Winning a Tournament by Any Means Necessary

Sushmita Gupta[1], **Sanjukta Roy**[2], **Saket Saurabh**[1,2] and **Meirav Zehavi**[3]
[1] University of Bergen, Bergen, Norway
[2] The Institute of Mathematical Sciences, HBNI, Chennai, India
[3] Ben-Gurion University, Beersheba, Israel
sushmita.gupta@gmail.com, sanjukta@imsc.res.in, saket@imsc.res.in, meiravze@bgu.ac.il

Abstract

In a tournament, n players enter the competition. In each round, they are paired-up to compete against each other. Losers are thrown, while winners proceed to the next round, until only one player (the winner) is left. Given a prediction of the outcome, for every pair of players, of a match between them (modeled by a digraph D), the competitive nature of a tournament makes it attractive for manipulators. In the Tournament Fixing (TF) problem, the goal is to decide if we can conduct the competition (by controlling how players are paired-up) so that our favorite player w wins. A common form of manipulation is to bribe players to alter the outcome of matches. Kim and Williams [IJCAI 2015] integrated such deceit into TF, and showed that the resulting problem is NP-hard when $\ell < (1-\epsilon)\log n$ alterations are possible (for any fixed $\epsilon > 0$). For this problem, our contribution is fourfold. First, we present two operations that "obfuscate deceit": given one solution, they produce another solution. Second, we present a combinatorial result, stating that there is always a solution with all reversals incident to w and "elite players". Third, we give a closed formula for the case where D is a DAG. Finally, we present exact exponential-time and parameterized algorithms for the general case.

1 Introduction

In a (cup) tournament, we have a set N of n players who enter the competition, but only one can be the winner. In each round, the players are paired-up to compete against each other. Losers are thrown away, while winners proceed to the next round, until only one lucky player is left. This is a standard form of competition in sports, popular culture, elections and decision making schemes. Significant attention has been given to tournaments from the viewpoints of both artificial intelligence and economics and operation research [Rosen, 1986; Tullock, 1980; Laslier, 1997]. In real life, we can often predict (with high success rate) the outcome, for every pair of players, of a match between them. Given such tentative information, the inherent highly competitive nature of a tournament makes it attractive for manipulators. Here, we focus on the combination (introduced by Kim and Williams [Kim and Williams, 2015]) of two common forms of manipulation. We present "obfuscation operations", combinatorial results, polynomial-time algorithms for special cases, and exact exponential-time algorithms for the general case.

To explain how a tournament is conducted and manipulated formally, we need the following definitions. By *fixing* a tournament we refer to a bijective labeling (by N) of the leaves of a full binary tree T with n leaves (where n is a power of 2). The labeling of the internal vertices of T is determined as follows. While we have an unlabelled vertex v with two labeled children x and y, we label v by the winner of a match between the players corresponding to the labels of x and y. The winner of the tournament is the player corresponding to the label of the root of T. Clearly, this process is a precise model for the execution of a tournament. The predictive information we have at hand is modeled by a digraph D with $V(D) = N$ and where for every pair of players $u, v \in N$, either $(u, v) \in A(D)$ (which means that u beats v) or $(v, u) \in A(D)$ (which means that v beats u). Using terms from graph theory, D is called a *tournament* (not to be confused with the competition, also called a tournament).

It is natural to ask whether, given N, D and $w \in N$, there is a way to fix the tournament so that w wins. This problem, called TOURNAMENT FIXING (TF), was introduced by Vu et al. [Vu *et al.*, 2009]. From a computational point of view, Aziz et al. [Aziz *et al.*, 2014] showed that TF is NP-hard, solvable in time $\mathcal{O}(2.83^n)$ and space $\mathcal{O}(1.76^n)$ (or time $4^{n+o(n)}$ and polynomial space), and that parameterized by the feedback arc set number k of D, it is in XP (specifically, solvable in time $\mathcal{O}(n^{2k}\log n)$). Later, Kim and Williams [Kim and Williams, 2015] gave an $\mathcal{O}(2^n)$-time and space algorithm for TF. These algorithms can also count the number of fixings that make w win. The parameter k has a natural interpretation as follows. In real world competitions, there is a ranking of players' strengths (e.g., think of Tennis rankings), so that a player of a certain rank is expected to beat players of a lower rank. Indeed, this assumption is the basis of probabilistic models that study tournaments (e.g., see [Stanton and Williams, 2011b]). The number of matches where we guess (before the competition begins) that a player of a certain rank will beat a player of higher rank is generally very small. This number is an upper bound on k, implying that k is generally significantly smaller than n. In light of this, it is reasonable to seek a fixed-parameter tractable (FPT) algorithm, i.e. an algorithm whose time complexity is $f(k) \cdot n^c$ for some function f only of k and some constant $c \geq 0$ independent of k. In particular, when $f(k) = n^{\mathcal{O}(1)}$, the algorithm runs in polynomial time. Using integer linear programming in fixed dimension, Ramanujan and Szeider [Ramanujan and Szeider, 2017] developed an FPT algorithm for TF that runs in time $2^{\mathcal{O}(k^2 \log k)} n^{\mathcal{O}(1)}$. From the viewpoint of parameterized complexity, also Aronshtam et al. [Aronshtam *et al.*, 2017] studied (several variants of) TF with appropriate parameterizations. For example, they studied the question of whether we can fix the tournament so that w wins at least k matches (but possibly not all matches). With respect to the parameter d, the maximum number of players that any player can beat, they obtained an FPT algorithm that runs in time $d^{2^d} n^{\mathcal{O}(1)}$.

In addition, TF received significant attention from a structural point of view—an array of works exhibited structural properties of D that guarantee that w wins [Kim *et al.*, 2016; Kim and Williams, 2015; Stanton and Williams, 2011b; 2011a; Williams, 2010]. Let us review some of the results that are closer to our study (with emphasis on non-probabilistic structural statements), and refer the reader to the survey [Williams, 2016] for more information. Williams [Williams, 2010] studied the case where w is a *king* (resp. *super king*), that is, every player is either beaten by w or beaten by a player that is beaten by w (resp. $\log n$ players that are beaten by w). In particular, she showed that for a king w that beats at least half the players, there always exists a fixing that makes w the winner, while for a king w that beats less than half the players, there *might* not exist a fixing that makes w the winner. In addition, she showed that if w is a *super king*, then there exists a fixing that makes w win.

Later, Stanton and Williams [Stanton and Williams, 2011a] strengthened the work by Williams [Williams, 2010] with respect to her results for a probabilistic model, and by extending her condition that ensures that a king can always win. In a different work, Stanton and Williams [Stanton and Williams, 2011b] exhibited sufficient and necessary conditions that ensure that for *any* player w among the K players that have the largest out-degrees in D (for certain choices of K), there exists a fixing that makes w the winner. More recently, Kim and Williams [Kim and Williams, 2015] proved that TF is NP-hard even if w is a king that beats $n/4$ players, or if w is a so called *3-king* (that is not as strong as a king) that beats half the players. They also provided a sufficient condition to ensure that a 3-king can win the tournament. Later, Kim et al. [Kim *et al.*, 2016] further extended and generalized previous structural results, including the study of the power of 3-kings. We remark that a certain argument in one of our proofs may be viewed in the terms of a king (see Section 5), but apart from this curiosity, the notions we introduce are incomparable to those of kings and of the K best players by [Stanton and Williams, 2011b] (see Section 4).

In addition, several studies present empirical results. For example, Mattei and Walsh [Mattei and Walsh, 2016] showed that although TF is NP-hard, real world soccer and tennis instances (English Premier League, the German Bundesliga, and the ATP World Tour) are easily solvable. Indeed, real world instances are generally structured, hinting at the relevance of appropriate parameterizations to capture these structures. Another empirical study on TF was conducted by Russell and Beek [Russell and van Beek, 2011], which focused on restrictions on the space of possible seedings. Here, the motivation is that real cup competitions often place restrictions to ensure fairness and wide geographic interest, which make the problem harder to solve in practice.

Kim and Williams [Kim and Williams, 2015] introduced a more general question, called BRIBERY TF (BTF). Given N, D, $w \in N$ and $\ell \in \mathbb{N} \cup \{0\}$, BTF asks if it is possible to reverse at most ℓ arcs in D (that is, change the outcome of at most ℓ matches) so that afterwards there is a way to fix the tournament that makes w wins. TF is the special case of BTF where $\ell = 0$, which implies that BTF is NP-hard. Kim and Williams [Kim and Williams, 2015] proved that for any fixed $\epsilon > 0$, BTF is NP-hard even if $\ell < (1-\epsilon)\log_2 n$. On the other hand, they observed that if $\ell \geq \log_2 n$, then the input is a YES-instance.

Prior to [Kim and Williams, 2015], bribery has already been studied with respect to tournaments whose seeding is fixed in advance. The computational complexity of this problem is significantly simpler than that of TF. In particular, Russell and Walsh [Russell and Walsh, 2009] showed that this problem is solvable in polynomial time. In fact, this result was proved in a more general setting where we have a coalition (a set of players) such that all manipulations must involve a lose of one of its members. Additionally, Mattei et al. [Mattei *et al.*, 2015] showed that even in a probabilistic model, where we want to make w have a nonzero probability of winning, the problem is still solvable in polynomial time.

Our contribution is fourfold. First, we present two "obfuscation operations", whose input is a solution, i.e. a set of at most ℓ arcs to reverse and a fixing of the tournament (with these arcs reversed). If the operation is applicable, it returns (in polynomial time) a different solution. These operations are relevant when: *(i)* the deceit in the current solution is too conspicuous; *(ii)* the current solution could not be realized (e.g., some player refused to be bribed). Moreover, we use these two operations as building blocks in our proofs.

Secondly, we present a combinatorial result, which states that there always exists a solution with all reversals incident to w and "elite players". Roughly speaking, given a feedback arc set K of D with $k = |V(K)|$, we partition $V(D) \setminus V(K)$ into $k+1$ modules. Each of these modules induces a DAG, and an *elite club* roughly refers to a set of players of "highest rank" in the topological order of each of these DAGs (the formal definition is more general and robust, see Definition 2). Our definition of an elite club in incomparable to the notions discussed earlier (see the explanation in Section 4).

Third, we give a closed formula (based on our combinatorial result) that resolves the special case of BTF where D is a DAG. More precisely, the formula is satisfied if and only if the input instance of BTF is a YES-instance. The formula can be verified in polynomial time. Moreover, its proof is constructive (the satisfaction of the formula indicates that a certain candidate to be a solution is indeed a solution). In addition, we generalize our formula to characterize all YES-

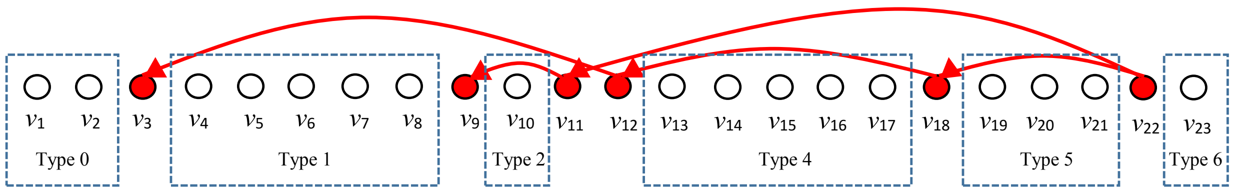
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10} v_{11} v_{12} v_{13} v_{14} v_{15} v_{16} v_{17} v_{18} v_{19} v_{20} v_{21} v_{22} v_{23}
Type 0
Type 1
Type 2
Type 4
Type 5
Type 6

Figure 1: Partition of $V(D) \setminus V(K)$ into types. The arcs in K and the vertices in $V(K)$ are colored red. For all v_i, v_j with $i < j$ such that $(v_j, v_i) \notin K$, we suppose that $(v_i, v_j) \in A(D)$ (not displayed).

instances also for the general case. Of course, the general characterization cannot be verified in polynomial time.

Finally, we present exact algorithms for the general case of BTF. (Recall that in its full generality, BTF is NP-hard.) Specifically, we develop an algorithm that runs in time $2^n n^{\mathcal{O}(1)}$ and has a polynomial space complexity (which improves upon the algorithm by Kim and Williams [Kim and Williams, 2015] in the special case of TF), as well as a parameterized algorithm that runs in time $2^{\mathcal{O}(k^2 \log k)} n^{\mathcal{O}(1)}$, where k is the feedback arc set number of D. The latter algorithm implies that BTF is FPT parameterized by k alone.

2 Preliminaries

Denote $[n] = \{1, \ldots, n\}$. Given a digraph D, denote its arc-set and vertex-set by $A(D)$ and $V(D)$, respectively. Given $U \subseteq V(D)$, by $D[U]$ denote the subdigraph of D induced by U. Given $R \subseteq A(D)$, denote $R^{\text{rev}} = \{(u, v) : (v, u) \in R\}$. Moreover, D^R denotes the digraph obtained from D by reversing all arcs in R, that is, $V(D^R) = V(D)$ and $A(D^R) = (A(D) \setminus R) \cup R^{\text{rev}}$. We say that R is a *feedback arc set* of D if D^R is a directed acyclic graph (DAG), i.e. D^R has no directed cycle. By $V(R)$ we denote the set of vertices incident to the arcs in R. The *feedback arc set number* of D is the minimum size of a feedback arc set of D.

A *topological order* of a DAG D is a total order $<$ on $V(D)$ such that for all $u, v \in V(D)$, if $u < v$ then $(v, u) \notin A(D)$. That is, if we denote $V(D) = \{v_1, v_2, \ldots, v_n\}$ such that $v_i < v_j$ whenever $i < j$, then each vertex v_i can only have outgoing arcs into vertices of lower rank (i.e. larger index). We say that v_i is the *predecessor* of v_{i+1}, and v_{i+1} is the *successor* of v_i, for all $i \in \{1, 2, \ldots, n\}$. Note that if D is an acyclic tournament, then $<$ is uniquely defined.

Given a tournament D and a feedback arc set K of D, the *type* of a vertex $v \in V(D) \setminus V(K)$ is $i \in \{0, 1, \ldots, k\}$ if there are exactly i vertices $u \in V(K)$ such that $u < v$ with respect to the topological order $<$ of D^K (see Fig. 1).

Given a rooted tree T and $v \in V(T)$, we let T_v denote the subtree of T rooted at v. Towards a reformulation of BTF, we need the concept of a binomial arborescence (see Fig. 2).

Definition 1. *A* binomial arborescence *is defined recursively:*

- *A single node v is a binomial arborescence rooted at v.*
- *Given two vertex disjoint binomial arborescences of equal size, T_v rooted at v and T_u rooted at u, adding an arc from v to u results in a binomial arborescence rooted at v.*

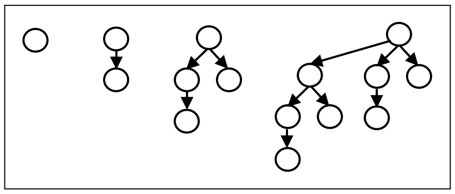

Figure 2: Binomial arborescnences of sizes $2^0, 2^1, 2^2$ and 2^3 (unique up to isomorphism).

Given a tournament D, we say that a binomial arborescence T is *spanning* if $V(T) = V(D)$. The proposition below gives rise to a graph theoretic interpretation of BTF.

Proposition 1 ([Williams, 2010]). *For a tournament D with $w \in V(D)$, there is a fixing of D that makes w win if and only if D has a spanning binomial arborescence T rooted at w.*

In particular, if we have a fixing of D where w wins, then by taking all arcs in D corresponding to the matches that took place in the competition, we obtain a spanning binomial arborescence T rooted at w. On the other hand, having a spanning binomial arborescence T rooted at w, the tournament can be fixed so that w wins and all matches that take place correspond to the arcs in T (see [Williams, 2010]). Having Proposition 1 at hand, BTF is equivalent to the following question: Is there $R \subseteq A(D)$ of size at most ℓ such that D^R has a spanning binomial arborescence rooted at w? Unless written otherwise, we take this point of view.

Finally, we state (without proof) immediate properties of binomial arborescences.

Observation 1. *Let T be a binomial arborescence on $n \geq 2$ vertices. Then, for every vertex $v \in V(T)$, it holds that T_v is a spanning binomial arborescence of $D[V(T_v)]$ and in particular $|V(T_v)|$ is a power of 2. Moreover, the root of T has exactly $\log n$ children, $v_1, v_2, \ldots, v_{\log n}$, with $|V(T_{v_i})| = 2^{i-1}$ for all $i \in \{1, 2, \ldots, \log n\}$.*

3 Obfuscation Operations

We give two operations to turn one solution into another. In each operation, we have a tournament D, a subset $R \subseteq A(D)$ of size at most ℓ, and a spanning binomial arborscence T of D^R rooted at w. Without loss of generality, suppose $R^{\text{rev}} = A(T) \setminus A(D)$. We say that an operation is *valid* if it returns a subset $R' \subseteq A(D)$ of size at most ℓ, and a spanning binomial arborscence T' of $D^{R'}$ rooted at w.

Subtree clean-up. Our first operation is called "subtree clean-up" (see Fig. 3). We say that this operation is *appli-*

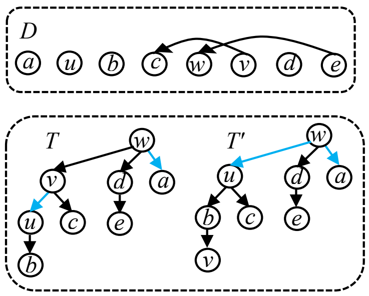

Figure 3: An application of the subtree clean-up operation. In D, displayed arcs are backward arcs; all other arcs are forward arcs. The sets R^{rev} and R'^{rev} are the sets of blue arcs in T and T', respectively. In T', the operation is not applicable.

cable if there exists a vertex $v \in V(T) \setminus \{w\}$ such that $A(T_v) \cap R^{\text{rev}} \neq \emptyset$. Given such a vertex $v \in V(T) \setminus \{w\}$, the operation is performed as follows.

First, since $|V(T_v)|$ is a power of 2 (see Observation 1), we have that $D[V(T_v)]$ admits a spanning binomial arborescence T'_u, rooted at some vertex u. To see this, arbitrarily fix a tournament on $D[V(T_v)]$, conduct the competition, and let T'_u be the spanning binomial arborescence obtained by taking all arcs of $D[V(T_v)]$ that correspond to the matches that took place. Note that the root u of T'_u might be v itself. However, $T'_u \neq T_v$ because T'_u is a subtree of D while $A(T_v) \cap R^{\text{rev}} \neq \emptyset$ (which means that T_v is not a subtree of D). Let p be the parent of v in T (since $v \neq w$, this parent exists). Now, if $(p, u) \in A(D)$ then define $R' = R \setminus A(T_v)^{\text{rev}}$, and otherwise (if $(u, p) \in A(D)$) define $R' = (R \setminus A(T_v)^{\text{rev}}) \cup \{(u, p)\}$. We define T' from T as follows: remove the tree T_v and the arc (p, v), and then add the tree T'_u and the arc (p, u).

Because T_v is a spanning binomial arborescence of $D^R[V(T_v)]$ (see Observation 1), we have that T'_u is isomorphic to T_v. Having this observation in mind, one can easily verify that T' is a spanning binomial arborescence of $D^{R'}$ rooted at w. Moreover, $|R'| \leq |(R \setminus A(T_v)^{\text{rev}}) \cup \{(u, p)\}| \leq |R| - |R \cap A(T_v)^{\text{rev}}| + 1 \leq |R|$. Thus, the validity of our operation follows.

Lemma 1. *The subtree clean-up operation is valid.*

Cyclic shift. Our second operation is called "cyclic shift" (see Fig. 4), where we assume that we are given a feedback arc set K of D with $|V(K)| = k$, and the topological order $<$ of D^K. We say that a directed subpath of T, denoted by $P = p_1 \to p_2 \to \cdots \to p_q$, *enables a cyclic shift* if $p_1, p_2, \ldots, p_q \notin V(K) \cup \{w\}$

1. are vertices of the same type $t \in \{0, 1, \ldots, k\}$,
2. which are consecutive (that is, p_i is the predecessor of p_{i+1} for all $i \in \{1, 2, \ldots, q-1\}$), and
3. the predecessor of p_1 exists, it is not the parent of p_1 in T, and it is also of type t.

We say that the cyclic shift operation is *applicable* if we have a directed subpath of T that enables a shift. Given such a subpath $P = p_1 \to p_2 \to \cdots \to p_q$, the operation is performed as follows. Let p_0 denote the predecessor of p_1. Then,

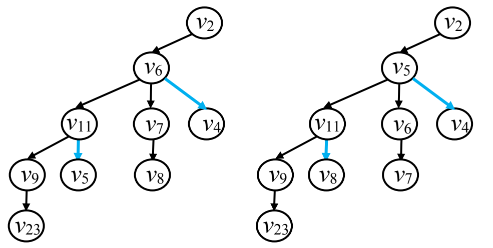

Figure 4: An application of the cyclic shift clean-up operation with D being the graph in Fig. 1 (add vertices to ensure its size is a power of 2, whose display is irrelevant here). Only parts of the trees T and T' are displayed. Reversed arcs are colored blue. Here, $P = v_6 \to v_7 \to v_8$, $p_0 = v_5$, $p_1 = v_6$, $p_2 = v_7$ and $p_3 = v_8$.

as the name of the operation suggests, we define T' from T as follows: for all $i \in \{0, 1, \ldots, q\}$, replace the vertex p_i by the vertex $p_{(i-1) \bmod (q+1)}$. Denote $R' = (A(T') \setminus A(D))^{\text{rev}}$. We now argue why this operation is valid.

Lemma 2 (*). [1] *The cyclic shift operation is valid.*

4 Combinatorial Result

In this section, we show that if the input instance of BTF is a YES-instance, then it admits a solution with two very specific properties. Apart from being a combinatorial result of independent interest, it is also useful practically as it significantly shrinks our search space. Only the second property assumes the presence of a feedback arc set K. If one is interested only in the first property, s/he can set $K = A(D)$, and otherwise K can be obtained by either a polynomial-time approximation scheme (PTAS) [Kenyon-Mathieu and Schudy, 2007], or an $2^{\mathcal{O}(\sqrt{k})} n^{\mathcal{O}(1)}$-time parameterized algorithm [Karpinski and Schudy, 2010; Feige, 2009].

To formulate the second property, we introduce the following definition (examples are given below).

Definition 2. *Let D be a tournament with a feedback arc set K. Let $<$ be the topological order of D^K. A set $C \subseteq V(D)$ is an* elite club *if for all types $i \in \{0, \ldots, |V(K)|\}$, there do not exist two vertices u, v of type i such that $u < v$, $u \notin C$ and $v \in C$. That is, if a vertex v of type i belongs to C, then all vertices u of type i that beat v also belong to C.*

*For $w \in V(D)$, an elite club C is a w-*elite club *if $(v, w) \in A(D)$ for all $v \in C$. That is, all members of C beat w.*

For example, in Fig. 1, $C_1 = \{v_1, v_3, v_4, v_5, v_{13}, v_{14}, v_{15}, v_{18}\}$ is an elite club, while $C_2 = \{v_1, v_3, v_4, v_6, v_{13}, v_{14}, v_{15}, v_{18}\}$ is not since it contains v_6 but not its predecessor v_5 that belongs to the same type. It is easy to see that our notion of an elite club is incomparable to both the notion of kings and the best players defined by [Stanton and Williams, 2011b]. For example, a vertex v *last* of its type is a king if it beats, say, a vertex that beats any other vertex, and it is "likely to be" a best player if it belongs to one of the first types, but it belongs to an elite club C only if *all* vertices of the same type as v belong to C. Similarly, a vertex in an elite club (even if it is first of its type) can be neither a king nor a best player.

[1]Due to lack of space, proofs of results marked by * are omitted.

Our main combinatorial result says that if there is a way to make w win, then there is also such a way where all matches necessary to alter involve w itself. Moreover, it further says that all of the other players that are involved in these matches belong to a group of "top players" (among those of each type). We think these two findings are of philosophical interest as well. Formally, this is stated as follows.

Theorem 1 (*). *Let (D, w, ℓ) be a* YES*-instance of* BTF. *Let K be a feedback arc set of D with $|V(K)| = k$, and let $<$ be the topological order of D^K. Then, there exists $R \subseteq A(D)$ with $|R| \leq \ell$ and a spanning binomial arborscence T of D^R rooted at w, which satisfy both properties below.*

1. *Every arc in R has w as an endpoint.*
2. *There exists a w-elite club C of size at most ℓ such that every arc in R has an endpoint in C.*

5 Characterization of YES-Instances

In this section, we present two (related) characterizations of YES-instances, along with one implication. First, based on Theorem 1 and Observation 1, we present a closed formula that resolves the special case of BTF where D is a DAG. To this end, we need the following definition.

Definition 3. *Let D be a tournament. The* victory count *of a vertex $v \in V(D)$, denoted by $\mathsf{victory}(v)$, is defined by*

$$\mathsf{victory}(v) = |\{u \in V(D) : (v, u) \in A(D)\}|.$$

That is, $\mathsf{victory}(v)$ is the number of players beaten by v.

If D is acyclic, then $\mathsf{victory}(v) = |\{u \in V(D) : v < u\}|$.

We are now ready to state our formula. While the statement is existential, the proof is constructive and implies how a solution, if one exists, can be found.

Theorem 2. *Let (D, w, ℓ) be an instance of* BTF *where D is a DAG. Then, (D, w, ℓ) is a* YES*-instance if and only if the expression $[\mathsf{victory}(w) \geq \frac{n}{2^\ell} - 1]$ is true. In case the expression is true, a solution can be found in polynomial time.*

Proof sketch. In one direction, suppose that $\mathsf{victory}(w) \geq \frac{n}{2^\ell} - 1$ is true. Let S be a set of exactly $\frac{n}{2^\ell} - 1$ vertices beaten by w. Arbitrarily partition S into $\log n - \ell$ sets, $U_1, \ldots, U_{\log n - \ell}$, such that $|U_i| = 2^{i-1}$ for all $i \in \{1, \ldots, \log n - \ell\}$. Since $\sum_{i=1}^{\log n - \ell} 2^{i-1} = \frac{n}{2^\ell} - 1$, this is possible. Moreover, arbitrarily partition $V(D) \setminus (U \cup \{w\})$ into ℓ sets, $U_{\log n - \ell + 1}, \ldots, U_{\log n}$, such that $|U_i| = 2^{i-1}$ for all $i \in \{\log n - \ell + 1, \ldots, \log n\}$. Now, for all $i \in \{1, \ldots, \log n\}$, fix a tournament (competition) for $D[U_i]$, and let T_i denote the corresponding spanning binomial arborescence of $D[U_i]$ rooted at some vertex w_i. Define T by $V(T) = V(T_1) \cup \ldots \cup V(T_{\log n}) \cup \{w\}$ and $A(T) = A(T_1) \cup \ldots \cup A(T_{\log n}) \cup \{(w, w_i) : i \in \{1, \ldots, \log n\}\}$. Moreover, define $R = \{(w_i, w) \in A(D) : i \in \{\log n - \ell + 1, \ldots, \log n\}\}$. Then, $|R| \leq \ell$, and it is easy to verify that T is a spanning binomial arborescence of D^R rooted at w. The proof also indicates how to construct a solution in polynomial time.

In the other direction, suppose that (D, w, ℓ) is a YES-instance. Let (R, T) be a solution that satisfies property 1 in Theorem 1. Let X be the set of children v of w in T such that $(w, v) \in A(D)$, and denote $U = \bigcup_{v \in X} V(T_v)$. Since $|R| \leq \ell$, we have that $|X| \geq \log n - \ell$. Thus, by Observation 1, $|U| = \sum_{v \in X} |V(T_v)| \geq \sum_{i=1}^{\log n - \ell} 2^{i-1} = \frac{n}{2^\ell} - 1$. Moreover, since (R, T) satisfies property 1, we have that $A(T_v) \cap R^{\text{rev}} = \emptyset$ for all $v \in X$. Thus, since $(w, v) \in A(D)$ for all $v \in X$ and D is a DAG, we have that w beats all the vertices in U. ☐

In our proof of the forward direction above, it can be shown that w is made a king by reversing ℓ arcs. The spirit of the proof of Theorem 2 is not limited to the special case of DAGs. Indeed, we have a characterization of YES-instances also for general tournaments. However, this characterization is not nice in the sense that it cannot be verified in polynomial time. Nevertheless, we find it of theoretical interest, and it will be used later. In this context, recall that BTF is NP-hard even when $\ell = 0$, and hence we do not expect it to admit an easy to verify characterization.

Theorem 3. *Let (D, w, ℓ) be an instance of* BTF. *Then, (D, w, ℓ) is a* YES*-instance if and only if there exists $U \subseteq V(D)$ of size exactly $\frac{n}{2^\ell} - 1$ so that there is a fixing of $D[U \cup \{w\}]$ that makes w win. Given such U with its fixing, a solution can be found in polynomial time.*

Proof sketch. In one direction, suppose that there exists $U \subseteq V(D)$ of size exactly $\frac{n}{2^\ell} - 1$ so that there is a fixing of $D[U \cup \{w\}]$ that makes w win, and let T' be the corresponding binomial arborescence of $D[U \cup \{w\}]$ rooted at w. Arbitrarily partition $V(D) \setminus (U \cup \{w\})$ into ℓ sets, $U_{\log n - \ell + 1}, \ldots, U_{\log n}$, such that $|U_i| = 2^{i-1}$ for all $i \in \{\log n - \ell + 1, \ldots, \log n\}$. For all $i \in \{\log n - \ell + 1, \ldots, \log n\}$, fix a tournament (competition) for $D[U_i]$, and let T_i denote the corresponding spanning binomial arborescence of $D[U_i]$ rooted at some vertex w_i. Define T by $V(T) = V(T_{\log n - \ell + 1}) \cup \ldots \cup V(T_{\log n}) \cup V(T')$ and $A(T) = A(T_{\log n - \ell + 1}) \cup \ldots \cup A(T_{\log n}) \cup \{(w, w_i) : i \in \{\log n - \ell + 1, \ldots, \log n\}\} \cup A(T')$. Moreover, define $R = \{(w_i, w) \in A(D) : i \in \{\log n - \ell + 1, \ldots, \log n\}\}$. Similarly to the proof of Theorem 2, the claim follows.

In the other direction, suppose that (D, w, ℓ) is a YES-instance. Let (R, T) be a solution that satisfies property 1 in Theorem 1. Let X' be the set of children v of w in T such that $(w, v) \in A(D)$. Since $|R| \leq \ell$, we have that $|X'| \geq \log n - \ell$, and we denote by X some subset of X' of size exactly $\log n - \ell$. Order $X = \{x_1, \ldots, x_{\log n - \ell}\}$ such that $|V(T_{x_i})| \leq |V(T_{x_{i+1}})|$ for all $i \in \{1, \ldots, \log n - \ell - 1\}$. Since (R, T) satisfies property 1, we know that T_{x_i} is a spanning binomial arborescence of $D[V(T_{x_i})]$ for all $i \in \{1, \ldots, \log n - \ell\}$. By Observation 1, it can be verified that there exists a subtree T'_{x_i} of T_{x_i} on exactly 2^{i-1} vertices that is a spanning binomial arborescence of $D[V(T'_{x_i})]$ rooted at x_i (the idea is to truncate the subtrees of the children of x_i that are largest). Now, denote $U = V(T'_{x_1}) \cup \ldots \cup V(T'_{x_{\log n - \ell}})$. Observe that $|U| = \sum_{i=1}^{\log n - \ell} |V(T'_{x_i})| = \sum_{i=1}^{\log n - \ell} 2^{i-1} = \frac{n}{2^\ell} - 1$. Thus, by taking all the arborescences T'_{x_i} and adding the arcs (w, x_i) (which belong to $A(D)$ by our choice of X), we exhibit a spanning binomial arborescence of $D[U \cup \{w\}]$ that is rooted at w. ☐

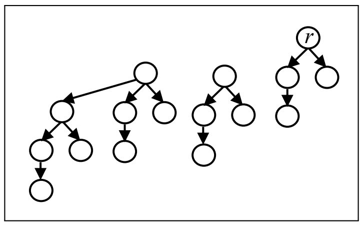

Figure 5: A 2-removed binomial arborescence on 2^4 vertices.

In our proof of the forward direction above, it can be shown that w is not necessarily made a king by reversing ℓ arcs. As a corollary of Theorem 3, we have the following one-way statement (which is constructive).

Theorem 4. *Let (D, w, ℓ) be an instance of* BTF. *If* $\mathsf{victory}(w) \geq \frac{n}{2^\ell} - 1$, *then (D, w, ℓ) is a* YES-*instance. In this case, a solution can be found in polynomial time.*

6 Exact Algorithms

Our exact algorithms are based on translating the characterization in Theorem 3 into the language of spanning binomial arborscences. To this end, we introduce a new definition (see Fig. 5). The validity of the sizes mentioned in this definition follows from Observation 1.

Definition 4. *Let $\ell \in \mathbb{N}$. Let T be a binomial arborescence of size n with $\log n \geq \ell$. Let the children of its root r be v_i, $i \in [\log n]$, where $|V(T_{v_i})| = 2^{i-1}$ for all $i \in \{1, \ldots, \log n\}$. An* ℓ-removed binomial arborescence *T' of size n is the digraph obtained from T by deleting the arcs (r, v_i) for all $i \in \{\log n - \ell + 1, \ldots, \log n\}$. The vertex r is the* root *of T'.*[2]

Note that up to isomorphism, an ℓ-removed binomial arborescence of size n is unique. We now give the promised translation of Theorem 3.

Lemma 3. *Let (D, w, ℓ) be an instance of* BTF. *Then, (D, w, ℓ) is a* YES-*instance if and only if D contains a spanning ℓ-removed binomial arborescence T rooted at w.*

Proof sketch. In one direction, suppose that D contains a spanning ℓ-removed binomial arborescence T rooted at w. Let T' denote the tree of T that is rooted at w. Then, $|V(T') \setminus \{w\}| = n - 1 - \sum_{i=\log n-\ell+1}^{\log n} 2^{i-1} = \frac{n}{2^\ell} - 1$ (by Observation 1) and T' is a spanning binomial arborescence of $D[V(T')]$ rooted at w. Denoting $U = V(T') \setminus \{w\}$, the claim follows from the reverse direction of Theorem 3.

In the other direction, suppose that (D, w, ℓ) is a YES-instance. By the forward direction of Theorem 3, there exists $U \subseteq V(D)$ of size exactly $\frac{n}{2^\ell} - 1$ so that there is a fixing of $D[U \cup \{w\}]$ that makes w win. Let T' be a spanning binomial arborescence of $D[U \cup \{w\}]$ rooted at w. Arbitrarily partition $V(D) \setminus (U \cup \{w\})$ into ℓ sets, $U_{\log n-\ell+1}, \ldots, U_{\log n}$, such that $|U_i| = 2^{i-1}$ for all $i \in \{\log n - \ell + 1, \ldots, \log n\}$. For all $i \in \{\log n - \ell + 1, \ldots, \log n\}$, fix a tournament (competition) for $D[U_i]$, and let T_i denote the corresponding spanning binomial arborescence of $D[U_i]$ rooted at some vertex w_i. Define T by $V(T) = V(T_{\log n-\ell+1}) \cup \ldots \cup V(T_{\log n}) \cup V(T')$ and $A(T) = A(T_{\log n-\ell+1}) \cup \ldots \cup A(T_{\log n}) \cup A(T')$, and let w be its root. Then, it is easy to verify that T is a spanning ℓ-removed binomial arborescence of D rooted at w. □

[2]Although T' contains $\ell + 1$ vertices of in-degree 0, there is only one vertex that we define as the root.

Towards the presentation of our first algorithm, we show how to detect ℓ-removed binomial arborescences.

Lemma 4. *Let D be a digraph, $w \in V(D)$, and $\ell \in \mathbb{N}$. We can decide in time $2^n n^{\mathcal{O}(1)}$ and polynomial space if D contains a spanning ℓ-removed binomial arborescence T rooted at w. If the answer is positive, we output T.*

Proof sketch. Amini et al. [Amini *et al.*, 2012] proved the following result. Suppose that we have two digraphs, D on n vertices and H on k vertices, such that the treewidth of the underlying undirected graph of H is t. Moreover, suppose that each vertex in D has a color from a set of k colors. Then, it can be decided in time $2^k n^{t+\mathcal{O}(1)}$ and polynomial space if D has a colorful subgraph isomorphic to H. This algorithm can be immediately adapted to also solve the case where each digraph among D and H has a distinguished vertex, say, d and h, so that the isomorphism must match d and h. In our setting, H is the (unique) ℓ-removed binomial arborescence T on n vertices (that is, $k = n$), whose underlying undirected graph is a forest and hence has treewidth 1. The coloring is simply the identity function (that is, every vertex in D gets a unique color), $d = w$ and h is the root of T. As the algorithm in [Amini *et al.*, 2012] also outputs a copy of H in D (if one exists) in the same running time bound, the correctness of our lemma follows. □

Having Lemmata 3 and 4, we are ready to present our exponential-time algorithm.

Theorem 5. BTF *can be solved in time $2^n n^{\mathcal{O}(1)}$ and polynomial space. In the case of a* YES-*instance, a solution can be found in the same running time bound.*

Proof sketch. Given an instance (D, w, ℓ) of BTF, call the algorithm in Lemma 4 to decide if D contains an ℓ-removed binomial arborescence T' of size n rooted at w. If the answer is negative, return NO. Else, let T' be the output. Define T as the binomial arborescence obtained from T' by adding arcs from w to all other vertices in T' of in-degree 0. Moreover, define $R = (A(T) \setminus A(D))^{\text{rev}}$, and observe that $|R| \leq \ell$. The correctness of the algorithm then follows from Lemma 3. □

Slight modification of the algorithm in [Ramanujan and Szeider, 2017], which detects if D has a spanning binomial arborescence rooted at w, proves the following.

Lemma 5 (*). *Let D be a tournament with feedback arc set number k, $w \in V(D)$ and $\ell \in \mathbb{N}$. We can decide in time $2^{\mathcal{O}(k^2 \log k)} n^{\mathcal{O}(1)}$ and polynomial space whether D contains an ℓ-removed binomial arborescence T of size n rooted at w. If the answer is positive, we output T.*

Replacing Lemma 4 by Lemma 5 proves the following.

Theorem 6. BTF *can be solved in time $2^{\mathcal{O}(k^2 \log k)} n^{\mathcal{O}(1)}$ and polynomial space, where k is the feedback arc set number of the input tournament. In case of a* YES-*instance, a solution can be found in the same running time bound.*

7 Conclusion

In this paper, we studied BTF and gave combinatorial results, polynomial-time algorithms for special cases, and exact exponential-time algorithms for the general case. One of our results imply that the problem is FPT parameterized by k, the size of a feedback arc set of the input tournaments. A natural question, therefore, is whether the problem admits a polynomial kernel. Another open question is whether BTF (or even TOURNAMENT FIXING) is FPT when parameterized by the size of a feedback vertex set of the input tournament.

Acknowledgements

We would like to thank anonymous referees for valuable comments that helped improve the article.

References

[Amini *et al.*, 2012] Omid Amini, Fedor V. Fomin, and Saket Saurabh. Counting subgraphs via homomorphisms. *SIAM J. Discrete Math.*, 26(2):695–717, 2012.

[Aronshtam *et al.*, 2017] Lior Aronshtam, Havazelet Cohen, and Tammar Shrot. Tennis manipulation: can we help Serena Williams win another tournament? - or can we control a knockout tournament with reasonable complexity? *Ann. Math. Artif. Intell.*, 80(2):153–169, 2017.

[Aziz *et al.*, 2014] Haris Aziz, Serge Gaspers, Simon Mackenzie, Nicholas Mattei, Paul Stursberg, and Toby Walsh. Fixing a balanced knockout tournament. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada. (Accepted to Artificial Intelligence Journal (AIJ).)*, pages 552–558, 2014.

[Feige, 2009] Uriel Feige. Faster FAST (Feedback Arc Set in Tournaments). *CoRR*, abs/0911.5094, 2009.

[Karpinski and Schudy, 2010] Marek Karpinski and Warren Schudy. Faster algorithms for feedback arc set tournament, Kemeny rank aggregation and betweenness tournament. In *Proceedings of 21st International Symposium Algorithms and Computation ISAAC 2010*, pages 3–14, 2010.

[Kenyon-Mathieu and Schudy, 2007] Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 95–103, 2007.

[Kim and Williams, 2015] Michael P. Kim and Virginia Vassilevska Williams. Fixing tournaments for kings, chokers, and more. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI*, pages 561–567, 2015.

[Kim *et al.*, 2016] Michael P. Kim, Warut Suksompong, and Virginia Vassilevska Williams. Who can win a single-elimination tournament? In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 516–522, 2016.

[Laslier, 1997] J. F. Laslier. Tournament solutions and majority voting. *Springer-Verlag*, 1997.

[Mattei and Walsh, 2016] Nicholas Mattei and Toby Walsh. Empirical evaluation of real world tournaments. *CoRR*, abs/1608.01039, 2016.

[Mattei *et al.*, 2015] Nicholas Mattei, Judy Goldsmith, Andrew Klapper, and Martin Mundhenk. On the complexity of bribery and manipulation in tournaments with uncertain information. *J. Applied Logic*, 13(4):557–581, 2015.

[Ramanujan and Szeider, 2017] M. S. Ramanujan and Stefan Szeider. Rigging nearly acyclic tournaments is fixed-parameter tractable. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3929–3935, 2017.

[Rosen, 1986] S. Rosen. Prizes and incentives in elimination tournaments. *The American Economic Review*, 76(4):701–715, 1986.

[Russell and van Beek, 2011] Tyrel Russell and Peter van Beek. An empirical study of seeding manipulations and their prevention. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 350–356, 2011.

[Russell and Walsh, 2009] Tyrel Russell and Toby Walsh. Manipulating tournaments in cup and round robin competitions. In *Algorithmic Decision Theory, First International Conference, ADT 2009, Venice, Italy, October 20-23, 2009. Proceedings*, pages 26–37, 2009.

[Stanton and Williams, 2011a] Isabelle Stanton and Virginia Vassilevska Williams. Manipulating stochastically generated single-elimination tournaments for nearly all players. In *Internet and Network Economics - 7th International Workshop, WINE 2011, Singapore, December 11-14, 2011. Proceedings*, pages 326–337, 2011.

[Stanton and Williams, 2011b] Isabelle Stanton and Virginia Vassilevska Williams. Rigging tournament brackets for weaker players. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 357–364, 2011.

[Tullock, 1980] G. Tullock. Toward a theory of the rent-seeking society. *Texas A&M University Press*, 1980.

[Vu *et al.*, 2009] Thuc Vu, Alon Altman, and Yoav Shoham. On the complexity of schedule control problems for knockout tournaments. In *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary, May 10-15, 2009, Volume 1*, pages 225–232, 2009.

[Williams, 2010] Virginia Vassilevska Williams. Fixing a tournament. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010.

[Williams, 2016] Virginia Vassilevska Williams. Knockout tournaments. In *Handbook of Computational Social Choice*, pages 453–474. Cambridge University Press, 2016.