Fixing Tournaments for Kings, Chokers, and More *

Michael P. Kim and Virginia V. Williams
Computer Science Department
Stanford University
{michael.kim,virgi}@cs.stanford.edu

Abstract

We study the tournament fixing problem (TFP), which asks whether a tournament organizer can rig a single-elimination (SE) tournament such that their favorite player wins, simply by adjusting the initial seeding. Prior results give two perspectives of TFP: on the one hand, deciding whether an arbitrary player can win any SE tournament is known to be NP-complete; on the other hand, there are a number of known conditions, under which a player is guaranteed to win some SE tournament. We extend and connect both these lines of work. We show that for a number of structured variants of the problem, where our player is seemingly strong, deciding whether the player can win any tournament is still NP-complete. Dual to this hardness result, we characterize a new set of sufficient conditions for a player to win a tournament. Further, we give an improved exact algorithm for deciding whether a player can win a tournament.

1 Introduction

Consider the following natural problem called *tournament fixing* (TFP): Suppose we are given n players V, a favorite player $v \in V$ and for every choice of players $x, y \in V$, reliable information about who would win the match between x and y, were they to play. Then, can we find a seeding of a balanced single-elimination tournament, so that if all match outcomes turn out as predicted, player v will win?

Single-elimination tournaments, also called knockout tournaments, are prevalent in many diverse scenarios. The most common occurrence is in sports tournaments such as Wimbledon or March Madness, but they are also used in drug trials, in patent races [Lita, 2008], hiring employees from a pool of candidates, and also as voting protocols. The latter application assumes that there are n candidates and the match outcomes are determined via pairwise majority preferences of the voter population. The tournament fixing problem in this case is just a question of whether an election chairman can influence the outcome of a *binary cup* election by fixing the order in which candidates are compared. Computationally, we are interested in whether there is an efficient algorithm for TFP. This question falls in the general line of work on the complexity of agenda control for voting introduced by [Bartholdi *et al.*, 1989; 1992].

Over the last decade, there have been many papers studying the complexity of TFP [Lang *et al.*, 2007; Hazon *et al.*, 2007; 2008; Vu *et al.*, 2009; Vassilevska Williams, 2010; Stanton and Vassilevska Williams, 2011a; 2011b; Aziz *et al.*, 2014]. There are roughly two approaches to the problem. The first is to prove worst-case hardness results. This approach culminated in a recent paper by [Aziz *et al.*, 2014] that showed that determining if there is a winning seeding for the favorite player is NP-complete [1] even when the match outcome information is completely accurate.

The second approach to the problem is to find a general class of inputs for which an efficient solution exists. This approach is based on the premise that the instances of TFP that one sees in practice are not really worst-case, so that the hardness results may not apply to the real world. This approach has produced results of the following form: suppose that the match outcomes are actually produced by a process that takes the abilities of players into account but adds random noise, then for several very interesting models of this form, in almost all cases one can make any player a winner, i.e. a winning seeding exists with high probability over the randomness of the noise [Vassilevska Williams, 2010; Stanton and Vassilevska Williams, 2011a]. In this paper we extend our knowledge along both approaches.

Results. The input to the problem is an n-node tournament[2] graph where the nodes are the players and there is an edge from i to j if and only if i would beat j in a match. A king is a player that has distance at most 2 to every other player in the tournament graph (i.e. for each j, it either beats j or beats another k that beats j). A result [Vassilevska Williams, 2010] from the second approach to attacking TFP is that if the favorite player v is a king and it can beat at least half the players, then there is always a winning seeding for v. Could it

*This research was supported by a Stanford School of Engineering Hoover Fellowship, NSF Grant CCF-1417238 and BSF Grant BSF:2012338.

[1] TFP is clearly in NP; given a seeding, we can compute the winner in $n-1$ matches. The challenge was to show NP-hardness.

[2] A tournament graph is a directed graph for which for every i, j, exactly one of (i, j) and (j, i) is an edge.

be that there is always an efficient algorithm for determining that a king player can win, regardless of its outdegree? In the known NP-hardness result for TFP [Aziz *et al.*, 2014], the favorite player is far from being a king; moreover, it only beats $\log n$ players – the minimum needed to be able to win.

We resolve the king TFP question as follows. In Theorem 1 we show that TFP is still NP-hard, even when the favorite player is a king that beats $n/4$ players. This result is close to tight, as kings that beat $n/2$ players always have a winning seeding [Vassilevska Williams, 2010]. In addition, if the favorite player has distance at most 3 to all other players in the tournament graph (a "3-king"), then TFP is NP-hard even if the 3-king can beat half of the players. In all, we prove NP-hardness for TFP under even slight perturbations of most known structural results where the favorite player can always win, showing that there are natural parameters for which the player can always be made a winner, but if the parameters are slightly off, it is NP-hard to determine whether she can.

We also address a scenario where the tournament organizer can bribe some players to intentionally lose a match. It is clear that with only $\log n$ bribes, any player can be made the winner (just bribe the players he plays against). We show that it is NP-hard to find a way to bribe $(1-\varepsilon)\log n$ players for any constant $\varepsilon > 0$ in order for the favorite player to win.

For the second type of approach to TFP, we extend the work of [Stanton and Vassilevska Williams, 2011b; 2011a] to the case of 3-kings by presenting sufficient conditions under which a winning seeding for a 3-king can always be found efficiently. We then propose an application of this structural results to the following scenario.

One setting, in which an organizer might be interested in manipulating the seeding, arises when a given player is quite skilled, but tends to "choke under pressure" and lose against significantly weaker players. The organizer may have incentive to allow the player to survive into the later rounds in order to increase the appeal of final matches (amongst stronger, more popular players). The notion of "choking" is a well-known phenomenon in sports and other competitions, and has been studied as a broader psychological phenomenon [Baumeister, 1984; Beilock and Carr, 2001].

To model player choking, we introduce a new generative model for tournaments based on a model proposed by Condorcet (see e.g. [Braverman and Mossel, 2008; Stanton and Vassilevska Williams, 2011a]). Just as with Condorcet's model, we assume that there is an underlying total ordering of the player $v_1, \ldots, v_n$ where v_i is stronger than v_{i+1}. In Condorcet's model, there is a probability $p < 1/2$ for which a weaker player v_j beats a stronger player v_i, $i < j$, and stronger players beat weaker players with probability $1-p$. In the new model, we have two parameters, $p \le q < 1/2$, and a threshold m. For every i and j, $i < j$, such that both $i, j < m$ or $i, j \ge m$, v_i beats v_j with probability $1-p$ (and loses with probability p). If however $i < m$ and $j \ge m$, then v_j beats v_i with probability $q \ge p$.

Intuitively, the last $n-m$ players are the weakest, and these are the ones that stronger players typically choke against. More complicated models of choking can be defined, and similar results can be proven for them, but here we opt for simplicity. We show that our result on 3-kings implies that, for almost all tournament graphs generated by this model, if $\Omega(\ln n/n) \le q < 1/3 - o(1)$, then even if p is very small (even 0), any of the top $n/3 - o(n)$ players have a winning seeding, as long as the players anyone chokes against are among the last $n/3$ players. Such a result cannot be obtained using the previously known structural results.

Finally, [Aziz *et al.*, 2014] also considered computing the number of winning seedings for a given player. They obtained a variety of algorithms for the problem, the best runtime of which was $O(2.8285^n)$. We obtain an improved runtime of $O(2^n \text{poly } n)$. The number of winning seedings is a natural notion of player strength – it is proportional to the probability that the player will win a randomly seeded tournament.

Notation
$\mathcal{N}_{out}(v) = \{u \mid (v,u) \in E\}$, $\mathcal{N}_{out,S}(v) = \mathcal{N}_{out}(v) \cap S$
$\mathcal{N}_{in}(v) = \{u \mid (u,v) \in E\}$, $\mathcal{N}_{in,S}(v) = \mathcal{N}_{in}(v) \cap S$
$out(v) = \lvert\mathcal{N}_{out}(v)\rvert$, $out_S(v) = \lvert\mathcal{N}_{out,S}(v)\rvert$
$in(v) = \lvert\mathcal{N}_{in}(v)\rvert$, $in_S(v) = \lvert\mathcal{N}_{in,S}(v)\rvert$
$G^{(i)} = G$ after $i \in [\log n]$ rounds of play $S^{(i)} = \{s \in S \mid s$ is alive after i rounds of play$\}$
$S_{j:\ell} = \{s_k \mid j \le k \le \ell\}$

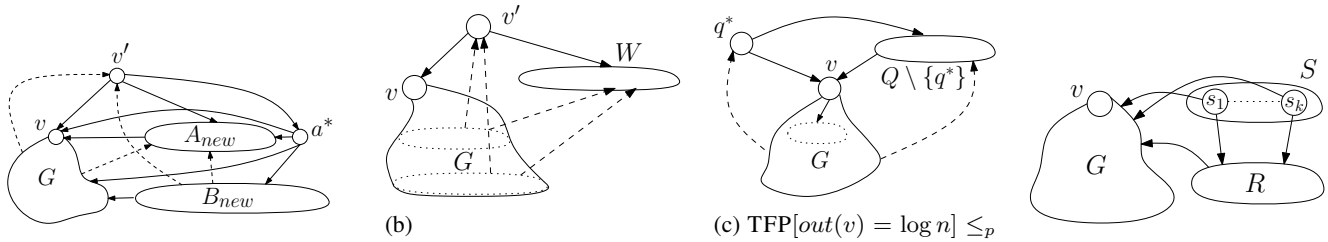
Table 1: Summary of the notation used in this paper.

1.1 Preliminaries and Notation

In this paper, we focus on *single-elimination (SE) tournaments*, which are played as follows. The organizer specifies a permutation of the players, which corresponds to a *seeding* or *bracket*. Then, players are matched according to this seeding, in disjoint pairs of consecutive players, advancing to the next round if and only if they win their match. Matches in subsequent rounds are predetermined by the original seeding, and play continues until only one player remains. This player wins the SE tournament.

We will call v an *SE winner* over G if one can efficiently, in polynomial time, construct a seeding to a balanced SE tournament where v wins. Showing that v is an SE winner over G is equivalent to finding a subgraph of G which is a spanning *binomial arborescence* rooted at v, which is defined recursively as follows: a single node a is an order 0 binomial arborescence rooted at a; a is the root of an order i binomial arborescence over S where $|S| = 2^i$ if a is the root of an order $i-1$ binomial arborescence over S_a and has a child b who is also the root of an order $i-1$ binomial arborescence over S_b, where $S_a \cap S_b = \emptyset$ and $|S_a| = |S_b| = 2^{i-1}$. In an arborescence, all edges point away from the root. We say v a *Condorcet winner* over S if v wins against all players in S.

In Table 1, we give a summary of the notation that will be used throughout this paper. We let v, u be any nodes in V and let $S \subseteq V$. We will abuse notation and use $\mathcal{N}(S)$ to be $\bigcup_{s \in S} \mathcal{N}(s)$, for in and out neighbors. In some subsets $S \subseteq V$, it will be useful to define a total order over the players in S, and to refer to groups of players by their indices; thus, for $S_{j:\ell}$, we require $j \ge 1$ and $\ell \le |S|$. All graphs will be tournament graphs over $n = 2^k$ players for some $k \in \mathbb{N}$, unless explicitly stated otherwise. This ensures that the SE tournament will be balanced.



(a) TFP $\leq_p$ KING-TFP (b) KING-TFP $\leq_p$ 3KING-TFP (c) TFP$[out(v) = \log n] \leq_p$ NEARLY-AS-STRONG-TFP (d) TFP $\leq_p$ BRIBERY-TFP

Figure 1: A visual summary of our reductions. We label the original (G, v) as well as the sets of nodes added by the reductions.

2 Hardness Results

Restrictions of TFP are NP-hard. Here, we extend the work of [Aziz *et al.*, 2014] to show, somewhat surprisingly, that many natural restrictions of TFP are still NP-hard by reducing TFP to the restricted versions. In particular, we show that a number of sufficient conditions for a player v to be an SE winner are fairly tight, in the sense that relaxing the conditions results in problem instances that are NP-hard to decide.

At a high-level, our reductions proceed according to the following framework. We start with an instance of TFP, (G, v) where $G = (V, E)$, and create a new graph over $G' = (V', E')$, which contains G as a subgraph. Then, we designate a desired winner v' and add a set of dummy players S. The players in S will play similarly against players outside of S (for instance, $\forall s \in S,\ \forall u \in V,\ (s, u) \in E'$). The additional players serve two purposes: they add structure to the instance (G', v') (for instance, making v' a king) and importantly, they enforce the property that v' can win in our new tournament if and only if v plays exclusively within G and wins the subtournament.

First, we show that, unless P = NP, any algorithm for deciding whether a king can win a tournament would not be efficient, even if the king wins against a quarter of the players.

Theorem 1. *TFP is* NP*-hard even when the player of interest v is a king with $out(v) = |V|/4$.*

Proof. Consider some instance of TFP, (G, v) on n nodes. From $G = (V, E)$, we construct $G' = (V', E')$ as follows: include all nodes and edges from G in G', and add a node v' with $(v', v) \in E'$. Add the following $3n - 1$ new players:

- $n - 1$ players called A_{new}, where $\forall a \in A_{new}$, $(v', a), (a, v) \in E'$ and $\forall u \in V \setminus \{v\}$, $(u, a) \in E'$
- $2n - 1$ players called B_{new}, where every player $b \in B_{new}$ wins against all players in $V \cup A_{new} \cup \{v'\}$
- a player called a^* who only loses to v'.

Let all unspecified edges point towards v' and be directed arbitrarily within A_{new} and B_{new}. Note that this makes G' a tournament graph over $4n$ players, where v' is a king with $out(v') = n$. We claim $(G', v') \in$ TFP $\iff (G, v) \in$ TFP.

Note that all $b \in B_{new}$ are Condorcet winners over everyone except a^*. Thus, each $b \in B_{new}$ must be eliminated either by another node in B_{new}, or by a^*. This means that a^* must be the winner of some subtournament containing all of B_{new}. Because $|B_{new} \cup \{a^*\}| = 2n$, this subtournament will take $\log n + 1$ rounds, and v' must not play against a^* until the final round. Next, note that if v' played v before the penultimate round, then some other $u^* \in V \setminus \{v\}$ would survive to play against v', because all $u \in V \setminus \{v\}$ are Condorcet winners over A_{new}, and v' would lose. Thus, for v' to win, v' must win over the n players in $A_{new} \cup \{v'\}$, while v must survive for $\log n$ rounds and win over G. □

We can use the fact that TFP is hard even when the player of interest is a king to show that it is hard even when the player of interest is a 3-king with out-degree at least $n/2$.

Theorem 2. TFP *is* NP*-hard, even when the player of interest v is a 3-king with $out(v) = |V|/2$.*

Proof. Consider some instance of TFP, (G, v), where v is a king on n nodes. Construct G' by adding v' with $(v', v) \in E'$ and $\forall u \in V \setminus \{v\}$, $(u, v') \in E'$. Also, add $n - 1$ other players, which we call W. All $w \in W$ will lose to every player in $V \cup \{v'\}$ (and play arbitrarily amongst themselves). We claim that $(G', v') \in$ TFP $\iff (G, v) \in$ TFP.

Note that v' is a 3-king in G' because v' wins over v who was a king over the original graph, and wins over all $w \in W$. Also note that if v' plays any $u \in V$ before playing v in the final round, then v' will lose the tournament. Thus, for v' to win, v' must survive through $W \cup \{v'\}$, while v survives through V. In other words, v must win over G. □

Another result from [Vassilevska Williams, 2010] shows that players who are stronger than the players who beat them are SE winners. It is natural to wonder what happens when we relax this condition, requiring that v be "nearly as strong" as the players who beat v. In the following theorem, we formalize this notion and give a corresponding hardness result.

Theorem 3. *For all constant $\varepsilon > 0$, TFP is* NP*-hard even when $\forall u \in \mathcal{N}_{in}(v)$, $out(u) \leq (1 + \varepsilon) out(v)$.*

Proof. Consider some instance of TFP (G, v) on n nodes, where we require v to have low out-degree[3], $out(v) = \log n$. From $G = (V, E)$, we will construct $G' = (V', E')$ as follows. We add $N \geq n$ nodes, called Q, where $\forall q \in Q$, $(q, v) \in E'$ and $\forall u \in V \setminus \{v\}$, $(u, q) \in E'$. We choose one $q^* \in Q$ and $\forall q \in Q \setminus \{q^*\}$, we add $(q^*, q) \in E'$. The rest of Q may play arbitrarily. Note that $out(q^*) = N$ and $\forall u \in \mathcal{N}_{in}(q^*)$, $out(u) < N + n$. Thus, for $N \geq n/\varepsilon$, then for all $u \in \mathcal{N}_{in}(q^*)$, $out(u) \leq (1 + \varepsilon) out(v)$. We claim $(G', q^*) \in$ TFP $\iff (G, v) \in$ TFP.

[3]By Aziz *et al.*'s original reduction, TFP on instances with $out(v) = \log n$ is known to be NP-hard.

Note that all players $u \in V \setminus \{v\}$ are Condorcet winners over Q. If any such u survives beyond v, then some player from V will win the tournament. This means, for q^* to win, v must win a subtournament over the players in V and possibly some from Q. But by our assumption, $out(v) = \log n$, so v can win a subtournament of at most n players – the number of players in V. It is easy to see that if v survives $\log n$ rounds, then q^* is a Condorcet winner over the remaining players. Thus, q^* wins over G' if and only if v wins over G. □

Limited bribery doesn't help. Beyond considering instances where our player is strong, we might be equally interested in studying what happens if we give the organizer more power, such as the ability to bribe players to throw a match. We define BRIBERY-TFP as follows: given a tournament $G = (V, E)$, a desired winner v, and a natural number k, decide whether v can win an SE tournament in G when we allow the organizer to bribe up to k players to throw a match that they would otherwise win. Clearly, when $k = 0$, the problem is equivalent to TFP. When $k \geq \log n$, then the tournament can certainly be fixed [4]. The question is what happens inbetween.

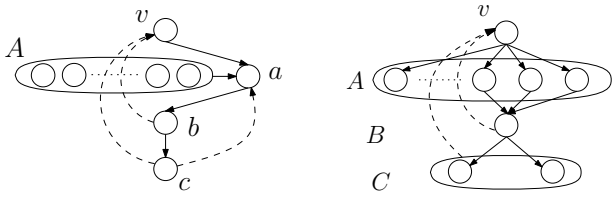
Theorem 4. *For all constant $\varepsilon > 0$,* BRIBERY-TFP *is* NP*-hard, when $k \leq (1 - \varepsilon) \log n$.*

Proof. Consider some instance of TFP, (G, v) on n nodes. We construct G_k as follows: add a set of $n(2^k - 1) - k$ players called R who are Condorcet winners over V, and add a set of k players called S who are Condorcet winners over $V \cup R$.

Suppose some player $t \in S \cup R$ was seeded to play against v in the first $\log n$ rounds. If $(t, v) \in E$, then v would lose. If t was bribed to lose, so that $(v, t) \in E$, and v survives past the first $\log n$ rounds, then $out(v) \leq k - 1$, with k rounds remaining. To see this, note that the only players who remain after $\log n$ rounds in $\mathcal{N}_{out}(v)$ are those from $R \cup S$ who were bribed to throw their matches to v and any other surviving players $u \in V \cap \mathcal{N}_{out}(v)$. But in order for any such u to survive $\log n$ rounds, some $t \in R \cup S$ must have been bribed to lose against u. Thus, because one bribed player t, has already lost, there are at most $k - 1$ remaining. This means, in order to win over G_k, v must first win over G, then win against the k bribed players $\subseteq S \cup R$. Thus, v can win over G_k with k bribes if and only if v can win over G. Thus, for any constant $\varepsilon > 0$, we can obtain a polynomial time reduction to a BRIBERY-TFP instance of size $N = n^{1/\varepsilon}$ and with $k = (1 - \varepsilon) \log N$ bribes. □

3 Structural Results

Previous work characterizes a variety of sufficient conditions for a player to win an SE tournament [Vassilevska Williams, 2010; Stanton and Vassilevska Williams, 2011b]. Many of these results are about players who are kings. In this section, we extend the notion of a king and describe a new set of sufficient conditions for a 3-king to be an SE winner. We start with a set of motivating examples, where v is a 3-king but cannot win, justifying that, in some sense, our conditions are necessary. When discussing 3-kings, we will generally let v be our 3-king and player of interest, and define the following disjoint subsets of the other players by their distance from v as $A = \mathcal{N}_{out}(v)$, $B = \mathcal{N}_{out}(A) \cap \mathcal{N}_{in}(v)$, and $C = \mathcal{N}_{in}(v) \setminus B$. Thus, $V = \{v\} \cup A \cup B \cup C$.



(a) $\mathcal{N}_{out}(a) = \{b\}$, $\mathcal{N}_{in}(b) = \{a\}$, $\mathcal{N}_{in}(c) = \{b\}$. No matter how big $|A|$ is, v cannot win, as there are not enough edges into b and c to defeat both players.

(b) Without a perfect matching onto C, even for large A, the players in A with edges into B may be eliminated before B is able to defeat C.

Figure 2: Examples of tournaments where v is a 3-king, but cannot win an SE tournament under any seeding.

Motivating Examples. To start, we might hope to borrow techniques from [Vassilevska Williams, 2010; Stanton and Vassilevska Williams, 2011b] and try to construct a seeding based on the out-degree of v or a matching into the players who win against v. In particular, perhaps for sufficiently large A, if there is a perfect matching[5] from B onto C, then v will win some SE tournament. This intuition turns out to be false. To see this, consider Figure 2a. In this simple example, v wins against all but two players, and there is a perfect matching from B onto C. Nevertheless, v still cannot win any SE tournament – either b will advance beyond a, or c advances beyond b. In either case, b or c will win every SE tournament.

Another idea from [Vassilevska Williams, 2010] would be to argue that if the players in B are no stronger than v (i.e. $\forall b \in B,\ out(b) \leq out(v)$), then if $|C|$ is sufficiently small, we can find a seeding where v will win. In Figure 2b, we show that without additional assumptions, this condition is not sufficient. Even for large A and small C, there are tournaments where the players in B are no stronger than v, but v cannot win any SE tournament. We will see that, while these conditions fail independently, in combination, they are sufficient for v to be made a winner.

3.1 Main Result

Theorem 5. *Consider a tournament $G = (V, E)$ where $v \in V$ is a 3-king. Let $A = \mathcal{N}_{out}(v)$, $B = \mathcal{N}_{out}(A) \cap \mathcal{N}_{in}(v)$, and $C = \mathcal{N}_{in}(v) \setminus B$. v is guaranteed to be an SE winner if the following conditions hold.*

1. $|A| \geq \frac{|V|}{3}$
2. $\forall b \in B,\ out(b) \leq out(v)$

[4] In fact, v can win for any $k \geq \log n - \lfloor \log d \rfloor$ where $d = out(v)$. To see this, consider matching players in $\mathcal{N}_{out}(v) \cup \{v\}$ until v is the sole survivor. v can last $\lfloor \log d \rfloor$ rounds playing $\mathcal{N}_{out}(v)$ and thus needs only $\log n - \lfloor \log d \rfloor$ more wins to win out.

[5] We say there is a perfect matching M from S onto T if M is a node-disjoint subset of edges $(s, t) \in S' \times T$ for some $S' \subseteq S$ where $|S'| = |T| = |M|$.

3. There is a perfect matching from B onto C

Our proof of Theorem 5 involves many technical pieces. For a detailed account of all of these pieces, please refer to the full version of this section. We start by outlining the algorithm we use to construct the initial matchups and give an intuition for why it works.

In the first round, we will match $B^{(0)}$ onto $C^{(0)}$ using the perfect matching (see Table 1 for the notation used here). While these players are playing, we will match $A^{(0)}$ against itself in a manner that ensures v will be a king over $V^{(1)}$. In particular, we will find a matching in $A^{(0)}$ such that for every player $b \in B^{(0)}$, there will be some surviving $a \in A^{(1)}$, where $(a, b) \in E$. Separately, we can then argue that for sufficiently large $A^{(0)}$, the resulting $A^{(1)}$ will be large enough for v to be a guaranteed SE winner.

We will say that $a \in A$ *covers* $b \in B$ if we select (a, a') in our matching, such that a survives to $A^{(1)}$, and $(a, b) \in E$. We assume all $b \in B$ participate in the perfect matching onto C. (If there is some extra $b' \in B$, we can match these players in the first round to some $a' \in A$ and remove the covered players from consideration.) Note that this means our bound on the out-degree of $b \in B$ by $out(v) = |A|$ implies $\forall b \in B,\ out_B(b) + 2 \leq in_A(b)$. If we rank the players $b \in B$ in ascending order by $in_A(b)$, we can see that our assumption implies the following condition.

$$\sum_{b \in B_{1:i}} in_A(b) \geq \binom{i}{2} + 2i \qquad \forall i \in \{1, \ldots, |B|\} \qquad (1)$$

Note that (1) implies every node in B starts with at least two in-edges from A. We claim that if (1) holds, we can cover the nodes in B. Intuitively, to cover some $b \in B$, we want to find an edge (a, a') where a beats b and a' doesn't have too many edges into other players in B. To accomplish this, we might start with the player $b_{min} \in B$ who has the minimum in-degree from A, and match two players a, a' who each beat b_{min}, repeating this for each $b \in B$ until all of B is covered.

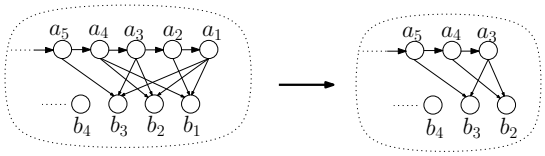


Figure 3: If (a_2, a_1) covers b_1, the invariant (1) is broken.

This intuition is born out precisely in our covering algorithm when there is a unique player with in-degree from A of 2; however, when the player of minimum in-degree has more than 2 in-edges, we need to be careful about which edge we use to cover b_{min}. Consider Figure 3, which depicts a possible subset of the players in A and B. In particular, note that the in-degree from A of b_1, b_2, b_3 all equal 3, and thus, satisfy (1). Nevertheless, if we choose an arbitrary edge to cover b_1, then in the resultant graph, we may not be able to cover both b_2 and b_3. For instance, suppose A is transitive (that is, $\forall j > i,\ (a_j, a_i) \in E$). Then if (a_2, a_1) is selected to cover b_1, v cannot be a king at the start of round 2 because either b_2 or b_3 will not be covered. Algorithm 1 handles cases like this by noting that if the minimum in-degree from A in B is d, then there are at most $2d-3$ nodes $b \in B$ where $in_A(b) = d$. This follows directly from (1). Our key insight is to look for a way to cover multiple nodes amongst these $2d-3$ possible "minima". We argue that if some $a \in A$ covers at least two of the first $2d-3$ nodes, then (1) is maintained. If not, then there is a matching that does not affect the in-degrees of other minimal nodes and maintains the invariant.

Algorithm 1 COVERB(A, B, E)

$M_A \leftarrow \emptyset$
while $B \neq \emptyset$ **do**
  Sort the players $b \in B$ by $in_A(b)$ (ascending)
  Let $d \leftarrow in_A(b_1)$
  if $\exists (a, a') \in E$ s.t. $a, a' \in \mathcal{N}_{in}(b_1)$ and $\mathcal{N}_{out}(\{a, a'\}) \cap B_{1:2d-3} = \{b_1\}$ **then**
    $e \leftarrow (a, a')$
  else $\{\exists\ a, a', b_j$ s.t. $(a, a'), (a, b_1), (a, b_j) \in E$ where $j \leq 2d-3\}$
    $e \leftarrow (a, a')$
  end if
  $M_A \leftarrow M_A \cup \{e\}$
  $A \leftarrow A \setminus \{a, a'\}$
  $B \leftarrow B \setminus \mathcal{N}_{out,B}(a)$
end while

We break the analysis of Algorithm 1 into two cases. First, we check if there is a pair of players in A who beat b_1 and beat none of the other $2d-3$ lowest-ranked players. When $in_A(b_1) = 2$, we always fall into this case, and match the two players who win against b_1. Otherwise, we know that $in_A(b_1) \geq 3$ and there is some edge (a, a') we can choose where $a \in in_A(b_1)$ covers b_1 and another $b_j \in B$ for some $j \leq 2d-3$. It should be clear these cases are exhaustive; moreover, we claim that in either case, (1) is maintained as a loop invariant. Proving this fact requires carefully accounting for the edges that could be lost by the covering in each iteration. A detailed analysis is provided in the full version.

To prove the full theorem, we need an additional lemma. The lemma is a strengthened version of a theorem from [Vassilevska Williams, 2010] about kings with large out-degrees.

Lemma 1. *Assume v is a king, where $A = \mathcal{N}_{out}(v)$ and $B = \mathcal{N}_{in}(v)$. Let k be the cardinality of the maximum matching from A to B. If $|A| > \frac{n}{2} - k$, then one can efficiently rig an SE tournament such that v wins.*

The proof of the lemma is straightforward and simply involves using the perfect matching in the first round to result in a king who wins against at least half the remaining players.

To complete the proof, we argue that in $G^{(1)}$, $out(v)$ and the size of the matching from $A^{(1)}$ to $B^{(1)}$ will be sufficiently large. In order to ensure a large matching, we introduce the notion of assignment. We will say that $b \in B$ is *assigned*, if some $a \in A$ covered b when b was the minimum element (i.e. not coincidentally). Then, we will run Algorithm 1 repeatedly on the unassigned players in B, until we have a maximal matching. By bounding the number of edges from A to B lost per assignment, we can show that if $|A|$ is initially

$|V|/3$, then the resultant graph will satisfy the conditions for Lemma 1.

3.2 Choking Under Pressure in Random Graphs

We can reasonably model the phenomenon of choking under pressure using the generative model proposed in the introduction. Recall, the model assumes that, in general, upsets are rare occuring with probability p, but when a player v_i plays against another player v_j weaker than some threshold $m < j$, v_i may be prone to choke against v_j with a greater probability q in a match up v_i should have otherwise won.

Under this model, we can argue that with high probability nearly a third of the players will be 3-kings satisfying the conditions in Theorem 5, and thus will be SE winners.

Theorem 6. *Let $\Delta = \sqrt{n \ln n}$. There exists an n_0 such that for all $n > n_0$ if $p \leq \Theta(\ln n/n) \leq q \leq (m - i - \Theta(\Delta))/n$, a player v_i will be an SE winner for any $i \leq n/3 - \Theta(\Delta)$ with high probability when $m \geq i + n/3 + \Theta(\Delta)$.*

Corollary 1. *For any v in the top $n/3 - o(n)$ players, v is an SE winner with high probability over a tournament generated by our model even when the top players choke against the bottom third of players with constant probability $q < 1/3 - o(1)$.*

The proof of Theorem 6 uses familiar concentration bounds and properties of random graphs as in [Stanton and Vassilevska Williams, 2011a] to show that the following three claims hold with high probability.

First, because $p \leq \Theta(\ln n/n)$, we know there will be at most $i + \Theta(\Delta) < n/3$ strong players who beat v_i. Second, if $m > i + n/3 + \Theta(\Delta)$, then v_i will win against at least $n/3$ strong players. Third, if $q \leq (m - i - \Theta(\Delta))/n$, then for every weak player w, $out(w) \leq out(v_i)$; moreover, because $q \geq \Omega(\ln n/n)$, there will be a perfect matching from these weak players onto the strongest players who beat v_i. In combination, these properties, which hold with high probability, correspond directly to the required conditions on our 3-king in Theorem 5. It follows that any such v_i will be an SE winner with high probability.

4 Exact Algorithm

We consider computing the number of seedings for which a particular player can win a balanced SE tournament in the more general case of incomplete information in which the input tournament graph may be missing some edges. This number is proportional to the probability that the player can win the tournament under a random seeding, and is of independent interest. We will prove two statements. The first is that if the given tournament graph is allowed to be missing some edges, then computing the number of winning seedings for a given player is $\#P$-hard. The second result is a $2^n \text{poly}\, n$ time algorithm for exactly solving this problem. We begin with the hardness result.

Proposition 1. *Given a directed graph $G = (V, E)$ such that for any $x, y \in V$, at most one of $\{(x, y), (y, x)\}$ is in E, and given a node $v \in V$, counting the number of spanning binomial arborescences of G rooted at v is $\#P$-complete. That is, given only partial information about match outcomes, determining the probability that a given player can win a random single-elimination tournament is $\#P$-hard.*

The proof proceeds via a reduction from bipartite perfect matching: given a bipartite graph on partitions S and T, direct the edges from S to T, and then add extra edges between the nodes of S to make a binomial arborescence spanning S rooted at some node $v \in S$. Then the number of tournaments that v can win is exactly the number of perfect matchings in the original graph.

Because of the above proposition, we do not expect a subexponential time algorithm for the problem. We now present a $2^n \text{poly}\, n$ time algorithm, improving on a result from [Aziz *et al.*, 2014]. We begin with the following fact.

Fact 1. *Let $f_v(S)$ be the number of spanning binomial arborescences rooted at node v in the subgraph of G induced by $S \subset V$. Then*

$$f_v(S) = \sum_{u \in \mathcal{N}_{out}(v)} \sum_{\substack{T \subseteq S \setminus \{v\} \\ |T| = |S|/2 \\ u \in T}} f_u(T) \cdot f_v(S \setminus T).$$

We will use the following lemma, the proof of which follows [Björklund *et al.*, 2007] closely and will appear in the full version of the paper.

Lemma 2. *Let $k = 2^i$ for some $i \geq 1$. Let f and g be functions taking $k/2$-sized subsets of $[n]$ to the integers. Let h be a function taking k-sized subsets of $[n]$ to the integers defined as follows.*

$$h(S) = \sum_{\substack{T \subseteq S \\ |T| = k/2}} f(T) \cdot g(S \setminus T)$$

Then, if given $O(1)$ time access to evaluating f and g, one can compute $f(S)$ for all $S \subseteq [n], |S| = k$ in time $O(k \cdot 2^n)$.

Theorem 7. *There is an $O(2^n \text{poly}\, n)$ time algorithm that computes for any given directed graph $G = (V, E)$ for which $(u, v) \in E$ implies $(v, u) \notin E$, and any $a \in V$, the number of spanning binomial arborescences of G rooted at a.*

Proof. Based on Fact 1, we define for each $i = 1, \ldots, \log n$, $j = 1, \ldots, n$, functions f_{ij}, g_{ij} that take as input a subset $S \subseteq [n]$ and return an integer. If $|S| \neq 2^i$, then $f_{ij}(S) = g_{ij}(S) = 0$. If $|S| = 2^i$:

$$f_{ij}(S) = \sum_{k \in \mathcal{N}_{out}(j)} \sum_{U \subseteq S} f_{(i-1)j}(U) \cdot f_{(i-1)k}(S \setminus U).$$

$f_{0v}(\{v\}) = 1$ and $f_{0v}(S) = 0$ if $S \neq \{v\}$.

Intuitively, $f_{iv}(S)$ represents the number of spanning binomial arborescences rooted at v in the graph induced by S when $|S| = 2^i$, and where V is identified with $[n]$.

Define g_{ij} as follows. If $|S| \neq 2^i$ or if $j \in S$, $g_{ij}(S) = 0$. Otherwise, $g_{ij}(S) = \sum_{k \in \mathcal{N}_{out}(j)} f_{ik}(S)$. That is, $g_{ij}(S)$ computes the number of binomial arborescences spanning S rooted at a neighbor of $j \notin S$. Thus, for any S with $|S| = 2^i$, we can write

$$f_{ij}(S) = \sum_{\substack{U \subseteq S \\ |U| = |S|/2}} f_{(i-1)j}(U) \cdot g_{(i-1)j}(S \setminus U).$$

The quantity we want to compute is $f_{(\log n)a}(V)$.

Fix i. Given $f_{ik}(S)$ for all S of size 2^i, one can compute all $g_{ij}(S)$ (for sets S of size 2^i and nodes j) in time $O(n2^i2^n)$: go through all $O(n2^n)$ pairs (S, j) and add $f_{ik}(S)$ to $g_{ij}(S)$ for all of the $\leq |S| \leq 2^i$ nodes $k \in S$ for which $(j, k) \in E$.

By Lemma 2, for each fixed i, j, if we are given $f_{(i-1)j}(S')$ and $g_{(i-1)j}(S')$ for all sets S' of size 2^{i-1}, we can compute $f_{ij}(S)$ on all sets S of size 2^i in $O(2^i2^n)$ operations.

To summarize, the algorithm goes through all i from 1 to $\log n$, and through j from 1 to n. For each fixed i, j, it computes $f_{ij}(S)$ and $g_{ij}(S)$ for all sets S of size 2^i, given the values of $f_{(i-1)j}(S')$ and $g_{(i-1)j}(S')$ for all sets S' of size 2^{i-1}, in time $O(2^i2^n)$, using Lemma 2. Then the output is $f_{(\log n)a}(V)$. The runtime is asymptotically $n2^n \sum_{i=1}^{\log n} 2^i \leq O(n^2 2^n)$. The integers that the algorithm computes with, however, can be as big as $n!$ – the number of binomial arborescences there can be in a graph. This adds an extra poly $\log(n!) \leq$ poly n factor to the running time, as operations on B bit integers take no more than poly B time. □

5 Conclusion and Future Work

In this work, we answer a number of open questions related to the complexity of TFP. We exhibit a new set of conditions that are sufficient to guarantee a player is an SE winner; moreover, we give an improvement to the best-known algorithm for calculating the number of seedings that a given player wins in an arbitrary tournament.

We also introduce two previously-unstudied variants of TFP – where the organizer is allowed to bribe players to throw their matches and where strong players may tend to choke against weaker players. While our results provide an initial understanding of these variants, the models open the door for interesting future work. In particular, studying both of these variants in probabilistic versions of TFP might provide further insight into how bribery and choking affect the manipulation of real-world tournaments.

Acknowledgments

We thank the anonymous reviewers for their comments.

References

[Aziz *et al.*, 2014] Haris Aziz, Serge Gaspers, Simon Mackenzie, and Nicholas Mattei. Fixing a balanced knockout tournament. In *Proc. AAAI*, pages 552–558, 2014.

[Bartholdi *et al.*, 1989] J. Bartholdi, C. Tovey, and M. Trick. The computational difficulty of manipulating an election. *Social Choice Welfare*, 6:227–241, 1989.

[Bartholdi *et al.*, 1992] J. Bartholdi, C. Tovey, and M. Trick. How hard is it to control an election? *Mathematical and Computer Modeling*, 16(8/9):27–40, 1992.

[Baumeister, 1984] Roy F Baumeister. Choking under pressure: self-consciousness and paradoxical effects of incentives on skillful performance. *Journal of personality and social psychology*, 46(3):610, 1984.

[Beilock and Carr, 2001] Sian L Beilock and Thomas H Carr. On the fragility of skilled performance: What governs choking under pressure? *Journal of experimental psychology: General*, 130(4):701, 2001.

[Björklund *et al.*, 2007] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. Fourier meets Möbius: fast subset convolution. In *Proc. STOC*, pages 67–74, 2007.

[Braverman and Mossel, 2008] M. Braverman and E. Mossel. Noisy sorting without resampling. In *Proc. SODA*, 2008.

[Hazon *et al.*, 2007] N. Hazon, P. E. Dunne, and M. Wooldridge. How to rig an election. In *Proc. BISFAI*, 2007.

[Hazon *et al.*, 2008] N. Hazon, P.E. Dunne, S. Kraus, and M. Wooldridge. How to rig elections and competitions. In *Proc. COMSOC*, 2008.

[Lang *et al.*, 2007] J. Lang, M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Winner determination in sequential majority voting. In *Proc. IJCAI*, pages 1372–1377, 2007.

[Lita, 2008] D. Lita. Method and apparatus for managing billiard tournaments. *US Patent App.*, 20080269925, Oct 2008.

[Stanton and Vassilevska Williams, 2011a] I. Stanton and V. Vassilevska Williams. Manipulating stochastically generated single-elimination tournaments for nearly all players. In *Proc. WINE*, pages 326–337, 2011.

[Stanton and Vassilevska Williams, 2011b] I. Stanton and V. Vassilevska Williams. Rigging tournament brackets for weaker players. In *Proc. IJCAI*, pages 357–364, 2011.

[Vassilevska Williams, 2010] V. Vassilevska Williams. Fixing a tournament. In *Proc. AAAI*, pages 895–900, 2010.

[Vu *et al.*, 2009] T. Vu, A. Altman, and Y. Shoham. On the complexity of schedule control problems for knockout tournaments. In *Proc. AAMAS*, pages 225–232, 2009.