Fixing a Balanced Knockout Tournament

Haris Aziz and **Serge Gaspers** and **Simon Mackenzie** and **Nicholas Mattei**
NICTA and UNSW, Sydney, Australia
{haris.aziz, serge.gaspers, simon.mackenzie, nicholas.mattei}@nicta.com.au

Paul Stursberg
Technische Universität München, Germany
paul.stursberg@ma.tum.de

Toby Walsh
NICTA and UNSW, Sydney, Australia
toby.walsh@nicta.com.au

Abstract

Balanced knockout tournaments are one of the most common formats for sports competitions, and are also used in elections and decision-making. We consider the computational problem of finding the optimal draw for a particular player in such a tournament. The problem has generated considerable research within AI in recent years. We prove that checking whether there exists a draw in which a player wins is NP-complete, thereby settling an outstanding open problem. Our main result has a number of interesting implications on related counting and approximation problems. We present a memoization-based algorithm for the problem that is faster than previous approaches. Moreover, we highlight two natural cases that can be solved in polynomial time. All of our results also hold for the more general problem of counting the number of draws in which a given player is the winner.

Introduction

Balanced knockout tournaments are one of the most widely-used formats for sports competitions (Horen and Riezman 1985; Connolly and Rendleman 2011; Groh et al. 2012). A prominent example is the Wimbledon Men's tennis tournament in which 128 players enter the tournament and the player who wins seven consecutive matches right from the first round to the final wins the tournament. The format is also used in certain elimination style election and decision making schemes and has received considerable interest in AI (Vu, Altman, and Shoham 2009; Vassilevska Williams 2010) as well as economics and operations research (Rosen 1986; Vu et al. 2013; Tullock 1980; Laslier 1997).

Consider the setting in which there is a set of players $N = [n]$ (we use the notation $[n] := \{1, \ldots, n\}$) where $n = 2^c$ for some integer c. Given N, an *ordered balanced knockout tournament* $\mathrm{T}(N, \pi)$ is defined as a balanced binary tree with n leaf nodes where the *seeding* π specifies the labelling of the leaf nodes with respect to N. All ordered balanced knockout tournaments that are isomorphic to each other (with respect to the labelling of the leaf nodes) are said to have the same *draw*. They are represented by a single (unordered) *balanced knockout tournament (BKT)* $\mathrm{T}(N, \sigma)$ where σ denotes the draw. The set of all draws is denoted by Σ. Whereas the total number of seedings is $n!$, the number of draws is $\frac{n!}{2^{n-1}}$ but even this grows very rapidly. For a tournament like Wimbledon, $n = 128$ and the number of distinct draws is $2.3 * 10^{177}$. This is significantly more than the number of atoms in the universe, or even a googol.

A BKT $\mathrm{T}(N, \sigma)$ is conducted in the following fashion. Players that correspond to sibling leaf nodes play a *match* against each other. The winner of the match proceeds up the tree to the next *round*. The *winner of* $\mathrm{T}(N, \sigma)$ is the player who reaches the root node. We are given a *pairwise comparison matrix* P such that $P_{jk} \in [0, 1]$ denotes the probability of player j beating player k in a pairwise elimination match and $0 \leq P_{ij} = 1 - P_{ji} \leq 1$. Given N, P and a draw σ, each player $i \in N$ has a certain probability $\mathrm{wp}(i, N, P, \sigma)$ of being the winner of $\mathrm{T}(N, \sigma)$. This probability can be computed in time $O(n^2)$ via a recursive formulation (Vu, Altman, and Shoham 2009). We denote by $\mathrm{mwp}(i, N, P) := \max_{\sigma \in \Sigma}(\mathrm{wp}(i, N, P, \sigma))$ the maximum possible winning probability of i in $\mathrm{T}(N, \sigma)$ taken over all draws $\sigma \in \Sigma$.

We consider the PROBABILISTIC TOURNAMENT FIXING PROBLEM (PTFP) in which the probability of each player beating another player is known and the goal is to find a draw that maximizes the probability of a certain player winning the BKT.

PROBABILISTIC TOURNAMENT FIXING PROBLEM (PTFP)
Instance: Player set N, pairwise comparison matrix P, a distinguished player $i^* \in N$, and target probability $q \in [0, 1]$.
Question: Does there exist a draw σ for the player set N for which the probability of i^* winning $\mathrm{T}(N, \sigma)$ is at least q?

PTFP was proposed by Vu, Altman, and Shoham (2009) and has been studied in numerous papers (see e.g., (Stanton and Vassilevska Williams 2011a; 2011c; 2011b)). It is a well-motivated problem in sports analytics (Silver 2011). Vu, Altman, and Shoham (2009) and Vassilevska Williams (2010) showed that PTFP is NP-hard for various restrictions. These results also imply that computing the maximum possible winning probability of a given player is NP-hard. Nevertheless, the computational complexity of a particularly natural and interesting problem,

the TOURNAMENT FIXING PROBLEM (TFP), has remained a major open question. In the TFP, P is *deterministic* i.e., $P_{jk} \in \{0,1\}$ for all players $j,k \in N$. We say that player j *beats* player k iff $P_{jk} = 1$. The winner of each match is deterministically known beforehand and the question is whether there exists a draw for which a given player can win in the corresponding BKT.

TOURNAMENT FIXING PROBLEM (TFP)
Instance: Player set N, deterministic pairwise comparison matrix P, and a distinguished player $i^* \in N$.
Question: Does there exist a draw σ for the player set N for which i^* is the winner of $\mathrm{T}(N,\sigma)$?

TFP is equivalent to checking whether there exists a seeding π for which i is the winner of $\mathrm{T}(N,\pi)$. We note that TFP is a special case of the problem with same name as defined in (Vassilevska Williams 2010).[1] TFP is also a special case of #TFP — the problem of *counting* the number of draws for which a given player is the winner. This count can be used to compute the probability of a player winning in a draw chosen uniformly at random. It can also be considered as the relative strength of the player.

Contributions. We first settle the computational complexity of TFP by showing that it is NP-complete. The problem was explicitly stated as an open problem a number of times (Vu, Altman, and Shoham 2009; Vassilevska Williams 2010; Russell and van Beek 2011; Stanton and Vassilevska Williams 2011b; 2011a; 2011c; Lang et al. 2012; Vu et al. 2013). As a corollary, we show that unless P = NP, there exists no polynomial-time approximation algorithm for computing the maximal winning probability of a player. The inapproximability result provides additional motivation for the line of work in which heuristic algorithms have been proposed for PTFP (Vu et al. 2013). Another corollary is that there exists no *fully polynomial time randomized approximation scheme* (FPRAS) for counting the number of draws for which a player is the winner.

In view of these intractability results, we identify two natural cases for which even #TFP can be solved in polynomial time. In the first case, the players can be divided into a constant number of player types. It appeals to the scenario where players can be divided into groups based on similar intrinsic ability. In the second case, there is a linear ordering on the ability of players with a constant number of exceptions where a player with lower ability beats a player with higher ability.[2] Finally, we provide an exact memoization-based algorithm to solve #TFP that is faster than known exact approaches to solve the problem: it runs in time $O(2.8285^n)$ and uses space $O(1.7548^n)$. If only polynomial space is available, the running time becomes $4^{n+o(n)}$, and we give a range of possible time-space trade-offs.

Related Work. After the work of Vu, Altman, and Shoham (2009), PTFP and TFP have been studied in a number of AI papers. Vassilevska Williams (2010) identified various sufficient conditions for a player to be a winner of a BKT. In a followup paper, Stanton and Vassilevska Williams (2011c) focused on when weak players can possibly win a BKT. Stanton and Vassilevska Williams (2011b; 2011a) identified conditions in a probabilistic model under which the tournament organizer can fix the tournament with high probability. In (Vu and Shoham 2011), the problem of designing 'fair' draws was considered. Lang et al. (2007) and Lang et al. (2012) examined winner determination in voting trees that need not be balanced.

TFP can also be considered as the problem of checking whether a player is a *possible winner* in a BKT. Computing possible winner for other voting rules where the information on the preferences is not complete has been studied extensively (Xia and Conitzer 2011; Aziz et al. 2012). Another related problem is checking whether a sports team can still win a round-robin competition when all the matches have not yet been completed (Kern and Paulusma 2004; Gusfield and Martel 2002).

TFP is NP-complete

In this section, we settle the complexity of TFP. For convenience, we will represent the pairwise comparison matrix P as a graph where an edge from i to j exists iff i beats j.

Theorem 1. *TFP is NP-complete.*

Proof. We reduce from the NP-hard variant of the 3SAT problem in which every literal appears at most twice. Given such a 3SAT instance $F = (X, C)$ where $X = \{x_1, \ldots, x_{|X|}\}$ is the set of variables and C the set of clauses, we build an instance of TFP where a draw exists such that a distinguished player m_1 wins iff F is satisfiable. The TFP instance consists of a set of players $N = \{1, \ldots, n\}$ where n is the smallest power of 2 greater than $32 \cdot |X|$. The resulting knock-out tournament will thus consist of $R := \lceil \log(32 \cdot |X|) \rceil - 1 \geq 5$ rounds where the first (lowest) four rounds will be used to store the gadgets while the later rounds will enforce certain outcomes for the gadgets. We can decompose the set of players N as follows

$$N = M \dot\cup S \dot\cup G^{\mathrm{X}} \dot\cup G^{\mathrm{CG}} \dot\cup G^{\mathrm{F}} \tag{1}$$

where players in G^{X} are used in the *choice gadgets* that will model the variable assignment, players in G^{CG} are used in the *clause/garbage gadgets* that will model the behaviour of the clauses, and players in G^{F} are used in *filler gadgets* that will be used to balance the BKT. Players in S are *special players* that will ensure the connection between choice and clause gadgets. Finally we will show that for m_1 to win the BKT, all $k := \frac{n}{16}$ players in $M = \{m_1, m_2, \ldots, m_k\}$ will have to proceed to the fifth round. We will use a total of

[1] In the version of TFP defined in (Vassilevska Williams 2010), there can be additional constraints in which certain matches are disallowed.

[2] The condition is quite natural since in many competitions there is a clear-cut ranking of the players according to their skills with only a few pairs of players for which the weaker player can beat the stronger player. For example, as of 15/01/2014, Nikolay Davydenko was the *only* tennis player among the men's top 64 who had a winning head-to-head record against Rafael Nadal.

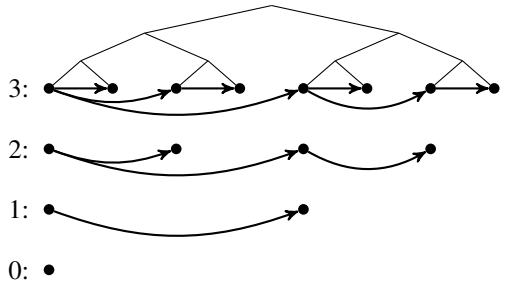

Figure 1: The spawning process. By the shown draw, the leftmost player can win the BKT.

k gadgets where each m_i will be associated with one particular gadget. This gadget will ensure that m_i can proceed to the fifth round. More precisely, we use $|X|$ choice gadgets numbered from 1 to $|X|$, $|X|$ clause/garbage gadgets (that may each contain multiple clauses) numbered from $|X|+1$ to $2|X|$ and $(k-2|X|)$ filler gadgets numbered from $(2|X|+1)$ to k. Using this numbering, we can further partition the above sets as follows:

$$G^{\mathrm{X}} = \dot{\bigcup_{j=1}^{|X|}} G_j; \qquad G^{\mathrm{CG}} = \dot{\bigcup_{j=|X|+1}^{2|X|}} G_j; \qquad G^{\mathrm{F}} = \dot{\bigcup_{j=2|X|+1}^{k}} G_j \qquad (2)$$

Note that the sets G_j have 10, 13 or 15 elements, depending on whether they are a subset of G^{X}, G^{CG} or G^{F}, respectively.

Set M. The relation between elements of the set of winners M is recursively defined via a linear ordering of players as follows: We start with player m_1. At each iteration, every player a *spawns* a new player b placed directly to his right. In the pairwise comparison graph, each player beats all players to his left in this construction except for the one that spawned him. This recursive construction is repeated until a total of k players are present (see Figure 1).

Lemma 1. *There is a draw σ such that m_1 wins the BKT $T(M,\sigma)$.*

Proof. Seeding all players in M from left to right according to the spawning process makes m_1 win the tournament (see Figure 1): Whenever two players meet, the left player in the match has spawned the right player, thus the left player wins. As the leftmost player, m_1 wins the tournament. □

Global structure. We now describe how the sets from (1) and (2) relate to each other. In many places, we will use the *right-left-rule*, that is elements from sets with a higher index will beat elements from sets with a lower index. For instance, for all $j > j'$ and elements $i \in \{m_j\} \cup G_j$, $i' \in \{m_{j'}\} \cup G_j$ where $(i,i') \neq (m_j, m_{j'})$ we have that i beats i'. All members of S beat all other players unless explicitly stated otherwise. The set S can be partitioned into subsets S_j corresponding to each variable of the SAT instance, such that $\forall j \in [|X|] : S_j = S_{x_j} \dot\cup S_{\bar{x}_j} \dot\cup \{s_j^*\}$ where

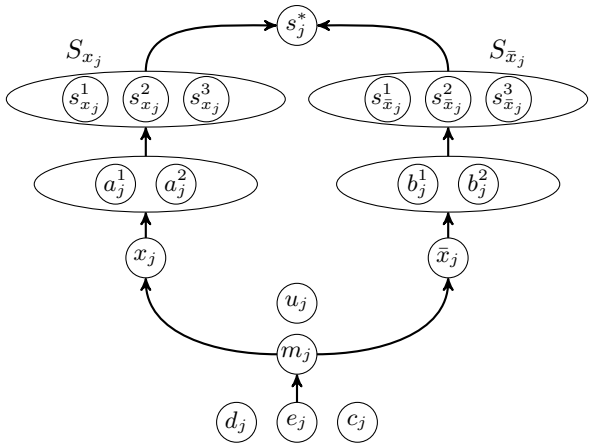

Figure 2: Pairwise comparisons in the j-th choice gadget. All arrows not shown in the figure run downwards, horizontal arcs run right to left. Vertices grouped in a component have the same relation with vertices outside the component.

$|S_{x_j}| = 3$ and $|S_{\bar{x}_j}| = 3$. We further define the set of *particles* (these will move between the choice and clause/garbage gadgets) by $S_p := \bigcup_{j\in[|X|]}(S_{x_j} \dot\cup S_{\bar{x}_j})$. The members of S_p follow the right-left rule between each other, i.e., for $j > j'$, elements from S_{x_j} and $S_{\bar{x}_j}$ beat elements from $S_{x_{j'}}$ and $S_{\bar{x}_{j'}}$. The players in $\{s_j^* \mid j \in [|X|]\}$ follow the right-left rule amongst themselves. For each $j \in [|X|]$, s_j^* is beaten by all other members of S_j and beats all members of $S_p \setminus S_j$.

Choice gadget. For each $j \in [|X|]$, the j-th choice gadget consists of player m_j, all ten elements of G_j and all of S_j. Note that some elements of S_j will appear again in the clause/garbage gadgets. The pairwise comparison graph for these elements (for fixed j) is shown in Figure 2. The choice gadget is structured in such a way that it is possible for m_j to win a subtournament composed of all elements in the gadget except two elements of either S_{x_j} or $S_{\bar{x}_j}$, as illustrated in Figure 4. We will also show that this is the only way in which m_j can reach the fifth round in a tournament won by m_1.

Clause/garbage gadget. We now describe the internal structure of the clause/garbage gadgets. The j-th gadget consists of m_j and the 13 elements of G_j, two of these are denoted c/g. The pairwise comparison graph for these players is shown in Figure 3. For each clause $c_i \in C$ we will call one of the players denoted c/g *associated with clause* c_i, all remaining players c/g are *garbage players*. All players shown in the figure are beaten by all players in S with the following exceptions: (i) garbage players beat all players from S_p, (ii) players associated with clause c_i beat all players from the set S_{x_j} or $S_{\bar{x}_j}$ if x_j or $\bar{x}_j$ occurs in clause c_i, respectively.

The clause gadget is structured in such a way that it is possible for m_j to win a subtournament composed of all elements in the gadget with the addition of a compatible S_p element for both clause/garbage players included. We will

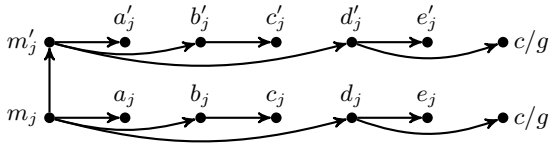

Figure 3: Pairwise comparison graph for a clause/garbage gadget. All arrows not shown in the figure run downwards, horizontal arcs run right to left.

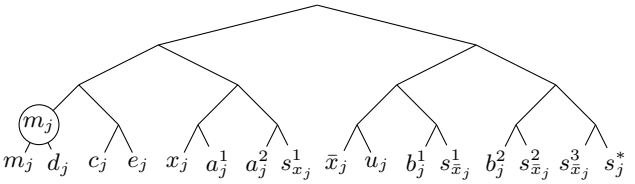

Figure 4: The subtournament for choice gadget j. In this case, x_j=true is selected.

also show that if we want m_1 to win the tournament, this is the only way in which m_j can reach the fifth round.

The filler gadget j consists of m_j and the players in set G_j. To create the pairwise comparison graph relating them to one another, we use the same method of spawning new players as for M, starting from player m_j (see Figure 1).

We note that for all $j \in [k]$, player m_j beats exactly four players from the set G_j.

Lemma 2. *For all $j \in [k]$ the following hold:*

a) Let σ be a draw where $\forall i < j$, player m_i reaches the fifth round by winning a 16-player subtournament that contains all players from G_i and m_j plays against all four players from G_j that he beats. Then, the 16-player subtournament by which m_j proceeds to the fifth round consists exactly of:

i) if the j-th gadget is a choice gadget: $\{m_j\} \dot\cup G_j \dot\cup S_j$ where two elements of either $S_{x_j} \dot\cup \{s_j^*\}$ or of $S_{\bar{x}_j} \dot\cup \{s_j^*\}$ are removed,

ii) if the j-th gadget is a clause/garbage gadget: $\{m_j\} \dot\cup G_j$ and for each of the c/g players in G_j one additional element of S_p that he can beat,

iii) if the j-th gadget is a filler gadget: $\{m_j\} \dot\cup G_j$.

b) If the 16-player subtournament for the first four rounds in which m_j is placed consists exactly of i), ii), or iii) for the respective type of the j-th gadget, then a draw for the subtournament exists by which m_j reaches the fifth round.

The proof of Lemma 2 is easy but lengthy and thus omitted. Figure 4 shows the relevant subtournament for a choice gadget. We can now show the following, which will also show that for all $j \in [k]$ the antecedent of Lemma 2a) is satisfied and thus the consequent holds.

Lemma 3. *For m_1 to win the tournament, all players $m_j \in M$ must reach the fifth round, winning a 16-player subtournament that contains all players from G_j by playing against all four players from G_j that they can beat.*

Proof. Let M_ℓ denote the set of 2^ℓ players from M that are present after the ℓ-th iteration of the spawning process. Furthermore, denote by ℓ_j the smallest ℓ such that $m_j \in M_\ell$. We use induction over the set M. The induction will proceed from left to right according to the order specified in the description of the spawning process. First, note that for all $j \in [k]$, player m_j beats exactly $(R-4-\ell_j)$ players from M that are to his right. The induction hypothesis is as follows.

> *In order for m_1 to win the tournament, the following must hold for all $j \in [k]$:*
> *(i) In some round, m_j plays against the player that spawned him, (ii) m_j must reach round $(R+1-\ell_j)$ and be beaten in that round, (iii) m_j plays against all players from G_j that he can beat, (iv) the subtournament consisting of the first four rounds that m_j wins contains all players from G_j.*

For the base case, consider m_1. For *(i)* and *(ii)* there is nothing to show, as m_1 must win the tournament. As m_1 only beats a total of R players, he needs to play against all of these, which proves Part *(iii)*. Part *(iv)* is implied by Lemma 2.

We now show that for a player $m_j \in M$, the induction hypothesis holds, provided that it holds for all players $m_{j'} \in M$ with $j' < j$. Consider the player m_{j^*} that spawned m_j.

For *(i)*, note that by the induction hypothesis, all players from G'_j with $j' < j^*$ have to be placed in the subtournament won by $m_{j'}$. Similarly, all players $m_{j'}$ with $j' < j^*$ have to play (and lose) against the player that spawned them, therefore they cannot play against m_{j^*} (as they would lose). Thus, m_{j^*} can only play against players from G_{j^*} and players m_r from M to m_{j^*}'s right and the respective sets G_r. He beats exactly four players in G_{j^*}, none in the other G_j and $(R-4-\ell_{j^*})$ from M to his right. By the induction hypothesis, he thus needs to play against all of these (including m_j) to proceed to round $(R+1-\ell_{j^*})$.

For *(ii)*, note that to the left of m_j there are $(R-4-\ell_j)$ other players that are spawned by m_{j^*}. Denote them by $m_{j_1^*}, m_{j_2^*}, \ldots, m_{j^*_{R-4-\ell_j}}$ and note that $R+1-\ell_{j_i^*} = i+4$. Thus, m_j cannot face m_{j^*} in round $5, 6, \ldots, R-\ell_j$ (as other players need to play against m_{j^*} in these rounds) and has to proceed at least until round $(R+1-\ell_j)$. By the same argument as for m_{j^*} above, before m_j faces m_{j^*}, he can only play against players from G_j and players m_r from M to m_j's right and the respective sets G_r. Player m_j only beats a total of $R-\ell_j$ players among these, thus m_j must be beaten in round $R-\ell_j$. Furthermore, he must play against all of the above players that he can beat, which settles Part *(iii)*.

Again, Part *(iv)* is implied by Lemma 2 as by the argument above, m_j needs to play against all four players from G_j that he can beat. □

We can now finally show the following lemma.

Lemma 4. *There exists a draw such that m_1 wins the tournament if and only if the SAT instance is satisfiable.*

Proof. First, note that by Lemma 1 and Lemma 3, m_1 can win the tournament iff all players from the set M reach the

fifth round. As only $|M|$ players can reach the fifth round in total, m_1 can win the tournament iff each player $m_j \in M$ can win a 16-player subtournament that consists exactly of the players stated in Lemma 2. We thus have to show that a draw in which each player $m_j \in M$ wins a 16-player subtournament with exactly the players stated in Lemma 2 exists iff the 3SAT instance is satisfiable.

($\Leftarrow$) Let the formula F be satisfied by a truth assignment A. We construct a BKT over N in which m_1 wins. For all j where the j-th gadget is a choice gadget, we place all players from $\{m_j\} \dot\cup G_j \dot\cup S_j$ with the exception of two players from S_{x_j} (if $A(x_j) =$ true) or $S_{\bar{x}_j}$ (if $A(x_j) =$ false) in one subtournament for the first four rounds. These are exactly 16 players and by Lemma 2b), we can choose the draw in such a way that m_j wins the subtournament.

Note that the choice gadgets use all players from S except for two players of each set S_{x_j} (or $S_{\bar{x}_j}$) where x_j (or $\bar{x}_j$) is a literal that evaluates to true. In other words, for every literal that evaluates to true, two players remain that can be beaten by every player associated with a clause in which the literal players appear.

For the clause/garbage gadgets, to use Lemma 2b) we need for each c/g player one additional player that he beats. If c/g is associated with a clause $c_i \in C$, pick one of the literals in c_i that evaluates to true, say $\bar{x}_j$, and assign one of the two players from $S_{\bar{x}_j}$ mentioned above to this c/g player. Note that at most two such players are required for any literal, as every literal only appears twice. To all garbage players, assign an arbitrary remaining player from set S. Note that, after composing the subtournaments for all choice gadgets, $2|X|$ players from the set S were left. As there are $2|X|$ c/g players, we can assign a player from S to each of these. For all j where the j-th gadget is a clause/garbage gadget, we now place all players from $\{m_j\} \dot\cup G_j$ in one subtournament for the first four rounds, together with the two players that are assigned to the c/g players in G_j. By Lemma 2b), we can chose a draw such that m_j wins this subtournament. For all j where the j-th gadget is a filler gadget, we place all players from $\{m_j\} \dot\cup G_j$ in one tournament for which, by Lemma 2b), we can find a draw where m_j wins.

($\Rightarrow$) Let a draw be given such that all players in M win a 16-player subtournament consisting exactly of the players stated in Lemma 2. First, note that whenever a player s_j^* is not placed in the j-th choice gadget, we get a contradiction as by Lemma 2, this player cannot be placed anywhere else while all players from M still reach the fifth round. The following truth assignment is thus well-defined: $A(x_j) =$ true if m_j's subtournament includes all players from S_j except two players from S_{x_j}, $A(x_j) =$ false if m_j's subtournament includes all players from S_j except two players from $S_{\bar{x}_j}$

We now show that A satisfies all clauses. Let $c_i \in C$ be a clause and c/g the player associated with that clause. Let j^* be the 16-player subtournament including c/g. By the assumption, m_{j^*} wins the subtournament, thus Lemma 2 implies that it contains a player that c/g beats, i.e. a player from set S_{x_j} or ($S_{\bar{x}_j}$) where x_j or ($\bar{x}_j$) occurs in clause C_i. By the definition of A, that player can only be used in the subtournament if the corresponding literal evaluates to true (as otherwise, the player is used in m_j's subtournament). □

This concludes the proof of the theorem. □

Implications of Theorem 1

Theorem 1 simultaneously implies a number of results from the literature, in particular Theorem 1, 2, and 3 in (Vu, Altman, and Shoham 2009) and Theorem 1 in (Vassilevska Williams 2010). It also yields some further corollaries. For PTFP and $\alpha \geq 1$, an algorithm is called an *α-approximation algorithm for the maximum winning probability* if it can find a draw in which the winning probability of the given player i is at least $\mathrm{mwp}(i, N, P)/\alpha$.

Corollary 1. *Unless P = NP, there exists no polynomial-time algorithm for PTFP that approximates the maximum winning probability of a player within any given factor.*

It immediately follows from Theorem 1 that #TFP is NP-hard. We next highlight that even randomisation is not very helpful for #TFP. Let Γ be a finite alphabet in which we agree to describe our problem instances and solutions. A *fully polynomial time randomized approximation scheme (FPRAS)* for a function $f : \Gamma^* \to \mathbb{Q}$ is an algorithm that takes input $x \in \Gamma^*$ and a parameter $\epsilon \in \mathbb{Q}_{>0}$, and returns with probability at least $\frac{3}{4}$ a number between $f(x)/(1+\epsilon)$ and $(1+\epsilon)f(x)$. Moreover, an FPRAS runs in time polynomial in the size of x and $1/\epsilon$. RP is the complexity class consisting of problems that can be solved in randomized polynomial time. The statement below follows from Proposition 8 in (Welsh and Merino 2000) and Theorem 1.

Corollary 2. *Unless NP = RP, there is no FPRAS for #TFP.*

Hence, unless NP = RP, there also does not exist an FPRAS for computing the probability of a player winning a BKT for a draw chosen uniformly at random.

Tractable Cases

We first examine a natural case in which players are divided into player *types*: All players of one type beat exactly the same subset of players of other types. The result of matches between players of the same type is irrelevant as we do not care which player in each type wins. In this variant that we call #TFP-types, the objective is to count the number of draws for which a player of a given *type* wins. We first adapt the concept of the pairwise comparison matrix. For two player types i and j, define $P_{ij} = 1$ if an i-player wins a match between i and j and $P_{ij} = 0$ otherwise. Note that this definition is chosen such that $P_{ii} = 1$ for all player types i.

Theorem 2. *#TFP-types is polynomial-time solvable if there are a constant number of player types.*

Proof. We will use a dynamic programming scheme. All vectorial inequalities assume equal dimensions and are meant component-wise. Let $N_x := \{\eta = (\eta_1, \ldots, \eta_k) \geq 0 \mid \eta_1 + \cdots + \eta_k = 2^x\}$ and denote by $\#\mathrm{TFP}(i, x, \eta \in N_x)$ the number of draws for an x-round tournament involving η_j players of type j for all $j \in [k]$ in which a player of type i wins the tournament. We will assume that the players of one type are not distinguishable. For fixed i and x, there are

$\binom{k+2^x}{k}$ such problems that potentially need to be considered. $\#\mathrm{TFP}(i,x,\eta)$ is computed via the following recursion.

$$\#\mathrm{TFP}(i,x,\eta) = \sum_{\substack{\eta' \in N_{x-1} \\ \eta' \leq \eta}} \#\mathrm{TFP}(i,x-1,\eta') \cdot \sum_{\substack{j \in [k] \\ P_{ij}=1 \\ \eta'_j < \eta_j}} \#\mathrm{TFP}(j,x-1,\eta-\eta') \quad (3)$$

The base cases are given by

$$\#\mathrm{TFP}(i,0,\eta) = \begin{cases} 1 & \eta_i = 1 \text{ and } \forall j \neq i : \eta_j = 0 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

for all $i \in [k]$. Eq. (3) only uses values of $\#\mathrm{TFP}(j,y,\eta')$ with $y < x$, thus every problem only needs to be solved once. For constant k, solving a problem requires $O(n^k)$ operations and for constant k it holds that $\binom{k+2^x}{k} \in O(n^k)$. Thus, $O(\log(n)n^{2k})$ operations are necessary to compute $\#\mathrm{TFP}(i,\log(n),(n_1,\ldots,n_k))$. □

The second tractable case that we identify is:

Theorem 3. *#TFP is polynomial-time solvable if there is a linear ordering of strengths of the players with at most a constant number of pairwise relations flipped.*

Proof. Let b be the number of backwards arcs that do not respect the linear ordering. We can show by induction that the problem can be reduced to #TFP-types with at most $4b+3$ player types. Since the number of player types is constant, the theorem follows from Theorem 2. □

An Exponential Time Algorithm for #TFP

#TFP can be trivially solved in time $2^{O(n \log n)}$ via a brute-force enumeration of all possible draws. In exponential time algorithmics (Fomin and Kratsch 2010), the aim is to design algorithms solving the problem exactly with worst-case running times outperforming the brute-force solution. In this section we give an algorithm for #TFP running in time $O(2.8285^n)$ using space $O(1.7548^n)$. If only polynomial space is available, the running time becomes $4^{n+o(n)}$, and we give a range of possible time-space trade-offs. The algorithm is based on the recursion formula (3) and memoization used at various levels of the recursion. We use $\mathsf{poly}(n)$ to stand for a polynomial function in n.

Theorem 4. *For every y, $2 \leq y \leq \log n$, there is an algorithm solving #TFP in time $T(n) = \mathsf{poly}(n) \cdot \prod_{p=0}^{y-1} \frac{n}{2^p} \cdot 2^{n/2^p}$ and space $S(n) = \mathsf{poly}(n) \cdot (2^y)^{n/2^y} \cdot \left(\frac{2^y}{2^y-1}\right)^{n-n/2^y}$.*

Proof Sketch. The algorithm recursively evaluates the formula (3) for the special case where each player type consists of one player, starting from $\#\mathrm{TFP}(i,\log n,(1,\ldots,1))$. It uses memoization for all recursive calls $\#\mathrm{TFP}(\cdot,x,\cdot)$ where $x \leq (\log n) - y$. That is, the algorithm uses a table indexed by players, level, and player type vector. To evaluate $\#\mathrm{TFP}(\cdot,x,\cdot)$ with $x \leq (\log n) - y$, the algorithms first checks in this table whether this recursive call has already been evaluated. Only if the value has not yet been computed, it computes the result recursively, and stores it in the table. Then, it returns the result that is stored in the table. The number of table entries used by memoization is upper bounded by n times the number of subsets of size at most $n/2^y$. Using Stirling's approximation for factorials, the space usage of the algorithm can be upper bounded by $S(n) = \mathsf{poly}(n) \cdot (2^y)^{n/2^y} \cdot \left(\frac{2^y}{2^y-1}\right)^{n-n/2^y}$. The running time of the algorithm is the time used for the recursion without memoization, $\mathsf{poly}(n) \cdot \Pi_{p=0}^{y-1} \frac{n}{2^p} \cdot 2^{n/2^p}$, plus the time for the part with memoization, which is upper bounded by $S(n) \cdot n \cdot 2^{n/4} = O(2.0868^n)$. Thus, for any y, $2 \leq y \leq \log n$, we obtain an algorithm with running time $T(n) = \mathsf{poly}(n) \cdot \Pi_{p=0}^{y-1} \frac{n}{2^p} \cdot 2^{n/2^p}$ using space $S(n) = \mathsf{poly}(n) \cdot (2^y)^{n/2^y} \cdot \left(\frac{2^y}{2^y-1}\right)^{n-n/2^y}$. □

y	time $T(n)$	space $S(n)$
2	$O(2.8285^n)$	$O(1.7548^n)$
3	$O(3.3636^n)$	$O(1.4576^n)$
4	$O(3.6681^n)$	$O(1.2634^n)$
$\log n$	$4^{n+o(n)}$	$\mathsf{poly}(n)$

Table 1: Running times and space requirements for the algorithm of Theorem 4 for various time-space trade-offs.

Various time and space requirements of the algorithm are reported in Table 1. Using the rule of thumb that for current computing architectures, the space requirements of an algorithm become a bottleneck if they exceed the square root of the time requirements, the analyses for $y = 2$ and $y = 3$ currently seem the most relevant.

Conclusions

In this paper we considered problems related to tournament fixing. Although being able to change draws is not always realistic, the computational problems that are considered have been analyzed in post analysis of tournament draws and also shed light on the relative strengths of players. Our main result is that TFP is NP-complete. We discussed a number of implications of the result. We complement the computational hardness result in the paper by presenting algorithms for #TFP — both for the general case as well as restricted cases. A possible future direction is to propose parametrized algorithms for TFP.

Acknowledgments

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program. Serge Gaspers is the recipient of an Australian Research Council Discovery Early Career Researcher Award (project number DE120101761).

References

Aziz, H.; Brill, M.; Fischer, F.; Harrenstein, P.; Lang, J.; and Seedig, H. G. 2012. Possible and necessary winners of partial tournaments. In *Proceedings of the 11th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 585–592. IFAAMAS.

Connolly, R. A., and Rendleman, R. J. 2011. Tournament qualification, seeding and selection efficiency. Technical Report 2011-96, Tuck School of Business.

Fomin, F. V., and Kratsch, D. 2010. *Exact Exponential Algorithms*. Springer.

Groh, C.; Moldovanu, B.; Sela, A.; and Sunde, U. 2012. Optimal seedings in elimination tournaments. *Economic Theory* 49(1):59–80.

Gusfield, D., and Martel, C. 2002. The structure and complexity of sports elimination numbers. *Algorithmica* 32(1):73–86.

Horen, J., and Riezman, R. 1985. Comparing draws for single elimination tournaments. *Operations Research* 33(2):249–262.

Kern, W., and Paulusma, D. 2004. The computational complexity of the elimination problem in generalized sports competitions. *Discrete Optimization* 1(2):205–214.

Lang, J.; Pini, M. S.; Rossi, F.; Venable, K. B.; and Walsh, T. 2007. Winner determination in sequential majority voting. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, 1372–1377.

Lang, J.; Pini, M. S.; Rossi, F.; Salvagnin, D.; Venable, K. B.; and Walsh, T. 2012. Winner determination in voting trees with incomplete preferences and weighted votes. *Journal of Autonomous Agents and Multi-Agent Systems* 25(1):130–157.

Laslier, J.-F. 1997. *Tournament Solutions and Majority Voting*. Springer-Verlag.

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

Russell, T., and van Beek, P. 2011. An empirical study of seeding manipulations and their prevention. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI 2011)*, 350–356. IJCAI/AAAI.

Silver, N. 2011. When 15th is better than 8th: The math shows the bracket is backward. On *Five-ThirtyEight Blog*, http://fivethirtyeight.blogs.nytimes.com/2011/03/15/when-15th-is-better-than-8th-the-math-shows-the-bracket-is-backward/. Last Accessed Feb. 2014.

Stanton, I., and Vassilevska Williams, V. 2011a. Manipulating single-elimination tournaments in the Braverman-Mossel model. In *IJCAI Workshop on Social Choice and Artificial Intelligence*.

Stanton, I., and Vassilevska Williams, V. 2011b. Manipulating stochastically generated single-elimination tournaments for nearly all players. In *Proceedings of 7th International Workshop on Internet and Network Economics (WINE)*, 326–337.

Stanton, I., and Vassilevska Williams, V. 2011c. Rigging tournament brackets for weaker players. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, 357–364.

Tullock, G. 1980. *Toward a Theory of the Rent-seeking Society*. Texas A&M University Press.

Vassilevska Williams, V. 2010. Fixing a tournament. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, 895–900. AAAI Press.

Vu, T.; Altman, A.; and Shoham, Y. 2009. On the complexity of schedule control problems for knockout tournaments. In *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 225–232.

Vu, T., and Shoham, Y. 2011. Fair seeding in knockout tournaments. *ACM Transactions on Intelligent Systems Technology* 3(1):1–17.

Vu, T.; Hazon, N.; Altman, A.; Kraus, S.; Shoham, Y.; and Wooldridge, M. 2013. On the complexity of schedule control problems for knock-out tournaments.

Welsh, D. J. A., and Merino, C. 2000. The potts model and the tutte polynomial. *Journal of Mathematical Physics* 41(3):1127–1152.

Xia, L., and Conitzer, V. 2011. Determining possible and necessary winners under common voting rules given partial orders. *Journal of Artificial Intelligence Research* 41(2):25–67.