On the Complexity of Schedule Control Problems for Knockout Tournaments[*]

Thuc Vu, Alon Altman, Yoav Shoham
Computer Science Department
Stanford University, California, 94305
{thucvu,epsalon,shoham}@stanford.edu

ABSTRACT

Knockout tournaments constitute a common format of sporting events, and also model a specific type of election scheme (namely, sequential pairwise elimination election). In such tournaments the designer controls the shape of the tournament (a binary tree) and the seeding of the players (their assignment to the tree leaves). In this paper we investigate the computational complexity of tournament schedule control, i.e., designing a tournament that maximizes the winning probability a target player. We start with a generic probabilistic model consisting of a matrix of pairwise winning probabilities, and then investigate the problem under two types of constraint: constraints on the probability matrix, and constraints on the allowable tournament structure. While the complexity of the general problem is as yet unknown, these various constraints – all naturally occurring in practice – serve to push to the problem to one side or the other: easy (polynomial) or hard (NP-complete).

Categories and Subject Descriptors

H.2 [**Computing Methodologies**]: Artificial Intelligence

General Terms

Tournament Design

Keywords

Tournament Design, Voting Theory, Election Control, Complexity

1. INTRODUCTION

Tournaments[1] constitute a very common social institution. Their best known use is in sporting events, which attract millions of viewers and billions of dollars annually. But tournaments also play a key role in other social and commercial settings, ranging from the employment interview process to patent races and rent-seeking contests (see [12, 15, 10] for details).

Tournaments constitute a strict subclass of all competition formats, and yet they still allow for many different variations. All tournaments consist of *stages* during which several *matches* take place, matches whose outcomes determine the set of matches in the next stage, and so on, until some final outcome of the tournament is reached. But tournaments vary in how many stages take place, which matches are played in each stage, and how the outcome is determined.

In this paper we focus on a narrower class of tournaments: *knockout* tournaments. In this very familiar format the players are placed at the leaf nodes of a binary tree. Players at sibling nodes compete against each other in a pairwise match, and the winner of the match moves up the tree. The player who reaches the root node is the winner of the tournament. We show an example in Figure 1.

While our motivation is anchored in the tournament world, the specific format of knockout tournaments is isomorphic to a particular class of elections studied within voting theory, namely sequential elimination voting with pairwise comparison [3, 9]. In such elections the players are the candidates, and the pairwise matching represents a pairwise election rather than a sporting match, but otherwise the process is identical.

What makes the connection to elimination election particularly striking is the problem we tackle in this paper, namely schedule control. In knockout tournaments, as in sequential elimination voting with pairwise comparison, the designer has limited control: The shape of the tournament/voting tree, and the assignment of players/candidates to its leaves. This is called the *schedule* of the tournament or the agenda of the election. The rest is outside the designer's control. The question we tackle is how a designer can best exercise this control in order to optimize a certain quantity. In this paper we focus on maximizing the winning probability of a given target player (this is usually viewed in a negative connotation, namely as biasing the tournament or rigging the election; of course by studying the difficulty of such manipulation we do not condone it, and the results can be used to prevent it rather than enable it). This is a natural question in the context of voting; there are several others in the context of tournaments, but they lie outside the scope of this paper. Thus, while in general the theory of tournaments is quite different from the theory of voting, they coincide when we speak about schedule control in knockout tournaments. This has not always been recognized, and specifically some literature on knockout tournaments makes no reference to voting, but we will appeal to the literature of both camps. For coherence, however, we will continue to use the sports/tournament terminology in the remainder of this paper.

Our problem may at first seem narrow – a very restricted class of tournaments, and a very specific design objective. But this seemingly simple question turns out to be surprisingly subtle and some of the answers are counter-intuitive. To begin with, note that the number of possible schedules grows extremely quickly with the

[*]This work is supported by NSF grants IIS-0205633-001 and SES-0527650.

[1]Here we use the term with the commonsense meaning, instead of indicating directed tournament graphs.

Cite as: On the Complexity of Schedule Control Problems for Knockout Tournaments, Thuc Vu, Alon Altman, Yoav Shoham, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

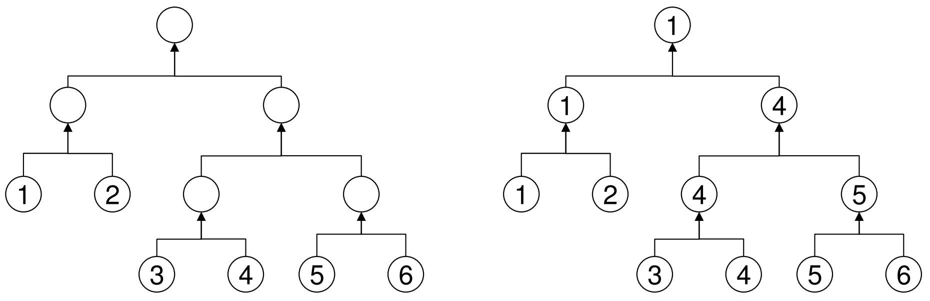

Figure 1: An example of a tournament structure for 6 players and one possible outcome

number of players, e.g., $O(\frac{n!}{2^{n-1}})$ when the possible schedule is limited to be of balanced tournament only. This means that even for a small number of players it can be hard to answer the question. For $n = 2, 4, 8, 16, 32$, the numbers of possible, non-duplicate schedules are $1, 3, 315, 638 \times 10^6, 122 \times 10^{24}$ respectively. But the asymptotic analysis is also not straightforward, since it is sensitive to several variations of the problem modeling. Our basic model, which appears in both the tournament literature and the voting literature, is that of a winning-probability matrix, the (i, j) entry of which represents the probability that player i wins over player j in a match between them (see [6, 4] for example). With no further constraints, it is unknown whether there exists an efficient algorithm to find the optimal structure. However, when we place certain natural constraints on the structure of the tournament or the winning-probability matrix, the problem becomes either provably easy (namely, polynomial) or provably hard (specifically, NP-complete). In this paper, we discuss these settings and analyze the complexity of the problem in each setting.

The rest of the paper is organized as follows. We first present the general model for tournament design in Section 2. Then in Section 3 we summarize the existing results from the literature. We then describe in Section 4 and 5 different constraints that can be placed on the model and our results for these settings. We summarize the results in Section 6 and suggest possible directions for future work.

2. THE GENERAL MODEL AND PROBLEM

We start out with the most general model of a knockout tournament. In this setting, there is no constraint on the structure of the tournament, as long as it only allows pairwise matches between players. We also assume that for any pairwise match, the probability of one player winning against the other is known. This probability can be obtained from past statistics or from some learning models. Here we do not place any constraints on the probabilities either, besides the fundamental properties. Thus there might be no transitivity between the winning probabilities, e.g., player i has more than 50% chance of beating player j, player j has more than 50% chance of beating player k, but player k also has more than 50% chance of beating player i.

We define a knockout tournament as the following:

Definition 1. **(General Knockout Tournament)** Given a set N of players and a matrix P such that P_{ij} denotes the probability that player i will win against player j in a pairwise elimination match and $0 \le P_{ij} = 1 - P_{ji} \le 1$ ($\forall i, j \in N$), a knockout tournament $KT_N = (T, S)$ is defined by:

- A tournament structure T which is a binary tree with $|N|$ leaf nodes
- A seeding S which is a one-to-one mapping between the players in N and the leaf nodes of T

We write KT_N as KT when the context is clear.

To carry out the tournament, each pair of players that are assigned to sibling leaf nodes with the same parent compete against each other in a pairwise elimination match. The winner of the match then "moves up" the tree and then competes against the winner of the other branch. The player who reaches the root of the tournament tree is the winner of the tournament.

Intuitively the probability of a player winning the tournament depends on the probability that it will face a certain opponent and win against that opponent. We formally define this quantity below:

Definition 2. **(Probability of Winning a Tournament)** Given a set N of players, a winning probability matrix P, and a knockout tournament $KT_N = (T, S)$, the probability of player k winning the tournament KT_N, denoted $q(k, KT_N)$ is defined by the following recursive formula:

1. If $N = \{j\}$, then $q(k, KT_N) = \begin{cases} 1 & \text{if } k = j \\ 0 & \text{if } k \neq j \end{cases}$

2. If $|N| \ge 2$, let $KT_{N_1} = (T_1, S_1)$ and $KT_{N_2} = (T_2, S_2)$ be the two sub-tournaments of KT such that T_1 and T_2 are the two subtrees connected to the root node of T, and N_1 and N_2 are the set of players assigned to the leaf nodes of T_1 and T_2 by S_1 and S_2 respectively. If $k \in N_1$ then

$$q(k, KT_N) = \sum_{i \in N_2} q(k, KT_{N_1}) \times q(i, KT_{N_2}) \times P_{ki}$$

and symmetrically for $k \in N_2$.

This recursive formula also gives us an efficient way to calculate $q(k, KT)$:

PROPOSITION 1. *Given a set N of players, a winning probability matrix P, and a knockout tournament $KT_N = (T, S)$, the complexity of calculating $q(k, KT)$ for a given $k \in N$ is $O(|N|^2)$.*

PROOF. First note that the number of operations is linear in the number of pairs (i, j) with $i, j \in N$ we consider. Moreover, for a given $i, j \in N$ we match up i and j only once. Thus the complexity is $O(|N|^2)$. □

Given a set of players N and the winning probabilities P between the players, the goal of the tournament designer is to come up with the tournament structure T and the seeding S that will maximize the probability of a given player $k \in N$ winning the tournament. This optimization problem has a decision version which

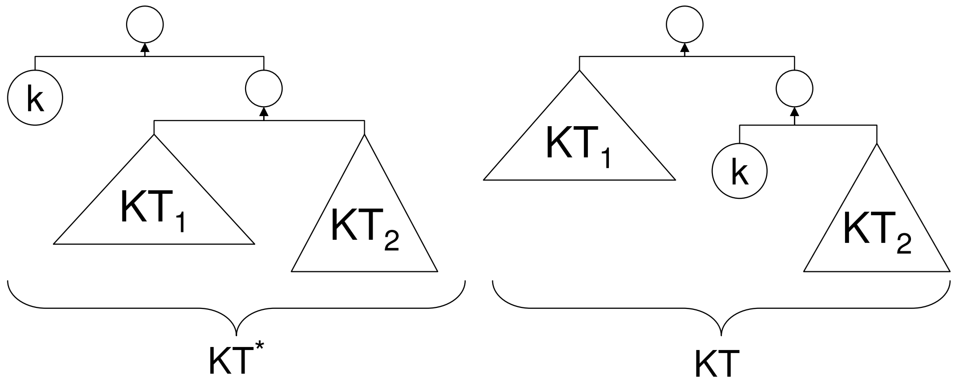

Figure 2: Biased knockout tournament KT^* that maximizes the winning chance of k and a general tournament KT

asks if there exist T and S such that the probability of k winning the tournament is greater than a given value θ.

The first intuition for the optimization problem is that the later any player plays in the tournament, the better chance she has of winning the tournament. We state and prove this intuition in the following proposition.

PROPOSITION 2. *Given a set of players N and the winning probability matrix P, the tournament structure that maximizes the probability of player $k \in N$ has the biased structure as KT^* in Figure 2 in which k has to play only the final match.*

PROOF. We prove this by induction.

Base case: When $|N| = 2$, there is only one possible binary tree with 2 leaf nodes.

Inductive step: Assume that the theorem holds for N with $|N| \leq n-1$. For any given $k \in N$, we will show that it also holds for N with $|N| = n$ by converting any tournament structure that does not have a biased structure to KT^* as in Figure 2 such that in KT^*, k has at least the same chance of winning.

Let's consider any given tournament structure KT that does not have the biased structure. Let KT_1 and KT'_2 be the two disjoint sub-tournaments that make up KT, and let N_1, N'_2 be the set of players assigned to KT_1, KT'_2 respectively. Assume wlog that $k \in N'_2$. Since $|N'_2| < |N|$, the chance of k winning the tournament is maximized when KT'_2 has the biased structure. Therefore we just need to compare the chance of k winning in KT with its chance in KT^* as shown in Figure 2:

$$q(k,KT) = \sum_{i \in N'_2 \setminus \{k\}} [P_{ki} \cdot q(i,KT_2)] \times \sum_{j \in N_1} [P_{kj} \cdot q(j,KT_1)]$$

$$q(k,KT^*) = \sum_{j \in N_1, i \in N'_2 \setminus \{k\}} P_{ki} \cdot q(i,KT_2) \cdot P_{ij} \cdot q(j,KT_1)$$
$$+ \sum_{j \in N_1, i \in N'_2 \setminus \{k\}} P_{kj} \cdot q(j,KT_1) \cdot P_{ji} \cdot q(i,KT_2)$$

$$q(k,KT^*) - q(k,KT) = \sum_{j \in N_1, i \in N'_2 \setminus \{k\}} [q(j,KT_1) \cdot q(i,KT_2) \cdot$$
$$(P_{kj}P_{ji} + P_{ki}P_{ij} - P_{ki}P_{kj})]$$

$P_{ij} + P_{ji} = 1 \Rightarrow$
$P_{kj}P_{ji} + P_{ki}P_{ij} \geq min\{P_{ki}, P_{kj}\} \geq P_{ki}P_{kj}$

Therefore we have $q(k,KT^*) \geq q(k,KT)$. □

Here we show that the biased structure in Figure 2 is optimal over any tournament structure, as opposed to a similar result in [4] that is only applicable for a very specific linear structure. Proposition 2 gives us the shape of the optimal tournament structure and lets us reduce the original problem to a smaller one. Yet it still remains an open question whether there exists an efficient algorithm to find the exact optimal schedule. Nevertheless, by placing certain natural constraints on the structures of the tournament or the winning probabilities of the players, we manage to get a better analysis of the problem. In the next section we will introduce the common constraints considered in the literature and the existing results in the settings with these constraints. We will also discuss the limitations of these results.

3. RELATED WORK

In the most common settings in the tournament design literature (see, e.g., [6, 1, 13]), the players are assumed to have intrinsic abilities and ranked based on these abilities. The abilities are unknown but the ranking is available to the designer. In this setting, the probability of one player winning against another is also known and is monotonic with regard to the rankings of the players, i.e., any player will have a higher chance of winning against a lower ranked player than winning against a higher ranked player. Besides this monotonicity constraint, the structure of the tournament is also restricted to be balanced binary tree. Most of the works in this setting focus on maximizing the winning probability of the highest ranked player. Yet the existing results are limited to very small cases of n, the number of players, such as $n = 4$ or $n = 8$. In our work, we generalize the objective function to maximizing the winning probability of any given player, not just the highest ranked one, and focus on asymptotic results instead.

Tournament design problems are also addressed in the context of voting. In [8], the candidates are competing in an election based on sequential majority comparisons along a binary voting tree. In each comparison, the candidate with more votes wins and moves on; the candidate with less votes is eliminated. Essentially, the candidates are competing in a knockout tournament in which the result of each match is deterministic. The probability of winning a match is either 0 or 1. In this setting, without any constraints on the structure of the voting tree, there is a polynomial time algorithm to decide whether there exists a voting tree that will allow a particular candidate to win the election. When the voting tree has to be a balanced binary tree, a modified version of the problem is NP-complete. In this version, there is a weight associated with each match between a pair of players, and the question becomes how to find the voting tree with the minimum weight that allows the target candidate to win the election.

The problem of finding the right voting tree (referred to as the control problem) is also addressed in [4] but with probabilistic comparison results instead. Here, the objective is finding a voting tree that allows a candidate to win the election with probability at least a certain value. Within this setting, the authors show that another modified version of the control problem is NP-complete. Besides the balanced tree constraint, the authors require the outcomes of the election to be "fair", i.e., the stronger candidate always wins each pairwise comparison. We provide a much more general result in our paper by not putting any restriction on the outcomes of the matches. They are determined solely by the winning probabilities between the players.

The computational aspects of other methods of controlling an election are also considered in [2, 5]. Here, the organizer of the election is trying to change the result of the election through controls (such as adding or deleting) of the voters or candidates. It has been shown that for certain voting protocols, some methods of control are computationally hard to perform. Nevertheless, our focus is not on using computational hardness to prevent manipulation but rather on providing an analysis on the complexities of tournament design problems.

4. A CONSTRAINT ON THE STRUCTURE OF THE TOURNAMENT

In Section 2, we have shown that the optimal general tournament structure is very unbalanced with the target player on one side and the rest of the players on the other side of the tree. One might say that this structure is unfair since the target player will have to compete only in the final match. One particular way to enforce fairness is to require the tournament structure to be a balanced binary tree (for simplicity, we assume that the number of players is a power of 2). This way, every player has to play the same number of matches in order to win the tournament.

Definition 3. **(Balanced Knockout Tournament)** Given a set N of players such that $|N| = 2^m$, a knockout tournament $KT = (T, S)$ is called balanced when T is a balanced binary tree.

Due to the attractiveness of this fairness between players, the balanced knockout tournament format has been widely addressed in the literature and is in fact the most commonly used format in practice. In this setting, since the structure of the tournament is fixed, the remaining control of the tournament designer is in the seeding of the tournament, i.e., the assignment of players to the leaf nodes of the tree. Thus our previous problem is reduced to finding the seeding that will maximize the winning probability of a particular player. Note that as we have mentioned in Section 1, even in this seemingly simple format, the number of different seedings to consider grows extremely fast with the number of players. Capturing this intuition, we have the following hardness result for the decision version of this problem:

THEOREM 1. *Given a set of players N and a winning probability matrix P, it is NP-complete to decide whether there exists a balanced knockout tournament KT such that $q(k, KT) \geq \delta$ for a given δ and $k \in N$.*

This theorem follows from Theorem 3. Therefore we will defer the discussion of the proof of this theorem to the next section. Since the decision version is NP-complete, it follows that optimization version of the problem is NP-hard. Note that the same result holds for any number of players (e.g., even when $|N|$ is not a power of 2). In this case, when there is an odd number of players at any round, we allow the tournament designer to let any player advance to the next round without competing. This allows certain bias, e.g., if the number of players is $2^m + 1$, there is an odd number of players at every round except the final, and the target player can actually advance straight to the final match. Nevertheless, it is still NP-hard to find the optimal structure for the target player. This can be proved by using a similar reduction in which we make sure that the target player is in fact the only plausible choice for the designer to advance to the next round when there is an odd number of remaining players.

5. CONSTRAINTS ON PLAYER MODEL

Besides the balance constraint on the structure of the tournament, we also address different constraints on the winning probabilities between the players. One such constraint is on the possible values that the probabilities can take, e.g., the deterministic constraint. Another constraint is a certain overall structure that the winning probability matrix need to satisfy, e.g., the monotonicity constraint. We will discuss both types of constraints below.

5.1 Win-Lose Match Results

The first constraint we consider is to require the result of each match to be deterministic, i.e., winning probabilities can only be either 0 or 1. As mentioned in Section 3, a knockout tournament in this setting is analogous to a sequential pairwise elimination election. Given a tournament structure, a player in the tournament will either win the tournament for certain (winning with probability 1) or will lose for certain (winning with probability 0). Note that the winning probability matrix can be any arbitrary binary matrix.

When there is no constraint on the structure of the tournament, as shown in [8], there exists a polynomial time algorithm to find the tournament structure that allows a target player k to win the tournament or decide that it is impossible for k to win. When the tournament has to be balanced, it is still an open problem.

We shall now discuss another problem model that we believe will be helpful for the understanding of the proof of Theorem 3. In this model, there is no constraint to the tournament tree, except that each player has to start from a pre-specified round. In other words, the tournament can take the shape of any binary tree, but each player has to start at certain depth of the tree.

Definition 4. **(Knockout Tournament with Round Placements)** Given a set N of players and a winning probability matrix P, a vector $R \in \mathbb{N}^{|N|}$, if there exists a knockout tournament KT such that in KT, player i starts from round R_i (the leaf nodes with the maximum depth in the tree are considered to be at round 1), then R is called a *feasible* round placement and such tournament KT is called a knockout tournament with round placement R. When there is an odd number of players at any given round, one player playing at that round can automatically advance to the next round.

Note that when all players have round placement 1, the tournament is balanced. We have the following hardness result:

THEOREM 2. *Given a set of players N, the winning probability matrix P such that $\forall i \neq j \in N$, $P_{ij} \in \{0, 1\}$, and a feasible round placement R, it is NP-complete to decide whether there exists a tournament structure KT with round placement R such that a target player $k \in N$ will win the tournament.*

PROOF. It is easy to show that the problem is in NP. We will show the problem is NP-complete using a reduction from the Vertex Cover problem.

Vertex Cover: Given a graph $G = \{V, E\}$ and an integer k, is there a subset $C \in V$ such that $|C| \leq k$ and C covers E?

Reduction method:
We construct a tournament $KT = (T, S)$ with a special player o and a round placement R such that o wins KT if and only if there exists a vertex cover of size at most k.

KT contains the following players[2]:

1. Objective player: o which starts at round 1.
2. Vertex players: $\{v_i \in V\}$ which start at round 1. There are $n = |V|$ such players.
3. Edge players: $\{e_i \in E\}$. There are $m = |E|$ such players. e_i starts at round $(n - k + i - 1)$.
4. Filler players: For each round r such that $(n - k + m) > r \geq (n - k)$, there is one filler player f^r that starts at round r. Thus there are a total of m of them. They are meant for player o.
5. Holder players: For each round r, there are a set of holder players h^r_i (i.e., multiple copies of h^r) that start at round r. These players are meant for the vertex players. The number of copies of h^r depends on the value of r:
   - If $1 \leq r \leq (n - k)$, there are $(n - r)$ copies
   - If $(n - k) < r \leq (n - k + m)$, there are $(k - 1)$ copies
   - If $(n - k + m) < r \leq (n + m)$, there are $(m + n - r)$ copies

The winning probabilities between the players are assigned as in Table 5.1. In a nutshell:

1. o only wins against v_i and f with probability 1 (always wins) and loses against all others with probability 1 (always loses).
2. v_i always wins against h^r, e_j that it covers, and $v_{i'}$ with $i' > i$. It always loses against all other players.
3. e_j always wins against h^r, f, $e_{j'}$ with $j' > j$.
4. Between two f^r players, the winner can be either one.
5. Between two h^r players, the winner can be either one.

The reduction is polynomial since the numbers of players in the tournament is polynomial.

We first need to show how to construct a schedule KT that allows o to win the tournament if there exists a vertex cover C of size at most k. The desired KT is composed of three phases:

Phase 1: Phase 1 is the first $(n - k)$ rounds. In this phase, we eliminate all vertex players that are not in C while keeping the remaining vertex players, and o. At each round r, match up o with $v' \notin C$ and let each of the $(n - r)$ holder players h^r match up with the remaining v_i. Notice that after each round, one vertex player gets eliminated and there is one less h^r. After $(n - k)$ rounds, there are k vertex players left corresponding to the vertices in C.

Phase 2: Phase 2 is the following m rounds. In this phase, we eliminate all edge players. For each round, we match up o with f^r. At each round r, there will be one edge player e starting at that round. We match e against $v_i \in C$ that covers it. For the remaining vertex players, we match them up with $(k - 1)$ holder players h^r. After m rounds, all of the edge players will be eliminated (since the k vertex players left form a vertex cover). The remaining players at the end of this phase are k vertex players and o.

Phase 3: Phase 3 is the final k rounds after Phase 2. In this phase, we eliminate the remaining vertex players. At each round, the number of new holder players starting at that round is one less than the number of remaining vertex players. We match up the vertex players with h^r, and o with the remaining v. At the end of this phase, only o remains.

For the other direction, we need to prove that o can win the tournament only if there is a vertex cover C of size k. First note that during Phase 1, for o not to get eliminated, it has to play against a vertex player v. Thus after the first $(n-k)$ rounds, there are at most k vertex players remaining (there can be less if two vertex players play against each other).

During Phase 2, the only way that an edge player e can be eliminated is to play against v that covers it or play against another edge player e' which started at an earlier round. If e is eliminated by e', there must be either h^r, v, or f^r that was eliminated earlier by an edge player e'' (which can possibly be e'). Since there is only $(k - 1)$ holder players at each round, if h^r was eliminated by e'', two vertex players must have played against each other and one of them must have been eliminated. If f^r was eliminated by e'', at that round r, o must have played against some v to advance. Thus for all cases, there is at least one v that got eliminated. Note that in this phase, at any round, there are only $(k + 1)$ new players. Therefore, at the end of this phase, there are exactly $(k + 1)$ players remaining including o. If all edge players get eliminated by vertex players, there are k vertex players remaining. If there is at least one e which did not get eliminated or got eliminated by another edge player but not a vertex player, there are less than k vertex players remaining.

Now during Phase 3, for o to win the tournament, o can only play against a vertex player. Thus the number of vertex players is reduced each round by 1. Moreover, since there are $(k - 1)$ holder players h^r starting at the first round of the phase, and one less for each round after that, if there are less than k vertex players at the beginning of Phase 3, there will be at least one non-vertex player remaining. If that is the case, at the last round of Phase 3, there must be at least one edge or holder player remaining and o will lose the tournament.

Therefore, for o to win the tournament, there must be k vertex players at the beginning of Phase 3. This implies all edge players must have been eliminated by vertex players during Phase 2. So each edge player must be covered by at least one of the remaining vertex players after Phase 1. Since there are at most k of them after Phase 1, these remaining vertex players form a vertex cover of size at most k. $\square$

After placing this constraint on the structure of the tournament tree, the tournament design problem has changed from easy to hard. This gives an indication that the design problem for balanced knockout tournament within this setting is probably also hard.

5.2 Win-Lose-Tie Match Results

When the match results are deterministic, it is an open problem whether there exists an efficient algorithm to find the optimal balanced knockout tournament for a given player. Surprisingly, when we allow there to be a tie between two players (each has equal chance of winning), the problem becomes provably hard.

THEOREM 3. *Given a set of players N, a winning probability matrix P such that $P_{ij} \in \{0, 1, 0.5\}$, it is NP-complete to decide*

[2]We overload some notations here but the given the context, it should be clear

	v_j	e_j	f^r	h_i^r
o	1	0	1	0
v_i	1 if $i \le j$, 0 otherwise	1 if v_i covers e_j, 0 otherwise	0	1
e_i	-	1 if $i \le j$, 0 otherwise	1	1
f^r	-	-	arbitrary	1
h_i^r	-	-	-	arbitrary

Table 1: The winning probabilities of row players against column players in KT

whether there exists a balanced knockout tournament KT such that $q(k, KT) \geq \delta$ for a given δ and $k \in N$.

The proof of Theorem 3 is similar to the proof of Theorem 2 with two modifications to the reduction:
1. We need to construct some gadgets that simulate the round placements, i.e., if player i starts from round r, player i will not be eliminated until round r. In order to achieve this, we will introduce $(2^r - 1)$ filler players that only player i can beat. This will keep player i busy until at least round r
2. We need to make sure that the round placement for any player is at most $O(log(n))$ with n equal to the size of the Vertex Cover Problem so that the size of the tournament is still polynomial.
The details of the proof for this theorem is included in the appendix.

Since Win-Lose-Tie match results is a special case of general winning probabilities, we can reduce the problem of finding the optimal balanced knockout tournament with Win-Lose-Tie match results to the problem of finding the optimal general balanced knockout tournament. This constitutes the proof for Theorem 1.

When there is no constraint on the structure of the tournament, there exists a polynomial time algorithm to either find a schedule that allows the target player to win with probability 1 or decide that such a schedule does not exist. This algorithm is a modification the algorithm introduced in [8] to compute possible winners. In the modified version, when there is a tie between 2 players, we remove all the edges between them in the tournament graph. We will then proceed to finding all the winning paths from the target player to other players in the tournament. If there is a winning path to all other players, there exists a schedule to make the target player win, and it is the binary tree formed by combining the winning paths.

5.3 Monotonic Winning Probabilities

Another natural constraint is to require a certain overall structure of the winning probability matrix P. One of the most common models in the literature is the monotonic model (see for example [6, 11, 7, 14]). In this model, the players are ranked from 1 to n in descending order of unknown intrinsic abilities. The tournament designer only know the rankings and the winning probabilities between the players, which are correlated to the intrinsic abilities.

Definition 5. **(Knockout Tournament with Monotonic Winning Probabilities)** A knockout tournament $KT = \{N, T, S, P\}$ has monotonic winning probabilities when the winning probability matrix P satisfies the following constraints:

1. $P_{ij} + P_{ji} = 1$
2. $P_{ij} \geq P_{ji} \qquad \forall(i, j) : i \leq j$
3. $P_{ij} \leq P_{i(j+1)} \qquad \forall(i, j)$

As in other settings, we can place the balance constraint on the structure of the tournament. However, similarly to the case of deterministic match results, when we require the tournament to be balanced, the complexity of finding the optimal tournament becomes unknown. Yet, when we relax this condition to allow small violation, we can obtain certain hardness result. We call the new condition ϵ-monotonicity.

Definition 6. **(Knockout Tournament with ϵ-Monotonic Winning Probabilities)** A winning probability matrix P is ϵ-monotonic with $\epsilon > 0$ when P satisfies the following constraints:

1. $P_{ij} + P_{ji} = 1$
2. $P_{ij} \geq P_{ji} \qquad \forall(i, j) : i \leq j$
3. $P_{ij} \leq P_{ij'} + \epsilon \qquad \forall(i, j, j') : j' > j$

As ϵ goes to 0, the winning probability matrix P will gets closer to being monotonic. Note that we only relax the second requirement of monotonicity. In this setting, the problem of finding the optimal balanced structure is provably hard:

THEOREM 4. *Given a set of players N, an ϵ-monotonic winning probability matrix P with $\epsilon > 0$, it is NP-complete to decide if there exists a balanced knockout tournament KT such that $q(k, KT) \geq \delta$ for a given δ and $k \in N$.*

The proof for the theorem is included in the appendix.

Here we assume ϵ and δ have the same precision. Since ϵ can be arbitrarily small, this result suggests that there does not exist an efficient algorithm to find the optimal balanced tournament for a target player in the setting with monotonic winning probabilities.

6. CONCLUSION AND FUTURE WORK

In this paper we have investigated the computational aspect of schedule control for knockout tournaments. We have considered several modelings of the problem based on different constraints that can be placed on the structure of the tournament or the model of the players. In particular, we have shown that when the tournament has to be balanced, the structure control problem is NP-hard, even when the match results can only be win, lose, tie, or when the winning probabilities between the players have to be ϵ-monotonic. We have also charaterized the optimal structure for general knockout tournaments. The results are summarized in Table 2 with new results in bold face.

When the match results are deterministic, the complexity of the control problem remains an open problem for future work. Other directions include finding optimal structure for other objective functions such as fairness or "interestingness" of the tournament, or considering other constraints on the tournament structure and player models.

7. REFERENCES

[1] D. R. Appleton. May the best man win? *The Statistician*, 44(4):529–538, 1995.

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

	General Prob.	Win-Lose-Tie	Win-Lose	ϵ-Mono	Mono
General Struct	Open	$\mathbf{O(n^2)}$	$O(n^2)$ [Lang'07]	Open	Open
Balanced Struct	**NP-hard**	**NP-hard**	Open	**NP-hard**	Open
Round-placements	**NP-hard**	**NP-hard**	**NP-hard**	**NP-hard**	Open

Table 2: Summary of the complexity results

[3] S. J. Brams and P. C. Fishburn. Voting procedures. In K. J. Arrow, A. K. Sen, and K. Suzumura, editors, *Handbook of Social Choice and Welfare*.

[4] N. Hazon, P. E. Dunne, S. Kraus, and M. Wooldridge. How to rig elections and competitions. In *COMSOC'08*, 2008.

[5] E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. Anyone but him: The complexity of precluding an alternative. *Artif. Intell.*, 171(5-6):255–285, 2007.

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

[7] F. K. Hwang. New concepts in seeding knockout tournaments. *The American Mathematical Monthly*, 89(4):235–239, apr 1982.

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

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

[10] G. C. Loury. Market structure and innovation. *The Quarterly Journal of Economics*, 93(3):395–410, August 1979.

[11] J. W. Moon and N. J. Pullman. On generalized tournament matrices. *SIAM Review*, 12(3):384–399, jul 1970.

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

[13] D. Ryvkin. The predictive power of noisy elimination tournaments. Technical report, The Center for Economic Research and Graduate Education - Economic Institute, Prague, Mar. 2005.

[14] A. J. Schwenk. What is the correct way to seed a knockout tournament? *The American Mathematical Monthly*, 107(2):140–150, feb 2000.

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

APPENDIX

PROOF OF THEOREM 3. Similar to the Proof of Theorem 2, we show here a reduction from the Vertex Cover problem.

Reduction method:
We construct a tournament $KT = (T, S)$ with a special player o such that o wins KT with probability 1 if and only if there exists a vertex cover of size at most k.

KT contains the following players:

1. Objective player: o
2. Vertex players: $\{v_i \in V\}$ and an extra special vertex v_0 which does not cover any edge. If we let $n = |V|$ then there are $n + 1$ vertex players.
3. Edge players: $\{e_i \in E\}$. There are $m = |E|$ edge players.
4. Filler players: For each round r such that $0 < r \leq \lceil log(n - k)\rceil$, there are k filler players $f^r_{v,i}$, i.e., there are k copies of f^r_v. These players are meant to keep at least k vertex players advancing to the next round. For each round r such that $\lceil log(n-k)\rceil < r \leq \lceil log(n-k)\rceil + \lceil log(m)\rceil$, there are k filler players $f^r_{e,i}$. These are meant for the edge players. We might refer to both types of filler players as f^r_i or simply f^r.
5. Holder players: For player e_i, there are $2^{\lceil log(n-k)\rceil} - 1$ edge holder players $h^l_{e_i}$. These will make sure no edge player will be eliminated before reaching round $\lceil log(n - k)\rceil + 1$. For each filler player f^r_i, there are $2^r - 1$ holder players $h^l_{f^r_i}$ that will make sure no filler player will be eliminated before reaching round r. There are also
$$K = 2^{\lceil log(n-k)\rceil + \lceil log(m)\rceil + \lceil log(k+1)\rceil + 1} - 1$$
special holder players h^l_o that will allow player o to advance to the final match.

The winning probabilities between the players are assigned as in Table 3. In a nutshell:

1. o only wins against v_i and h_o with probability 1 (always wins) and loses against all others with probability 1 (always loses).
2. v_i always wins against f^r (both f^r_v and f^r_e), e_j that it covers, and $v_{i'}$ with $i' > i$. It always loses against all other players. The special vertex player v_0 does not win against any edge player but wins against any other vertex player.
3. e_j always wins against h_{e_j}, f^r, and wins with probability 0.5 against another $e_{j'}$.
4. For the holder players, each of them only loses to the player it is meant for. For example, edge holder player $h^l_{e_j}$ only loses to the edge player e_j. Holder players tie when playing against each other or against an edge player (except for the edge holder players they are meant for). They always win against o and vertex players.

The reduction is polynomial since the number of players in the tournament is $O(K)$. Without loss of generality, we assume that the number of total players is a power of 2 because we can always add more h_o players and this will not affect the reduction shown below. Note that we consider the first round as round 1.

First we need to show how to construct a structure KT that let o win with probability 1 if there exists a vertex cover C of size at most k. The desired KT is composed of two phases:

Phase 1: Phase 1 is the first $\lceil log(n-k)\rceil$ rounds. In this phase, we eliminate all $v \notin C$ except the special vertex player v_0 while keeping o and all edge players. During this phase, for each player that has corresponding holder players, we will match them together. This will help each edge player e to get to round $\lceil log(n-k)\rceil + 1$, and each filler player f^r to get to round r. We also match o with the holder player h_o to help o advancing to the final round. At round $1 \leq r \leq \lceil log(n - k)\rceil$, if the vertex v_i is in C, we match the vertex player v_i with the filler player f^r. Otherwise we match it with another vertex player that is not in C. At the end of this phase , there are only $k + 1$ vertex players remaining. One of them is the special vertex player v_0.

Phase 2: Phase 2 is the following $\lceil log(m)\rceil + \lceil log(k+1)\rceil + 1$ rounds. In this phase, we eliminate all the edge players by repeat-

	v_j	e_j	$f_j^{r'}$	$h_{e_j}^{l'}$	$h_{f_j^{r'}}^{l'}$	$h_o^{l'}$
o	1	0	0	0	0	1
v_i	1 if $i < j$, 0 otherwise	1 if v_i covers e_j, 0 otherwise	1	0	0	0
e_i	-	0.5	0.5	1 if $i = j$, 0.5 otherwise	1	1
f_i^r	-	-	0.5	0.5	1 if $f_i^r = f_j^{r'}$, 0.5 otherwise	1
$h_{e_i}^l$	-	-	-	0.5	0.5	1
$h_{f_i^r}^l$	-	-	-	-	0.5	1
h_o^l	-	-	-	-	-	0.5

Table 3: The winning probabilities of row players against column players in KT

	v_j	e_j	$f_j^{r'}$	$h_{e_j}^{l'}$	$h_{f_j^{r'}}^{l'}$	$h_o^{l'}$
o	1	$1-\epsilon$	$1-\epsilon$	$1-\epsilon$	$1-\epsilon$	1
v_i	1 if $i < j$, 0 ow.	1 if v_i covers e_j, $(1-\epsilon)$ ow.	1	$1-\epsilon$	$1-\epsilon$	$1-\epsilon$
e_i	-	0.5	1	1 if $i = j$, $(1-\epsilon)$ ow.	1	1
f_i^r	-	-	0.5	0.5	1 if $f_i^r = f_j^{r'}$, 0.5 otherwise	1
$h_{e_i}^l$	-	-	-	0.5	0.5	1
$h_{f_i^r}^l$	-	-	-	-	0.5	1
h_o^l	-	-	-	-	-	0.5

Table 4: The ϵ-monotonic winning probabilities of row players against column players in KT

edly matching each vertex player with the edge players that it covers. If there are more edge players than vertex players, we will match the remaining edge players who are covered by the same vertex player with each other. If there is any edge player that does not have a match, we will match it with the edge filler player f_e^r. Note that there are at most k edge players that do not have a match. After each round, at least half of the edge players will be eliminated. Thus after at most $\lceil log(m) \rceil$ rounds, all the edge players will be eliminated. There will be only vertex players v, o and h_o remaining. We just need to match them up until o is the only player left since o wins against v and h_o with probability 1.

For the other direction, we need to prove that o can win the tournament with probability 1 only if there is a vertex cover C of size k. We need to show that if o wins with probability 1, after Phase 1, there will be only $k+1$ vertex players remaining including the special vertex v_0, and during Phase 2, all edge players will be eliminated by one of those remaining vertex players, i.e., no edge players gets eliminated during phase 1.

First note that for o to win with probability 1, no holder players except h_o can reach the final. Thus in the first $\lceil log(n-k) \rceil$ rounds, no edge player can get eliminated since there are $(2^{\lceil log(n-k) \rceil} - 1)$ holder players for each edge player. Also no filler player f^r can get eliminated before round r. At round r such that $r \leq \lceil log(n-k) \rceil$, the only way for a vertex player to advance to the next round is either playing against a filler player f^r or another vertex player. It cannot advance by playing against an edge player that it covers, since that would eliminate that edge player too early. It cannot advance by playing a filler player $f^{r'}$ with $r' > r$ either, since that would eliminate $f^{r'}$ too early. Therefore, besides k vertex players playing against f^r, at least half of the remaining vertex players will be eliminated after each round. At the end of round $\lceil log(n-k) \rceil$, there can be only at most $k+1$ vertex players remaining. Note that since vertex player v_0 wins against any other vertex players, it must still remain.

For o to win the tournament with probability of 1, o must not play against any edge players either. Moreover, note that when two edge players play against each other, each of them has a 50% chance moving on to the next round. Therefore the only way that an edge player gets eliminated is to play against a vertex player that covers it. So, each edge player must be covered by at least one of the remaining k vertex players (since v_0 does not cover any edge). Consequently the set of k remaining vertex players forms a vertex cover of size k. $\square$

PROOF OF THEOREM 4. To prove this theorem, we show a reduction from the Vertex Cover Problem to the tournament design problem in this setting. The reduction is similar to the proof of Theorem 3 with the same set of players but with slightly different winning probabilities (shown in Table 4). Essentially, we convert the probabilities of a vertex player winning against edge players that it does not cover from 0 to $(1-\epsilon)$. Similarly for o, it now either wins with probabilities 1 as before or with probabilities $(1-\epsilon)$. The new winning probabilities are ϵ-monotonic with this ordering of players in descending strengths: o, v_0... v_n, e_1... e_m, f^r, h_e, h_{f^r}, h_o. Note that for o to win the tournament with probability 1, she can only play against v and h_o. Thus all players of other types must be eliminated with probability 1. This allows the proof of Theorem 3 to hold in this setting. $\square$