set Z (Proposition 1).

For the second statement, suppose that there is such a binomial arborescence T' which is not rooted at q. Let $u \in \mathbf{Z}$ be such that $(u, q) \in A(T')$. It cannot be the case that $q \in V(T_j) \setminus \{r_j\}$ for any $j \in [\ell]$ since by our assumption, T_j is required to be a leaf-subtree of T'. Hence, the only remaining vertices of $\mathbf{Z}$ are the vertices in Z_0 and the vertices $r_1, \ldots, r_\ell$, all of which are beaten by q, a contradiction to our assumption that q is not the root of T'. It is straightforward to see that there is at least one binomial arborescence spanning the set $\mathbf{Z}$ and containing the binomial arborescences $T_1, \ldots, T_\ell$ as leaf-subtrees. Such a binomial arborescence (rather the corresponding tournament) can be constructed by starting with the tournaments corresponding to the trees $T_1, \ldots, T_\ell$ and arbitrarily adding matches between surviving players until we construct a knockout tournament on $\mathbf{Z}$. This completes the proof of the lemma. $\square$

LCA closure. For an arborescence T and vertex set M in $V(T)$ the least common ancestor-closure (LCA-closure) $\mathsf{LCA}(M)$ is obtained by the following process. Initially, set $M' = M$ and as long as there are vertices x and y in M' whose least common ancestor w is not in M' , add w to M'. When the process terminates, output M' as the LCA-closure of M. See Figure 2 for an illustration of the LCA-closure. We will require the following simple fact regarding the LCA-closure.

Observation 1. *Let T be an arborescence and $M \subseteq V(T)$. Then, $|\mathsf{LCA}(M)| \leq 2|M|$.*

We now define a set $\mathbf{B}_F^T$ which depends on $\mathbf{A}_F$ and T which is an s.b.a rooted at $v^\star$.

Definition 5. *We first set $\mathbf{B}_F^T = \mathsf{LCA}(\mathbf{A}_F)$. For every pair of vertices $u, v \in \mathbf{A}_F$ such that (a) $u \in \mathsf{Desc}_T(v) \setminus \mathsf{Child}_T(v)$ and (b) there is no other vertex $w \in \mathsf{LCA}(\mathbf{A}_F)$ such that $w \in \mathsf{Desc}_T(v) \cap \mathsf{Ansc}_T(w)$, we pick an arbitrary vertex*

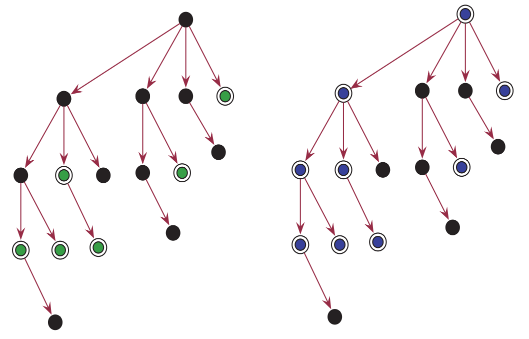

Figure 2: An illustration of the LCA-closure on a binomial arborescence on 16 vertices. The circled vertices in the first figure depict the set M and those in the second figure depict $\mathsf{LCA}(M)$. Observe that in the second figure, the least common ancestor of any pair of circled nodes is also circled.

$z \in \mathsf{Desc}_T(v) \cap \mathsf{Ansc}_T(u)$ and add it to $\mathbf{B}_F^T$. That is, we pick an arbitrary vertex on the u-v path in T and add it to $\mathbf{B}_F^T$. Finally, we set $\mathbf{B}_F^T := \mathsf{LCA}(\mathbf{B}_F^T)$.

When T is clear from the context, we simply refer to the set $\mathbf{B}_F^T$ as $\mathbf{B}_F$.

Definition 6. *Let T be an s.b.a rooted at $v^\star$ and let $X \subseteq V(T)$ such that $v^\star \in X$ and $\mathsf{LCA}(X) = X$. Let T' be a tree constructed as follows. We begin with T and delete all nodes which do not lie on a $v^\star$-u path in T for any $u \in X$. We then repeatedly short-circuit each node of degree 2 which is not in X. That is, we repeat the following step until we cannot do it anymore. Pick a vertex $u \in V(T) \setminus X$ with exactly one in-neighbor u^- and one out-neighbor u^+. Delete the vertex u and add the arc (u^-, u^+). Let T' be the tree that remains when the above step can no longer be performed. We call T' the* **topology** *of X in T.*

Since $\mathsf{LCA}(X) = X$, it follows that $V(T')$ is in fact equal to X. Also, when we talk about a vertex $u \in V(T')$, we can also talk about the same vertex u in T since $V(T') \subseteq V(T)$. Clearly, the topology of the set X in T must be one of at most $|X| \cdot \rho(|X|)$ trees where $\rho(p)$ denotes the number of labeled trees on p vertices. We will make use of the following well known theorem of Cayley.

Proposition 3 (Cayley 1889)**.** $\rho(p) = p^{p-2}$.

The algorithm. Our algorithm has 4 main phases. We will give a brief description of each phase and give formal proof of correctness at the end. We let π denote the ordering witnessing F and γ the ordering of $\mathbf{B}_F$ that respects π. For the rest of the description, we fix a hypothetical s.b.a T rooted at $v^\star$ and let H_T^F denote the topology of $\mathbf{B}_F$ in T.When F and T are clear from the context, we simply call it H.

Phase I: By definition, $V(H) = \mathbf{B}_F$ and due to Observation 1 we know that $|V(H)| \leq 16k + 8$. Hence, in this phase we 'guess' H by iterating over all topologies derived from a set of $k^{\mathcal{O}(k)}$ rooted labeled trees on at most $16k + 8$ vertices. Note that although we know the set $\mathbf{A}_F$, we do not know the set $\mathbf{B}_F$. However, we do not need to know this set at this point and will deal with this fact in the next step. We now state the following observation regarding H which is a direct consequence of the definition of the set $\mathbf{B}_F$ and is crucial for the correctness of our algorithm.

Observation 2. *For every $u \in V(H)$ and $v \in \mathsf{Child}_H(u) \setminus \mathsf{Child}_T(u)$, either $u \notin \mathbf{A}_F$ or $v \notin \mathbf{A}_F$. Consequently, $(u, v) \in A(D) \setminus F$.*

Phase II: In the second phase, we compute the images of the vertices of $V(H)$ under τ. Note that computing this immediately gives us a (injective) mapping from the vertices in $\mathbf{A}_F$ to the vertices of H. Formally speaking, we will guess the following function (Definition 7) by going over all functions from $V(H)$ to Types.

Definition 7. *Let $\psi_H^F : V(H) \to \mathsf{Types}$ be the function that maps vertices of $\mathbf{A}_F$ in $V(H)$ to themselves and maps the vertices of $V(H) \setminus \mathbf{A}_F$, to their image under τ_γ^π. That is, $\psi_H^F(v) = v$ if $v \in \mathbf{A}_F$ and $\psi_H^F(v) = \tau_\gamma^\pi(v)$ otherwise.*

Observe that there are only $k^{\mathcal{O}(k)}$ functions from $V(H)$ to Types and hence we can 'guess' ψ_H^F by simply iterating over all these functions.

Phase III: In this phase, we guess the number of descendants rooted at each vertex in $V(H)$, in T. This number is the size of the largest subtournament of the tournament given by T which is won by this vertex. We remark that even though at first glance, the number of guesses required here seems too big ($n^{\mathcal{O}(k)}$), it can in fact be bounded by the running time stated in Theorem 1. This is a crucial part of our FPT algorithm.

Let $sz_H^F : V(H) \to [n]$ denote the function where $sz_H^F(v) = |\mathsf{Desc}_T(v)|$ for every $v \in V(H)$. That is, this function gives the size of the largest subtournament (in T) won by each vertex of $V(H)$. Since we are by definition only interested in balanced knockout tournaments, it follows that the range of sz_H^F is the set $\{2^i | i \in [\log n] \cup \{0\}\}$. Hence, we can represent the function sz using a function $\chi_H^F : V(H) \to [\log n] \cup \{0\}$. This function is formally defined as follows.

Definition 8. *For every $u \in V(H)$, the number of descendants of u in the s.b.a T is $2^{\chi_H^F(u)}$.*

Hence we have $(\log n)^{\mathcal{O}(k)}$ choices for the function χ_H^F, which is $(k \log k)^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ due to Lemma 1. This completes the description of this phase.

Before we move to the description of the final phase, we define the notion of X-decompositions of an arborescence T for some set $X \subseteq V(T)$. We believe that a good understanding of this notion will the give the reader a better intuitive understanding of the final phase.

Definition 9. *Let T be an arborescence and let $X \subseteq V(T)$ such that it contains the root and let $|X| = \ell$. We say a set $\mathcal{T} = \{T_1, \ldots, T_\ell\}$ of subtrees of T is the X-***decomposition** *of T if these are precisely the subtrees of T that we obtain by deleting all arcs $(u, v) \in A(T)$ where $v \in X$.*

We have the following straightforward consequence of the above definition.

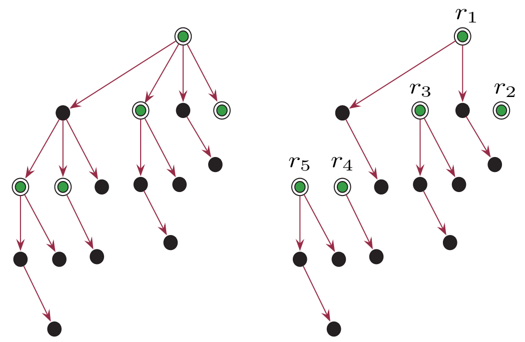

Figure 3: An illustration of the X-decomposition of the arborescence T. The vertices of X are green and represented as concentric circles.

Observation 3. *Let T be an arborescence and let $X \subseteq V(T)$ such that it contains the root and let $|X| = \ell$. Let $\mathcal{T} = \{T_1, \dots, T_\ell\}$ be the X-**decomposition** of T. Then, the vertex sets $V(T_1), \dots, V(T_\ell)$ form a partition of $V(T)$, and for all $i \in [\ell]$ there is a vertex $r_i \in X$ such that T_i is an arborescence rooted at r_i .*

Let $\mathcal{T} = \{T_1, \dots, T_\ell\}$ denote the $\mathbf{B}_F$-decomposition of T, where $\ell = |\mathbf{B}_F|$. Recall that the trees in $\mathcal{T}$ are pairwise vertex disjoint and each tree is rooted at a vertex of $\mathbf{B}_F$. Furthermore, since each tree contains exactly one vertex of $\mathbf{B}_F$ (which is the root of the subtree), it follows that no arc from the set F occurs *inside* any of these subtrees. We now have the following useful lemma describing the structure of each subtree. Recall that r_i is the root of the subtree T_i.

Lemma 4. *For each $i \in [\ell]$, $V(T_i) \subseteq \bigcup_{i \in \mathsf{Types}_{\geq}(r_i)} \tau^{-1}(i)$.*

Proof. In order to prove the lemma, it suffices to show that if $r_i \in \mathbf{A}_F$, then $\tau(u) > \gamma(r_i)$ for every $u \in V(T_i) \setminus \{r_i\}$ and if $r_i \notin \mathbf{A}_F$, then $\tau(u) \geq \tau(r_i)$ for every $u \in V(T_i)$.

For the first statement, suppose that for some $u \in V(T_i) \setminus \{r_i\}$, $\tau(u) \leq \gamma(r_i)$. That is, u beats r_i and so $\pi(u) < \pi(r_i)$. However, since T_i is an arborescence rooted at r_i and $u \in V(T_i)$, it follows that r_i has a path to u in T_i and hence it must be the case that there is an arc (x, y) along this path 'violating' the ordering π (that is, $\pi(x) > \pi(y)$), implying that the arc (x, y) is in the set F, a contradiction to the fact that no T_i contains an arc in F. The argument for the second statement is analogous. This completes the proof of the lemma. □

Intuitively, the structure guaranteed by the previous lemma implies that we no longer need to concern ourselves with the *actual* vertices inside each subtree but simply the *types* of the vertices and the *number* of vertices of each type. This allows us to phrase the remaining problem as a knapsack-type problem and write an ILP-FEASIBILITY instance with few variables.

Phase IV: In the final phase, we will construct an instance of ILP-FEASIBILITY corresponding to the guesses made so far, solve it using Proposition 2 and return YES if and only if the answer to this query is YES. We will show that the given instance of TFP is a YES instance if and only if at least one of the constructed ILP-FEASIBILITY instances is a YES instance.

Definition 10. *We say that a tuple (L, ψ, χ) is* valid *if L is an arborescence rooted at $v^\star$, $V(L) \supseteq \mathbf{A}_F$, $\psi : V(L) \to \mathsf{Types}$ such that $\psi(u) = u$ for every $u \in \mathbf{A}_F$, $\psi(u) \in \mathsf{Types} \setminus \mathbf{A}_F$ for every $u \in V(L) \setminus \mathbf{A}_F$ and $\chi : V(L) \to [\log n] \cup \{0\}$.*

For each valid tuple (L, ψ, χ), we define an instance $\mathbf{I}_{L,\psi,\chi}$ of ILP-FEASIBILITY over a set of $\mathcal{O}(k^2)$ variables as follows. For each $i \in \mathsf{Types}$ and $u \in V(L)$, we have a variable x_i^u. We now describe the constraints. We have four sets of constraints, which we denote by $\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3$ and $\mathbf{C}_4$.

- The constraints in $\mathbf{C}_1$ are defined as follows. Recall that for each $i \in \mathsf{Types}$, $\mathcal{P}_i(\tau)$ is the set of vertices whose image under τ is i. For each $i \in \mathsf{Types}$, let P_i denote the size of the set $\mathcal{P}_i(\tau)$. For each $i \in \mathsf{Types}$, we have a constraint
$$\sum_{u \in V(H)} x_i^u = P_i.$$
- The constraints in $\mathbf{C}_2$ are defined as follows. For every $u \in V(L) \setminus \mathbf{A}_F$, we have the constraint $x^u_{\psi(u)} \geq 1$.
- The constraints in $\mathbf{C}_3$ are defined as follows. For each $u \in V(L)$, we have a constraint
$$\sum_{i \in \mathsf{Types}_{\geq}(u)} x_i^u = 2^{\chi(u)} - \sum_{x \in \mathsf{Child}_H(u)} 2^{\chi(x)}.$$
- The constraints in $\mathbf{C}_4$ are defined as follows. For each $u \in V(L)$ and $i \in \mathsf{Types}_{<}(u)$, we have a constraint $x_i^u = 0$.

This completes the description of the instance $\mathbf{I}_{L,\psi,\chi}$ of ILP-FEASIBILITY. The meaning of the variables and constraints is as follows. *Suppose* that $L = H$, $\psi = \psi_H^F$ and $\chi = \chi_H^F$. For each $u \in \mathbf{B}_F$, let Q_u denote the subtree of T rooted at u and contained in the $\mathbf{B}_F$-decomposition of T. The value of x_i^u will be the number of those vertices in the subtree Q_u whose image under τ is i.

The constraints essentially require us to 'pack' vertices of each type into the subtrees $\{Q_u | u \in \mathbf{B}_F\}$ subject to the type and size restrictions imposed by the functions ψ, χ and λ.

- The set $\mathbf{C}_1$ says that vertices in each $\tau^{-1}(i)$ are partitioned across the subtrees $\{Q_u | u \in \mathbf{B}_F\}$.
- The set $\mathbf{C}_2$ says that for every subtree Q_u rooted at a vertex $u \in \mathbf{B}_F$, there is *at least* one vertex of type $\tau(u)$ which is contained in Q_u, which we know from earlier is the root.
- To understand the set $\mathbf{C}_3$, observe that for every $u \in \mathbf{B}_F$, the set of vertices in Q_u are precisely those vertices which are descendants of u in T but *not* descendants of v for any $v \in \mathsf{Child}_L(u)$. Hence, the *total number* of vertices in Q_u is given by the function χ and this is precisely $2^{\chi(u)} - \sum_{x \in \mathsf{Child}_H(u)} 2^{\chi(x)}$. Furthermore, by Lemma 2, we know that the vertices in Q_u only come from those

types in $\mathsf{Types}_{\geq}(u)$. Hence the constraints in the set $\mathbf{C}_3$ ask to assign vertices from only those types in $\mathsf{Types}_{\geq}(u)$ so that the number adds up to the size of Q_u.

- The set $\mathbf{C}_4$ merely complements the set $\mathbf{C}_3$ by ensuring that we do not assign *any* vertex of type $\mathsf{Types}_{<}(u) = \mathsf{Types} \setminus \mathsf{Types}_{\geq}(u)$ to the tree Q_u.

Lemma 5. *The given instance of* TFP *is a* YES *instance if and only if for some valid tuple (H, ψ, χ), the instance* $\mathbf{I}_{H,\psi,\chi}$ *is a* YES *instance of* ILP-FEASIBILITY.

Proof. We first consider the forward direction. Suppose that the given instance is a YES instance and let T be an s.b.a rooted at $v^\star$. Let H be the topology of $\mathbf{B}_F$ in T. Let $\psi : V(H) \to \mathsf{Types}$ be the function ψ_H^F (see Definition 7) and let χ be the function χ_H^F (Definition 8). For each $u \in \mathbf{B}_F$, let Q_u denote the tree rooted at u in the $\mathbf{B}_F$-decomposition of T. We now argue that the instance $\mathbf{I}_{H,\psi,\chi}$ is a YES instance. For this, we define the assignment $x_i^u = |\{q|\tau(q) = i, q \in V(Q_u)\}|$ for every $i \in \mathsf{Types}$ and $u \in \mathbf{B}_F$. That is, we assign to variable x_i^u the number of vertices in Q_u whose image under τ is i. The fact that this assignment satisfies all four sets of constraints can be seen by a straightforward examination. Indeed the explanation of the constraints given prior to this lemma shows precisely this. This completes the forward direction of the proof and we now argue the more involved converse direction.

Let (H, ψ, χ) be a valid tuple such that $\mathbf{I}_{H,\psi,\chi}$ is a YES instance. Let $\alpha^\star$ be the assignment of the variables which satisfies the given instance and for each i, u, let α_i^u denote the value of x_i^u given by a feasible solution to this instance. We will argue that the given instance of TFP is a YES instance by constructing an s.b.a T rooted at $v^\star$.

For each $u \in V(H)$ and $i \in \mathsf{Types}$, we assign a *unique* subset of $\mathcal{P}_i(\tau)$ to the pair (u, i). To formally capture this, we define a function $\zeta : V(H) \times \mathsf{Types} \to 2^{V(D)} \cup \{\bot\}$. Now, the mapping given by this function has to be defined carefully by performing a bottom up processing of H. Initially, we set $\zeta(u, i) = \bot$ for every $u \in V(H)$ and $i \in \mathsf{Types}$. We also mark all vertices in $V(D)$ as *free*. We then exhaustively execute the following procedure and along the way we change the marking of vertices of $V(D)$ from *free* to *taken* whenever we assign a set containing them to some u, i under ζ.

If there is a vertex $u \in V(H)$ and an element $v \in \mathsf{Child}_H(u)$ such that $\zeta(v, j) \neq \bot$ for every $j \in \mathsf{Types}$ (if u is a leaf of H then it vacuously satisfies this condition), then for every $i \in \mathsf{Types}$, we set $\zeta(u, i)$ to be the *weakest* α_u^i vertices in $\mathcal{P}_i(\tau)$ among those which are *free* and mark these vertices as *taken*. If $\alpha_u^i = 0$, then we set $\zeta(u, i) = \emptyset$.

The fact that ζ exists follows from the fact that the set of constraints in $\mathbf{C}_1$ are satisfied by the values $\{\alpha_i^u | u \in V(H), i \in \mathsf{Types}\}$. For each $u \in V(H)$, we define R_u to be the set $\bigcup_{i \in \mathsf{Types}} \zeta(u, i)$ and we define Y_u to be the set $\bigcup_{v \in \mathsf{Desc}_H(u)} R_v$.

It follows from the definition of ζ that the vertex sets $\{R_u | u \in V(H)\}$ form a partition of $V(D)$. Furthermore, since the constraints in C_2, $\mathbf{C}_3$ and $\mathbf{C}_4$ are satisfied by $\alpha^\star$, it follows that for every $u \in V(H)$, R_u contains at most one vertex of $V(F)$, R_u contains at least one vertex w of type $\psi(u)$ ($\tau(w) = \psi(u)$) and $R_u \subseteq \bigcup_{i \in \mathsf{Types}_{\geq}(w)} \zeta(u, i)$, implying that the strongest vertex in R_u has type $\psi(u)$. Furthermore, if u is a leaf of H, then $R_u = \sum_{i \in \mathsf{Types}_{\geq}(u)} x_i^u = \sum_{i \in \mathsf{Types}} x_i^u = 2^{\chi(u)}$. We will define the s.b.a T by performing a bottom up parse of H. During this parse, for every $u \in V(H)$, we will define a binomial arborescence T_u which spans Y_u and is rooted at r_u which is the strongest vertex of type $\psi(u)$ which is contained in Y_u. To keep track of the vertices of H which we have processed and those we have not, we will use a boolean function $\mathsf{Acc} : V(H) \to \{T, F\}$. Initially, $\mathsf{Acc}(u) = F$ for every $u \in V(H)$.

Constructing T. For each $u \in V(H)$ which is a leaf, we invoke the first statement of Lemma 3 to conclude that there is a binomial arborescence T_u spanning R_u and rooted at r_u, which is the strongest element of type $\psi(u)$ in R_u. We then set $\mathsf{Acc}(u) = T$. We now exhaustively repeat the following procedure for the rest of the vertices.

Pick a vertex u such that $\mathsf{Acc}(u) = F$ and $\mathsf{Acc}(v) = T$ for every $v \in \mathsf{Child}_H(u)$. Let $c_1, \ldots, c_\ell$ denote the children of u in H and let $T_{c_1}, \ldots, T_{c_\ell}$ denote the already defined arborescences where T_{c_j} spans Y_{c_j} and is rooted at r_{c_j} which is the strongest vertex of type $\psi(c_j)$ in Y_{c_j}.

Define the set $Z_0 = R_u$ and for each $j \in [\ell]$, the set Z_j as $V(T_{c_j})$. Define the trees $T_1, \ldots, T_\ell$ as $T_j = T_{c_j}$ and the root r_j of T_j as $r_j = r_{c_j}$. Since the sets $\{Z_j\}_{0 \leq j \leq \ell}$, the trees $\{T_j\}_{j \in [\ell]}$ and vertices $\{r_j\}_{j \in [\ell]}$ satisfy the premises of the second statement of Lemma 2, we conclude that there is a binomial arborescence T_u rooted at the strongest vertex of $Z_0 = R_u$ and spanning the set Y_u. Since we have already argued that the strongest vertex of R_u is of type $\psi(u)$, we conclude that T_u is rooted at the strongest vertex of type $\psi(u)$ in Y_u. We now set $\mathsf{Acc}(u) = T$ and continue. This procedure terminates when $u = v^\star$ and we will have demonstrated the existence of an arborescence rooted at $v^\star$ and spanning $V(D)$. This completes the proof of the lemma. $\square$

Theorem 1 follows from Lemma 5, the fact that there are $(k \log k)^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ valid tuples, and Proposition 2 combined with the fact that the number of variables in each constructed instance of ILP-FEASIBILITY is $\mathcal{O}(k^2)$.

Concluding remarks. Our fixed-parameter tractability result for the TOURNAMENT FIXING PROBLEM is but the first step in the formal study of the parameterized complexity of this problem. The challenge going forward is to chart the parameterized complexity landscape of the TFP with respect to stronger parameters for tournaments, e.g., feedback vertex set, cutwidth and treewidth. A second direction would be the study of the *probabilistic* version of the problem where the input is a probability matrix giving the probability of any player beating any other and the objective is to compute a seeding such that the probability of a special player wining the resulting knockout tournament exceeds a given threshold.

Acknowledgments. The authors wish to thank the anonymous reviewers for their helpful comments and acknowledge support by the Austrian Science Fund (FWF, project P26696).

References

Aziz, H.; Gaspers, S.; Mackenzie, S.; Mattei, N.; Stursberg, P.; and Walsh, T. 2014. Fixing a balanced knockout tournament. In Brodley, C. E., and Stone, P., eds., *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, 552–558. AAAI Press.

Betzler, N.; Bredereck, R.; Chen, J.; and Niedermeier, R. 2012. Studies in computational aspects of voting - A parameterized complexity perspective. In Bodlaender, H. L.; Downey, R.; Fomin, F. V.; and Marx, D., eds., *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *Lecture Notes in Computer Science*, 318–363. Springer Verlag.

Bevern, R. v.; Komusiewicz, C.; Molter, H.; Niedermeier, R.; Sorge, M.; and Walsh, T. 2016. h-index manipulation by undoing merges. In Kaminka, G. A.; Fox, M.; Bouquet, P.; Hüllermeier, E.; Dignum, V.; Dignum, F.; and van Harmelen, F., eds., *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, 895–903. IOS Press.

Cayley, A. 1889. A theorem on trees. *Quart. J. Math.* 23:376–378.

Chen, J.; Faliszewski, P.; Niedermeier, R.; and Talmon, N. 2015. Elections with few voters: Candidate control can be easy. In Bonet, B., and Koenig, S., eds., *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, 2045–2051. AAAI Press.

Cygan, M.; Fomin, F. V.; Kowalik, L.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer.

Dey, P.; Misra, N.; and Narahari, Y. 2015. Kernelization complexity of possible winner and coalitional manipulation problems in voting. In Weiss, G.; Yolum, P.; Bordini, R. H.; and Elkind, E., eds., *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, 87–96. ACM.

Downey, R. G., and Fellows, M. R. 2013. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer.

Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Berlin: Springer Verlag.

Fradkin, A. O., and Seymour, P. D. 2013. Tournament pathwidth and topological containment. *J. Comb. Theory, Ser. B* 103(3):374–384.

Frank, A., and Tardos, É. 1987. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica* 7(1):49–65.

Hemaspaandra, L. A.; Lavaee, R.; and Menton, C. 2016. Schulze and ranked-pairs voting are fixed-parameter tractable to bribe, manipulate, and control. *Ann. Math. Artif. Intell.* 77(3-4):191–223.

Kannan, R. 1987. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.* 12(3):415–440.

Kim, M. P., and Vassilevska Williams, V. 2015. Fixing tournaments for kings, chokers, and more. In Yang, Q., and Wooldridge, M., eds., *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 561–567. AAAI Press.

Kim, M. P.; Suksompong, W.; and Vassilevska Williams, V. 2016. Who can win a single-elimination tournament? In Schuurmans, D., and Wellman, M. P., eds., *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, 516–522. AAAI Press.

Lenstra, H. W., and Jr. 1983. Integer programming with a fixed number of variables. *Math. Oper. Res.* 8(4):538–548.

Lindner, C., and Rothe, J. 2008. Fixed-parameter tractability and parameterized complexity, applied to problems from computational social choice. In *Mathematical Programming Glossary. Informs Computing Society*, 1–15.

Stanton, I., and Vassilevska Williams, V. 2011. Rigging tournament brackets for weaker players. In Walsh, T., ed., *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, 357–364. IJCAI/AAAI.

Vassilevska Williams, V. 2010. Fixing a tournament. In Fox, M., and Poole, D., eds., *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010.* AAAI Press.

Vu, T.; Altman, A.; and Shoham, Y. 2009. On the complexity of schedule control problems for knockout tournaments. In Sierra, C.; Castelfranchi, C.; Decker, K. S.; and Sichman, J. S., eds., *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary, May 10-15, 2009, Volume 1*, 225–232. IFAAMAS.

Yang, Y. 2014. Election attacks with few candidates. In Schaub, T.; Friedrich, G.; and O'Sullivan, B., eds., *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, 1131–1132. IOS Press.