Artificial Intelligence 262 (2018) 1–14

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Fixing balanced knockout and double elimination tournaments ☆

Haris Aziz [a,*], Serge Gaspers [a], Simon Mackenzie [b], Nicholas Mattei [c], Paul Stursberg [d], Toby Walsh [a]

[a] Data61, CSIRO and UNSW Sydney, Australia
[b] Carnegie Mellon University, USA
[c] IBM Research, USA
[d] Technische Universität München, Germany

ARTICLE INFO

Article history:
Received 9 February 2017
Received in revised form 24 April 2018
Accepted 19 May 2018
Available online 25 May 2018

Keywords:
Knockout tournaments
Seedings of a tournament
Computational complexity

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. Finally, we show that our main NP-completeness result extends to a variant of balanced knockout tournaments called double-elimination tournaments.

1. Introduction

Balanced knockout tournaments are one of the most widely-used formats for sports competitions [7,10,13]. A prominent example is the Wimbledon Men's Singles 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 the field of artificial intelligence [12,16,17,23,34,36,37,27,1] as well as social sciences [5,21,22,25,33]. Knockout tournaments which are balanced are of particular interest, as they are considered to be fair [20] and allow a large number of matches to be played in parallel.

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.[1] Given N, an *ordered balanced knockout tournament* $\mathrm{T}(N, \pi)$ is defined as a balanced binary tree with n

☆ This is a revised and expanded version of Aziz et al. [3] with additional proof details as well as new results for tournaments with kings and double elimination tournaments.

* Corresponding author.
E-mail addresses: haris.aziz@unsw.edu.au (H. Aziz), serge.gaspers@data61.csiro.au (S. Gaspers), simonm@andrew.cmu.edu (S. Mackenzie), N.Mattei@ibm.com (N. Mattei), paul.stursberg@ma.tum.de (P. Stursberg), toby.walsh@data61.csiro.au (T. Walsh).

[1] The setting is general enough to cover the case where some players get byes in the first round. In that case we can consider a dummy player who always loses to the player who gets a bye.

https://doi.org/10.1016/j.artint.2018.05.002
0004-3702/

leaf nodes where the *seeding* π specifies the labeling of the leaf nodes with respect to N. All ordered balanced knockout tournaments that are isomorphic to each other (with respect to the labeling 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 $\frac{n!}{2^{n-1}}$ as all pairwise matchups in the leaf nodes are the same if adjacent elements of the seeding are swapped, but even this grows very rapidly. For a tournament like Wimbledon, $n = 128$ and the number of distinct draws is $\approx 2.2665 \cdot 10^{177}$. 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 $P_{ij} \in [0, 1]$ denotes the probability of player i beating player j in a pairwise elimination match and $0 \leq P_{ij} = 1 - P_{ji} \leq 1$. We call P *deterministic* if $P_{ij} \in \{0, 1\}$ for all players $i, j \in N$. In this case, we say that player i *beats* player j if $P_{ij} = 1$. 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 [36]. 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 can now define 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 et al. [36] and has been studied in numerous papers (see e.g., [29–31]). It is a well-motivated problem in sports analytics [28]. PTFP has been shown to be NP-hard for various restrictions, including the case where the entries of P are restricted to $\{0, \frac{1}{2}, 1\}$ [36] and the case where the matrix P is deterministic and certain matches are not allowed [34].

Nevertheless, the computational complexity of a particularly natural and interesting special case, the Tournament Fixing Problem (TFP), has remained a major open question. In the TFP, the matrix P is deterministic and all matches are allowed. The winner of each match is deterministically known beforehand and the question is whether there exists a draw for which a given player wins 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 the same name as defined in Vassilevska Williams [34], where there can be additional constraints by which certain matches are disallowed. 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 interpreted as the relative strength of the player.

Contributions We first settle the computational complexity of TFP by showing that it is NP-complete. The question was explicitly stated as an open problem a number of times [12,19,35,26,29–31,34,36,37]. As a corollary, we show that unless $\mathrm{P} = \mathrm{NP}$, there exists no polynomial-time approximation algorithm for computing the maximal winning probability of a player. This inapproximability result provides additional motivation for the line of work in which heuristic algorithms have been proposed for PTFP [37]. 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 (and hence also TFP) can be solved in polynomial time. In the first case, the players can be divided into a constant number of player types. This setting 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.

Finally, we consider double-elimination tournaments which are a variant of knockout tournaments in which the losers get a second chance to win the overall tournament. We show that TFP is NP-complete for this problem as well thereby answering another open problem [32].

[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. Russell and van Beek [26] as well as Mattei and Walsh [24] provide an extended discussion and empirical data on these phenomena.

Related work After the work of Vu et al. [36], PTFP and TFP have been studied in a number of research papers. Vassilevska Williams [34] identified various sufficient conditions for a player to be a winner of a BKT, e.g. if he is a king who beats half of all players. In a followup paper, Stanton and Vassilevska Williams [31] focused on when weak players can possibly win a BKT.

The authors of this paper proved Theorem 1 in a conference paper [3] and provided two additional tractable cases as well as an exponential-time algorithm. Subsequently, Kim and Vassilevska Williams [17] showed that TFP remains NP-hard if the distinguished player is a king that beats $n/4$ players. They also prove that TFP remains NP-hard for 3-kings who can beat $n/2$ players. These structural results about kings are enclosed in one general theorem in Kim et al. [16]. Furthermore, Kim and Vassilevska Williams [17] present an algorithm that improves the (exponential) running time of that in Aziz et al. [3], albeit at the expense of requiring exponential space.

Stanton and Vassilevska Williams [29,30] identified conditions in a probabilistic model under which the tournament organizer can fix the tournament with high probability. In Vu and Shoham [38], the problem of designing 'fair' draws was considered. Lang et al. [20] and Lang et al. [19] examined winner determination in voting trees that need not be balanced. In more recent work, Chatterjee et al. [6] examine the robustness of winning probability with respect to small errors in the pairwise winning probabilities.

TFP can also be considered as the problem of checking whether a player is a *possible winner* in a deterministic BKT. Computing possible winners for other voting rules where the information on the preferences is not complete has been studied extensively [2,40]. 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 [11,15].

2. 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 directed 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 [4]. 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 with a distinguished player who can win the tournament by some draw if and only if 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 or equal to $32 \cdot |X|$. The resulting knock-out tournament will thus consist of $R := \log(n) = \lceil \log(32 \cdot |X|) \rceil \geq 5$ 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.

In the following, we use the notation $A \dot\cup B$ for the union of two sets A and B which are disjoint, i.e. $A \cap B = \emptyset$. 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 behavior 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.

There will be a total of $k := \frac{n}{16}$ gadgets, each associated with one of the players from the set $M = \{m_1, m_2, \ldots, m_k\}$. The player m_1 will be our distinguished player. We will show that for m_1 to win the BKT, all players from M will have to proceed to the fifth round. The structure of the gadgets will make sure that this can happen if and only if the 3SAT instance F is satisfiable.

More precisely, we use $|X|$ choice gadgets numbered from 1 to $|X|$, $|X|$ clause/garbage gadgets (that may each model up to two clauses) numbered from $|X| + 1$ to $2|X|$ and $(k - 2|X|)$ filler gadgets numbered from $(2|X| + 1)$ to k. As every literal may appear at most twice, note that the number of clauses is bounded by $|C| \leq \frac{4|X|}{3} \leq 2|X|$ and hence the $|X|$ clause/garbage gadgets provide enough space to model all clauses in C.

Using this numbering, we can further partition the above sets as follows:

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

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 spawning process* 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 *spawns* a new player 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. We denote by M_ℓ the

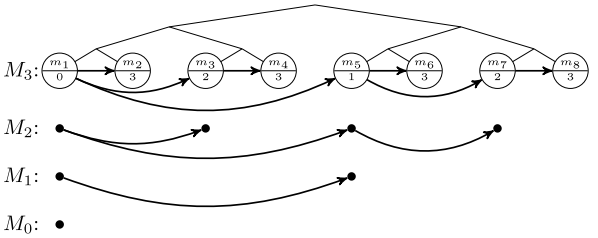

Fig. 1. The spawning process for the case $k = 8$, the lower part of each node shows the corresponding value of l_j. By the draw shown on top of M_3, the leftmost player can win the BKT.

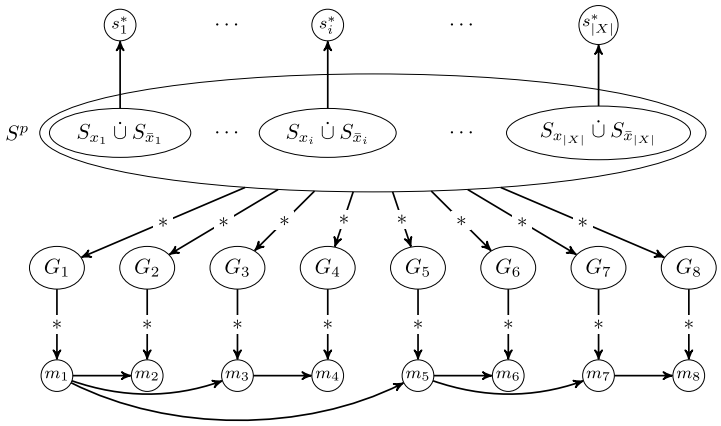

Fig. 2. Global structure for the case $k = 8$. All arrows not shown in the figure run downwards, horizontal arcs run right to left. Vertices grouped in a component have the same relation with vertices outside the component (unless specified otherwise). The relation of components connected by a starred edge has a few exceptions that are discussed in detail for the individual gadgets: Every element m_i beats exactly four of the elements from G_i and some G_i contain elements that beat some elements from S^p.

set of 2^ℓ players from M that are present after the ℓ-th iteration of the spawning process. Furthermore, for any $j \in [k]$, let ℓ_j be the smallest ℓ such that $m_j \in M_\ell$ (see Fig. 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 Fig. 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. □

We will show later that this is the *only* way for m_1 to win the tournament. In other words, m_1 can win the tournament if and only if all players from M can reach the fifth round simultaneously.

Global structure We now describe how the sets from (1) and (2) relate to each other. In general it is true that players from S beat all other players and players from $G^{\mathrm{X}} \dot\cup G^{\mathrm{CG}} \dot\cup G^{\mathrm{F}}$ beat players from M. There are a few exceptions to this rule, which will be detailed below.

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 $j, j' \in [k]$ with $j > j'$ and elements $i \in G_j$, $i' \in G_{j'}$ we have that i beats i'.

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 $|S_{x_j}| = |S_{\bar{x}_j}| = 3$. We further define the set of *particles* (these will move between the choice and clause/garbage gadgets) by $S^p := \dot\bigcup_{j\in[|X|]}(S_{x_j} \dot\cup S_{\bar{x}_j})$. 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 $S^p \setminus S_j$. The structure is visualized in Fig. 2.

The relation of the members of M among themselves is given by the spawning process (see Fig. 1).

The only exceptions to the general rule of S beating all other players and $G^{\mathrm{X}} \dot\cup G^{\mathrm{CG}} \dot\cup G^{\mathrm{F}}$ beating players from M are listed below, they will be detailed in the description of the individual gadgets.

- Some sets G_i contain elements that beat some elements from S^p.
- Every element m_i beats exactly four of the elements from G_i.

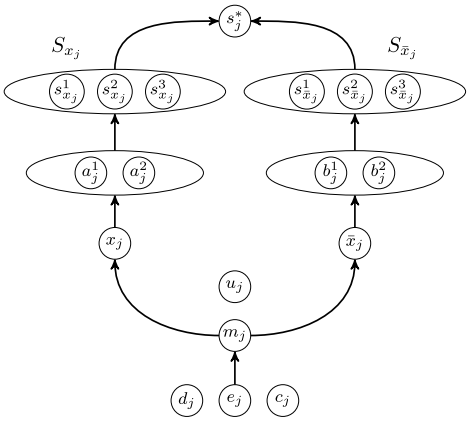

Fig. 3. Pairwise comparisons in the j-th choice gadget. All arrows not shown in the figure run downwards, horizontal arcs run right to left. Vertices grouped in a component have the same relation with vertices outside the component.

Lemma 2. *For m_1 to win the tournament, all players $m_j \in M$ must reach the fifth round.*

Proof. 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, recall that ℓ_j denotes the smallest ℓ such that m_j is present after the ℓ-th iteration of the spawning process. Further note that for all $j \in [k]$, player m_j beats exactly $(R-4-\ell_j) = (\log k - \ell_j)$ 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 $j \in [k]$:
- *If m_j was spawned by some player m_{j^*}, then m_j needs to play against m_{j^*} in round $(R+1-\ell_j)$.*

For the base case, consider m_1. Player m_1 was spawned by no other player, hence there is nothing to show.

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 $j' < j$. Denote by m_{j^*} the player that spawned m_j and note that $j^* < j$ and hence the induction hypothesis holds for j^*.

First, note that m_{j^*} beats all players $m_{j'}$ with $j' < j^*$ except the one that spawned him. However, by the induction hypothesis they all have to play (and lose) against the player that spawned *them*, therefore they cannot play against m_{j^*} (as they would lose). Thus, the only players that m_{j^*} can and is allowed to beat are players from G_{j^*} and players m_r from M with $r > j^*$. He beats exactly four players in G_{j^*} and $(R-4-\ell_{j^*})$ (including m_j) from M to his right. By the induction hypothesis, m_{j^*} needs to reach round $(R+1-\ell_{j^*})$, he thus has to play against all of these players (including m_j). This proves that m_j must play against the player that spawned him.

To see why this has to happen in round $(R+1-\ell_j)$, 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 $(R+1-\ell_j)$.

The original claim now follows from the induction hypothesis because $\ell_j \leq \log_2(k) = \log_2(\frac{n}{16}) = \log_2(n) - 4 = R-4$ and hence $R+1-\ell_j \geq R+1-(R-4) = 5$. □

We now describe the design of the gadgets. These will allow us to conclude that all players from M can reach the fifth round if and only if the 3SAT instance F is satisfiable. Note that all choice gadgets consist of 18 players ($\{m_j\} \dot\cup G_j \dot\cup S_j$), clause/garbage gadgets consist of 14 players ($\{m_j\} \dot\cup G_j$) and filler gadgets consist of 16 players ($\{m_j\} \dot\cup G_j$). In the first four rounds of a BKT, some players from the choice gadgets thus have to compete in a subtournament against the players from clause/garbage gadgets. The exact way in which this is possible while making sure that m_1 wins the tournament will be used to encode the 3SAT instance.

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 Fig. 3. 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 Fig. 6. 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. For each $j \in \{|X|+1, |X|+2, \ldots 2|X|\}$, 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

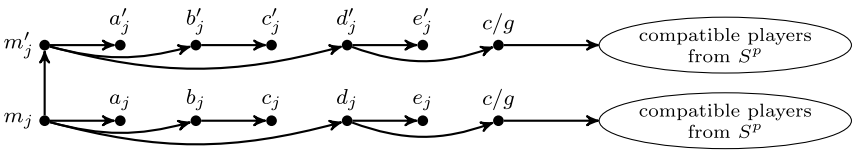

Fig. 4. Pairwise comparison graph for a clause/garbage gadget. All arrows not shown in the figure run downwards, horizontal arcs run right to left.

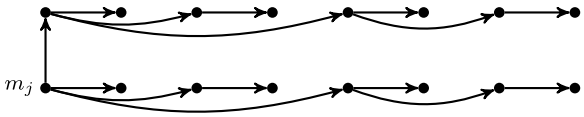

Fig. 5. Pairwise comparison graph for a filler gadget. All arrows not shown in the figure run downwards, horizontal arcs run right to left.

graph for these players is shown in Fig. 4. 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 element from S^p for both c/g-players included. We will 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.

Filler gadget Finally, for each $j \in \{2|X|+1, 2|X|+2, \ldots, k\}$, the j-th filler gadget consists of m_j and 15 players in set G_j. The pairwise comparison graph relating them to one another is shown in Fig. 5, the structure is the same as among the players from M.

We note that for all $j \in [k]$, player m_j beats exactly four players from outside of the set M. This allows us to prove the following lemma.

Lemma 3. *For all $j \in [k]$ the following hold:*

(i) If all players from M reach the fifth round, then the 16-player subtournament taking m_j to the fifth round consists exactly of:
  (a) 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,
  (b) 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,
  (c) if the j-th gadget is a filler gadget: $\{m_j\} \dot\cup G_j$.
(ii) If the 16-player subtournament for the first four rounds in which m_j is placed consists exactly of (i)a, (i)b, or (i)c for the respective type of the j-th gadget, then a draw for the subtournament exists by which m_j reaches the fifth round.

Proof. We prove the lemma by induction over j. Assume that the statement holds for all $i < j$ (this also takes care of the case $j = 1$ were we do not assume anything).

Note that by induction, all players from $G_i, i < j$ have to be seeded in the i-th subtournament and are hence no longer available for the j-th subtournament. Also, no other player from M may be seeded in the subtournament of m_j, as otherwise not all players from M can reach the fifth round.

In our analysis we consider positions in the subtournament one after another. We say that a player is no longer available when in order for m_j to advance we need that he be placed in one of the positions already considered.

We first consider the case where the j-th gadget is a choice gadget. In order to proceed to the fifth round, player m_j has to be matched against all four of the players in G_j that he can beat. In Fig. 3 these are players $\{d_j, c_j, x_j, \bar{x}_j\}$. Of those four players, d_j beats no player that is still available. Therefore, d_j cannot proceed to the second round and must face m_j in the first round. It is therefore no longer available elsewhere. In rounds two to four, m_j thus has to be matched against c_j, x_j, and $\bar{x}_j$. Of these, c_j beats only one player that is still available (e_j) and can therefore only proceed to the second round. By the same argument as before, m_j must be matched against c_j in the second round, thus c_j and e_j are no longer available for consideration in later positions. For the third and fourth round we can either match m_j first against x_j and then against $\bar{x}_j$ or the other way around since both those players are capable of reaching the fourth round. Since both x_j and $\bar{x}_j$ beat exactly three of the available players and u_j is among them for both x_j and $\bar{x}_j$, the player who will be matched against m_j in the fourth round must beat u_j. This match must happen in the first round, as u_j beats none of the available players. No matter whether we match m_j against x_j or $\bar{x}_j$ first, x_j must beat a_j^1 and a_j^2 whilst $\bar{x}_j$ must beat b_j^1 and b_j^2. Depending on

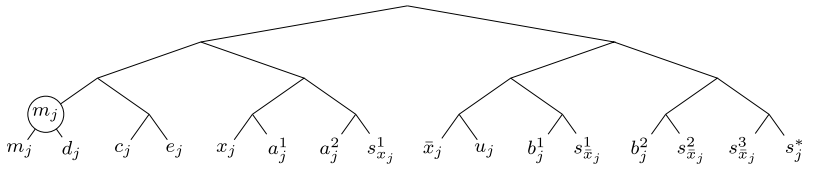

Fig. 6. The subtournament for choice gadget j. In this case, x_j=true is selected.

which option we chose, a_j^1 and a_j^2 will have to lose either in the second and third round and b_j^1 and b_j^2 in the first and second round or vice versa. Note that, among the players still available, a_j^1 and a_j^2 only beat members of S_{x_j} whereas b_j^1 and b_j^2 only beat members of $S_{\bar{x}_j}$. This means that if we select a_j^1 and a_j^2 to reach the second and third round, their first- and second-round matches must all be against players from S_{x_j} by which all three members of S_{x_j} lose against either a_j^1 or a_j^2 in the first or second round. On the other hand, if a_j^1 and a_j^2 only had to get to first and second round, they would only have to play against one member of S_{x_j} and the other two could be removed from the subtournament. The same holds for b_j^1, b_j^2 and $S_{\bar{x}_j}$, respectively. This situation is shown in Fig. 6. This proves (i), (ii) is easily verified if one observes that one of the players in S_j^p that has to make it to the second round can do so by being matched against s_j^* or a player from S_i^p with $i < j$.

Now let us consider the case where the j-th gadget is a clause gadget, the argument is very similar. Again, player m_j has to be matched against all four of the players in G_j that he can beat. In Fig. 4 these are the players $\{a_j, b_j, d_j, m'_j\}$. Of those four players, a_j beats no player that is still available. Therefore, a_j cannot proceed to the second round and must face m_j in the first round. This makes a_j unavailable to be used in other positions. In rounds two to four, m_j thus has to be matched against b_j, d_j, and m'_j. Of these, b_j beats only one player that is still available (c_j) and can therefore only proceed to the second round. The same argument as before forces m_j to be matched against b_j in the second round removing b_j and c_j from availability. Player m_j now beats d_j and m'_j amongst the available players. Player d_j only beats two other players, therefore m_j will beat him in the third round. Player d_j beats e_j and a c/g player. Player e_j beats none of the available players, therefore d_j must beat him in the first round. This makes e_j unavailable. The c/g player beats compatible players from S^p and can therefore go to the second round if one of those is available. Player m_j is left with only one player to beat to get to the fifth round which is m'_j. Player m'_j can and needs to reach the fourth round. The sub-tournament enabling m'_j to reach the fourth round is structured exactly the same way as the one just described. Therefore m_j can go to the fifth round if and only if we can add to the subtournament consisting of m_j and G_j one compatible (beatable) player from S^p for each c/g player, which proves both (i) and (ii).

The filler gadget is built by the spawning process. If the j-th gadget is a filler gadget, this guarantees that m_j beats only four players, one of which beats no one in the gadget. m_j therefore needs to beat him to get to the second round and therefore that player is no longer available. The spawning process also guarantees there is a second player that beats only one other available player in the gadget and can only get to the second round. m_j must beat him in the second round. We can continue using the same argument until the fifth round to show that m_j can go to the fifth round if and only if all players from G_j are placed in the subtournament together with him, which proves both (i) and (ii). □

Note that in light of Lemma 2, it follows by induction from Lemma 3 that m_1 can win the tournament if and only if the composition of all 16-player subtournaments covering the first four rounds is exactly as prescribed. We can now finally prove the following lemma.

Lemma 4. *Let $F = (X, C)$ be a 3SAT instance where each literal appears at most twice. There exists a set of players N with pairwise comparison matrix P and a distinguished player $m_1 \in N$ such that F is satisfiable if and only if we can find a draw σ such that m_1 wins the tournament* $\mathrm{T}(N, \sigma)$.

Proof. Let N, m_1 and P be as specified above. First, note that by Lemma 1 and Lemma 2, m_1 can win the tournament if and only if all players from the set M reach the fifth round. This is possible if and only if each player $m_j \in M$ can win a 16-player subtournament that consists exactly of the players stated in Lemma 3. 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 3 exists if and only if 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 $A(x_j) = \text{true}$) or $S_{\bar{x}_j}$ (if $A(x_j) = \text{false}$) in one subtournament for the first four rounds. These are exactly 16 players and by Lemma 3(ii), 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.

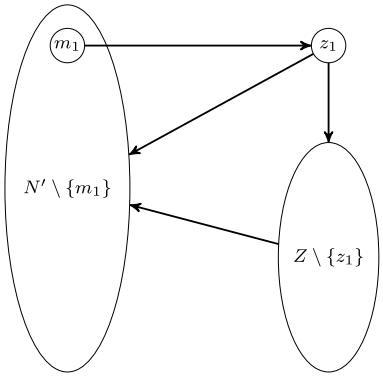

Fig. 7. Pairwise comparison among members of the set N from Lemma 5.

For the clause/garbage gadgets, to use Lemma 3b) 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 3(ii), we can choose 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 3(ii), 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 3. First, note that whenever a player s_j^* is not placed in the j-th choice gadget, we get a contradiction as by Lemma 3, 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: $A(x_j) = \text{true}$ if m_j's subtournament includes all players from S_j except two players from S_{x_j}, $A(x_j) = \text{false}$ 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 the corresponding clause gadget. Let j^* be the index of that gadget. By the assumption, m_{j^*} wins the corresponding subtournament, thus Lemma 3 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 Theorem 1. □

The following Lemma will be required for Theorem 6. We prove it here, as it can be used to show that it remains computationally hard to decide whether a particular player can win a BKT, even if we already know that this player is a particularly strong player: A player a is called a *king* if for every player b that a does *not* beat, there exists a third player c such that a beats c and c beats b. Using the next lemma, we will show that TFP remains NP-hard if restricted to instances where m_1 is a king.

A related theorem has previously been proved by Kim and Vassilevska Williams [17] using our Theorem 1. The result from [17] is slightly more general in that it requires m_1 to be a particularly strong king (one that directly beats $\frac{1}{4}$ of all players). Leaving out this additional requirement, we can make a simpler argument and only require an instance half the size.

Lemma 5. *Let $k \in \mathbb{N}$ and N' a set of players with $|N'| = 2^k$ and a distinguished player $m_1 \in N'$.*

(i) Let Z be another (disjoint) player set with $|Z| = |N'|$ and $z_1 \in Z$ a player from Z. Let P be a pairwise comparison matrix on $N = N' \dot\cup Z$ such that
- *z_1 beats z for any $z \in Z \setminus \{z_1\}$*
- *z beats p for any pair $(z, p) \in (Z \times N') \setminus \{(z_1, m_1)\}$*
- *m_1 beats z_1 (see Fig. 7).*

Then, there is a draw σ of N such that m_1 wins the tournament $\mathrm{T}(N, \sigma)$ if and only if there is a draw σ' of N' such that m_1 wins the tournament $\mathrm{T}(N', \sigma')$.

(ii) Let P' be a pairwise comparison matrix on N'. There exists a set of players Z and pairwise comparison matrix P on $N' \dot\cup Z$ satisfying the conditions from (i) such that P coincides with P' on N'.

Proof. For part (i), we prove both directions separately.

($\Rightarrow$) Assume that there exists a draw σ' such that m_1 wins the tournament $\mathrm{T}(N', \sigma')$. Construct σ in such a way that the final is a match between the winners of two sub-tournaments A and B where in A, all players from N' are seeded according to σ' and in B all players from Z are seeded in an arbitrary way. The sub-tournament B is won by z_1 (as it beats all other players that compete in B) and sub-tournament A is won by m_1 (by Lemma 4 as A is exactly the tournament $\mathrm{T}(N, \sigma')$). In the final, m_1 wins against z_1 and hence m_1 wins the tournament $\mathrm{T}(N, \sigma)$. ($\Leftarrow$) Assume that there is no draw σ' such that m_1 wins the tournament $\mathrm{T}(N', \sigma')$. Again, let σ be a draw for N such that the final is a match between the winners of two sub-tournaments A and B and assume w.l.o.g. that m_1 is seeded in A. We differentiate three cases:

(i) If z_1 is also seeded in A, then there is at least one player from $Z \setminus \{z_1\}$ that is seeded in B (because only $|N'| = |Z|$ players can compete in A). Hence a player $z \in Z \setminus \{z_1\}$ will win the sub-tournament B and even if m_1 wins A, he will be beaten by z in the final.
(ii) If z_1 is not seeded in A but some other players from Z are, then one of them will win the tournament A and hence m_1 cannot win.
(iii) If no player from Z is seeded in A, then by the assumption there is no draw such that m_1 can win the sub-tournament A.

Hence, there is no draw σ such that m_1 wins the tournament $\mathrm{T}(N, \sigma)$ if m_1 cannot win the tournament among the players in N'.

For part (ii) of Lemma 5, a suitable instance can easily be constructed, as no assumptions are made on the structure of P restricted to N'. □

Theorem 2. *TFP is NP-complete even if the distinguished player is a king.*

Proof. First, note that in any instance satisfying the conditions of Lemma 5, m_1 is a king (by beating z_1 who in turn beats every other player).

Let $F = (X, C)$ be a 3SAT instance where each literal appears at most twice. By Lemma 4, we can find a player set N', pairwise comparison matrix P' and distinguished player m_1 such that F is satisfiable if and only if there is a draw σ such that m_1 can win $T(N', \sigma)$.

By Lemma 5(ii), we can obtain a suitable instance such that m_1 is a king and the statement follows from Lemma 5(i). □

2.1. Implications of Theorem 1

Theorem 1 simultaneously implies a number of results from the literature, in particular Theorem 1, 2, and 3 in Vu et al. [36] and Theorem 1 in Vassilevska Williams [34]. 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 deterministic polynomial-time algorithm for PTFP that approximates the maximum winning probability of a player within any given positive factor.*

Proof. The approximation algorithm can be used to check in polynomial time whether the maximum winning probability is zero or non-zero. If the algorithm is deterministic, it can hence solve TFP in polynomial time. □

It immediately follows from Theorem 1 that #TFP is NP-hard. We next highlight that even randomization 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 $\frac{1}{\epsilon}$. 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 [39] and Theorem 1.

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

Proof. The statement follows from Theorem 1 and the well-known fact (see for e.g., Proposition 8 in [39]) that if $f : \Gamma^* \longrightarrow \mathbb{N}$ is such that deciding whether f is non-zero or not is NP-hard, then there is no FPRAS for f unless NP = RP. □

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.

3. Tractable cases

We first examine a natural case in which players are divided into player *types*: the outcome of any match is determined only by the type of the respective players. In other words, given two types i and j, it is either the case that all players from i win against all players from j or the other way round. This is an interesting case as in sports tournaments it may be that many players are of the same or similar quality, i.e., of the same type. 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 3. *#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 k be the number of different player types. We denote the number of players of type i by η_i. 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 $\binom{k+2^x-1}{k-1}$ such problems that potentially need to be considered. This is because for positive integers p, q there are $\binom{p+q-1}{q-1}$ weak compositions of p into q parts, i.e., there are $\binom{p+q-1}{q-1}$ distinct ways to write a sum of q non-negative integers that evaluates to p. The number $\#\mathrm{TFP}(i, x, \eta)$ is computed via the following recursion.

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

The recursion considers all possible opponents of a player of type i at round x, as well as all possibilities for the $(x-1)$-round seedings of the subtournaments that the players of type i and j need to win in order to reach the final round.

The base cases are given by

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

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 $\binom{k+2^x-1}{k-1} \in O(n^k)$. 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 the case when the players can be linearly ordered by strength. This is a common rule of thumb used when guessing who will win a tournament and is supported by empirical results in the literature [26].

Theorem 4. *#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. For $b = 0$, we simply divide the players into the 3 types $N^-(i^*)$, $\{i^*\}$, and $N^+(i^*)$, where $N^-(i^*)$ and $N^+(i^*)$ are the set of players that beat i^* and are beaten by i^*, respectively. By induction, assume the statement holds for all $b' < b$. To show that it holds for b backwards arcs, select an arbitrary backwards arc $a = (u, v)$, reverse it, and group the players into $4(b-1)+3$ player types. We partition the players of u's type into three new types: $\{u\}$, those players that beat u, and those that are beaten by u. We do the same for v, and now we can reverse the arc a again, as it originally was. Thus we have $4(b-1)+3+4$ player types for the instance. Since the number of player types is constant, the theorem follows from Theorem 3. □

4. 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 [9], 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

Table 1
Running times and space requirements for the algorithm of Theorem 5 for various time-space trade-offs.

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	$O(3.6681^n)$	$O(1.2634^n)$
$\log n$	$4^{n+o(n)}$	$\text{poly}(n)$

only polynomial space is available, the running time becomes $4^{n+o(n)}$, and we give a range of possible time-space trade-offs. After the publication of our conference paper, Kim and Vassilevska Williams [17] improved our running time to $O(2^n\text{poly}(n))$ using $O(2^n\text{poly}(n))$ space. However, our algorithm also works if less space is available. The algorithm is based on the recursion formula (3) and memoization used at various levels of the recursion. We use $\text{poly}(n)$ to stand for a polynomial function in n.

Theorem 5. *For every y, $2 \leq y \leq \log n$, there is an algorithm solving #TFP in time $T(n) = \text{poly}(n) \cdot \prod_{p=0}^{y-1} \frac{n}{2^p} \cdot 2^{n/2^p}$ and space $S(n) = \text{poly}(n) \cdot (2^y)^{n/2^y} \cdot \left(\frac{2^y}{2^y-1}\right)^{n-n/2^y}$.*

Proof. We start with the polynomial space algorithm. The algorithm recursively evaluates the formula (3) for the special case where each player type consists of one player. We say that player i corresponds to player type η_i. The evaluation is top-down, i.e., to evaluate $\#\text{TFP}(i, x, \eta)$ the algorithm performs $O(|N_x| \cdot 2^{|N_x|})$ recursive calls to itself solving for each $\eta' \in N_{x-1}$ where $\eta' \leq \eta$ the subinstance $\#\text{TFP}(i, x-1, \eta')$ and the subinstances $\#\text{TFP}(j, x-1, \eta-\eta')$ for all $j \in [n]$ such that $P_{ij} = 1$ and $\eta_j - \eta'_j = 1$. The base cases are as in the proof of Theorem 3, and the algorithm returns $\#\text{TFP}(i, \log n, (1, \ldots, 1))$.

For n players, the number of recursive calls of the algorithm is upper bounded by $R(n) \leq n \cdot 2^n \cdot R(\frac{n}{2})$, which evaluates to $R(n) = \Pi_{p=0}^{\log n} \frac{n}{2^p} \cdot 2^{n/2^p} = 4^{n+o(n)}$. The running time $T(n)$ of the algorithm is within a polynomial factor of n of the number of recursive calls.

Using memoization, we can avoid repeating calculations at the expense of exponential space usage. The algorithm uses a table indexed by players, level, and player type vector. We say that we use memoization at level x if, when evaluating $\#\text{TFP}(j, x', \eta)$ with $x' \leq x$, 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. To evaluate $\#\text{TFP}(j, x', \eta)$ with $x' > x$, it does not use the table, and computes it recursively as in the polynomial space algorithm.

At level $\log n$, only one evaluation is made. At level $(\log n) - 1$, $O(n2^n)$ evaluations are made, but none of them is evaluated more than once. Therefore, we will use memoization at level $(\log n) - y$, where y varies from 2 to $\log n$. 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, and a binary representation of integers from 0 to $n!$ using $O(n \log n)$ bits [18], the space usage of the algorithm is upper bounded by $S(n) = \text{poly}(n) \cdot (2^y)^{n/2^y} \cdot \left(\frac{2^y}{2^y-1}\right)^{n-n/2^y}$. The running time of the algorithm is the time used for the recursion without memoization, $\text{poly}(n) \cdot \prod_{p=0}^{y-1} \frac{n}{2^p} \cdot 2^{n/2^p}$, plus the time for the part with memoization, which is upper bounded by $S(n) \cdot n \cdot 2^{n/4} = O(2.0868^n)$, since at most $S(n) = O(1.7548^n)$ entries are computed and each computation retrieves $O(n \cdot 2^{n/4})$ table values. Thus, for any y, $2 \leq y \leq \log n$, we obtain an algorithm with running time $T(n) = \text{poly}(n) \cdot \prod_{p=0}^{y-1} \frac{n}{2^p} \cdot 2^{n/2^p}$ using space $S(n) = \text{poly}(n) \cdot (2^y)^{n/2^y} \cdot \left(\frac{2^y}{2^y-1}\right)^{n-n/2^y}$ □

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.

5. Double-elimination tournaments

In double-elimination tournaments there are two brackets: *the winners bracket* where the competition proceeds exactly as in a BKT and *the losers bracket* where the losers from the BKT are mapped into a new knockout tournament. After (or concurrently) with the conclusion of the winners bracket, the losers bracket tournament is run. The champion of the losers bracket plays (possibly again) the champion of the winners bracket (the BKT winner); in a deterministic setting the winner of this final match is the winner of the overall tournament. Just like BKTs, we can define TFP for double-elimination tournaments. Stanton and Vassilevska Williams [32] state the complexity of the TFP for double-elimination tournaments as an open problem. We show that the problem is NP-complete.

Double elimination tournaments [8,14] have been present in sports for decades, and were mathematically defined by Stanton and Vassilevska Williams [32]. In the definition by Stanton and Vassilevska Williams [32] a double elimination

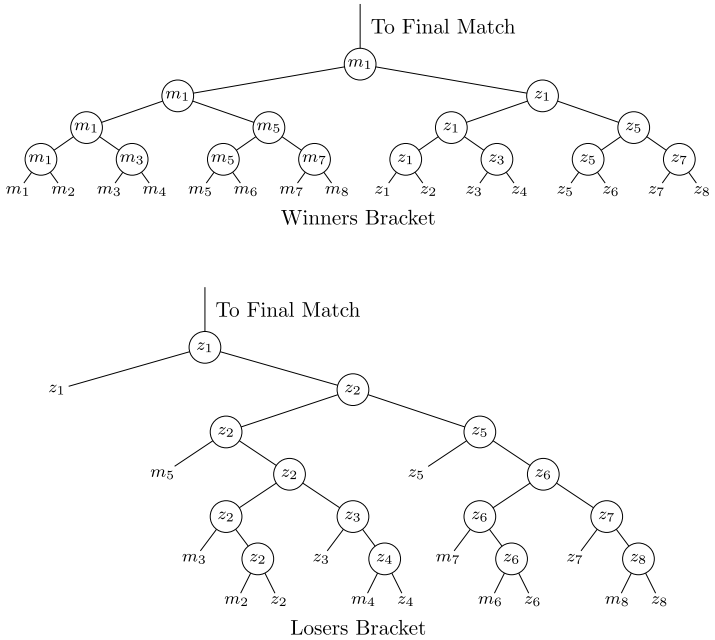

Fig. 8. Example of a winners bracket (above) and losers bracket (below) where each left hand player wins each round in their respective (m and z) maximal sub-tournaments. The champion of the losers bracket plays the champion of the winners bracket in a final match.

tournament is defined by the shape of the losers bracket and a *link function* that maps losers from the BKT into the losers bracket. This function maps matches in the BKT to leaves of the losers bracket. Stanton and Vassilevska Williams [32] also develop several desiderata by which the losers bracket can be evaluated including: *balance*, that the losers bracket should be a balanced tournament; *round-fairness*, that losers should compete against other losers who lost in (nearly) the same round; and *repeat-avoidance*, link functions should minimize the possibility of two teams meeting both in the winners and losers bracket.

Consider the 16 entrant example of a winners bracket, (implicit) link function, and corresponding losers bracket illustrated in Fig. 8. Our winners bracket consists of two maximal sub-tournaments, one starting below the circled m_1 node and one below the circled z_1 node which we will call N' and Z following the convention from Lemma 5. The depicted losers bracket is both round-fair and balanced according to the definitions given by Stanton and Vassilevska Williams [32].

Imagine that the tournament graph of the players $\{m_1, \ldots, m_8, z_1, \ldots, z_8\}$ is generated using the pairwise relation in Lemma 5 with the additional constraint that lower number elements of Z defeat higher number elements of Z. Hence, in the winners bracket the left hand player of every leaf pairing would advance, as illustrated in Fig. 8. To construct our losers bracket, we use the fair, balanced bracket described by Stanton and Vassilevska Williams [32].

We can extend the construction depicted in Fig. 8 to BKTs with any number of players. Note also that we can use any conceivable loser bracket setup and link function, in particular the following theorem also holds for double elimination tournaments with balanced losers brackets and round-fair link functions.

Theorem 6. *The tournament fixing problem is NP-complete for double elimination tournaments, regardless of any restrictions on losers bracket setup and link function.*

Proof. In Lemma 5 we define the conditions on two sub-tournaments of equal size, N' and Z, where N' corresponds to a TFP instance. By Lemma 5, within the Z bracket z_1 beats everyone and everyone in Z beats everyone in N' with the exception that m_1 defeats z_1. We will show that m_1 can win the double elimination tournament if and only if we can find a solution to the TFP instance defined on N'.

Formally, let $F = (X, C)$ be a 3SAT instance where each literal appears at most twice. By Lemma 4 we can find a player set N', pairwise comparison matrix P' and distinguished player m_1 such that F is satisfiable if and only if there is a draw σ' such that m_1 can win $\mathrm{T}(N', \sigma')$. Using Lemma 5(ii), we can extend the instance to the set of players $N := N' \dot\cup Z$ with pairwise comparison matrix P that satisfies the properties defined in Lemma 5 part (i). By Lemma 5(i), F is satisfiable if and only if there is a draw σ such that m_1 can win $\mathrm{T}(N, \sigma)$. It remains to show that there is a draw σ such that m_1 can win $\mathrm{T}(N, \sigma)$ if and only if there is a draw σ^* such that m_1 wins the double elimination tournament.

($\Rightarrow$) We show that if m_1 wins $\mathrm{T}(N, \sigma)$ then he wins the double elimination tournament by the same draw $\sigma^* = \sigma$.

By Lemma 5, the winners' bracket is won by m_1. This means that at some point, player z_1 is eliminated from the winners' bracket and enters the losers' bracket. As z_1 beats all players except m_1 (and hence in particular all players in the

losers' bracket), z_1 wins the losers' bracket and faces m_1 in the final, where he loses and m_1 wins the double elimination tournament.

($\Leftarrow$) Suppose that m_1 wins the double elimination tournament by some draw σ^*. If m_1 proceeds to the final as winner of the winner's bracket (which is a BKT of N), then he also wins $T(N, \sigma^*)$.

Suppose therefore that by the draw σ^*, player m_1 is at some point eliminated from the winners' bracket. This means that the winners' bracket will be won by some player from Z. We show that m_1 cannot win the double elimination tournament in this case.

If m_1 does not win the winners' bracket, then z_1 must win the winners' bracket for m_1 to win the tournament, since m_1 is beaten by all other players from Z. Therefore, z_1 will never enter the losers' bracket. This means that some player from $Z \setminus z_1$ will win the losers' bracket, as every player from $Z \setminus z_1$ beats all other players (those from N', including m_1) that enter the losers' bracket at some point. Hence, m_1 cannot win the losers' bracket and therefore cannot win the tournament.

This proves that m_1 cannot win the double elimination tournament as winner of the losers' bracket. Therefore, m_1 can only win the double elimination tournament resulting from the draw σ^* by advancing through the winner's bracket. This means that he wins $T(N, \sigma^*)$ which concludes the only-if part of the proof. □

As noted before Lemma 5, a similar argument can be constructed using the result from [17]. It follows from Theorem 6, that the probabilistic generalization of the problem is NP-hard as well. Similarly, we get similar corollaries as Corollaries 1 and 2 for double elimination tournaments.

6. 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 parameterized algorithms for TFP.

Acknowledgements

Haris Aziz is supported by a CSIRO Julius Career Award. Serge Gaspers is the recipient of an Australian Research Council (ARC) Future Fellowship (FT140100048) and acknowledges support under the ARC's Discovery Projects funding scheme (DP150101134).

References

[1] L. Aronshtam, H. Cohen, T. Shrot, Tennis manipulation: can we help Serena Williams win another tournament?, Ann. Math. Artif. Intell. 80 (2) (2017) 153–169.
[2] H. Aziz, M. Brill, F. Fischer, P. Harrenstein, J. Lang, H.G. Seedig, Possible and necessary winners of partial tournaments, in: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), IFAAMAS, 2012, pp. 585–592.
[3] H. Aziz, S. Gaspers, S. Mackenzie, N. Mattei, P. Stursberg, T. Walsh, Fixing a balanced knockout tournament, in: Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI), 2014, pp. 552–558.
[4] P. Berman, M. Karpinski, A.D. Scott, Approximation hardness of short symmetric instances of MAX-3SAT, Electron. Colloq. Comput. Complex. 049 (2003).
[5] J. Brown, D.B. Minor, Selecting the best? Spillover and shadows in elimination tournaments, Manag. Sci. (2014) 3087–3102.
[6] K. Chatterjee, R. Ibsen-Jensen, J. Tkadlec, Robust draws in balanced knockout tournaments, in: Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI), AAAI Press, 2016, pp. 172–179.
[7] R.A. Connolly, R.J. Rendleman, Tournament Qualification, Seeding and Selection Efficiency, Tech. Rep. 2011-96, Tuck School of Business, 2011.
[8] C.T. Edwards, Double-elimination tournaments: counting and calculating, Am. Stat. 50 (1) (1996) 27–33.
[9] F.V. Fomin, D. Kratsch, Exact Exponential Algorithms, Springer, 2010.
[10] C. Groh, B. Moldovanu, A. Sela, U. Sunde, Optimal seedings in elimination tournaments, Econ. Theory 49 (1) (2012) 59–80.
[11] D. Gusfield, C. Martel, The structure and complexity of sports elimination numbers, Algorithmica 32 (1) (2002) 73–86.
[12] N. Hazon, Y. Aumann, S. Kraus, M. Wooldridge, Evaluation of election outcomes under uncertainty, in: Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2008, pp. 959–966.
[13] J. Horen, R. Riezman, Comparing draws for single elimination tournaments, Oper. Res. 33 (2) (1985) 249–262.
[14] L. Huang, Prize and incentives in double-elimination tournaments, Econ. Lett. 147 (2016) 116–120.
[15] W. Kern, D. Paulusma, The computational complexity of the elimination problem in generalized sports competitions, Discrete Optim. 1 (2) (2004) 205–214.
[16] M.P. Kim, W. Suksompong, V. Vassilevska Williams, Who can win a single-elimination tournament?, in: Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI), 2016, pp. 516–522.
[17] M.P. Kim, V. Vassilevska Williams, Fixing tournaments for kings, chokers, and more, in: Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI), 2015, pp. 561–567.
[18] D.E. Knuth, Fascicle 1: bitwise tricks & techniques; binary decision diagrams, in: The Art of Computer Programming, vol. 4, Addison–Wesley Professional, 2009.
[19] J. Lang, M.S. Pini, F. Rossi, D. Salvagnin, K.B. Venable, T. Walsh, Winner determination in voting trees with incomplete preferences and weighted votes, Auton. Agents Multi-Agent Syst. 25 (1) (2012) 130–157.
[20] J. Lang, M.S. Pini, F. Rossi, K.B. Venable, T. Walsh, Winner determination in sequential majority voting, in: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI), 2007, pp. 1372–1377.

[21] J.-F. Laslier, Tournament Solutions and Majority Voting, Springer-Verlag, 1997.
[22] E. Marchand, On the comparison between standard and random knockout tournaments, J. R. Stat. Soc. 51 (2) (2002) 169–178.
[23] N. Mattei, J. Goldsmith, A. Klapper, M. Mundhenk, On the complexity of bribery and manipulation in tournaments with uncertain information, J. Appl. Log. 13 (4) (2015) 557–581.
[24] N. Mattei, T. Walsh, Empirical evaluation of real world tournaments, CoRR, arXiv:1608.01039, http://arxiv.org/abs/1608.01039, 2016.
[25] S. Rosen, Prizes and incentives in elimination tournaments, Am. Econ. Rev. 76 (4) (1986) 701–715.
[26] T. Russell, P. van Beek, An empirical study of seeding manipulations and their prevention, in: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI), 2011, pp. 350–356.
[27] T. Russell, T. Walsh, Manipulating tournaments in cup and round robin competitions, in: Proceedings of the 1st International Conference on Algorithmic Decision Theory, 2009, pp. 26–37.
[28] N. Silver, 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/ (Accessed February 2014).
[29] I. Stanton, V. Vassilevska Williams, Manipulating single-elimination tournaments in the Braverman–Mossel model, in: IJCAI Workshop on Social Choice and Artificial Intelligence, 2011.
[30] I. Stanton, V. Vassilevska Williams, Manipulating stochastically generated single-elimination tournaments for nearly all players, in: Proceedings of 7th International Workshop on Internet and Network Economics (WINE), 2011, pp. 326–337.
[31] I. Stanton, V. Vassilevska Williams, Rigging tournament brackets for weaker players, in: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI), 2011, pp. 357–364.
[32] I. Stanton, V. Vassilevska Williams, The structure, efficacy, and manipulation of double-elimination tournaments, J. Quant. Anal. Sports 9 (4) (2013) 319–335.
[33] G. Tullock, Toward a Theory of the Rent-Seeking Society, Texas A&M University Press, 1980.
[34] V. Vassilevska Williams, Fixing a tournament, in: Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI), AAAI Press, 2010, pp. 895–900.
[35] V. Vassilevska Williams, Fixing tournaments, in: R.J. Lipton, K.W. Regan (Eds.), People, Problems, and Proofs: Essays From Gödel's Lost Letter: 2010, Springer, 2013, pp. 319–322, Ch. 61.
[36] T. Vu, A. Altman, Y. Shoham, On the complexity of schedule control problems for knockout tournaments, in: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2009, pp. 225–232.
[37] T. Vu, N. Hazon, A. Altman, S. Kraus, Y. Shoham, M. Wooldridge, On the Complexity of Schedule Control Problems for Knock-Out Tournaments, Working paper, 2013.
[38] T. Vu, Y. Shoham, Fair seeding in knockout tournaments, ACM Trans. Intell. Syst. Technol. 3 (1) (2011) 1–17.
[39] D.J.A. Welsh, C. Merino, The Potts model and the Tutte polynomial, J. Math. Phys. 41 (3) (2000) 1127–1152.
[40] L. Xia, V. Conitzer, Determining possible and necessary winners under common voting rules given partial orders, J. Artif. Intell. Res. 41 (2) (2011) 25–67.