KNOCKOUT TOURNAMENT DESIGN:
A COMPUTATIONAL APPROACH

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Thuc D. Vu
August 2010

© 2010 by Thuc Duy Vu. All Rights Reserved.

Re-distributed by Stanford University under license with the author.

This work is licensed under a Creative Commons Attribution-Noncommercial 3.0 United States License.
http://creativecommons.org/licenses/by-nc/3.0/us/

This dissertation is online at: http://purl.stanford.edu/qk299yx6689

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Yoav Shoham, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Matthew Jackson

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Timothy Roughgarden

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Preface

Knockout tournaments constitute a very common and important form of social institution. They are perhaps best known in sporting competitions, but also play a key role in other social and commercial settings as they model a specific type of election scheme (namely, sequential pairwise elimination election). In such tournaments the organizer controls the shape of the tournament (a binary tree) and the seeding of the players (their assignment to the tree leaves).

A tournament can involve millions of people and billions of dollars, and yet there is no consensus on how it should be organized. It is usually dependent on arbitrary decisions of the organizers, and it remains unclear why one design receives precedence over another. The question turns out to be surprisingly subtle. It depends among other things on (a) the objective, (b) the model of the players, (c) the constraints on the structure of the tournament, and (d) whether one considers only ordinal solutions or also cardinal ones.

We investigate the problem of finding a good or optimal tournament design across various settings. We first focus on the problem of tournament schedule control, i.e., designing a tournament that maximizes the winning probability of a target player. While the complexity of the general problem is still unknown, various constraints – all naturally occurring in practice – serve to push the problem to one side or the other: easy (polynomial) or hard (NP-complete).

We then address the question of how to find a fair tournament. We consider two alternative fairness criteria, adapted from the literature: envy-freeness and order preservation. For each setting, we provide either impossibility results or algorithms (either exact or heuristic) to find such a fair tournament. We show through experiments that our heuristics are both efficient and effective.

Finally, using a combination of analytic and experimental tools we investigate the optimality of ordinal solutions for three objective functions: maximizing the predictive power, maximizing the expected value of the winner, and maximizing the revenue of the tournament. The analysis relies on innovative upper bounds that allow us to evaluate the optimality of any seeding, even when the number of possible seedings is extremely large.

Acknowledgements

First and foremost, I would like to thank my advisor, Yoav Shoham, for his advice and guidance over the years. I have learned many things from him, not just in academics but also things about life and myself. I am grateful to have met him and to be able to work with him. I would also like to thank other members of my committee: Matthew Jackson, Tim Roughgarden, Jean-Claude Latombe, and Balaji Prabhakar. I would like to especially thank my past mentors and advisors: Manuela Veloso, and Nguyen T. Hung. Thay Hung introduced me to CS in high school and taught me most of what I know about algorithms, and Manuela jump-started my passion for doing research while I was at CMU.

I also wish to thank past and current members of the multi-agent group (which includes Mike, Chris, Nicolas, Ashton, Sam, and Rob) for the invaluable aids and supports they have provided. I will especially miss all the long and stimulating research discussions, as well as all the (very) late nights frantically trying to finish the papers before the deadlines. Special thanks to Mike; he has been not only a great officemate but also a true friend and companion. The journey we have shared this far has been long, and challenging, but Mike has made it a lot less painful than it could have been.

I would like to also thank my friends (too many of them to name here) who have shared various parts of my life. They have made my life much more enjoyable than I could ever hope for. Special thanks to My, who has not only brought joys and meanings to my life but also touched my heart beyond words. I am truly lucky to have met her. And last but also most importantly, I would like to thank my family for their unconditional love, confidence, and support. They have been my inspirations and motivations (and pressures too, sometimes). Without them, I would have never made this far. This thesis is in fact dedicated to them.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Tournaments[1] are a very common and very important form of social institution. They are perhaps best known in sporting competitions such as the Wimbledom, NCAA college basketball, or the FIFA playoffs. These competitions attract millions of television viewers and make billions of dollars annually. In addition to their role in sporting competitions, 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 [26, 30, 20] for details and further discussion).

Although tournaments represent just one type of competition format, many different variations of tournaments are possible. In general, 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 tournament outcome 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 our work we focus on *knockout* tournament. The knockout tournament is among the simplest tournament formats; perhaps for this reason it is also among the most popular. In such tournaments, players are initially 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.

[1]Throughout this thesis, when we write "tournament", we intend the everyday meaning of the term, rather than its graph theoretic interpretation [22, 23].

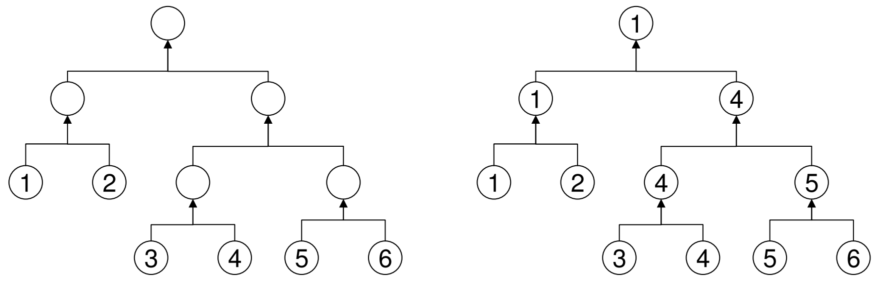

Figure 1.1: An example of a tournament structure for 6 players and one possible outcome

We show an example in Figure 1.1. The tournament organizer can only change the shape of the binary tree (which is the *tournament structure*) and the arrangement of the players along the leaves of the tree (which is known as the *tournament seeding*).

Intuitively, the structure and the seeding of a tournament can significantly influence the outcome. Yet it is unclear why one particular design should be favored over another, or what is the optimal way to organize a tournament. These seemingly simple questions turn out to be surprisingly subtle and some of the answers are counter-intuitive. First of all, we need to specify our objective function, i.e., what we are trying to optimize for. It can be the probability that the strongest player will win the tournament, the revenue of the tournament, or something else. Moreover, we also need to be clear about other aspects of the setting such as the model of the players[2] (which decides the match outcomes), the information available to the organizer, the constraint on the structure of the tournament, or the size of the tournament. Since there are several choices for each of these quantities, the result is a large space of design problems.

Over the past 40 years this space has been explored very partially. Most of the previous work was limited to 4 or 8 players, to specific probabilistic models of match outcomes, to ordinal solutions (where the only information available to the tournament designer is the ordering among players in terms of strength), and to one specific objective function: maximizing the predictive power (that is, the probability that the strongest player will win the tournament). We will be more precise about these prior results in Chapter 3 after we

[2]We use the terms “player model” and “winning probability model” interchangeably throughout the text

introduce the formal model in Chapter 2, but this crude description is sufficient to explain, in general terms, our contributions.

We dramatically broaden the scope of the analysis along several dimensions with the main focus on the objective function. First we generalize maximizing the predictive power to maximizing the winning probability of any given player. This is also known as the schedule control problem. In Chapter 4, we provide an analysis of the computational complexity of the problem in various settings. We start with the most general model, and then investigate the problem under two types of constraints: constraints on the player model, and constraints on the allowable tournament structure. The various constraints we consider – all naturally occurring in practice – serve to make the problem either easy (polynomial time computable) or hard (NP-complete).

In Chapter 5, we consider the existence of fair seeding in knockout tournaments. We define two fairness criteria, both adapted from the literature: envy-freeness and order preservation. We show how to achieve the first criterion in tournaments whose structure is unconstrained, and prove an impossibility result for balanced tournaments. For the second criterion we have a similar result for unconstrained tournaments, but not for the balanced case. We provide instead a heuristic algorithm which we show through experiments to be efficient and effective. This suggests that the criterion is achievable also in balanced tournaments. However, we prove that it again becomes impossible to achieve when we add a weak condition guarding against the phenomenon of tournament dropout.

In Chapter 6, we address a set of three objective functions: the predictive power, the expected strength of the winner, and the revenue of the tournament. We provide both worst-case and average-case analysis; since the number of distinct seedings is large, this analysis relies on a novel algorithm for computing an upper bound on optimal seeding. We consider how the optimality of different seedings varies across different objectives. We also propose an efficient method to improve the results when additional information is available.

Chapter 2

Formal Descriptions of the Settings

2.1 The Most General Setting

We start 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.

Definition 1. (General Winning Probability Model) *Given a set of n players, the winning probabilities between the players form a matrix P such that P_{ij} denotes the probability that player i will win against player j, $\forall (i \neq j) : 1 \leq i, j \leq n$, and P satisfies the following constraints:*
1. $P_{ij} + P_{ji} = 1$
2. $0 \leq P_{ij}, P_{ji} \leq 1$

Given the winning probabilities between the players, we define a knockout tournament as follows:

Definition 2. (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 $0 \leq P_{ij} = 1 - P_{ji} \leq 1$ ($\forall i, j \in N$), a knockout tournament $KT_N = (T, S)$ is defined by:*

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

We use KT to denote KT_N when the context is clear.

To carry out the tournament, each pair of players that are assigned to sibling leaf nodes will 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 sub-tournament branch that shares the same parent node. 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 3. (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*
$$q(k, KT_N) = \begin{cases} 1 & \textit{if } k = j \\ 0 & \textit{if } k \neq j \end{cases}$$

2. *If $|N| \geq 2$, 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 time 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 $O(|N|^2)$. □

2.2 Constraints on the Tournament Structure

Knockout tournaments usually do not have an unconstrained tournament structure in practice. The reason is that it can be very unfair, e.g., one player might be advanced straight to the final match. One particular way to enforce fairness is to require the tournament structure to be a balanced binary tree when it is possible, i.e., when 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 4. (Balanced Knockout Tournament) *Given a set N of players such that $|N| = 2^m$, a knockout tournament $KT = (T, S)$ is a balanced knockout tournament 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 (e.g., see [15, 1, 27, 12]) and is in fact the most commonly used format in practice.

When $|N|$ is not a power of 2, however, there must be some rounds in which the number of remaining players is odd. Thus the tournament tree cannot be a balanced binary tree anymore. In this case, to maintain a degree of fairness in the tournament, we require that at most one player can advance to the next round without competing, i.e., at most one player receives a “bye”. However, the tournament organizer can pick any player to give that bye to.

Definition 5. (Generalized Balanced Knockout Tournament) *Given a set N of players, a knockout tournament $KT = (T, S)$ is called balanced when at every round:*

1. *If the number of the remaining players is even, all of them must compete in a match and the winners will advance to the next round; and*
2. *If the number of the remaining players is odd, only one of them can advance to the next round without competing.*

Note that when the number of the players is $2^m + 1$, it is possible for the organizer to arrange for a player to advance straight to the final match without competing. One can place additional constraints on how the bye’s are given to further ensure fairness (e.g., no bye is given to the same player in two consecutive rounds). However, such a discussion lies beyond the scope of our text.

2.3 Constraints on the Winning Probability Model

Besides the constraints on the structure of the tournament, in the literature there are several other constraints on the winning probabilities between the players. The constraints can be either on the possible values that the probabilities can take or on the overall structure of the winning probability matrix.

For the first type of constraint, we consider the deterministic model, in which the winning probabilities can only be either 0 or 1. A knockout tournament in this setting is analogous to a sequential pairwise elimination election. We will discuss this connection in more details in Chapter 3. 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).

For the second type of constraint, we consider the monotonic model. This model is popular and well known in the literature (see for example [24, 15, 16, 28]). The players are assumed to have unknown but fixed intrinsic strengths or abilities. They are numbered from 1 to n in descending order of their strengths and the winning probabilities between the players reflect these rankings.

Definition 6. (Monotonic Winning Probability Model) *Given a set of n players, the winning probabilities between the players form a matrix P such that P_{ij} denotes the probability that player i will win against player j, $\forall(i \neq j) : 1 \leq i, j \leq n$, and P satisfies the following constraints:*
1. $P_{ij} + P_{ji} = 1$
2. $0 \leq P_{ij}, P_{ji} \leq 1$
3. $P_{ij} \leq P_{i(j+1)}$
4. $P_{ji} \geq P_{(j+1)i}$ *(which is actually implied by (1) and (3))*

In other words, the monotonic condition means it is always easier for players to win against opponents with worse rankings than the ones with better rankings. This is often the basis on which major sport tournaments decide on their seeding, inheriting player ranking from the relevant sporting association.

Chapter 3

Related Work

There are two main types of approaches in tournament design. The first one is axiomatic: different criteria are proposed for a seeding to be judged as a “good” seeding. The second one is qualitative: maximizing a specific quantity of the tournament. We describe these two types in Section 3.1 and Section 3.2. Tournament design problems are also discussed in economics literature and addressed under the context of voting theory. We present the connections in Section 3.3 and 3.4 respectively.

3.1 Axiomatic Approaches

Within axiomatic approaches, different criteria are proposed for seeding evaluations. Most of the work here focuses on balanced knockout tournaments with the monotonic winning probability model and ordinal solution (which is generated by using only the ordering of the players). In [28], three axioms are proposed to specify what a good seeding should satisfy. The axioms are called “Delayed Confrontation”, “Sincerity Rewarded”, and “Favoritism Minimized”:

- *Delayed Confrontation:* Two players rated among the top 2^j players shall never meet until the number of players has been reduced to 2^j or fewer.
- *Sincerity Rewarded:* A higher-ranked player should never be penalized by being given a schedule more difficult than that of any lower ranked.

- *Favoritism Minimized:* The schedule should minimize favoritism to any particular rank.

In order to achieve these axioms, the author in [28] introduces a seeding method in which for $n = 2^m$ players, they are divided into m cohorts with cohort C_i $(1 \leq i \leq m)$ consisting of players in the set $\{2^{i-1} + 1, ..., 2^i\}$. The players within each cohort will then be randomly placed at the pre-assigned positions of the cohort. The randomized seeding is shown to satisfy all three axioms on expectation. However, for a chosen seeding after the randomization, it can be grossly unfair.

Alternatively, in [16], a different property called "Monotonicity" is used as a requirement for a good seeding:

- *Monotonicity:* The probability of a given player winning the tournament is higher than all of the weaker players.

To achieve this property, the players are re-seeded after each round such that the strongest player remained faces the weakest, the second-strongest faces the second-weakest, so on and so forth.

3.2 Qualitative Approaches

In the second type of approach, rather than placing axiomatic constraints on the design, the goal is to optimize certain quantity. The most common objective function is to find the design that maximizes the winning probability of the best player. This probability is also called the predictive power of the tournament. This has been the focus of much work (see, e.g., [15, 1, 27, 12]). They all focus on balanced tournament and monotonic winning probability model. However, all the results are limited to very small cases of n (up to 8 players).

In [15, 12], the authors show that with the monotonic model, for $n = 4$, the seeding sequence $(1, 4, 3, 2)$ will maximize the predictive power, and the expected strength of the winner (regardless of the actual probability and strength values as long as they are monotone). Here we encode each seeding for a tournament of n players by a permutation of the numbers between 1 and n. The k^{th} number is the ranking of the player that will be placed at the k^{th} leaf node of the binary tournament tree, from left to right.

For $n = 8$, it is shown in [15] that there is no longer one solution that is always optimal for maximizing the predictive power. The optimal solution can in fact be any one of the following 8 seeding sequences:

$(1, 8, 7, 6, 2, 3, 4, 5)$	$(1, 7, 5, 6, 2, 4, 3, 8)$
$(1, 8, 5, 7, 2, 3, 4, 6)$	$(1, 8, 5, 6, 2, 4, 3, 7)$
$(1, 8, 5, 6, 2, 3, 4, 7)$	$(1, 8, 5, 7, 2, 4, 3, 6)$
$(1, 7, 5, 6, 2, 3, 4, 8)$	$(1, 8, 6, 7, 2, 4, 3, 5)$

Which sequence in the set will be optimal depends on the winning probability matrix P. They also show examples in which each of the sequences is the optimal seeding.

In general, the optimality of S_1^n for MaxP is lost when there is a disruption in the "smoothness" of the winning probabilities, e.g., two players have an equal chance of beating each other but one of them has a much better chance of winning against a third team. In [27], smoothness is enforced by making the assumption that each player i has an ability v_i drawn from certain distributions and that the winning probabilities are specified through the Thurstone-Mosteller Model: $P_{ij} = \Phi(v_i - v_j)$ where $\Phi(\cdot)$ is the CDF of the standard normal distribution. With these assumptions, the probability of the best player winning the tournament is calculated numerically for each of the possible seeding sequences for $n = 8$ and $(1, 8, 6, 7, 2, 3, 4, 5)$ (a generalized version of $(1, 4, 3, 2)$) is shown through experiments to be indeed the optimal seeding.

3.3 Tournament Design in Economics Literature

In economics literature, one of the most common objective functions is to maximize the total effort of the players (e.g., see [12, 9, 21, 26]). There are one or more prizes associated with matches in the tournaments. The values of the prizes to the players are common knowledge and might not be equal. The result of each match is decided based on the efforts of the players (who incur some cost based on the amount of effort they exert). The players will then strategically decide their effort levels to maximize their utilities, given the values of the prizes and the efforts of other players. The goal of the organizers is to find a prize structure and/or a tournament schedule that maximize the total efforts of the players.

Several variations of this setting are addressed in the literature. The tournament can be a single prize winner-take-all contest [30], or a multi-prize contest [21]. Different cost functions and utility functions are also considered. Most of the work here focus on providing an equilibrium analysis for tournaments of very small sizes (up to 4 players). For example, it is shown in [12] that when the tournament is single-prize, the seeding [1,3,2,4] maximizes both the total effort across the tournament and also the probability of a final among the two top players. Other objective functions such as maximizing competitive balance, or uncertainty of outcome (hence the "interestingness" of the tournament) are also addressed (see [29] for an extensive overview).

3.4 Connections to Social Choice Theory

Tournament design has a strong connection with social choice theory – the theory of collective decision-making [2]. They correspond to a particular class of elections studied within voting theory, namely sequential elimination voting with pairwise comparison [4, 19]. In such elections, the players are the candidates, and each match represents a pairwise election. The result of each match is decided based on the votes. The winner of each match proceeds to the next round and the last candidate remains is the winner of the election.

Specifically, the problem of finding the optimal knockout tournament is addressed in the social choice context as a type of election control problems, namely the *agenda control* problem. With an election control, the organizer attempts to achieve strategic results by influencing the way in which the election is held, e.g., by adding, or removing candidate(s) or voter(s), or arranging the agenda of the pairwise comparisons. When the goal is to make sure that a given candidate will win the election (in the case that the pairwise comparisons are deterministic), or to maximize the probability that a given candidate will win (probabilistic pairwise comparisons), the agenda control problem is analogous to knockout tournament design problem with the same objective function. Indeed, both cases are addressed in [18] and [13] respectively. In these papers, the authors focus on analyzing the complexity of finding the optimal schedule. They show some modified versions of the control problem are NP-hard.

Chapter 4

Maximizing Winning Probability

4.1 Introduction

Although knockout tournaments are perhaps best known for their role in sporting competitions, they also play an important role in social choice theory: they correspond to sequential elimination elections with pairwise comparison [4, 19]. In such elections, the players are the candidates, and each match represents a pairwise election, rather than a sporting match; but otherwise the process is identical.

What makes the connection to elimination elections particularly striking is the objective function we consider in this chapter: maximizing the winning probability of a given player. This is also known as the schedule control problem. In knockout tournaments, as in sequential elimination voting with pairwise comparison, the tournament organizer is able to control the schedule and the seeding of the tournament. Everything else is outside the control of the tournament designer. The question we tackle is how an organizer can best exercise this limited control in order to optimize a certain quantity. Specifically, we focus on maximizing the winning probability of a given target player.

Of course, such control is usually viewed as undesirable, as it suggests biasing the tournament or rigging the election. In studying the difficulty of such manipulation, we do not condone it, and indeed the results can be used to prevent manipulation rather than enable it. This is a natural question in the context of voting; there are several other objectives which we address in the following chapters. Thus, while in general the theory of tournaments is quite different from the theory of voting, they coincide when we speak about schedule control to maximize winning probability 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 number of players, i.e., $O(\frac{n!}{2^{n-1}})$ 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 the results highly depend on the modeling of the problem. 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 [15, 13] 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 (polynomial time computable) or hard (specifically, NP-complete). In this paper, we discuss these settings and analyze the complexity of the problem in each setting.

The remainder of the chapter is organized as follows. We describe our problem using the general model for tournament in Section 4.2 and the related work in Section 4.3. We discuss different constraints that can be placed on the model in Sections 4.4 and 4.5, and describe our results for these settings. We summarize the results of this chapter in Section 4.6 and suggest some possible directions for future work.

4.2 The General Model and Problem

We start with the most general model of a knockout tournament as described in Chapter 2. In this setting, there is no constraint on the structure of the tournament, as long as it only allows pairwise matches between players. There is also no constraint on the winning probabilities between the players though we assume that all the probabilities are known.

Given a set of players N, the winning probability matrix P, and a given target player $k \in N$, the aim of the schedule control problem is to find a tournament structure T and seeding S that will maximize the probability of player k winning the tournament. Note that this is an optimization problem, which has a natural decision version that asks if there exist T and S such that the probability of k winning the tournament is greater than a given value δ.

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.

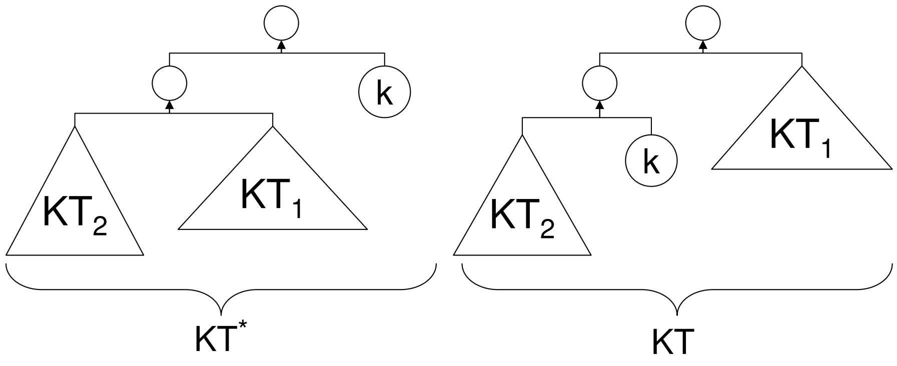

Figure 4.1: Biased knockout tournament KT^* that maximizes the winning chance of k and a general tournament KT without the biased structure

Proposition 2. *Given a set of players N and the winning probability matrix P, the tournament structure that maximizes the winning probability of player $k \in N$ has the biased structure as KT^* in Figure 4.1 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 4.1 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 4.1:

$$
\begin{aligned}
q(k, KT) &= \sum_{i \in N_2' \setminus \{k\}} [P_{ki} \times q(i, KT_2)] \times \sum_{j \in N_1} [P_{kj} \times q(j, KT_1)] \\
q(k, KT^*) &= \sum_{j \in N_1, i \in N_2' \setminus \{k\}} P_{ki} \times q(i, KT_2) \times P_{ij} \times q(j, KT_1) \\
&\quad + \sum_{j \in N_1, i \in N_2' \setminus \{k\}} P_{kj} \times q(j, KT_1) \times P_{ji} \times q(i, KT_2)
\end{aligned}
$$

$$
\begin{aligned}
q(k, KT^*) - q(k, KT) = \sum_{j \in N_1, i \in N_2' \setminus \{k\}} & [q(j, KT_1) \times q(i, KT_2) \\
& \times (P_{kj}P_{ji} + P_{ki}P_{ij} - P_{ki}P_{kj})]
\end{aligned}
$$

  Since $P_{ij} + P_{ji} = 1$, we have $P_{kj}P_{ji} + P_{ki}P_{ij} \geq min\{P_{ki}, P_{kj}\} \geq P_{ki}P_{kj}$.

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

□

Here we show that the biased structure in Figure 4.1 is optimal over *any* tournament structure, as opposed to a similar result in [13] that is only applicable for a very specific *linear* structure. Even though it still remains an open question whether there exists an efficient algorithm to find the exact optimal schedule, Proposition 2 indicates that the optimal schedule will be heavily biased towards the target player. This makes the problem of finding the optimal general schedule become less interesting since such a biased schedule is very undesirable in practice. Indeed, there are natural constraints that can be placed on the structure of the tournament to increase the fairness. 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.

4.3 Related Work

The work in this chapter lies in the general domain of *computational social choice theory*, a research area at the intersection of AI, theoretical computer science, and social choice theory. Computational social choice theory has attracted much interest recently [7]. Much of this interest stems from the possibility that computational complexity may provide a "solution" to some impossibility results in voting theory [4].

Specifically, a very well-known result in voting theory is the *Gibbard-Satterthwaite theorem*, which, crudely put, says that any voting protocol that is not a dictatorship must inherently be susceptible to *strategic manipulation* by voters. In other words, in any non-dictatorial voting protocol, there will be situations in which voters can benefit by lying about their preferences. However, the Gibbard-Satterthwaite theorem only says that voters can benefit from manipulation by misrepresenting their preferences *in principle*: it does not say that manipulation is *computationally feasible*. This observation led [3] to consider whether there were voting protocols in which manipulation by misrepresenting preferences is computationally difficult (NP-hard or worse). They were able to answer this question in the affirmative, showing that a voting protocol called "second-order copeland" was computationally hard to manipulate. This work subsequently led to many other researchers studying the circumstances under which voting protocols are computationally easy or computationally hard to manipulate. For example, [5] considered how many candidates are required in order for manipulation to be possible or impossible, while [6] discussed general approaches to designing hard-to-manipulate voting procedures, based on the idea of combining protocols. The average case complexity of manipulating elections (as opposed to asymptotic complexity) is considered in [25].

Considering tournaments, in the most common settings in the tournament design literature (see, e.g., [15, 1, 27]), 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 complexity results instead.

Tournament design problems have also been considered in the context of voting. In [18], 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 [13] 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 [3, 14]. 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.

4.4 A Constraint on the Structure of the Tournament

In Section 4.2, 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. This way, every player has to play the same number of matches in order to win the tournament. Note that the number of players might not be a power of 2. We consider both cases below.

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. As noted in Section 4.1, even in this more constrained format, the number of different seedings to consider still grows extremely fast with the number of players (in the order of $O(\frac{n!}{2^{n-1}})$) Capturing this intuition, we have the following hardness result for the decision version of this problem:

Theorem 3. *Given a set of players N such that $|N| = 2^m$ for some $m \geq 1$ 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 $k \in N$.*

In fact, this theorem is a special case of Theorem 7, which we prove later in this chapter. We therefore defer the discussion of the proof of this theorem to the next section. Since the decision version is NP-complete, it follows that the optimization version of the problem is NP-hard.

When $|N|$ is not a power of 2, we use the generalized balanced knockout tournament as defined in Chapter 2. We require that at most one player can advance to the next round without competing, i.e., at most one player receives a “bye”. However, the tournament organizer can pick any player to give that bye to. Let $|N| = 2^m + l$. Notice that the shape of the tournament can vary significantly for different values of l. We therefore redefine the scheduling problem dependent on l as the following.

Problem 1. **(l-BKT)** *Given a set N of players such that $|N| = 2^m + l$ and $l < 2^m$, and the winning probability matrix P, does there exist a balanced knockout tournament $KT = (T, S)$ such that $q(k, KT) \geq \delta$ for a given δ and $k \in N$?*

When $l = 1$, it is possible for the organizer to arrange for a particular player to advance straight to the final match. However, surprisingly, the scheduling problem is still hard in this case.

Theorem 4. *For any fixed l, the l-BKT problem is NP-complete.*

We prove this theorem by using Theorem 3 and the following lemma:

Lemma 1. *For any fixed l, the 0-BKT problem can be reduced to the l-BKT problem in polynomial time.*

Proof. Given a set N_1 of players with $|N_1| = 2^m$, the winning probabilities between the players, and a target player o_1 in the 0-BKT problem, we show how to construct a set N of players and a special player $o_2 \in N$ such that $|N| = 2^m + l$, and there exists a tournament schedule for N that allows o_2 to win with probability at least δ if and only if there exists a schedule for N_1 that allows o_1 to win with probability at least δ. Let N_2 be a set of l players. One of them we call o_2. We design the winning probabilities between the players as the following:

- All players in N_2 lose with probability 1 to all players in N_1 except o_1

- o_1 loses to all players in N_2
- o_2 wins with probability 1 against all other players in N_2
- The winning probabilities between other players in N_2 are arbitrary

The first direction is easy. Given a tournament KT_1 for N_1 such that o_1 wins with probability δ, we construct the tournament KT as in Figure 4.2. The tournament contains two sub-tournaments: KT_1 for N_1 and KT_2 for N_2. Since the size of N_1 is a power of 2, after o_2 eliminates all other players in N_2, the number of remaining players is $2^{m'}+1$. This allows o_2 to advance to the final match, where it can play against o_1 who has a probability of at least δ getting to the final match. As o_2 wins against o_1 with probability 1, o_2 will win the tournament KT with probability at least δ.

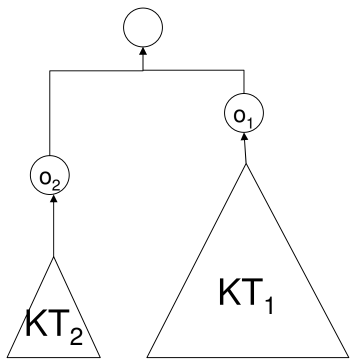

Figure 4.2: The tournament KT in the proof of Lemma 1

For the other direction, first notice that if there are some players in N_1 who play against some players in N_2, those players in N_1 will advance to the next round and the number of remaining players in N_1 will not be a power of 2 anymore in the subsequent rounds unless all the players in N_2 are eliminated. To see this let's assume we have $2^{m'}+k_1$ players of N_1 (with $0 \leq k_1 < 2^{m'}$) and l' players of N_2 remaining. Since there is at most one player who can get a bye at each round, $2^{m'}+k_1+l' \leq 2^{m'+1}$. If there are k_2 players in N_1 playing k_2 players in N_2 in the next round, the remaining players of N_1 after the next round is: $n_1 = 2^{m'-1} + \frac{k_1+k_2}{2}$. Since $2^{m'}+k_1+l' \leq 2^{m'+1}$ and $k_2 \leq l'$, we have $2^{m'-1} \leq n_1 \leq 2^{m'}$. Thus if n_1 is a power of 2, it can only be either $2^{m'-1}$ when $k_1 = k_2 = 0$, or $2^{m'}$. For the second case, when $n_1 = 2^{m'}$, this means $k_1 + k_2 = 2^{m'}$. However since $k_1 + k_2 \leq k_1 + l' \leq 2^{m'}$, it must be the case the $k_2 = l'$, and therefore, all the players in N_2 must be eliminated.

Since $l < 2^m$, if o_2 wins the tournament with probability at least δ, when all other players in N_2 are eliminated, some players in N_1 must remain. o_1 must be one among them otherwise o_2 will win the tournament with probability 0. If some players in N_1 have played against some other players in N_2 in previous rounds, the number of remaining players in N_1 must be $2^{m'} + k_1$ with $2^{m'} > k_1 > 0$. If the total number of remaining players is even, player o_2 will have to play against some player in N_1 but not o_1 in either this round or the next round and will lose the tournament for sure. If it is odd, after at most $\log_{k_1}$ rounds, there will be an even number of players with at least 3 players in N_1 remaining. Player o_2 will have no chance of winning the tournament.

For o_2 to win the tournament KT with probability at least δ, we have shown that no players in N_1 can play against players in N_2. Thus in KT, o_1 must have won with probability at least δ the sub-tournament KT_1 that consists only of players in N_1. $\square$

This implies the optimization version of the l-BKT problem is NP-hard. Note that the result is stronger than showing that it is NP-hard to find the optimal tournament for a set of players of any size.

It is also hard to approximate the optimal value within any constant factor.

Theorem 5. *For any fixed l, and t, it is NP-hard to approximate the optimal value OPT of an l-BKT problem with a factor at least r for any given $r \geq \frac{1}{e^t}$. In other words, given a set N of players such that $|N| = 2^m + l$ and $2^m > l$, and a winning probability matrix P, it is NP-hard to find a balanced knockout tournament KT such that $\frac{q(k,KT)}{OPT} \geq \frac{1}{e^t}$.*

Since this theorem follows from Theorem 8, we will also defer the discussion of the proof of this theorem to the next section.

4.5 Constraints on Player Model

Here we consider both of the constraints we described in Chapter 2: deterministic and monotonic winning probability model.

4.5.1 Win-Lose Match Results

The first constraint we consider is to require that winning probabilities can only be either 0 or 1, and so the result of each match is deterministic. As we will discuss in Section 4.3, 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).

When there is no constraint on the structure of the tournament, as shown in [18], 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, finding such a seeding 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 7. In this model, there is no constraint to the tournament tree, except that each player must 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. This is different to having a bye in Section 4.4 in which the organizer has a choice of whom receiving the bye. Here, the organizer has to follow the prescribed placements of the players.

Definition 7. (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 the 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 6. *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 well-known Vertex Cover problem [11, p.190]:

> **Vertex Cover:** Given a graph $G = (V, E)$ and an integer k, does there exist a subset $C \subseteq V$ such that $|C| \leq k$ and C covers E (i.e., each edge in E is incident to at least one vertex in C)?

Given an instance $\{(V, E), k\}$ of Vertex Cover problem, our reduction is as follows. We construct a set of players N with a special player o and a round placement R such that there exists a tournament KT that allows o to win if and only if there exists a vertex cover of size at most k.

N contains the following players[1]:

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 filler players. They are created to help player o advance.
5. Holder players: For each round r, there is a set of holder players h_i^r that start at round r. These players have the same winning probabilities and can be viewed as multiple copies of h^r, which are created to help the vertex players advance. 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

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

- 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 4.5.1. 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.

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

Table 4.1: The winning probabilities of row players against column players in KT

The reduction is polynomial since the number 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 involves 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 some vertex player v_i. 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 are only $(k-1)$ holder players at each round, if h^r was eliminated by e'', either two vertex players must have played against each other and one of them must have been eliminated, or one of them has to play against f^r. In that case o must have played against some v to advance. If f^r was eliminated by e'', at that round r, o also must have played against some vertex player v. 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 vertex players.

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. □

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.

4.5.2 Win-Lose-Tie Match Results

As we have already mentioned, when all probabilities P_{ij} are either 0 or 1, then it is an open problem whether there exists an efficient algorithm to find the optimal balanced knockout tournament for a given player. However, when we allow the possibilities of ties between players (each has equal chance of winning), the problem becomes hard.

We define the problem as the following.

Problem 2. **(l-BKTWLT)** *Given a set N of players such that $|N| = 2^m + l$ and $l < 2^m$, and the winning probability matrix P such that $P_{ij} \in \{0, 1, 0.5\}$ for all $i, j \in N$, does there exist a balanced knockout tournament KT such that $q(k, KT) \geq \delta$ for a given δ and $k \in N$.*

Theorem 7. *For any fixed l, the l-BKTWLT problem is NP-complete.*

Since Lemma 1 still applies here, we only need to show the proof for 0-BKTWLT. It is similar to the proof of Theorem 6 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.

Proof of Theorem 7. As in the Proof of Theorem 6, we again give a reduction from Vertex Cover. Given an instance $\{(V, E), k\}$ of Vertex Cover problem, we construct a set of players N with a special player o such that there exists a balanced tournament KT that allows o to win with probability at least 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 created to help the edge players advance. We might refer to both types of filler players as f^r_i or simply f^r.

5. Holder players: For each 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 $2^{r-1} - 1$ 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 4.2. 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_v^r and f_e^r), 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.

	v_j	e_j	$f_j^{r'}$	$h^{l'}_{e_j}$	$h^{l'}_{f_j^{r'}}$	$h^{l'}_o$
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	1	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
$h^l_{e_i}$	-	-	-	0.5	0.5	1
$h^l_{f_i^r}$	-	-	-	-	0.5	1
h^l_o	-	-	-	-	-	0.5

Table 4.2: The winning probabilities of row players against column players in KT

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 balanced tournament KT that lets 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 a corresponding holder player, we will match the player with its holder player. 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 advance 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 repeatedly matching each vertex player with the edge players that it covers. If there are more vertex players than edge players, we match the free vertex players with each other. 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. □

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 3.

Similarly, the following theorem also leads to Theorem 5.

Theorem 8. *For any fixed l, and t, it is NP-hard to approximate the optimal value OPT of an l-BKTWLT problem with a factor at least r for any given $r \geq \frac{1}{e^t}$.*

We prove this theorem by using Lemma 1 and the following two lemmas:

Lemma 2. *Given a vertex cover problem, when the answer is "No", the winning probability of the player o in any balanced tournament for the set of players N constructed as in the proof of Theorem 7 is at most $1 - \frac{2}{|N|}$.*

Proof. Since the answer is "No", there must exist an edge player e that could not be matched with a vertex player who covers it. Moreover, the winning probabilities of e against all other players are either 1 or $\frac{1}{2}$. After $\log|N| - 1$ rounds, the probability of e getting to the final match is at least $(\frac{1}{2})^(\log|N| - 1) = \frac{2}{|N|}$. Since e wins against o with probability 1, o can only win with probability at most $1 - \frac{2}{|N|}$. □

Lemma 3. *If there exists a polynomial time algorithm Alg to approximate the optimal value of any 0-BKTWLT problem with a factor at least r for a given $r > 0$, given a 0-BKTWLT problem I for a set of players N, we can approximate the optimal value OPT_I of I with a factor at least $\sqrt{r}$ in polynomial time using a tournament of size $2|N|$.*

Proof. Given Alg and I, we show how to approximate the optimal value of I within $\sqrt{r}$ in polynomial time. Let N_1 be the set of players, and o_1 the target player in I. We duplicate the players in N_1 and their winning probabilities. Let N_2 be the set of the cloned players, and o_2 the cloned version of o_1. We design the winning probabilities between the players in N_1 and N_2 as the following:

- All players in N_2 lose with probability 1 to all players in N_1 except o_1
- o_1 loses to all players in N_2

Let I' be the 0-BKTWLT problem for the set of players $N = N_1 \cup N_2$, and the target player o_2. We apply the algorithm Alg on I' to find a tournament KT such that o_2 wins with probability within a factor r of $OPT_{I'}$.

Using the same arguments as in the proof of Lemma 1, we can show that for o_2 to win KT with any probability greater than 0, no player in N_1 plays against another player in N_2. Thus KT is composed of two sub-tournaments KT_1 and KT_2 for the players in N_1 and N_2 respectively. Moreover, $q(o_2, KT) = q(o_1, KT_1) \times q(o_2, KT_2)$ and $\frac{q(o_2,KT)}{OPT_{I'}} \geq r$. Therefore we have either $\frac{q(o_1,KT_1)}{OPT_I} \geq \sqrt{r}$ or $\frac{q(o_2,KT_2)}{OPT_I} \geq \sqrt{r}$ because otherwise $q(o_2, KT) = q(o_1, KT_1) \times q(o_2, KT_2) < r \times OPT_I \times OPT_I \leq r \times OPT_{I'}$. Thus either KT_1 or KT_2 is the tournament we are trying to find. □

Given the lemmas above, the proof of Theorem 8 is straightforward.

Proof. We show how to reduce the Vertex Cover problem to the problem of approximating the optimal value OPT of an 0-BKTWLT problem within a factor of r for any fixed t and $r > \frac{1}{e^t}$. First, given a vertex cover problem VC, we construct a set of players N and a target player o corresponding to VC as in the proof of Theorem 7. There is a solution to VC if and only if there exists a tournament KT for N such that o wins KT with probability 1. From Lemma 2, we know that if there is no solution, the winning probability of o is at most $1 - \frac{2}{|N|}$. Thus if we have a polynomial algorithm to approximate the optimal winning probability of o with a factor more than $1 - \frac{1}{|N|}$, we can decide in polynomial time whether there is a solution to VC. If and only if the algorithm returns at most $1 - \frac{2}{|N|}$, then the VC problem has no solution.

Following from Lemma 3, we can approximate the optimal winning probability of o with a factor at least $r^{\frac{1}{t|N|}}$ by repeating the process $\log(t|N|)$ times. The size of the new tournament is $t|N|^2$, which is still polynomial. Since $r > \frac{1}{e^t}$, we have $r^{\frac{1}{t|N|}} > (\frac{1}{e^t})^{\frac{1}{t|N|}}$. Moreover we have $\lim_{|N|\to\infty}(1 - \frac{1}{|N|})^{t|N|} = \frac{1}{e^t}$. Thus we can approximate the optimal winning probability of o with a factor of more than $1-\frac{2}{|N|}$ using a tournament of polynomial size in $|N|$. □

4.5.3 Monotonic Winning Probabilities

Another natural constraint is to require a certain overall structure for the winning probability matrix P. One of the most common models in the literature is the monotonic model as we described in Chapter 2. In this model, the players are ranked from 1 to n in descending order of unknown intrinsic abilities. The tournament designer only knows the rankings and the winning probabilities between the players, which are correlated to the intrinsic abilities.

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 slightly, we can obtain a hardness result. We call the new condition ϵ-monotonicity.

Definition 8. (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 hard:

Theorem 9. *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 $k \in N$.*

Proof. 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 7 with the same set of players but with slightly different winning probabilities (shown in Table 4.3).

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

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 7 to hold in this setting. $\square$

Here we assume ϵ and δ have the same precision representation (otherwise if ϵ is much more accurate than δ, the ϵ-monotonic setting becomes equivalent with the original monotonic setting). Since ϵ can become 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 (assuming $P \neq NP$).

4.6 Discussion

In this chapter we investigated the computational aspects of schedule control for knockout tournaments. We considered several variants of the schedule control 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 schedule 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. Our results are applicable even for cases in which the number of players is not a power of 2. We have also shown that it is NP-hard to approximate the optimal solution of schedule control for any reasonable ratio in the balanced setting. For the general knockout tournaments, we have characterized the optimal structure in which it is biased toward the target player. The results are summarized in Table 4.4 with new results in bold face.

	General Prob.	Win-Lose-Tie	Win-Lose	ϵ-Mono	Mono
General Struct	Biased	Open	$O(n^2)$ [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 4.4: Summary of the complexity results

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.

Our hardness results of Theorems 4 and 5 may lead us to the conjecture that a designer cannot benefit from having the probability matrix P, since is is hard to find an optimal seeding even with this input. However, in practice, a worst-case analysis is not enough. Indeed the problem could be very well solvable for average-case. We provide such an analysis in Chapter 6 to show that in fact using a simple heuristic function can help us achieve very good results on average.

Chapter 5

Fair Seedings in Knockout Tournaments

5.1 Introduction

As we have shown in the previous chapter, the seeding can significantly influence the result of a tournament. This has been also demonstrated in several papers, but most of them focus on how to find a seeding that maximizes the winning probability of the strongest player (so-called "predictive power") (see, e.g., [15, 1, 27]). This usually means giving the strongest player the easiest schedule, while making it harder for the remaining players. An example of such seedings is given in Figure 5.1. In this example, players are numbered based on their strengths with player 1 being the strongest player. This seeding is rarely seen in practice however, because it seems to be very unfair to other players, especially the strong ones such as 2 or 3.

The most popular seeding actually used in major sport tournaments is the one that pairs up in the first stage the strongest player with the weakest player, the second strongest player with the second weakest player, so on and so forth. An example for 8 players is shown in Figure 5.2. One possible rationale is that it seems fair: confrontations between strong players are delayed until later rounds, and that helps increase the chance that one of those strong players will win the tournament. This raises several questions: How do we capture this intuition about fairness? Is there any other seeding that is also fair? Can we always find such a fair seeding?

To answer these questions, we consider two alternative fairness criteria adapted from

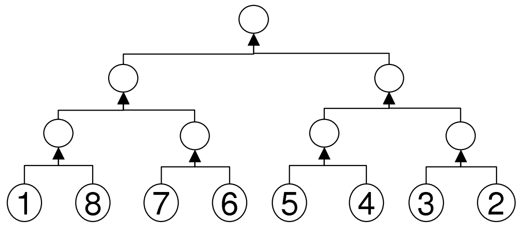

Figure 5.1: A biased seeding that maximizes winning probability of player 1

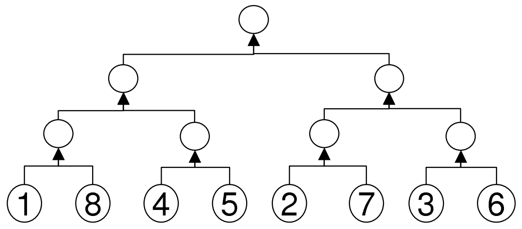

Figure 5.2: A popular seeding with 8 players

the literature: envy-freeness and order preservation. We analyze separately each criterion across different tournament settings, providing both possibility and impossibility results. To explain our results we need to lay out the settings with all their variants, which we do in the next section. In Section 3 we summarize the existing results from the literature. We then analyze envy-freeness and order preservation in Section 4 and 5 respectively, and conclude in Section 6.

5.2 Settings

We consider two different types of tournament structures and two different models of winning probabilities between the players. Regarding the tournament structure, we consider both balanced and unconstrained tournaments (as described in Chapter 2).

Among the unconstrained tournaments, we want to draw attention to a special structure called "caterpillar" tree. In a tournament with a caterpillar structure, the players are arranged in some order. The first two players will compete against each other. The winner of the match will then advance forward and compete with the third player, so on and so forth.

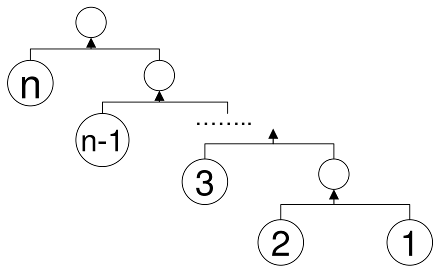

Figure 5.3: A caterpillar tournament for n players

An example is shown in Figure 5.3. This structure has been studied in the social choice literature as a voting tree structure and has been shown to have interesting properties (e.g., see [8]).

Among balanced tournaments, S_n^* is a seeding formed by matching in the first stage the strongest player with the weakest player, the second strongest player with the second weakest player, and so on and so forth. For the successive rounds, the sub-tree containing the strongest player is combined with the sub-tree containing the weakest players possible, so on and so forth.

$$S_n^* = \Big[[1, n], ..., [i, (n - i + 1)], [(\frac{n}{2} - i + 1), (\frac{n}{2} + i)], ...\Big]$$

Here we encode the seeding by a permutation of the numbers between 1 and n. The k^{th} number is the ranking of the player that will be placed at the k^{th} leaf node of the binary tournament tree, from left to right. An example of S_n^* when $n = 8$ is $S_8^* = [1, 8, 4, 5, 2, 7, 3, 6]$ as shown in Figure 5.2.

Most knockout tournaments used in practice are balanced. Although the caterpillar tournament approximates some real-world competition structures (for example the world championship organized by the World Boxing Association), the main reason to consider unbalanced tournaments in addition to balanced ones is to highlight the special properties of the latter.

Regarding the winning probability models, we consider both of the General and Monotonic Winning Probability Model (which are also described in Chapter 2). We restate their definitions below.

Definition. (General Winning Probability Model) *Given a set of n players, the winning probabilities between the players form a matrix P such that P_{ij} denotes the probability that player i will win against player j, $\forall(i \neq j) : 1 \leq i, j \leq n$, and P satisfies the following constraints:*
1. $P_{ij} + P_{ji} = 1$
2. $0 \leq P_{ij}, P_{ji} \leq 1$

Definition. (Monotonic Winning Probability Model) *Given a set of n players, the winning probabilities between the players form a matrix P such that P_{ij} denotes the probability that player i will win against player j, $\forall(i \neq j) : 1 \leq i, j \leq n$, and P satisfies the following constraints:*
1. $P_{ij} + P_{ji} = 1$
2. $0 \leq P_{ij}, P_{ji} \leq 1$
3. $P_{ij} \leq P_{i(j+1)}$
4. $P_{ji} \geq P_{(j+1)i}$ *(which is actually implied by (1) and (3))*

Since the monotonic model is a special case of the general model, many of our results are specifically proved for the monotonic model and then generalized to the general model as corollaries.

5.3 Past Results

Our work is most related to the axiomatic approaches in the literature. Specifically, our fairness criteria are generalized and adapted from “Sincerity Rewarded” (proposed in [28]) and “Monotonicity” (proposed in [16]).

- *Sincerity Rewarded:* A higher-ranked player should never be penalized by being given a schedule more difficult than that of any lower ranked.

- *Monotonicity:* The probability of a given player winning the tournament is higher than all of the weaker players.

In these two papers, the authors only focus on monotonic winning model. Here our criteria are also applicable for the general model. The second difference is that unlike past work which uses probabilistic seedings [28] or dynamic seedings [16], we focus on deterministic and static seedings instead, since they are more often used in practice. Interestingly, in [15, 12] it is shown that for a balanced tournament of size 4 with monotonic winning probability model, the seeding $[1\ 4\ 2\ 3]$ always achieves "Monotonicity". Tournament of size 4 is a special case due to its extremely small size. For our paper, we focus on tournaments of sizes 8 or larger.

Fairness for other tournament formats such as round robin or Swiss-system are also addressed in the literature (see [17] for an extensive survey). Most of the work here focuses on graph theoretic and combinatorial properties of the tournament (e.g., no team plays on the same court two consecutive rounds, or no team plays against extremely weak or extremely strong teams in consecutive rounds) as opposed to the winning probabilities of the players.

5.4 First Fairness Criterion: Envy-freeness

In [28], it is argued that a good seeding should not give any strong player a harder tournament schedule than a weaker player. One of the main reasons is that if strong players are given harder schedules, they will have incentives to under-perform during the pre-season period or lie about their strengths in order to get lower rankings instead. We formalize this intuition to define envy-freeness as the first criterion for a seeding to be fair: no player envies the seeding position of another player weaker than him.

First we need to specify when a player is considered to be stronger than another. In the monotonic winning probability model, the notion of a stronger player is clear. For any pair of players (i, j) such that $i < j$, player i is always stronger than player j. In the general model, that is no longer the case. Player i might be stronger than j while playing against some opponents but not the others. Similarly, the winning probability between i and j by itself is not a good indicator either. Player i may have much higher chance to win in the match against j, but has much lower chance than j while playing against other players. In this case, it is unclear which player should be considered stronger.

To avoid these disputable cases, we define a player to be stronger between the two if he dominates the second player when playing against all other players.

Definition 9. (Dominance) *Player i dominates another player j if and only if the following holds: (a) $P_{ij} \geq P_{ji}$, and (b) $\forall k : 1 \leq k \leq n$ $(k \neq i, j)$, it is the case that $P_{ik} \geq P_{jk}$.*

This definition is natural and intuitive. Notice that it also applies to the monotonic model; here domination is equivalent to being higher ranked.

When there is a tie between two players (i.e., all of their winning probabilities are equal), we need a tie-breaking rule. Our results hold for any monotonic tie-breaking rule. WLOG, we assume the tie is resolved based on lexicographic ordering: for any pair of players (i, j) such that $\forall k : 1 \leq k \leq n$ $(k \neq i, j)$, $P_{ik} = P_{jk}$ and $P_{ij} = P_{ji} = 0.5$, i is considered to dominate j if and only if $i < j$.

Given the definition of Dominance, we define Envy-freeness as the following.

Definition 10. (Envy-free Seeding) *Given a set N of players, a winning probability matrix P, and a knockout tournament $KT_N = (T, S)$, the seeding S is envy-free if and only if $\forall (i, j) \in N$ such that i dominates j, player i does not have a higher probability of winning the tournament by swapping his position in the seeding S with j's, i.e., $q(i, (T, S)) \geq q(i, (T, S' = S_{i \leftrightarrow j}))$.*

Notice that in the general winning probability model, if there is no pair of players such that one dominates the other, any seeding will trivially satisfy the requirement above. On the other extreme, in the monotonic model, an envy-free seeding will have to satisfy the requirement for all pairs of players.

5.4.1 Envy-freeness for Unconstrained Tournaments

When there is no constraint on the structure of the tournament, envy-freeness can be achieved by the following tournament structure and seeding method. Note that the result holds for the general model (and hence the monotonic model also).

Theorem 10. *Given a set N of players, a winning probability matrix P, envy-freeness can be achieved by the knockout tournament with the caterpillar structure and the seeding S such that for every pair of players (i, j), if i dominates j then j is placed below i on the tournament tree.*

The seeding above can be achieved easily by first constructing a directed acyclic graph in which each node represents a player and there is a directed edge from player i to player j ($\forall(i,j) \in N$) if and only if i dominates j. We then perform a topological sort of the graph and use this ordering to place players on the tournament tree. We are then guaranteed that dominated players are placed lower on the tournament tree and the above condition is satisfied.

Proof of Theorem 10. We prove the theorem by induction on the number of players.
Base case: When $|N| = 2$, there is only one possible binary tree with 2 leaf nodes which is trivially envy-free.

Inductive step: Assume that the theorem holds for all N' such that $|N'| = n - 1$, we need to show for all N such that $|N| = n$, the seeding we construct as above is envy-free.

Let S be a seeding which satisfies the condition that for any pair of players (i, j), if i dominates j then j is placed below i on the tournament tree. Let k^* be the player at the root node of the tournament tree using the seeding S. Let $N' = N \setminus \{k^*\}$. Due to the special property of S, there is no player $k' \in N$ such that k' dominates k^*.

Since N' has size $n - 1$, based on the inductive hypothesis, the seeding S' for N' (as shown in Figure 5.4) is envy-free. Thus to show that S is also envy-free, we just need to show that k^* is not envious of any player placed below him. In other words, k^* will not have a higher winning probability when swapping his seeding position with any player $k_i \in N'$.

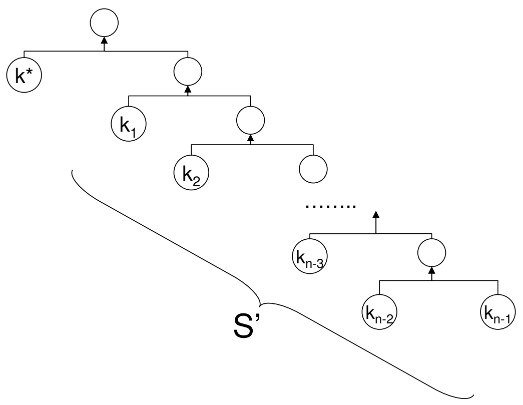

Figure 5.4: The seeding S composed of k^* and S'

Using the same notation as in Section 5.2, we denote S as:

$$[k^*, k_1, k_2, ..., k_{i-1}, k_i, k_{i+1}, ..., k_{n-1}]$$

Let S_i be the seeding created from S by swapping k^* with k_i:

$$S_i = [k_i, k_1, k_2, ..., k_{i-1}, k^*, k_{i+1}, ..., k_{n-1}]$$

Notice that in S_i, regardless of the ordering of the players k_j on the left of k^*, $\forall 1 \leq j \leq i$, the winning probability of k^* will stay the same.

Formally, let $p(k_j)$, $\forall j : i < j \leq n-1$, be the probability that player k_j will get to play against k^* (in other words, the probability that k_j will win the sub-tournament among $\{k_{i+1}, ..., k_{n-1}\}$). The probability that k^* will win the tournament with the seeding S_i ($q(k^*, S_i)$), is the same as with the seeding $S'_i = [k_1, k_2, ..., k_{i-1}, k_i, k^*, k_{i+1}, ..., k_{n-1}]$ because:

$$q(k^*, S_i) = \sum_{j:i<j\leq n-1} [p(k_j) \times P_{k^*,k_j}] \times \prod_{l:1\leq l\leq i} P_{k^*k_l} = q(k^*, S'_i)$$

Let′s consider the winning probability of k^* in S'_{i-1}, which is created by swapping k^* and k_i (i.e., $S'_{i-1} = [k_1, k_2, ..., k_{i-1}, k^*, k_i, k_{i+1}, ..., k_{n-1}]$).

$$\begin{aligned} q(k^*, S'_{i-1}) &= \sum_{j:i<j\leq n-1} [p(k_j)P_{k_jk_i}P_{k^*k_j} + p(k_j)P_{k_ik_j}P_{k^*k_i}] \times \prod_{l:1\leq l\leq i-1} P_{k^*k_l} \\ &= \sum_{j:i<j\leq n-1} p(k_j)[P_{k_jk_i}P_{k^*k_j} + P_{k_ik_j}P_{k^*k_i}] \times \prod_{l:1\leq l\leq i-1} P_{k^*k_l} \\ &\geq \sum_{j:i<j\leq n-1} p(k_j) \times \min(P_{k^*k_j}, P_{k^*k_i}) \times \prod_{l:1\leq l\leq i-1} P_{k^*k_l} \\ &\geq \sum_{j:i<j\leq n-1} p(k_j) \times P_{k^*k_j} \times P_{k^*k_i} \times \prod_{l:1\leq l\leq i-1} P_{k^*k_l} \\ &= \sum_{j:i<j\leq n-1} p(k_j) \times P_{k^*k_j} \times \prod_{l:1\leq l\leq i} P_{k^*k_l} \\ &= q(k^*, S'_i) \end{aligned}$$

The first inequality holds because $P_{k_j k_i} + P_{k_i k_j} = 1$. The second inequality holds because $0 \leq P_{k^* k_j}, P_{k^* k_i} \leq 1$. This means when we swap the position of k^* with the position of the player right in front of k^* in the seeding, the winning probability of k^* does not decrease. Thus if we keep swapping the position of k^* with k_j in S'_j sequentially for $j = (i-1) \to 1$, we have the following sequence of inequalities:

$$q(k^*, S_i) = q(k^*, S'_i) \leq q(k^*, S'_{i-1}) \leq ... \leq q(k^*, S'_2) \leq q(k^*, S'_1) \leq q(k^*, S)$$

This shows that k^* will not have a higher winning probability by swapping its seeding position with player k_i. Thus S is envy-free. □

5.4.2 Envy-freeness for Balanced Tournaments

When the tournament structure has to be a balanced binary tree, it is no longer possible to always achieve envy-freeness even when the winning probabilities between players are monotonic. Here we assume n, the number of players, is a power of 2.

Theorem 11. *For any $n = 2^r \geq 8$, there exists a set of n players with a monotonic winning probability matrix P such that it is not possible to find an envy-free seeding S for the balanced knockout tournament between these n players.*

Proof. We first prove the result for the case when $n = 8$, and then for general n.

The case of $n = 8$: We construct a tournament between 8 players with the winning probability matrix as shown in Table 5.1 (note that we only show the top half of the matrix). Since there are only 315 unique seedings for a tournament of 8 players, we can exhaustively check all of them using a computer program. Given this winning probability matrix, for each of those 315 seedings, there is always a pair of players whose probabilities of winning the tournament violate the criterion.

The case of $n = 2^r > 8$: To construct a tournament between n players, we add $(n-8)$ dummy players to the set of 8 players above. The dummy players lose to the original 8 players with probability 1. Note that player 7 and 8 can be regarded as the dummy players to the top 6 players since player 7 and 8 lose with probability 1 to these players. Let M denote the set of the top 6 players from 1 to 6. Essentially the tournament between n players

	2	3	4	5	6	7	8
1	0.5	0.5	0.5	$0.5+2\epsilon$	$0.5+2\epsilon$	1	1
2	-	0.5	0.5	0.5	$0.5+\epsilon$	1	1
3	-	-	0.5	0.5	$0.5+\epsilon$	1	1
4	-	-	-	0.5	$0.5+\epsilon$	1	1
5	-	-	-	-	0.5	1	1
6	-	-	-	-	-	1	1
7	-	-	-	-	-	-	0.5

Table 5.1: The winning probabilities between 8 players

is reduced to a tournament between the players in M as if we take out all of the dummy players and player 7 and 8.

Let's consider all possible tournament trees T_M for 6 players. We first prove the following property of T_M: For any seeding to be envy-free, in the *reduced* tournament tree T_M of the players in M, there exists no pair of players $(i, j) \in M$ such that the difference between their depths is greater than 1 as in Figure 5.5.

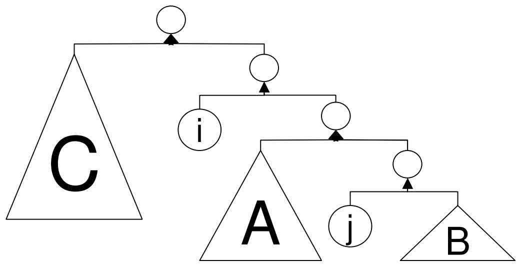

Figure 5.5: The *reduced* tournament structure that does not satisfy envy-freeness

This is true since we can always swap j with the dummy player who plays against i in the *original* tournament and improve the winning probability of j. To show this, let's first calculate the winning probability of j before the swap:

$$Q_1 = \sum_{l \in B} [q(l, B) \times P_{jl}] \times \sum_{k \in A} [q(k, A) \times P_{jk}] \times P_{ji} \times \sum_{h \in C} [q(h, C) \times P_{jh}]$$

Recall that $q(k, A)$ is the probability that player k will win the sub-tournament A.

After the swap, the winning probability of j becomes:

$$
\begin{aligned}
Q_2 &= \sum_{l\in B}\sum_{k\in A} q(l,B)\times q(k,A)[P_{jl}P_{lk}+P_{jk}P_{kl}]\times P_{ji}\times\sum_{h\in C}[q(h,C)\times P_{jh}] \\
&\geq \sum_{l\in B,k\in A} q(l,B)\times q(k,A)\times \min(P_{jl},P_{jk})\times P_{ji}\times\sum_{h\in C}[q(h,C)\times P_{jh}] \\
&> \sum_{l\in B,k\in A} q(l,B)\times q(k,A)\times (P_{jl}\times P_{jk})\times P_{ji}\times\sum_{h\in C}[q(h,C)\times P_{jh}] \\
&= \sum_{l\in B} q(l,B)\times P_{jl}\times\sum_{k\in A} q(k,A)\times P_{jk}\times P_{ji}\times\sum_{h\in C}[q(h,C)\times P_{jh}] = Q_1
\end{aligned}
$$

The first inequality holds because $P_{lk}+P_{kl}=1$. The second inequality is strict because for j, l, and k in the top 6 players, P_{jl} and P_{jk} is less than 1. Note that we do not assume any relationship between i and j here since our focus is on j being envious of the dummy player who got matched up with i in the original tournament.

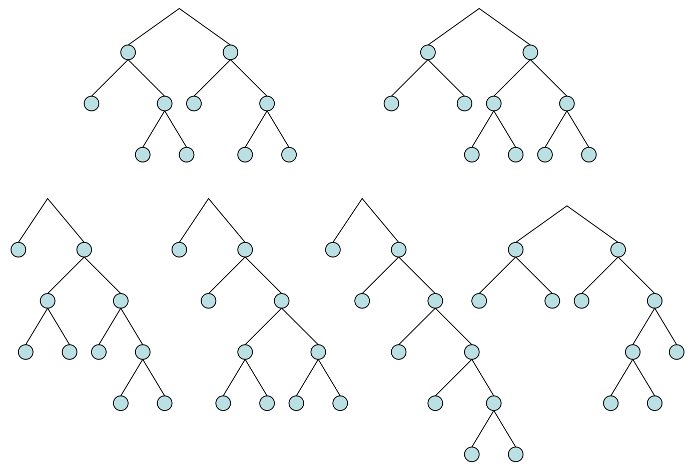

Figure 5.6: Possible tournament trees for 6 players

There are only 6 possible knockout tournament trees for 6 players as shown in Figure 5.6. However, only the top two trees satisfy the condition that there is no pair of players with a difference in their depths greater than 1. Thus if the seeding of the original tournament is envy-free, the reduced tournament tree T_M has to be one of the top two trees.

For each of these trees, notice that for the two players who are placed at a smaller depth than the rest, they must have played against dummy players in the original tournament before they can advance to their current round. Notice that with the current winning probability matrix, player 7 and 8 play the same role as the dummy players. Thus if there existed an envy-free seeding for the original tournament, we would be able to find an envy-free seeding for the set of the top 8 players. However, we have shown in the first part that there does not exist such a seeding. This implies there is no envy-free seeding for the tournament of size $n > 8$ either. $\square$

Note that the same result does not apply for unconstrained tournaments. In the proof above, we make use of the fact that any player must play against some player (which can be a dummy player) to advance to the next round. In the case of unconstrained tournament, a player can advance directly to the final round without competing.

Since the monotonic model is a special case of the general model, we have the following corollary.

Corollary 12. *For any given $n = 2^r \geq 8$, with the general winning probability model, it is not always possible to find an envy-free seeding for a balanced knockout tournament of size n.*

Given this impossibility result for balanced tournaments, in the next section we discuss an alternative fairness criterion that seems intuitively weaker, with the hope that it can be more easily satisfied.

5.5 Second Fairness Criterion: Order Preservation

Instead of focusing on the seeding positions of the players, we place a requirement on their probabilities of winning the tournament instead: stronger players must have higher probabilities of winning the tournament than weaker players. This requirement seems to be less restrictive than the envy-freeness criterion, yet it still encourages the players to perform accordingly to their strengths.

Definition 11. (Order-Preserving Seeding) *Given a set N of players, a winning probability matrix P, and a knockout tournament $KT_N = (T, S)$, the seeding S is order preserving* *if and only if* $\forall(i, j) \in N$ *such that* i *dominates* j*, player* i *does not have a lower probability of winning the tournament than player* j*, i.e.,* $q(i,(T,S)) \geq q(j,(T,S))$.

The criterion is adapted and generalized from the notion of "Monotonicity" first introduced in [16].

5.5.1 Order Preservation for Unconstrained Tournaments

When there is no constraint on the structure of the tournament, it is easy to see that the same tournament structure and seeding in Theorem 10 will achieve order preservation.

Theorem 13. *Given a set* N *of players, a winning probability matrix* P*, order preservation can be achieved by the knockout tournament with the caterpillar structure and the seeding* S *such that for every pair of players* (i, j)*, if* i *dominates* j *then* j *is placed below* i *on the tournament tree.*

Proof. Recall from Theorem 10 that the seeding S created as above is envy-free. Using proof by contradiction, we assume S is not order preserving, and show that this implies it will not be envy-free either.

If S is not order preserving, there must exist two players i, and j such that i dominates j, and the winning probability of i is smaller than j, i.e., $q(i, S) < q(j, S)$. Since i dominates j, i must be placed above j on the tournament tree. Let A be the set of players placed above i on the tournament tree, B between i and j, and C below j. The winning probabilities of j given the seeding S can be calculated as the following:

$$q(j, S) = \prod_{k \in A} P_{jk} \times P_{ji} \times \prod_{k \in B} P_{jk} \times \prod_{k \in C} [q(k, C) \times P_{jk}]$$

Here we use $q(k, C)$ to denote the probability that player $k \in C$ will win the sub-tournament between the players in C and advance to the match against player j.

Let S' be the seeding created by swapping the position of i and j in S. Since S is envy-free, the winning probability of i with the seeding S must not be smaller than with S', i.e.,

$q(i, S) \geq q(i, S')$. The winning probability of i with S' can be calculated as the following:

$$q(i, S') = \prod_{k \in A} P_{ik} \times P_{ij} \times \prod_{k \in B} P_{ik} \times \prod_{k \in C} [q(k, C) \times P_{ik}]$$

However, since i dominates j, $\forall k \in N$, $P_{ik} \geq P_{jk}$ and $P_{ij} \geq P_{ji}$. Therefore we have:

$$q(i, S') \geq \prod_{k \in A} P_{jk} \times P_{ji} \times \prod_{k \in B} P_{jk} \times \prod_{k \in C} [q(k, C) \times P_{jk}] = q(j, S) > q(i, S)$$

This is a contradiction. Thus S must be also order-preserving. □

Notice that the method above does not take into consideration the actual values of the winning probabilities between players. The criterion can be achieved as long as the relative ordering of the players are known. It would be very useful if the same result applied for the case of balanced tournament. Unfortunately, as we show in the next section, no fixed seeding is always order preserving for balanced tournaments, even when the winning probabilities are monotonic.

5.5.2 Order Preservation for Balanced Tournaments

If the seeding S_n^* as in Figure 5.2 could be shown to always satisfy order preservation, this would justify its popularity in practice. However, as we show in the following theorem, there are cases in which this seeding violates the criterion.

Theorem 14. *For any* fixed *seeding S of a balanced tournament of size $n = 2^r \geq 8$, there exists a monotonic winning probability matrix P such that S is not order preserving.*

Proof. We first show by induction that if there exists a fixed seeding S that always satisfies order preservation, then S has to be S_n^* (as defined in Section 2).

Base case: When $n = 2$, $S_2^* = [1\ 2]$ is the only seeding.

Inductive case: Assume that the claim holds for $n = m$, we need to show that it also holds for $n = 2m$. We will present a sequence of winning probability matrices that all impose different constraints on an order preserving seeding. Since we are looking for a fixed seeding that is order preserving for *any* given winning probabilities, any seeding that does not satisfy all of these constraints can be discarded.

Let's first consider the following winning probabilities between the players:

- $\forall i, j : 1 \leq i, j < 2m$, $P_{ij} = 0.5$
- $\forall i : 1 \leq i < 2m$, $P_{i(2m)} = 1$

Note that due to monotonic tie-breaking rule, player 1 dominates all other players here. The only order preserving seedings in this case are the ones that pair up player 1 and $2m$ in the first round. Otherwise whoever gets to play against player $2m$ will have a winning probability of $\frac{2}{m}$, whereas player 1 only wins with probability $\frac{1}{m}$. Thus we can now only consider the seedings that match up player 1 and $2m$ in the first round.

Let us consider another set winning probabilities between the players:

- $\forall i, j : 1 \leq i, j < 2m - 1$, $P_{ij} = 0.5$
- $\forall i : 1 \leq i < 2m - 1$, $P_{i(2m)} = P_{i(2m-1)} = 1$
- $P_{(2m)(2m-1)} = 0.5$

Since we already know that the remaining seedings under consideration must match up player 1 with player $2m$, player 2 must then be matched up with player $2m - 1$ to satisfy order preservation.

We can take this argument and generalize it to show for each value of $k = 3 \to m$, player k has to be matched up with player $(2m - k + 1)$ by using the following set of winning probabilities:

- $\forall i, j : 1 \leq i, j \leq 2m - k$, $P_{ij} = 0.5$
- $\forall i, j : 1 \leq i \leq 2m - k, 2m - k + 1 \leq j \leq 2m$, $P_{ij} = 1$
- $\forall i, j : 2m - k + 1 \leq i, j \leq 2m$, $P_{ij} = 0.5$

In other words, the top $(2m - k)$ players tie with each other, and win with probability 1 against the bottom k players. The bottom k players also tie among themselves. In this case, since the top $k - 1$ players are already matched up with the bottom $k - 1$ players, player k

has to be matched up with player $(2m-k+1)$ in order to have a higher winning probability than the players from $(k+1)$ to $(2m-k)$.

Now let us consider the case in which the top m players win against the bottom m players with probability 1. We have shown above that player k, $\forall k : 1 \leq k \leq m$, has to be paired up with player $(2m-k+1)$. Therefore, all the top m players will advance to the next round with probability 1. The current tournament essentially becomes a tournament between these m players. Based on the inductive hypothesis, they must be seeded by S^*_m. This proves that the seeding for $2m$ players must be S^*_{2m}.

Above we have used induction to show that the fixed seeding that satisfies order preservation for n players could only possibly be S^*_n. Now we just need to find a winning probability matrix such that S^*_n violates the criterion. Consider the following winning probabilities:

- $\forall i, j : 1 \leq i < 6 < j \leq n$, $P_{ij} = 1$
- $\forall i : 9 \leq i \leq n$, $P_{7i} = P_{8i} = 1$
- $\forall i, j : 1 \leq i, j \leq 5$, $P_{ij} = 0.5$
- $P_{36} = P_{46} = P_{56} = 0.5$, and $P_{16} = P_{26} = 1$
- All other winning probabilities are 0.5

In other words, the top 8 players win against all other players with probability 1. The top 5 players tie with each other and win against player 7, and 8 with probability 1. Player 6 loses to 1 and 2 with probability 1 but ties with player 3, 4, and 5.

Let's consider the last 3 rounds of the tournament. Only the top 8 players make to these rounds and they are seeded by $S^*_8 = [1, 8, 4, 5, 2, 7, 3, 6]$. Intuitively, since player 3 has only 50% chance of winning against player 6, player 2 has more chance of getting into the final match than player 1. If we carry out the calculations, we have $q(1, S^*_n) = 0.25 < q(2, S^*_n) = 0.375$, which violates order preservation. Thus S^*_n is not always order preserving either. $\square$

Since the monotonic model is a special case of the general model, we have the following corollary.

Corollary 15. *For any given $n = 2^r \geq 8$, there is no* fixed *seeding S that is always order preserving for any balanced tournaments of size n with the general winning probability model.*

The results above do not preclude the existence of a fair seeding for any *given* set of players and winning probabilities. While such existence remains an important open problem, we propose a heuristic algorithm to find an order preserving seeding and show through experiments that the algorithm is indeed efficient and effective. The pseudocode is provided in Algorithm 1.

Algorithm 1 Find-fair-seed (S: Initial seeding)

```
if $S$ satisfies fairness then
    Return $S$;
end if
$done \leftarrow$ false;
while $!done$ do
    $done \leftarrow$ true;
    Find-fair-seed (the first half of $S$);
    Find-fair-seed (the second half of $S$);
    $N \leftarrow$ the set of players in $S$;
    $wp \leftarrow$ winning probabilities of the players in $N$;
    $max\_dif \leftarrow 0$;
    for $i, j \in N : j - i > max\_dif$ do
        if $wp(i) < wp(j)$ then
            $done \leftarrow$ false;
            $max\_dif \leftarrow j - i$;
            $i^* \leftarrow i$;
            $j^* \leftarrow j$;
        end if
    end for
    if $!done$ then
        Swap the seeding positions of $i^*$, $j^*$ in $S$;
    end if
end while
Return $S$;
```

The recursive algorithm attempts to find a fair seeding for each half of the initial seeding. It then checks to see if there are any pairs of players that violate the order preserving condition. If there are, it will pick the pair with the maximum difference in ranking (on the dominance graph) between the players and swap the seeding positions of those players. It will then repeat the whole process.

To test the algorithm, we generate 100k test cases of monotonic winning probability model for each of the values of n between 8 and 256. For each test case, we randomly generate the winning probability matrix for n players and then use the algorithm to find a seeding that satisfies order preservation. We use S_n^* (defined in Section 2) as the initial seeding input.

It is not trivial to generate the monotonic winning probability matrix since we need to make sure the matrix satisfies the monotonicity while ensuring certain randomness. The process is composed of the following steps:

1. Generate the winning probabilities of player 1 by sampling $(n-1)$ values uniformly from [0.5,1] and sort them in ascending order.
2. Assign $P_{ii} = 0.5 \, \forall i$.
3. Generate P_{ij} $(\forall (i,j) : i < j)$ by sampling uniformly between $P_{i(j-1)}$ and $P_{(i-1)j}$, and then assign $P_{ji} = 1 - P_{ij}$.

We show in Figure 5.7 a graph of the average and maximum number of swaps that are made before a fair seeding is found. The graph shows that both quantities grow linearly with n. And even in the case of $n = 256$ players, the maximum number of swaps needed is still very small.

Based on the empirical success of the algorithm, we conjecture that order-preservation is always achievable. However, the criterion becomes provably impossible to achieve when a relatively weak condition is added. We discuss this additional condition and how it interacts with the order preservation criterion in the next section.

5.5.3 Robustness against Dropout

The experimental results of the heuristic algorithm in the previous section suggest that perhaps it is easy and always possible to find an order preserving seeding given the winning probability matrix. However, this optimism is quite fragile. For example, in this section

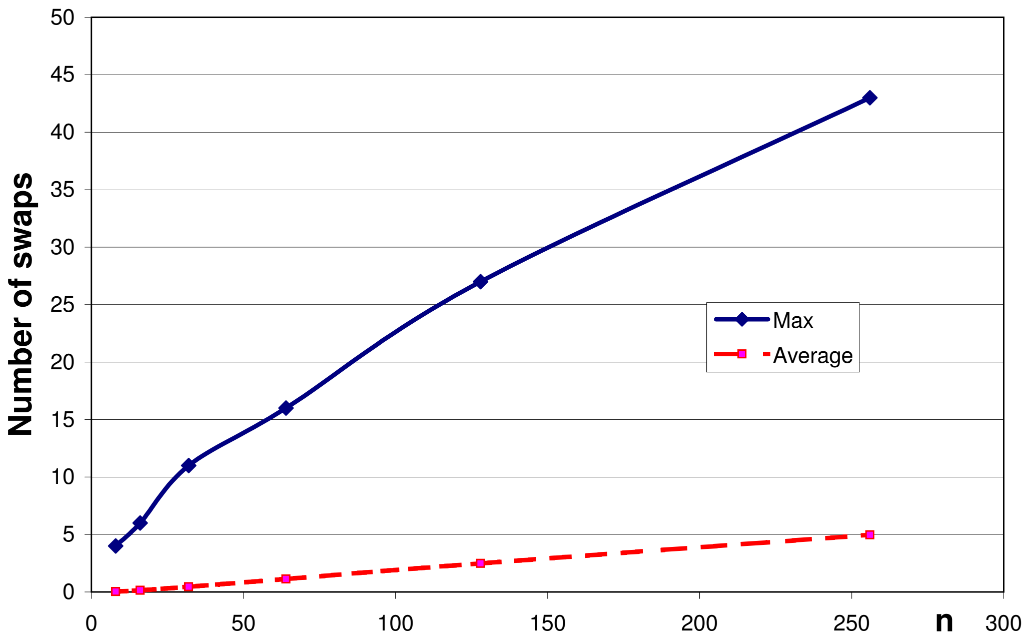

Figure 5.7: The average and maximum number of swaps over 100k tournaments for each n

we show that when we add a reasonable requirement of robustness against dropout, the criterion becomes provably impossible to achieve in certain cases.

A dropout occurs when a player forfeits his participation in a tournament due to exogenous factors such as injuries or illness, or in order to manipulate the outcome of the tournament. This phenomenon can significantly influence the winning probabilities of the remaining players. We define robustness against dropout as the following: there does not exist a pair of players (i, j) such that i dominates j, and j gains while i loses in winning probability after some player drops out. We allow the re-seeding of the remaining players after a dropout and if the tournament has to be balanced, a dummy player will be added.

Definition 12. (Robustness against Dropout) *Given a set N of players, a winning probability matrix P, and a knockout tournament $KT = (T, S)$. The seeding S is robust against dropout if for any dropout of player $k \in N$, there exists a seeding S' for the remaining $|N|-1$ players such that $\forall(i, j) \in N\backslash\{k\}$, if i dominates j and $q(i,(T, S'))-q(i,(T, S)) \leq 0$ then $q(j,(T, S')) - q(j,(T, S)) \leq 0$.*

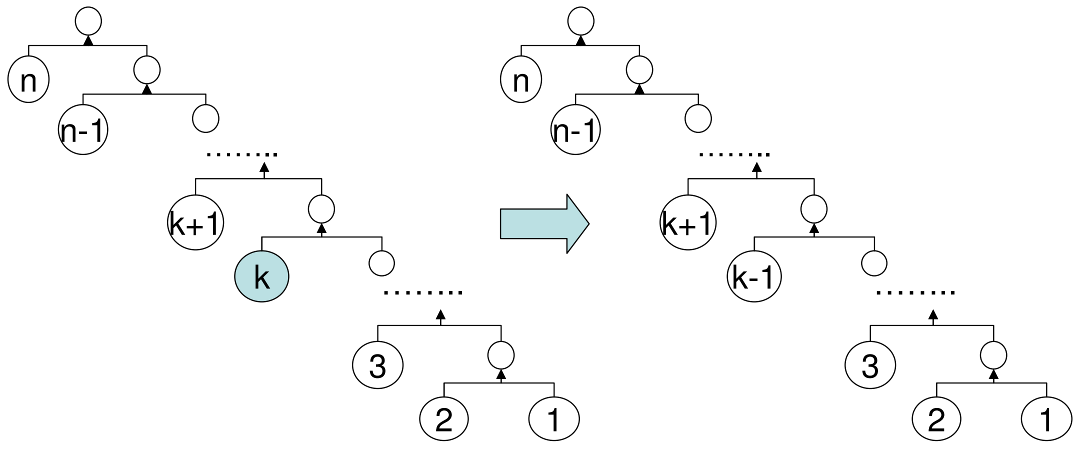

Figure 5.8: A tournament robust to the dropout of any player k

It is easy to show that given a winning probability matrix, the robustness requirement is achievable for both balanced and unconstrained tournaments (e.g., the tournament shown in Figure 5.8). Indeed the condition is quite weak when used alone. There are other stronger conditions that can be used. For example, no weak player gains in winning probability more than the players stronger than him (this in fact implies the requirement we proposed). However, when coupled with the order preservation requirement, this weak requirement is already enough to make it no longer always possible to find a seeding that can satisfy simultaneously both (even when the winning probabilities are monotonic).

Theorem 16. *For any $n = 2^r \geq 8$, there exists a set of n players with a monotonic winning probability matrix P such that it is not possible to find a seeding S such that S satisfies simultaneously the order preservation requirement and the condition for robustness against dropout.*

Proof. For a given n, we show how to construct the winning probability matrix P. There will be 3 top players and $n - 3$ dummy players. The winning probabilities between the top 3 players are: $P_{1,2} = 0.5$, $P_{1,3} = 0.6$, and $P_{2,3} = 0.5$. The dummy players will lose against the top 3 players with probability 1. Thus the tournament is reduced to a tournament between these 3 players. There are only three possible seedings: $[1\ vs.\ 2]\ vs.\ 3$, $[1\ vs.\ 3]\ vs.\ 2$, or $[2\ vs.\ 3]\ vs.\ 1$. The first two seedings are not order preserving (e.g., for the first seeding S_1, $q(1, S_1) = 0.3 < 0.45 = q(3, S_1)$). The last seeding does not satisfy the condition of robustness: when player 3 drops out, the winning probability of player 2 increases from 0.25 to 0.5, whereas the winning probability of player 1 decreases from 0.55 to 0.5. Note that this proof works for both balanced and unbalanced tournament. $\square$

5.6 Discussion

Fairness in tournament designs is an important consideration and to date, the literature on tournament designs has not adequately addressed it. In this chapter we have introduced two alternative fairness criteria: envy-freeness and order preservation. We showed that when there are no constraints on the structure of the tournament, fairness can be achieved easily for both criteria. For balanced tournaments, we provided two impossibility results: one for any seeding under the first criterion, and the other for any fixed seeding under the second criterion. When the seeding can vary depending on the actual values of the winning probabilities, we proposed a heuristic algorithm to find a seeding that satisfies order preservation. We showed that the algorithm is efficient and effective through experiments.

However, our hope of being able to find a fair seeding in all cases was dimmed by the fact that when we included a requirement of robustness, the criterion became provably impossible to satisfy for all cases. This suggests one should try to find the conditions of the winning probabilities between players that will guarantee the existence of a fair seeding. This remains an interesting open problem.

Besides the axiomatic approaches to fairness, one can also use a quantitative approach by defining how fair a seeding is based on some function over the pairwise inversions. The function here can be inversely proportional to the maximum value, the average, or the root mean square of the pairwise inversions. Each pairwise inversion in turn can be defined as the difference in winning probabilities when a weaker player has a higher probability of winning the tournament than another stronger player. The objective here then becomes maximizing this fairness measure.

Chapter 6

Optimality of Ordinal Solutions

6.1 Introduction

In the previous two chapters, we have mainly focused on the case in which the exact winning probabilities between the players are known and use this information to optimize or improve our seedings. Though these quantities can be obtained from past statistics, in reality, they are usually not available and can be inaccurate. The most common way of seeding a tournament is often based on the relative rankings of the players instead. We call this an ordinal solution. As an example, we show in Figure 6.1 one of the seedings most often used in practice. In this chapter we shift our focus to ordinal solution to analyze whether this is the right approach to seeding a tournament, or what ordinal seeding is optimal.

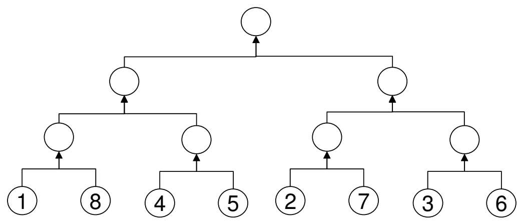

Figure 6.1: A tournament with 8 players

Over the past 40 years these questions have only been partially explored. For the questions to be well defined, we need to be clear about the objective function being optimized.

Most of the previous work focused on maximizing the so-called *predictive power* of the tournament (the probability that the strongest player will win) for up to 8 players. We significantly broaden the scope of the analysis along several dimensions. First, we introduce two additional objective functions: maximizing the expected value of the winner, and maximizing the revenue of the tournament, which we define in a way that correlates to the players' strengths and competitiveness. We will call these three objective criteria MaxP, MaxE and MaxR, respectively. Second, we provide a worst-case analysis of the optimality of ordinal seedings for the first two objectives. Third, we investigate the average-case performance for tournaments with size up to 128 players regarding all three objectives.

The biggest challenge in going beyond 8 players is the rapid growth of the number of distinct seedings as a function of the number of players (specifically, $O(\frac{n!}{2^{n-1}})$). As an example, for 16 players, this number is already 638×10^6. The interdependencies between different parts of the tournament make it very difficult to find the optimal seedings analytically. And the number of possible seedings is prohibitively large for analyzing the optimality of any seeding empirically. To overcome this challenge, we propose an easy-to-compute upper bound for each objective function. The upper bounds are provably correct, and allow us to evaluate different ordinal solutions and show how some can in fact approximate well the optimal values on average.

Finally, we also compare ordinal solutions to cardinal ones. Cardinal solutions are obtained by using all available information (e.g., the values of the winning probabilities between the players). We introduce a simple heuristic algorithm that takes as input an ordinal seeding and applies a simple hill-climbing search to improve on it. The most interesting result here is that even when given an input seeding that is quite suboptimal, the heuristic algorithm can still yield a close-to-optimal seeding.

To explain our results we need to lay out our settings, which we do in the next section. In Section 6.3 we summarize the existing results from the literature. We first focus on MaxP objective, and provide an analysis of the worst-case performance in Section 6.4, and average-case performance in Section 6.5. We then provide a similar analysis for MaxE and MaxR in Section 6.6. We present our cardinal solutions and experimental results for them in Section 6.7, and conclude in Section 6.8.

6.2 The Model

We divide our discussion of the formal setting into three parts: the tournament model, the objective function, and the type of solution allowed based on the information available.

6.2.1 Tournament Model

In this chapter, we focus on balanced binary tree (assuming the number of players is a power of 2) and the monotonic winning probability model. They are described in Chapter 2. In the monotonic model, the players are numbered from 1 to n in descending order of their unknown intrinsic strengths or abilities. Only the winning probabilities between the players are known and they reflect the rankings of the players. We restate the model below.

Definition. (Monotonic Model) *Given a set of n players, the winning probabilities between the players form a matrix P such that P_{ij} denotes the probability that player i will win against player j, $\forall(i \neq j) : 1 \leq i, j \leq n$, and P satisfies the following constraints:*
1. $P_{ij} + P_{ji} = 1$
2. $P_{ij} \leq P_{i(j+1)}$
3. $P_{ji} \geq P_{(j+1)i}$ (which is actually implied by (1) and (2))

Besides the winning probability, each player i also has a value v_i. This value should be interpreted as the strength or ability of the player, but in the definition of MaxR below we will assume that it is also a good proxy for the players' popularities or size of fan base. The only restriction on the values of v is that they are also monotonic with regard to the ordering of the players, i.e., $\forall(1 \leq i < j \leq n) : v_i \geq v_j$.

6.2.2 Objective Functions

We consider three objective functions for the tournament organizer. Recall from Chapter 2 that given the seeding of a tournament, the probability of any player i reaching any round can be calculated efficiently in $O(n^2)$. Let $\mathbb{S}$ be the set of all possible seeding sequences. Let $q_S^r(i)$ be the probability that player i will win round r in the tournament with the seeding $S \in \mathbb{S}$ (note that the final round is $\log n$). Let $Q_S^r(i)$ be the probability that i will reach round

$(r+1)$, i.e., $Q_S^r(i) = \prod_{k=1}^{r} q_S^k(i)$. We consider the following three different objective functions.

1. *MaxP:* Maximizing the predictive power:

$$\max_{S \in \mathbb{S}} Q_S^{\log n}(1)$$

2. *MaxE:* Maximizing the expected value of the winner:

$$\max_{S \in \mathbb{S}} \sum_{i=1}^{n} Q_S^{\log n}(i) \times v_i$$

3. *MaxR:* Maximizing the expected revenue of a tournament. We define the total revenue of a tournament as the sum of the revenues of all matches:

$$\max_{S \in \mathbb{S}} \sum_{r=1}^{\log n} \sum_{m=(i,j) \in M^r} Q_S^{r-1}(i) \times Q_S^{r-1}(j) \times Rev(m,r)$$

   where M^r is the set of all possible matches that can happen in round r, and $Rev(m,r)$ is the revenue made by having the match m in round r.

The first criterion, MaxP, is in fact a special case of the objective function we considered in Chapter 4 (i.e., maximizing the winning probability of a target player). Here the best player (player 1) is the target player. This has been the focus of much work (see, e.g., [15, 1, 27, 12]). MaxE has also appeared in the literature [15], while MaxR is novel. Note that MaxP is in fact a special case of MaxE when $v_i = 0$ $(\forall i > 1)$ and $v_1 = 1$. Since the value of v_i is monotonic and can be used to model the strengths, or popularities of the players, MaxE has natural interpretations. It can be viewed as maximizing the expected strength of the winner, or maximizing the chance to get a favorable winner.

Regarding MaxR, there are many different ways to model the revenue of a single match. Here we make the following assumptions: (1) A match at a later round should generate more revenue per ticket sale; (2) A team with a higher value (strength, popularity) would attract a bigger audience; (3) A more competitive match would also attract more viewers.

Based on these assumptions, we define the revenue of each match as the following:

$$Rev(m = (i, j), r) = r \times [(v_i + v_j) - |P_{ij} - P_{ji}|]$$

Within the square brackets, the first term corresponds to the popularity of the two teams, and the second term indicates the competitiveness of the match between them. Higher values of the teams will increase the revenue of the match, while bigger gap in the winning probabilities will decrease it.

6.2.3 Solution Types

In our work, we consider two types of seeding algorithms: Ordinal vs. Cardinal. For ordinal solutions, the tournament organizer only uses the rankings of the players. Thus these solutions are fixed seeding sequences that are applied for any ordered set of players regardless of the actual winning probabilities between them. We encode each seeding for a tournament of n players by a permutation of the numbers between 1 and n. The k^{th} number is the ranking of the player that will be placed at the k^{th} leaf node of the binary tournament tree, from left to right.

We want to differentiate two different categories of ordinal seedings:

- **Category 1:** Player 1 plays against players in the set $\{(n-2^{r-1}+1), ..., (n-2^r+2)\}$ at round r
- **Category 2:** All the remaining seedings

One special seeding belongs to the first category is $S_1^n = [1, n, (n-1), (n-2), ..., 2]$. For example, $S_1^8 = [1, 8, 7, 6, 5, 4, 3, 2]$. Intuitively, this seeding aims to achieve MaxP by matching player 1 with the weakest player at each stage while letting strong competitors of player 1 match up with each other earlier.

The second special seeding, called S_2^n, is a seeding of Category 2 formed by matching in the first stage the strongest player with the weakest player, the second strongest player with the second weakest player, and so on and so forth. For the successive stages, the sub-tree containing the strongest player is combined with the sub-tree with the weakest players possible, so on and so forth.

$$S_2^n = [...i, (n-i+1), (\frac{n}{2}-i+1), (\frac{n}{2}+i), ...]$$

An example is $S_2^8 = [1, 8, 4, 5, 2, 7, 3, 6]$ shown in Figure 6.1. This seeding is widely used in practice, especially in major sport tournaments. One possible rationale is that it can delay the confrontation between strong players until later rounds, and intuitively helps to improve the expected strength of the winner.

Most of the past results focus on ordinal solutions [15, 16, 28]. This type of solution provides us with a simple universal solution that we can apply to any set of ranked players without knowing the real strength of each player. This property makes it very useful in the cases when the actual winning probabilities between the players are hard to obtain. Nevertheless, when these values are known, they can be used to improve the optimality of the solutions. The seedings decided based on winning probabilities and values of the players are called cardinal solutions. We will show a simple yet effective heuristic algorithm for finding cardinal solutions, and compare these two types of solutions in later sections.

6.3 Past Results

The problem of finding an optimal tournament has been investigated in very limited settings. In [15], the authors show that with monotonic model, for $n = 4$, the seeding sequence $S_1^4 = S_2^4 = (1, 4, 3, 2)$ will achieve both MaxP and MaxE. This accords with our intuition as discussed in the previous section. However, for $n = 8$, they show that S_1^8 is no longer the universal optimal seeding to achieve MaxP. They prove that the optimal solution can in fact be any one of the following 8 seeding sequences:

$(1, 8, 6, 7, 2, 3, 4, 5)$	$(1, 7, 5, 6, 2, 4, 3, 8)$
$(1, 8, 5, 7, 2, 3, 4, 6)$	$(1, 8, 5, 6, 2, 4, 3, 7)$
$(1, 8, 5, 6, 2, 3, 4, 7)$	$(1, 8, 5, 7, 2, 4, 3, 6)$
$(1, 7, 5, 6, 2, 3, 4, 8)$	$(1, 8, 6, 7, 2, 4, 3, 5)$

We denote this set of seeding sequences by A. Which sequence in the set is optimal depends on the winning probability matrix P. They also show examples in which each of the sequences is the optimal seeding.

In general, the optimality of S_1^n for MaxP is lost when there is a disruption in the "smoothness" of the winning probabilities, e.g., two players have an equal chance of beating each other but one of them has a much better chance of winning against a third team. In [27], smoothness is enforced by making the assumptions that each player i has an ability v_i drawn from certain distributions and that the winning probabilities are specified through the Thurstone-Mosteller Model: $P_{ij} = \Phi(v_i - v_j)$ where $\Phi(\cdot)$ is the CDF of the standard normal distribution. With these assumptions, the probability of the best player winning the tournament is calculated numerically for each of the possible seeding sequences for $n = 8$ and S_1^8 is shown through experiments to be indeed the optimal sequence.

We significantly extend upon these past results in both theoretical and experimental directions. We first focus on MaxP, and provide a worst-case analysis of the optimality of ordinal solutions. We then propose a method that allows us to evaluate different ordinal solutions for tournaments of size up to 128 players. We also investigate other objective functions and introduce a robust and effective heuristic algorithm to find cardinal solutions. To the extent of our knowledge, this is the first time a cardinal solution is proposed and evaluated extensively.

6.4 MaxP: Worst-Case Performance

In this section, we discuss the worst-case performance of ordinal solutions for MaxP (maximizing the predictive power) with a special focus on the seeding S_1^n. We divide the discussion into two parts: when $n = 8$, and when $n > 8$.

6.4.1 Results for $n = 8$

When $n = 8$, even though it has been shown in [15] that S_1^8 ([1 8 7 6 5 4 3 2]) is not always the optimal seeding for maximizing the predictive power, intuitively it is still a very good candidate. In S_1^8, strong players (besides player 1) are matched up against each other in the early stages and only have a chance to play against player 1 in very late stages. This significantly increases the chance of player 1 playing against a weaker player at every round and consequently improves his chance to win the tournament.

This suggests that even though S_1^8 is not always the optimal solution, it can still be close to optimal. Indeed we have the following result on how far off S_1^8 can be from the optimal solution in the worst case.

Theorem 17. *For a balanced knockout tournament of size 8, the difference between the value of the seeding S_1^8 and the optimal values is at most $\frac{1}{8}$. Let $\mathbb{P}$ be the set of all possible winning probability matrices, and $\mathbb{S}$ be the set of all possilbe seedings for 8 players, we have the following.*

$$\forall P \in \mathbb{P}, \forall S' \in \mathbb{S}, Q_{P,S'}^{\log n}(1) - Q_{P,S_1^8}^{\log n}(1) \leq \frac{1}{8}$$

Proof. First note that from [15], the optimal seeding can only be one of the 8 sequences in the set A:

$[1, 8, 6, 7, 2, 3, 4, 5]$ $[1, 7, 5, 6, 2, 4, 3, 8]$
$[1, 8, 5, 7, 2, 3, 4, 6]$ $[1, 8, 5, 6, 2, 4, 3, 7]$
$[1, 8, 5, 6, 2, 3, 4, 7]$ $[1, 8, 5, 7, 2, 4, 3, 6]$
$[1, 7, 5, 6, 2, 3, 4, 8]$ $[1, 8, 6, 7, 2, 4, 3, 5]$

Thus we only need to compare the difference between the objective values of these seedings and of S_1^8.

We slightly change the notation to use $q_k^r(S)$ to denote the probability that player k will win round r given the seeding S. The probability of player 1 winning the tournament is $Q_{P,S}^{\log n}(1) = \sum_{k=1}^{3} q_1^k(S)$. We can divide each seeding into two halves and compare each half separately. The first half represents the first two rounds of player 1, and the second half the last round.

There are four different seedings possible for the first half: [1,8,6,7], [1,7,5,6], [1,8,5,6], [1,8,5,7]. It is straightforward to show that the probability of player 1 winning in [1,8,6,7] is better than the other three. For example:

$$
\begin{aligned}
&q_1^1([1,8,6,7]) \times q_1^2([1,8,6,7]) = P_{18}(P_{16}P_{67} + P_{17}P_{76}) \\
&= P_{18}(P_{16} + (P_{17} - P_{16})P_{76}) \geq P_{18}P_{16} \\
&\geq P_{18}(P_{16} + (P_{15} - P_{16})P_{56}) \\
&= q_1^2([1,8,5,6]) \times q_1^2([1,8,5,6])
\end{aligned}
$$

For the second half of the seedings, we need to compare $q_1^3([2,3,4,5])$ to $q_1^3([2,3,4,k])$, and $q_1^3([2,4,3,k])$ with $k \geq 5$. Let's first consider the seeding [2,3,4,k].

Since $\frac{1}{2} \leq P_{24} \leq P_{25} \leq P_{2k} \leq 1$ ($\forall k \geq 5$), and $P_{ij} + P_{ij} = 1$, we have the following.

$$
\begin{aligned}
&q_2^2([2,3,4,5]) - q_2^2([2,3,4,k]) \\
&= P_{23}[P_{24}P_{45} + P_{25}P_{54}] - P_{23}[P_{24}P_{4k} + P_{2k}P_{k4}] \\
&= P_{23}[P_{24}(P_{45} - P_{4k}) + P_{25}(1 - P_{45}) - P_{2k}(1 - P_{4k})] \\
&= P_{23}[P_{24}(P_{45} - P_{4k}) - (P_{2k} - P_{25}) + (P_{2k} - P_{25})P_{4k} - P_{25}(P_{45} - P_{4k})] \\
&= P_{23}[(P_{24} - P_{25})(P_{45} - P_{4k}) + (P_{2k} - P_{25})(P_{4k} - 1)] \\
&\leq P_{23}(P_{45} - P_{4k})(P_{24} - P_{25}) \\
&\leq \frac{1}{2}P_{23}(P_{4k} - P_{45})
\end{aligned}
$$

Similarly: $q_3^2([2,3,4,5]) - q_3^2([2,3,4,k]) \leq \dfrac{1}{2}P_{32}(P_{4k} - P_{45})$

Let's consider the difference between the winning probabilities of player 1 for round 3 between the two seedings:

$$\begin{aligned}
q_1^3([2,3,4,k]) - q_1^3([2,3,4,5]) &= \sum_{i=2,3,4} P_{1i}[q_i^2([2,3,4,k]) - q_i^2([2,3,4,5])] \\
&\quad + P_{1k} \times q_k^2([2,3,4,k]) - P_{15} \times q_5^2([2,3,4,5]) \\
&= -P_{12} \times (q_2^2([2,3,4,5]) - q_2^2([2,3,4,k])) \\
&\quad - P_{13} \times (q_3^2([2,3,4,5]) - q_3^2([2,3,4,k])) \\
&\quad + P_{14} \times (q_4^2([2,3,4,k]) - q_3^2([2,3,4,5])) \\
&\quad + P_{1k} \times q_k^2([2,3,4,k]) - P_{15} \times q_5^2([2,3,4,5])
\end{aligned}$$

We know that the sum of winning probabilities for each round must be equal 1:

$$\sum_{i=2,3,4,k} q_i^2([2,3,4,k]) = 1 = \sum_{i=2,3,4,5} q_i^2([2,3,4,5])$$

Thus we have the following.

$$\begin{aligned}
&q_1^3([2,3,4,k]) - q_1^3([2,3,4,5]) = \\
&\qquad (P_{14} - P_{12}) \times (q_2^2([2,3,4,5]) - q_2^2([2,3,4,k])) \\
&\qquad + (P_{14} - P_{13}) \times (q_3^2([2,3,4,5]) - q_3^2([2,3,4,k])) \\
&\qquad + (P_{1k} - P_{14}) \times q_k^2([2,3,4,k]) \\
&\qquad + (P_{14} - P_{15}) \times q_5^2([2,3,4,5]) \\
&\le \frac{1}{2}\frac{1}{2}P_{23}(P_{4k} - P_{45}) + \frac{1}{2}\frac{1}{2}P_{32}(P_{4k} - P_{45}) \\
&\qquad + (P_{1k} - P_{14})(1 - P_{4k})(P_{k2}P_{23} + P_{k3}P_{32}) \\
&\le \frac{1}{4}(P_{4k} - P_{45}) + \frac{1}{4}(1 - P_{4k}) \\
&\le \frac{1}{4} - \frac{1}{4}P_{45} \le \frac{1}{8}
\end{aligned}$$

Now let's consider the seeding [2,4,3,k], we have:

$$\begin{aligned} q_3^2([2,4,3,k]) &\leq q_3^2([2,3,4,5]) \\ q_4^2([2,4,3,k]) &\leq q_4^2([2,3,4,5]) \\ q_k^2([2,4,3,k]) &\leq q_5^2([2,3,4,5]) \end{aligned}$$

Thus the only way for the seeding [2,4,3,k] to improve the winning probability of player 1 is to decrease probability of player 2 winning the second round. Let's consider the difference between the two winning probabilities:

$$\begin{aligned} q_2^2([2,3,4,5]) - q_2^2([2,4,3,k]) = &P_{23}(P_{24}P_{45} + P_{25}P_{54}) \\ &-P_{24}(P_{23}P_{3k} + P_{2k}P_{k3}) = (*) \end{aligned}$$

Since $P_{23} \leq P_{23}P_{3k} + P_{2k}P_{k3}$ then:

$$\begin{aligned} (*) &\leq P_{23}(P_{24}P_{45} + P_{25}P_{54} - P_{24}) \\ &\leq P_{23}(P_{25} - P_{23}) \leq P_{23}(1 - P_{23}) \leq \frac{1}{4} \\ \Rightarrow &q_1^3([2,3,4,k]) - q_1^3([2,3,4,5]) \\ &\leq (P_{12} - P_{1k}) * (q_2^2([2,3,4,k]) - q_2^2([2,4,3,5])) \leq \frac{1}{8} \end{aligned}$$

Putting the two halves together, we have the desired property since the probability of player 1 winning the first two rounds (from the first half) is at most 1. This bound is tight since we can come up with an example where the equality holds. $\square$

The result indicates that S_1^8 provides a good guarantee for the worst-case performance. However one might ask whether this is the best guarantee. The following theorem shows that it is indeed the case.

Theorem 18. *For a balanced knockout tournament of size 8, the difference between the values of any ordinal seeding and the optimal values is at least $\frac{1}{8}$ in the worst case. Let $\mathbb{P}$ be the set of all possible winning probability matrices, and $\mathbb{S}$ be the set of all possilbe* *seedings for 8 players, we have the following.*

$$\forall S \in \mathbb{S}, \exists P \in \mathbb{P}, S' \in \mathbb{S} : Q^{\log n}_{P,S'}(1) - Q^{\log n}_{P,S}(1) \geq \frac{1}{8}$$

Proof. To show the bound, for each seeding, we just need to point out a winning probability matrix P and a seeding S' such that the inequality holds. Let's consider two types of seeding:

1. Category 1: Player 1 plays against $\{8\}$, $\{7, 6\}$, $\{5, 4, 3, 2\}$ in round 1, 2, 3 respectively. For example: [1 8 6 7 2 3 5 4].
2. Category 2: All the remaining seedings

For the first type of seeding, there are three possible choices for the second half of the seeding: [2 3 4 5], [2 4 3 5], [2 5 3 4]. For each of the choices, we show an instance where the inequality holds:

1. For $[a_1, a_2, a_3, a_4, 2, 3, 4, 5]$: Let $P_{12} = P_{13} = P_{23} = P_{24} = P_{43} = P_{45} = \frac{1}{2}$, $P_{25} = P_{35} = P_{46} = 1$, and $P_{1k} = 1(\forall k \geq 4)$. The remaining probabilities are assigned such that monotonicity holds. $[a'_1, a'_2, a'_3, a'_4, 2, 3, 4, 6]$ (in which we swap the positions of player 5 and 6 with each other) increases the winning probability of player 1 by $\frac{1}{8}$.
2. For $[a_1, a_2, a_3, a_4, 2, 4, 3, 5]$: $P_{12} = P_{23} = P_{35} = \frac{1}{2}$, $P_{2k} = 1(\forall k \geq 4)$, and $P_{1k} = 1(\forall k \geq 3)$; and the seeding $[a_1, a_2, a_3, a_4, 2, 3, 4, 5]$.
3. For $[a_1, a_2, a_3, a_4, 2, 5, 3, 4]$: Similar as the above.

For the second type of seeding, there must be at least one player in the seeding that is out of order, e.g., [1 8 7 5 6 4 3 2] in which the player 5 is in the wrong set. Let k be that first player in the seeding starting from the left. We can use the following winning probabilities: $P_{1i} = 1(\forall i > k)$, $P_{1i} = \frac{1}{2}(\forall i \leq k)$, $P_{ji} = 1(\forall j \leq k < i)$, and $P_{ji} = \frac{1}{2}$ for the remaining winning probabilities that we have not specified yet. Given a seeding of the second type, the winning probability of player 1 is at most half of [1 8 7 6 5 4 3 2]. Note that since $k < 8$, $q^3_1(S^8_1) \geq \frac{1}{4}$ and hence the difference is at least $\frac{1}{8}$ □

Thus, the results in this section provide at least one justification for using S_1^8: It comes along with a constant-factor worst-case approximation guarantee, and this is the best worst-case guarantee possible among all ordinal seedings.

6.4.2 Results for $n > 8$

When n grows larger than 8, it turns out that the ordinal solutions including S_1^n can no longer approximate the optimal value well in the worst case. We have the following result.

Theorem 19. *When $n \geq 8$, the worst-case ratio between the optimal value and the objective value of an ordinal solution is at least $\frac{3n}{2(n+2)}$ if the solution is of Category 1, and at least 2 if it is of Category 2. Let $\mathbb{P}$ be the set of all possible winning probability matrices, $\mathbb{S}_1$ and $\mathbb{S}_2$ be the set of ordinal seedings of Category 1 and 2 respectively, we have the following.*

1. $\forall S \in \mathbb{S}_1, \exists P \in \mathbb{P}, \exists S' \in \mathbb{S}_1 \cup \mathbb{S}_2 : \frac{Q_{P,S'}^{\log n}(1)}{Q_{P,S}^{\log n}(1)} \geq \frac{3n}{2(n+2)}$
2. $\forall S \in \mathbb{S}_2, \exists P \in \mathbb{P}, \exists S' \in \mathbb{S}_1 \cup \mathbb{S}_2 : \frac{Q_{P,S'}^{\log n}(1)}{Q_{P,S}^{\log n}(1)} \geq 2$

Proof. The proof of this theorem is similar to the proof for part 2 of Theorem 17. We consider the two categories of seedings separately.

1. *Category 1*: Player 1 plays against players in the set $\{(n-2^{r-1}+1), ..., (n-2^r+2)\}$ at round r

For the ordinal seeding of Category 1, in the final round, player 1 will face one of the players in the set $\{2, ..., \frac{n}{2}+1\}$. In other words, the second half of the tournament tree will contain these players. Let's consider the following winning probabilities:

- $\forall i, j : 1 \leq i, j \leq \frac{n}{4}+1, P_{ij} = \frac{1}{2}$
- $\forall i, j : 1 \leq i \leq \frac{n}{4}+1, \frac{n}{4}+3 \leq j \leq n, P_{ij} = 1$
- $P_{1(\frac{n}{4}+2)} = 1$
- $\forall i : 2 \leq i \leq \frac{n}{2}+1, P_{i(\frac{n}{4}+2)} = \frac{1}{2}$
- $\forall j : \frac{n}{2}+2 \leq j \leq n, P_{(\frac{n}{4}+2)j} = 1$

In other words, player $(\frac{n}{4}+2)$ ties with the top $(\frac{n}{2}+1)$ players except player 1, whom he loses to with probability 1. The top $(\frac{n}{4}+1)$ players win with probability 1 against all other players except player $(\frac{n}{4}+2)$. The probability that player $(\frac{n}{4}+2)$ will get to the final is $\frac{2}{n}$ since he has to play against another $\frac{n}{2}-1$ players whom he ties with. In the final round, player 1 either plays against player $(\frac{n}{4}+2)$ or one of the players in $\{2, ..., (\frac{n}{4}+1)\}$ (since they win against all the remaining players with probability 1). Thus the winning probability of player 1 with a seeding of the first type is:

$$1 \times \frac{2}{n} + \frac{1}{2} \times \frac{n-2}{n} = \frac{(n+2)}{2n}$$

One can notice that a better way to seed the players is: $[(1, n), (n-1, n-2), ..(\frac{n}{2}+1, \frac{n}{2}, .., \frac{n}{4}+3), ((\frac{3n}{4}+1, .., \frac{n}{2}+2, \frac{n}{4}+2)(\frac{n}{4}+1, ..., 2))]$
Here we are matching up player $(\frac{n}{4}+2)$ with the players in the set $\{\frac{n}{2}+2, .., \frac{3n}{4}\}$. Since player $(\frac{n}{4}+2)$ will win against these players with probability 1, he will have a probability of 1 to make to the semi-final playing against one of the player in $\{\frac{n}{4}+1, ..., 2\}$. Thus the chance of player $(\frac{n}{4}+2)$ to get to the final is $\frac{1}{2}$. The winning probability of player 1 with this seeding is:

$$1 \times \frac{1}{2} + \frac{1}{2} \times \frac{1}{2} = \frac{3}{4}$$

Therefore the ratio of the optimal value over the objective value of any seedings of type 1 is at least $\frac{3}{4} \times \frac{2n}{n+2} = \frac{3n}{2(n+2)}$

2. *Category 2*: All the remaining seedings

For the ordinal seedings of Category 2, we also use a similar strategy. Let k be the first player that is in the wrong set. We can use the following winning probabilities: $P_{1i} = 1 (\forall i < k)$, $P_{1i} = \frac{1}{2} (\forall i \geq k)$, $P_{ji} = 1 (\forall j \geq k > i)$, and $P_{ji} = \frac{1}{2}$ for the remaining winning probabilities. The worst-case ratio in this case is at least 2. □

As n becomes large, the lower bound of the worst-case ratio for ordinal seedings of Category 1 gets close to $\frac{3}{2}$ quickly. The theorem implies that the ordinal seedings can be much worse than the optimal solution. However, worst-case analysis does not always give the complete picture. In the next section, we investigate the average-case performance of ordinal solutions with special focus on S_1^n, which is intuitively a good candidate for the optimal solution.

6.5 MaxP: Average-Case Performance

To analyze the average-case performance of the ordinal solutions for the MaxP objective, we generate random test cases and evaluate the solutions for each test case. In this section, we first discuss the setup of our experiments, and then we introduce our framework for evaluate different ordinal solutions, and finally provide the experimental results.

6.5.1 Experiment Setup

For each of the test cases, we generate the values of the players and the winning probabilities between them. It is simple to generate the values of v. They are just random i.i.d. samples from the uniform distribution between [0,1], sorted in descending order. It is more complicated to generate the winning probabilities since we need to make sure the matrix satisfies the monotonicity condition while ensuring certain randomness. The process is composed of the following steps:

1. Generate the winning probabilities of player 1 by sampling $(n-1)$ values uniformly from [0.5,1] and sort them in ascending order.
2. Assign $P_{ii} = 0.5 \, \forall i$.
3. Generate P_{ij} ($\forall i, j : 1 \leq i < j \leq n$) by sampling uniformly between $P_{i(j-1)}$ and $P_{(i-1)j}$, and then assign $P_{ji} = 1 - P_{ij}$.

For $n = 8$, there are only 315 different non-duplicate seedings. With this number, we can easily find the optimal solution for any objective function through exhaustive search. We can then evaluate our ordinal and cardinal solutions by comparing their objective values to the optimal values. However, for $n \geq 16$, it is not possible to find the optimal values through exhaustive search since there are simply too many seedings, e.g., for $n = 16$, there are 638×10^6 non-duplicate seedings. The interdependency of different components of the tournament makes it very difficult to efficiently find the optimal solution. To get a sense of the challenges here, note that the objective value is calculated as a summation of $2^{\frac{\log n(\log n-1)}{2}}$ terms, each of them is in turn a product of $(n-1)$ terms. Thus in order to evaluate our solutions, we propose an efficient algorithm to calculate the upper bound of the optimal value and then compare our solutions to this bound instead.

6.5.2 Results for $n = 8$

As we have shown in Section 6.4, S_1^8 ([1 8 7 6 5 4 3 2]) has the best worst-case guarantee. Our experiments show that it also has the best average-case performance. Out of $1M$ test cases, S_1^8 is optimal in 99.78% of the cases and achieves 100% of the optimal values on average (due to rounding). The ordinal seeding with the second highest chance of achieving optimality is [1 8 6 7 2 4 3 5] but with only 0.89% of the test cases. Note that the sum of the two values is over 100% since they can be optimal simultaneously. However, the actual numerical difference is still very small (99.14%).

We show the experimental results for some of the ordinal seedings in Table 6.1. The leftmost column indicates the ordinal seeding we consider, the second and third columns show respectively the percentage of the test cases in which the seeding is optimal, and the average percentage compared to the optimal value for each seeding. The seeding S_2^8 ([1 8 4 5 2 7 3 6]), which is the most commonly used seeding in practice, has very poor performance. It achieves optimalily for 0% of the test cases, and only 84.12% of the optimal value on average. A random ordinal seeding will achieve 71.88% of the optimal value, while the worst seeding in each case will achieve only 57.82%. This shows that the choice of seeding can significantly influence the optimality of the solution.

6.5.3 Results for $n > 8$

Since it is computationally infeasible to find the optimal solution for all the test cases, we come up with an efficient algorithm to find the upper bound of the predictive power given the winning probabilities between the players. We then compare the objective values of S_1^n, S_2^n and random seeding with this upper bound.

To find an upper bound, denoted MaxPB, for the objective MaxP, first note that the

	MaxP	
	Freq.	**Val.**
Average	NA	71.88
Worst	NA	57.82
S_2^8	0	84.12
S_1^8	99.78	100
18672435	0.89	99.14
18672534	0.22	98.74

Table 6.1: The average percentage compared to the optimal values and the percentage of test cases being optimal over 1M tournaments

probability of a target player t winning the tournament is equal to the product of the probabilities that t wins each round of the tournament. Thus for a given seeding, if we can find the upper bounds of the probabilities that t will win each round, we can find the upper bound of the probability that t will win the whole tournament.

Secondly, notice that any player k in the tournament has a chance to play against player t. For any given seeding, if players t and k can have a match at round r (i.e., they both share the same ancestor of height r), the chance of that match happening (assuming t will make to round r) is at least the lower bound of the probability that k will reach round r. Moreover, given a seeding S, at round r, there is a set N_S^r of opponents that player t may match up against, i.e., the set of players in the sub-tournament tree sharing a root with player t at round r. The probabilities of player t playing against each of them sum up to 1: $\sum_{k \in N^r} P(t \text{ plays k}) = 1$ (again assuming t will make to round r).

With these observations, we calculate MaxPB using Algorithm 2 as $upper\_bound(1 \text{ reaches round } (\log n + 1))$. In a nutshell, the algorithm calculates the upper bounds of the probabilities that the player t will survive each round, and then multiply those values together. For each round, the algorithm tries to match up the player against opponents as weak as possible. For example, assume we are calculating $upper\_bound(9, 4)$ in a tournament of 32 players. It will be the product of the upper bounds of the probabilities that player 9 will win against player 32, the set of players $\{31, 30\}$, and the set $\{29, 28, 27, 26\}$ in round 1, 2, and 3 respectively.

The upper bound of the probability that player t will win each round can then be calculated by the method presented from line 4 to line 17 in Algorithm 2. Here we set the probabilities of the stronger players playing against t to be the lower bound of the probabilities that those players will make to round r, weaker players to be the upper bound, and at most one player in between, such that the sum of the probabilities is 1.

In fact, given any seeding S, this method can be applied for the set of players N^r at any given round r of S to calculate the upper bound of the probability that player t will win against N^r at round r. We denote this as $UBW^r(t, N^r)$. And similarly, $LBW^r(t, N^r)$ denotes the lower bound probability.

Algorithm 2 Calculate $upper\_bound(t, r^*)$: upper bound of the probability t reaches round r^*

```
    $upb \leftarrow 1$;
    for $r \leftarrow 1$ to $r^*$ -1 do
3:      $sumP \leftarrow 0$;      $winProb \leftarrow 0$;
        for $k = n - 2^{r-1} + 1$ to $(n - 2^r + 2)$ do
            $addP \leftarrow lower\_bound(k, r)$;
6:          $winProb \leftarrow winProb + addP \times P_{tk}$ ;
            $sumP \leftarrow sumP + addP$;
        end for
9:      for $k = (n - 2^r + 2)$ to $n - 2^{r-1} + 1$ do
            $addP \leftarrow upper\_bound(k, r) - lower\_bound(k, r)$;
            $winProb{+}= min(addP, 1 - sumP) \times P_{tk}$;
12:         $sumP \leftarrow sumP + addP$;
            if $sumP \geq 1$ then
                Exit to outer loop;
15:         end if
        end for
        $upb \leftarrow upb \times winProb$;
18: end for
    return upb;
```

The function $lower\_bound(t, r)$ can be calculated in a method similar to $upper\_bound(t, r)$ but with the order of the players reversed so that the target player t will have to play against the strongest opponents first. We also reverse the steps from line 4 to line 17, so that stronger opponents will have probabilities playing against t equal to the upper bounds and weaker opponents lower bounds.

To show that $MaxPB$ is the correct upper bound of MaxP, we just need to show that $upper\_bound(t,r)$ and $lower\_bound(t,r)$ are correct for any given player t and round r. We have the following theorem.

Theorem 20. *The value calculated by $upper\_bound(t,r)$ is the upper bound of the probability that player* t *will reach round* r*, and $lower\_bound(t,r)$ the lower bound.*

Proof. We prove the correctness of the upper and lower bounds by using induction on the number of round r. We are showing the proof for the correctness of the upper bound here. The proof for the lower bound is similar.

Base case: When $r = 1$, the upper bound is 1, which is trivially correct.

Inductive step: Assume that the bounds are correct for r, i.e., $(\forall 1 \leq t \leq n)$, $upper\_bound(t,r)$ is higher than the probability that player t will reach round r for any seeding, and reversely for $lower\_bound(t,r)$. We need to show that they are also correct for $(r+1)$.

In order to show that the upper bound is correct, we need to show the two following statements.

1. Given a seeding S, let N_S^r be the set of players that player t will match up against at round r. We need to show that multiplying the upper bound of the probability that t will survive each round will give us the upper bound of the probability that t will reach round $r+1$.

$$Q_S^r(t) = \prod_{r'=1}^{r} q_S^{r'}(t) \leq \prod_{r'=1}^{r} UBW^{r'}(t, N_S^{r'}) = upper\_bound(t, r+1)$$

2. If we use the method above to calculate the upper bound, the seeding S_1^n gives us the largest upper bound among all possilbe seedings.

$$\forall S' \in \mathbb{S}, \prod_{r'=1}^{r} UBW^{r'}(t, N^{r'}{}_{S'}) \leq \prod_{r'=1}^{r} UBW^{r'}(t, N_{S_1^n}^{r'})$$

Part 1 is straightforward. Given a set N_S^r of players, we know that:

$$\sum_{i \in N^r} P(\text{k plays against i}) = 1$$

and

$$upper\_bound(i, r) \geq P(\text{k plays against i}) \geq lower\_bound(i, r)$$

By setting the probability of k playing against all players in N_S^r to be the lower bounds, and gradually increases these probabilities starting from the weakest players, $UBW^r(k, N_S^r)$ is at least the probability that k will survive round r. And by multiplying these terms together, we have the desired inequality.

Part 2 is much more complicated and we will show a sketch of the proof here. We need to show that for any given seeding S, using the set $N_S^{r'}$ $(\forall r' : 1 \leq r' \leq r)$ to calculate the upper bound will result in smaller value than using $N_{S_1^n}^r$ $(\forall r' : 1 \leq r' \leq r)$, in which at round r', $N_{S_1^n}^{r'} = \{(n - 2^{r'-1} + 1)...(n - 2^{r'} + 2)\}$. We show this by gradually converting S to S_1^n while not decreasing the value of the upper bound.

First note that since $S \neq S_1^n$, in S there must exist at least one inversion, i.e., $\exists i, r, r_1 < r_2 : i, (i+1) \in N_{S_1^n}^r$ but $i \in N_S^{r_1}, (i+1) \in N_S^{r_2}$. In other words, either i or $(i+1)$ (or both) belong to the wrong set(s). Let j be such a player with the smallest r_1. If there are several such players with the same value r_1, we pick the player with the highest number. For example, when S=[1 8 5 4 6 3 2 7], $j = 5$. We will then swap j with $j+1$, which is player 6 in this case. Let $N_S^{r_1}$ and $N_S^{r_2}$ be the sets that j and $(j+1)$ are in respectively. It is easy to show that

$$UBW^{r_1}(k, N_S^{r_1} \backslash \{j\} \cup \{j+1\}) \geq UBW^{r_1}(k, N_S^{r_1})$$

For $N_S^{r_2} \backslash \{j+1\} \cup \{j\}$, we need to consider two cases: (1) j is the player with the smallest number in N^{r_2}, (2) otherwise. For the first case, one can write out the formula for $UBW^{r_2}(k, N_S^{r_2} \backslash \{j+1\} \cup \{j\}) - UBW^{r_2}(k, N^{r_2})$ to show that the difference is non-negative. For the second case, notice that by swapping j with $(j+1)$, the probability of j playing against k will be higher than the previous probability of $(j+1)$. This means a decrease in the probability of other players $j' < j$ playing against k. Hence, this also increases the upper bound. □

Corollary 21. $MaxPB = upper\_bound(1, \log n + 1)$ *is the correct upper bound of MaxP.*

In Figure 6.2 we show a graph plotting the experimental results for MaxP. Here we generate $100k$ tournaments of size n for each $n \in [16, 128]$. The x-axis denotes the size of the tournament, and the y-axis denotes the average percentage of a particular solution when compared to the upper bound. As n grows exponentially, the objective values of our S_1^n solution remain close to the values of the upper bound. It achieves more than 90% of the upper bound on average.

This implies that our upper bound is relatively tight, and S_1^n is close to being optimal. Since the optimal value has to be in between the two values, using both of them allows us to have a good approximation of the optimal value. S_1^n might suffice as a solution used in practice if we do not have the additional information on winning probabilities available. S_2^n and randomized seedings perform significantly worse than S_1^n, as expected.

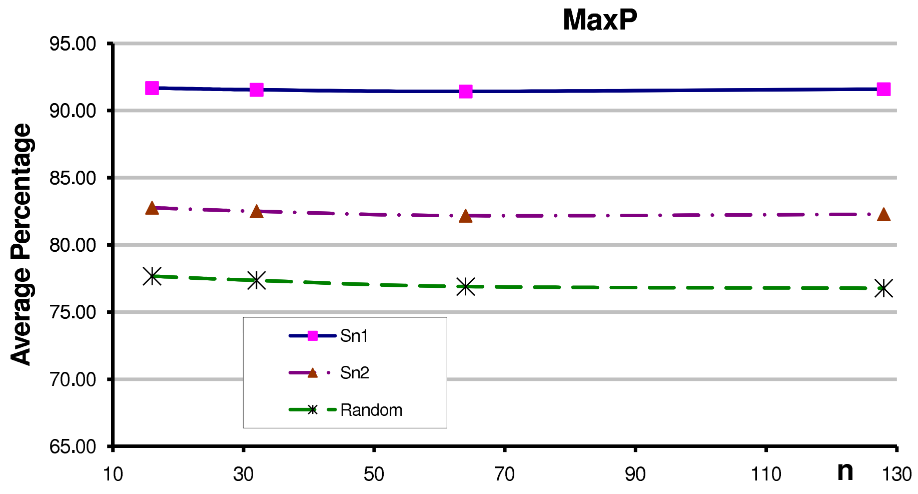

Figure 6.2: The average percentage of objective values of different solutions when compared to the upper bound for MaxP over 100k tournaments

6.6 Other objective functions: MaxE & MaxR

In this section, we extend our experiments to address the other two objective functions: maximizing the expected value of the winner, and maximizing the expected revenue of the tournament. Specifically, we aim to analyze how the objectives influence the optimality of the seeding solutions. Here we use the same experimental setup as in Section 6.5.

6.6.1 Results for $n = 8$

In Table 6.2, we present the experimental results for these two objective functions averaged over $1M$ tournaments. We only include here the seeding sequences that achieve optimality for at least 9% of the test cases for any objective. We also show the results for the two special seedings S_1^8 and S_2^8. Under each objective function, the first column shows the percentage of tournaments in which the seeding achieves optimality, and the second column of the table shows the percentage of the optimal value each seeding sequence achieves on average. We also include the average results over all seedings and input, and the results of the worst seeding averaged over all input.

	MaxE		**MaxR**	
	Freq.	**Val.**	**Freq.**	**Val.**
Ordinal				
Average	NA	94.54	NA	92.93
Worst	NA	87.36	NA	82.88
17452836	2.03	97.94	11.95	99.10
S_2^8	6.87	98.68	**22.60**	**99.50**
18562734	11.29	99.22	12.69	99.04
18572634	9.05	99.25	0.37	97.98
S_1^8	4.06	99.11	0.30	96.20
18672435	9.70	99.45	0.05	96.84
18672534	**40.07**	**99.63**	1.43	97.26

Table 6.2: The average percentage compared to the optimal values and the percentage of test cases with optimality over 1M tournaments

For MaxE, surprisingly, the sequence [1 8 6 7 2 5 3 4] has the best chance of being optimal. This is due to the cases in which there is a big gap between player 1 and player 2. Matching up player 1 against the weakest players earlier in the tournament will guarantee that player 1 appears in the final. S_1^8, however, despite having similar structure does not perform as well since players 2 and 3 are competing against each other too early.

For the MaxR objective, S_2^8 is the best seeding. It is interesting to note that the seeding [1 8 6 7 2 5 3 4] does not perform well for MaxR even though it is the best seeding for MaxE. In this seeding, player 1 is matched up with all weak opponents until it reaches the final. This reduces the competitiveness of the matches and hence the revenue of the tournament. The results also show that the widely used seeding S_2^8 actually helps to maximize the revenue of the tournament, rather than the expected strength of the winner. This justifies the use of this seeding from the organizer's perspective.

From the experimental data, we can make two observations: there is no ordinal solution that performs well across all three objective functions; for MaxE and MaxR, there is no seeding that achieves optimality with high frequency either, though the average value is very close to the optimal value. Thus ordinal solutions can be a good solution to use in practice, but one must be clear on what is being optimized for.

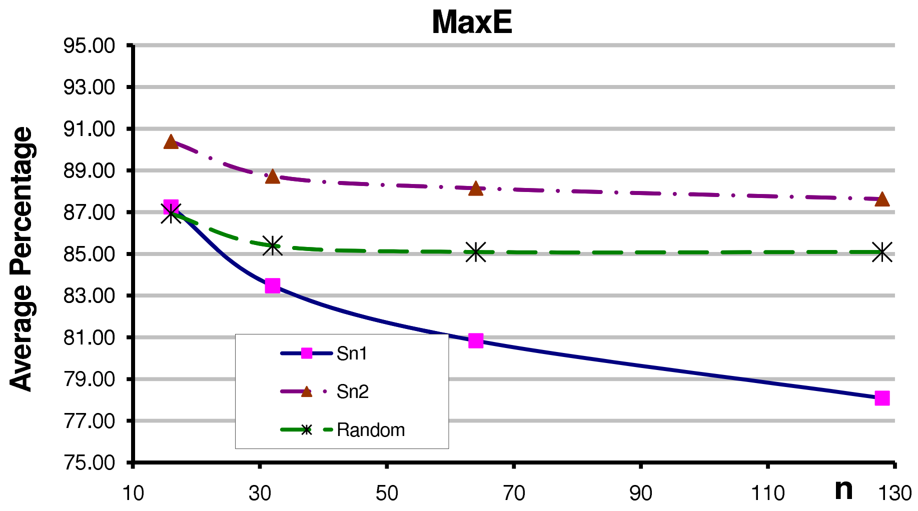

Figure 6.3: The average percentage of objective values of different solutions when compared to the upper bound for MaxE over 100k tournaments

6.6.2 Results for $n > 8$

First note that MaxP is a special case of MaxE in which the value of player 1 is 1 and the values of all other players are 0. Thus the same lower bound on the worst-case ratio of any ordinal seedings still applies here.

Corollary 22. *When $n \geq 8$, the worst-case ratio between the optimal value and the objective value of an ordinal solution is at least $\frac{3n}{2(n+2)}$ if the solution is of Category 1, and at*

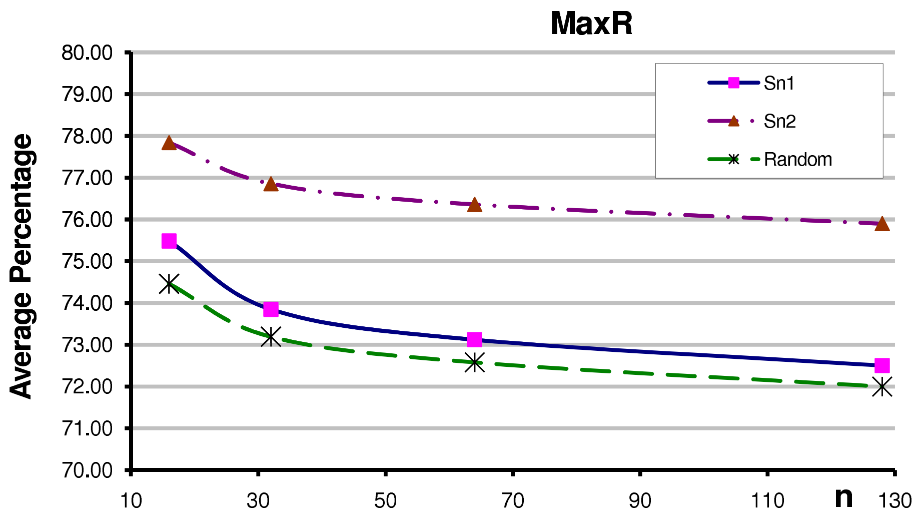

Figure 6.4: The average percentage of objective values of different solutions when compared to the upper bound for MaxR over 100k tournaments

least 2 if it is of Category 2.

The result indicates that the most commonly used S_2^n can in fact be much worse than the optimal seeding if one wants to maximize the expected value of the winner.

For the average-case performance, similarly to the experiments for MaxP, we need to use the upper bounds of the optimal values in order to evaluate the seeding solutions. Once we have the bounds for the probability that any player k will reach a certain round r, we can use them to find MaxEB, the upper bound of MaxE. Here we set the value of r to be $(\log n + 1)$, i.e., winning the tournament. If we set the winning probabilities of the stronger players to be the upper bound, weaker players to be the lower bound, and at most one player in between, such that the sum of the probabilities is 1, MaxEB is the expected value of the players based on these probabilities. We show the pseudocode for MaxEB in Algorithm 3.

To find an upper bound MaxRB of MaxR, notice that there are two components in MaxR formula: the expected value of the players and the competitiveness of each match. For the first component, we can use a similar method as in Algorithm 2 to find the upper bound on the expected value of the players at each round. For the second component, we can view each match between two players as a weighted edge in a graph with n nodes. The weight of edge (i, j) is the competitiveness between i and j. Thus to find the upper bound

Algorithm 3 Find MaxEB - MaxE Upper Bound

```
$MaxEB = 0;$        $sumP = 0$
for $k = n$ to $1$ do
    $addP \leftarrow lower\_bound(k, \log n + 1);$
    $MaxEB+ = addP \times v_k;$
    $sumP+ = addP;$
end for
for $k = 1$ to $n$ do
    $addP \leftarrow upper\_bound(k, \log n + 1)$
                $-lower\_bound(k, \log n + 1);$
    $addP \leftarrow min(addP, 1 - sumP);$
    $MaxEB+ = addP \times v_k;$
    $sumP+ = addP;$
    if $sumP \geq 1$ then
        Exit to outer loop;
    end if
end for
```

for the second component, for each round k, we can find the maximum-weight matching of size $\frac{n}{2k}$. This can be done in $O(n^3)$ by the algorithm described in [10] with addition of dummy nodes. The sum of the upper bounds for these two components from every round gives us MaxRB.

Theorem 23. *MaxEB and MaxRB are the upper bounds of MaxE and MaxR respectively.*

The proof for both of the bounds is fairly straightforward, and thus we do not present it here.

In Figure 6.3 and Figure 6.4, we show the experimental results for MaxE and MaxR respectively. We compare the values of our solutions to the upper bounds across $100k$ tournaments of size n for each value of $n \in \{16, ..., 128\}$. The results show a similar trend here. For MaxE, the values of S_2^n remain close to the upper bounds as n increases exponentially. It is interesting to note here that randomized seedings also work quite well. This is due to the fact that randomized seedings tend to spread the players uniformly and have characteristics similar to S_2^n.

For the MaxR objective function, the gap between the upper bound and the cardinal solution is slightly worse than with MaxP and MaxE. This is to be expected since the objective function is much more complicated here and makes it harder for the upper bound to remain tight.

6.7 Cardinal Solutions

The ordinal solutions we have addressed so far only use the relative rankings of the players as the input. There might be other information available such as the actual values of the winning probabilities and this information can be used to achieve better solutions. In this section, we investigate cardinal solutions by first proposing an effective and efficient heuristic algorithm to utilize the additional information, and compare the cardinal solutions generated by this algorithm to the ordinal ones.

6.7.1 Heuristic Algorithm for Cardinal Solutions

Our heuristic algorithm is based on the Hill-Climbing approach. Given an initial seeding sequence S, the algorithm will attempt to improve the seeding through $\log n$ rounds. For each round r with $r = \{1, ..., \log n\}$, the algorithm tries swapping every pairs of sub-tournament trees of height r to improve the objective value. At the end of the round, if there is a new seeding found with a better objective value, the whole round will be repeated. Otherwise the seeding with the best objective value so far will be used for the next round. When the algorithm is applied on the seeding S, we call the resulting seeding HC(S). This heuristic function is not only efficient but also applicable for any objective function, as long as the objective value of a given seeding can be calculated efficiently.

6.7.2 Results for $n = 8$

We show in Table 6.3 the experimental results of our heuristic algorithm applied on S_1^8, S_2^8 for all three objectives. Our cardinal solutions achieve optimality with very high percentages of test cases across all objectives. Their objective values are almost the same as the optimal values on average. This shows that the algorithm is indeed very effective. What makes it particularly interesting is that it can also greatly improve the optimality of a given seeding even when that seeding is not the best candidate for optimality, e.g., improving

S_2^8 from 6.87% to 94.99% for MaxE. This further demonstrates that the algorithm is very robust, regardless of the input seeding.

	MaxP		**MaxE**		**MaxR**	
	Freq.	**Val.**	**Freq.**	**Val.**	**Freq.**	**Val.**
Ordinal						
S_1^8	99.78	100	4.06	99.11	0.30	96.20
S_2^8	0	84.12	6.87	98.68	22.60	99.50
Cardinal						
$HC(S_1^8)$	**100.00**	**100.00**	**93.97**	**99.99**	**91.60**	**99.97**
$HC(S_2^8)$	**100.00**	**100.00**	**94.99**	**99.99**	**95.79**	**99.99**

Table 6.3: The percentage of test cases achieving optimality and the average percentage compared to the optimal values over 1M tournaments for different ordinal and cardinal solutions

One caveat is that the actual difference in values of the best ordinal seedings for each objective compared to those of the cardinal solutions is quite small. The results show for the case of $n = 8$ that cardinal methods are necessary if one really cares about the optimality of the solution. However from a different perspective, if one weighs in the additional cognitive and administrative burden of cardinal methods, as well as the possible uncertainty regarding the input numbers, ordinal methods seem to provide a very attractive alternative.

6.7.3 Results for $n > 8$

We show in Figure 6.5, 6.6, and 6.7 the improvements of our heuristic algorithm over the ordinal solutions for each of the objectives MaxP, MaxE, and MaxR respectively. The results here display the same trend as for the case of $n = 8$. Our cardinal solutions make improvements over all ordinal solutions. For ordinal solutions that are already performing well, the improvements are small. With MaxP and MaxE, our cardinal solution HC(S_1^8) and HC(S_2^8) can only make about 4% improvement when n is large. Yet this also allows the cardinal solutions to get even closer to the upper bounds (in the range of 95%). It shows that our upper bounds are in fact quite tight for MaxP and MaxE.

For the ordinal solutions that are not performing well, the improvements are much more significant. It is interesting that the end results of our heuristic algorithm applied on

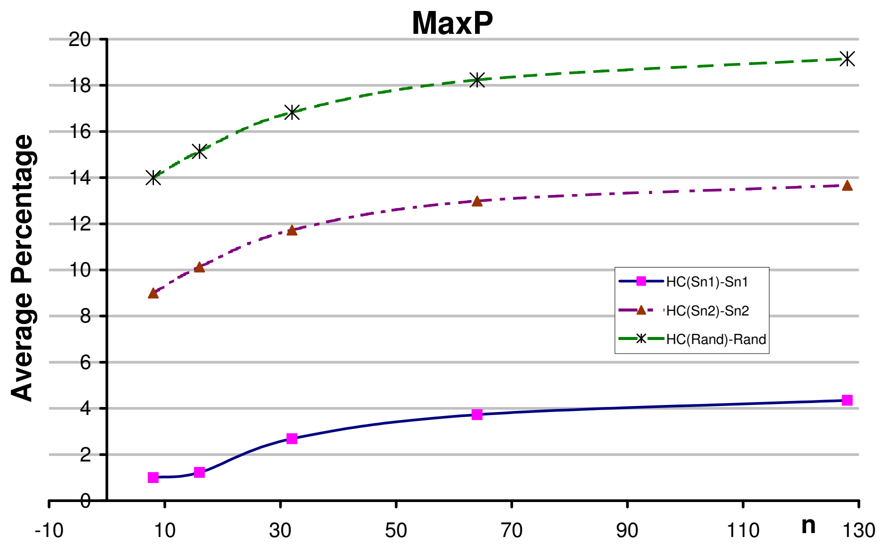

Figure 6.5: The improvement of cardinal solutions over ordinal solutions when compared to the upper bound for MaxP over 100k tournaments

different ordinal solutions are close to each other across all objective functions[1]. This once again confirms that our heuristic algorithm is in fact very robust and effective, regardless of the input seeding and objective functions.

The improvement of the cardinal solution is a bit smaller overall for MaxR. Here we optimize conflicting terms, e.g., increasing the value (strength) of one player might increase the revenue but might also decrease the competitiveness of the match if that player already has a better chance of winning. Hence the heuristic algorithm is not as effective as for the other objective functions.

6.8 Discussion

In this chapter we have investigated ordinal solutions for the problem of finding optimal seedings. We provided various theoretical and experimental analysis. We have substantially expanded the scope of the analysis from past work by considering three different objective functions, and tournaments of size up to 128. In order to perform experiments for

[1]We do not show the cardinal solutions explicitly but they can be inferred from Figure 6.2 and 6.5

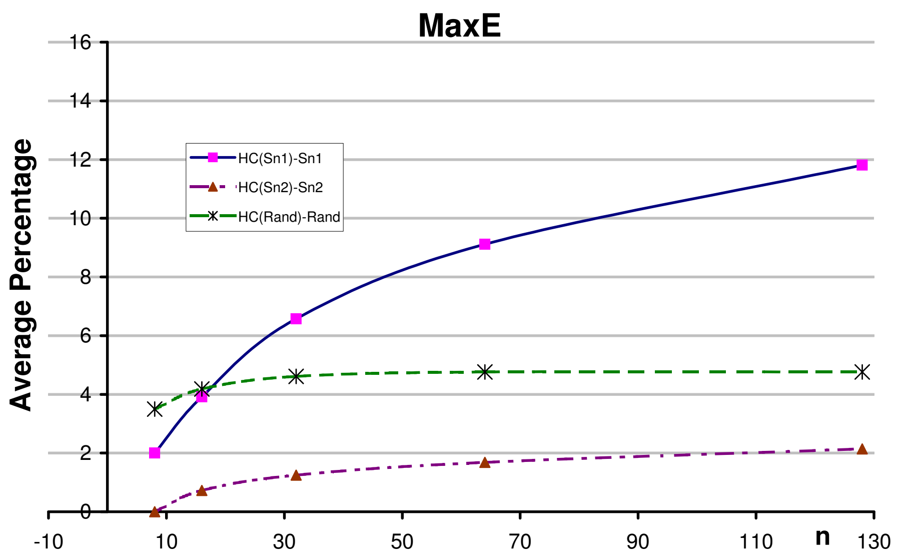

Figure 6.6: The improvement of cardinal solutions over ordinal solutions when compared to the upper bound for MaxE over 100k tournaments

tournaments of size larger than 8, we introduced a non-trivial upper bound for each objective function, and evaluated different solutions using these upper bounds. Across various settings, we showed that the optimal seeding sequence can vary significantly depending on the objective and the actual winning probabilities. An ordinal solution can be a good choice for one objective but not another.

We also proposed a simple and efficient heuristic algorithm for finding cardinal solutions. The results show that our heuristic can improve the objective values of ordinal solutions (more significantly when n is large). Thus it depends on how much one cares about the optimality of the solutions, and what information are available in order to decide if ordinal or cardinal solutions should be used. Moreover, the fact that our solutions approximate the upper bounds well shows that our solutions are close to optimal and our bounds are tight.

In future work, one possible extension would be to take into consideration other objective functions such as the interestingness or the fairness of the tournament. One can also use more complicated functions to better model the revenue of the tournament. Another extension is to relax the monotonicity condition of the winning probabilities between the

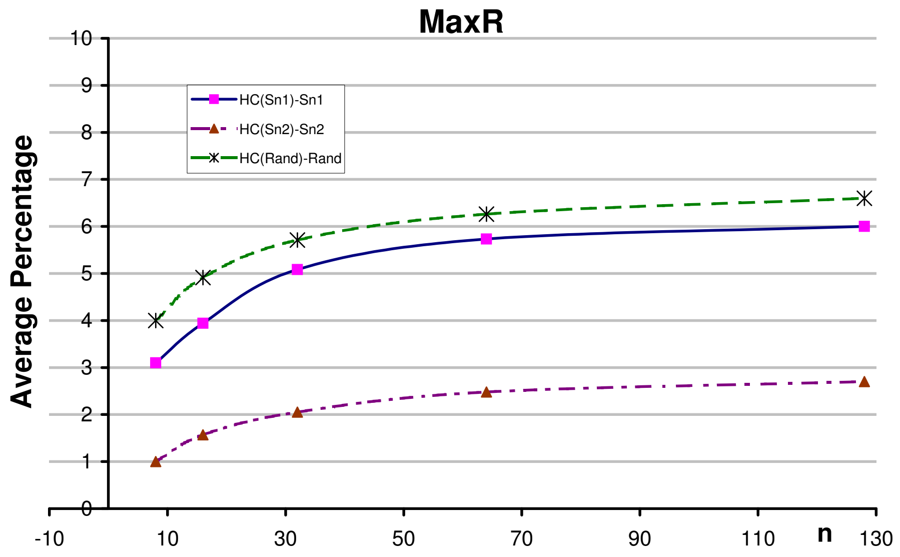

Figure 6.7: The improvement of cardinal solutions over ordinal solutions when compared to the upper bound for MaxR over 100k tournaments

players. Finally, it would be interesting to know whether there exist PAC bounds (Probably Approximately Correct) for the values of S_1^n with MaxP objective and S_2^n with MaxE objective, and what properties the winning probabilities must hold for such bounds to exist.

Chapter 7

Concluding Remarks

In this thesis, we have investigated the problems of designing knockout tournaments. We have significantly expanded the scope of the results in the literature along several dimensions: the objective function, the winning probability model, the solution type, and the size of the tournament. Specifically we focus on the following three problems:

- The complexity of the schedule control problem (i.e., finding the tournament structure and seeding that maximize the winning probability of a target player).
- The criteria for a seeding to be fair, and how to achieve them.
- The optimality of ordinal solutions in maximizing the following quantities: the predictive power of the tournament, the expected strength of the winner, and the revenue of the tournament.

In general, finding the desired design is hard and even impossible in some cases. For example, it is NP-hard to find the optimal solution for the schedule control problem when the tournament has to be balanced, or for some matrix of winning probabilities between the players, it is impossible to find an envy-free seeding. Moreover, one must be very clear about the objective function. There is no ordinal solution that performs well across all different objectives. The most common seeding used in practice (the NCAA seeding or S_2^n as in Figure 5.2) can be in fact very sub-optimal for maximizing the winning probability of the best player. Thus a knockout design must be chosen under the consideration of the objective function.

Besides ordinal solutions, we also address cardinal ones by proposing heuristic algorithms that can make use of the additional information available such as the winning probabilities between the players or their strengths. It turns out that these simple heuristics work quite well for randomly generated test cases. The experimental results show that our heuristic algorithm can improve over the objective values of ordinal solutions (more significantly when the number of players is large) and achieve close to optimal in most cases. Thus it depends on how much one cares about the optimality of the solutions, and what information are available in order to decide if ordinal or cardinal solutions should be used.

Bibliography

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

[2] K. J. Arrow, A. K. Sen, and K. Suzumura, editors. *Handbook of Social Choice and Welfare Volume 1*. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 2002.

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

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

[5] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? *Journal of the ACM*, 54(3), June 2007.

[6] E. Elkind and H. Lipmaa. Hybrid voting protocols and hardness of manipulation. In *Algorithms and Computation, 16th International Symposium, ISAAC 2005 (LNCS Volume 3827*. Springer-Verlag: Berlin, Germany, 2005.

[7] U. Endriss and J. Lang, editors. *Proceedings of the First International Workshop on Computational Social Choice Theory 2006 (COMSOC-2006)*. Amsterdam, The Netherlands, December 2006.

[8] Felix Fischer, Ariel D. Procaccia, and Alex Samorodnitsky. A new perspective on implementation by voting trees. In *EC '09*, pages 31–40, New York, NY, USA, 2009. ACM.

[9] Qiang Fu and Jingfeng Lu. The optimal multi-stage contest. *Economic Theory*, pages 1–32, 2009. 10.1007/s00199-009-0463-z.

[10] Harold N. Gabow and Robert E. Tarjan. Faster scaling algorithms for general graph matching problems. *J. ACM*, 38(4):815–853, 1991.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of* NP-*Completeness*. W. H. Freeman: New York, 1979.

[12] Christian Groh, Benny Moldovanu, Aner Sela, and Uwe Sunde. Optimal seedings in elimination tournaments. *Economic Theory*, 2009.

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

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

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

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

[17] Graham Kendall, Sigrid Knust, Celso C. Ribeiro, and Sebastin Urrutia. Scheduling in sports: An annotated bibliography. *Computers and Operations Research*, 37(1):1 – 19, 2010.

[18] Jérôme Lang, Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. Winner determination in sequential majority voting. In *IJCAI*, pages 1372–1377, 2007.

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

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

[21] Benny Moldovanu and Aner Sela. The optimal allocation of prizes in contests. *American Economic Review*, 91(3):542–58, 2001.

[22] J. W. Moon. *Topics on Tournaments*. Holt, Rinehart and Winston: New York, 1968.

[23] J. W. Moon. A problem on rankings by committees. *Econometrica*, 44(2):241–246, 1976.

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

[25] A. D. Procaccia and J. S. Rosenschein. Junta distributions and the average-case complexity of manipulating elections. *Journal of AI Research*, 28:157–181, 2007.

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

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

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

[29] Stefan Szymanski. The economic design of sporting contests. *Journal of Economic Literature*, 41(4):1137–1187, dec 2003.

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